Optimal Decision Trees for Interpretable and Constrained Clustering

POUYA SHATI, Department of Computer Science, University of Toronto, Canada and Vector Institute for Artificial Intelligence, Canada
YULIANG SONG, Department of Mechanical and Industrial Engineering, University of Toronto, Canada
ELDAN COHEN, Department of Mechanical and Industrial Engineering, University of Toronto, Canada
SHEILA A. MCILRAITH, Department of Computer Science, University of Toronto, Canada and Vector Institute for Artificial Intelligence, Canada

Constrained clustering is a semi-supervised approach to determining meaningful groupings of data that respect user-specified constraints. Such constraints are typically used to enforce desirable structural and domain-specific properties of the resulting clusters. Notably, such constraints can significantly improve the quality and accuracy of clustering. Data clustering solutions can take on many different forms. Decision trees are a particularly desirable solution form because of their inherent interpretability. Unfortunately, existing decision tree clustering approaches do not support clustering constraints and do not provide strong theoretical guarantees with respect to solution quality. To address the task of decision tree clustering with constraints, we present a novel SAT-based encoding that solves the problem to an approximated optimality in relation to a well-known bi-criteria objective. Our framework is the first exact approach for interpretable constrained clustering with decision trees. Experiments involving a range of real-world and synthetic datasets demonstrate that our approach can produce interpretable clustering solutions that are of superior quality compared to their non-interpretable counterparts, with or without the addition of constraints. We further provide new insights into the trade-off between interpretability and the satisfaction of user-specified constraints, presenting extensions to our clustering approach that treat the satisfaction of constraints as an additional optimization objective.

JAIR Track: Constraint Programming and Machine Learning

JAIR Associate Editor: Lars Kotthoff

JAIR Reference Format:
Pouya Shati, Yuliang Song, Eldan Cohen, and Sheila A. McIlraith. 2025. Optimal Decision Trees for Interpretable and Constrained Clustering. *Journal of Artificial Intelligence Research* 84, Article 4 (September 2025), 43 pages. DOI: 10.1613/jair.1.18144

1 Introduction

Clustering is an important data analysis technique whose objective is to find meaningful groupings of data within a corpora of heterogeneous data. It is a fundamental problem in ML that finds application in a diversity of settings including but not limited to medical sciences (Thalamuthu et al. 2006) and market analysis (Wu and Chou 2011). Criteria for defining data groupings vary, as do techniques for performing cluster analysis. As with many machine learning (ML) and data analysis techniques, there is increasing interest in making clustering human compatible, both by endeavoring to make the rationale for clustering of data human-interpretable, and,

Authors' Contact Information: Pouya Shati, pouya@cs.toronto.edu, Department of Computer Science, University of Toronto, Toronto, Canada and Vector Institute for Artificial Intelligence, Toronto, Canada; Yuliang Song, yl.song@mail.utoronto.ca, Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Canada; Eldan Cohen, eldan.cohen@utoronto.ca, Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Canada; Sheila A. McIlraith, sheila@cs.toronto.edu, Department of Computer Science, University of Toronto, Toronto, Canada and Vector Institute for Artificial Intelligence, Toronto, Canada.

DOI: 10.1613/jair.1.18144

where desirable, allowing domain- and/or data-analysis experts to provide additional constraints that inform the clustering process, so-called *constrained clustering*, e.g., (Babaki et al. 2014; Dinler and Tural 2016; Liu, Tao, et al. 2017; Wagstaff, Cardie, et al. 2001).

In this paper we address the problem of constrained clustering via decision trees, e.g., (Costa and Pedreira 2023; Linardatos et al. 2020; Quinlan 1986), with a view to providing mechanisms for human interpretability and intervention through the provision of clustering constraints. Clustering objectives encourage meaningful groups where points in each cluster are semantically close and points across clusters are distant. Most clustering algorithms perform a direct assignment of points to clusters or learn black-box functions akin to those employed for classification (such as deep neural models) which, depending on the classification technique, often provide little possibility of interpreting how the data is divided (Yang et al. 2021). Approaches that lack interpretability cannot easily accommodate applications that require a degree of transparency in decision making, nor can they easily accommodate applications where an expert practitioner wishes to interact in some way with the learning procedure.

Decision trees are inherently interpretable ML models that are easy for humans and computers alike to understand and to manipulate. In the past, decision trees have been used primarily for classification tasks, however recent approaches to clustering via decision trees have yielded a degree of interpretability through transparency by exposing clustering rationale in the branching structure of the tree (Bertsimas, Orfanoudaki, et al. 2021; Frost et al. 2020; Gamlath et al. 2021; Moshkovitz et al. 2020). Decision trees are interpretable in the context of clustering for the same reason that they are interpretable classifiers, i.e., they are sparse, simulatable, modular, and make informative decisions when the features are interpretable (Murdoch et al. 2019).

Approaches to learning decision trees are generally divided into those that employ local search with heuristics, e.g., (Breiman et al. 1984; Quinlan 2014, 1986), and those that employ combinatorial optimization, e.g., (Aglin et al. 2020; Avellaneda 2020; Bertsimas and J. Dunn 2017; Bessiere et al. 2009; Demirović et al. 2022; Hu et al. 2020; Narodytska et al. 2018; Verhaeghe et al. 2020; Verwer and Zhang 2019)). Local search approaches scale well in terms of performance but suffer from the lack of optimality guarantees that combinatorial optimization approaches can provide, given sufficient computational resources. A subcategory of combinatorial optimization approaches formulate the problem into a declarative standardized problem such as MaxSAT, Integer Programming (IP), and Constraint Programming (CP) and use off-the-shelf solvers to solve the encoded instances. The declarative nature of these so-called *formulation approaches* supports any constraint or specification that can be modelled in the target language (Aghaei et al. 2019), allowing, for example, the incorporation of clustering constraints.

While decision trees have proven themselves attractive models for classification and clustering, decision tree clustering techniques have not historically supported the addition of clustering constraints. Constrained clustering is a semi-supervised variant of clustering in which select domain-specific knowledge is provided to the learning algorithm as constraints to satisfy (Bilenko et al. 2004; Cohen et al. 2020; Liu, Tao, et al. 2017; Pelleg and Baras 2007). The constraints can be concerned with instance-level properties as well as more general high-level characteristics of clusters including their geometry and density. Instance-level constraints, including pairwise links, are surprisingly expressive given that a number of higher level constraints can be translated to them (Davidson and Ravi 2005b; Liu and Y. Fu 2015). Employing clustering constraints has also been shown to improve accuracy in addition to the control that it provides to the practitioner (Wagstaff and Cardie 2000; Wagstaff, Cardie, et al. 2001).

The declarative nature of formulation approaches to clustering is particularly well-suited to constrained clustering. Indeed, approaches that are based on combinatorial optimization formulations have been successful in tackling the non-tree variation of the constrained clustering problem (Babaki et al. 2014; Berg and Järvisalo 2017; Dao, Duong, et al. 2017; Davidson, Ravi, and Shamis 2010). It has also been shown that local search approaches can be used to solve decision tree clustering (Frost et al. 2020; Gamlath et al. 2021; Moshkovitz et al. 2020). However, they can lead to suboptimal solutions and lack support for constraints. Learning decision trees as

combinatorial optimization problems has proven to be an elusive goal in solving clustering or its constrained counterpart. Search-based, e.g., (Aglin et al. 2021), and formulation, e.g., (Bertsimas, Orfanoudaki, et al. 2021), methods have been proposed but shown to scale extremely poorly compared to analogous approaches for decision tree classification (Ignatiev et al. 2021).

We propose the first combinatorial optimization approach to learning decision trees that is effective for clustering problems, with or without clustering constraints. We present a novel SAT-based model that employs direct encoding of non-binary features based on the technique presented in Shati et al. 2023b. This approach optimizes towards a guaranteed approximation of a well-known bi-criteria clustering objective, and also supports pairwise clustering constraints. Our direct encoding of non-binary features is particularly well-suited to clustering problems whose instances involve numerical and categorical data. To reduce the size of the SAT instance, we interconnect the encoding of the objective and the encoding of the constraints, enabling us to deploy pruning techniques. Furthermore, we study feasibility against quality trade-offs and ways to address infeasibility by treating constraint satisfaction as an additional optimization objective. Lastly, we provide an iterative approach to more comprehensively optimize the bi-criteria approach.

Our main contributions are as follows:

(1) We present a MaxSAT encoding with a direct encoding of numerical features for solving the problem of decision tree clustering. Our encoding guarantees an ϵ-approximation of a Pareto optimal solution in maximum diameter (MD) and minimum split (MS) criteria. We add support for pairwise constraints as must-links and cannot-links to enable solving the constrained clustering problem.
(2) We integrate the encoding of pairwise links with that of the objectives to detect and prune infeasible and redundant clauses, in a novel method referred to as the Smart Pairs algorithm.
(3) We introduce soft pairwise constraints that are encouraged, rather than required, to be satisfied. Soft must-links and cannot-links help address the issue of infeasibility when no solution is found to satisfy all hard constraints in a given depth. We propose 1-stage, 2-stage, and 3-stage schemes to optimizing soft constraints.
(4) We detail how our encoding and support for constraints can be utilized towards producing approximations of all solutions that are Pareto optimal in $[\mathrm{MD}, \mathrm{MS}]$, rather than an arbitrary one.
(5) We perform an extensive set of experiments and show that: 1) Our approach is able to find interpretable solutions of high quality in short runtimes; 2) Employing tree clustering, the $[\mathrm{MD}, \mathrm{MS}]$ objective, or user-provided constraints improves the solutions in evaluation metrics, with a combination of two or all of them leading to an even more significant improvement; 3) There is a trade-off between solution accuracy and infeasibility in the presence of user-provided constraints; 4) Soft constraints enable us to benefit from a higher number of constraints and further improve accuracy; 5) Approximating the Pareto front as a whole enables us to find higher quality solutions in less constrained instances; 6) The objective approximation and Smart Pairs pruning algorithm significantly boost performance leading to higher quality solutions.

This paper significantly extends and enhances Shati et al. 2023a. It expands on the original work with support for soft constraints in multiple optimization schemes, complete Pareto front approximation, and a more comprehensive set of experimental studies.

2 Related Works

In this section, we review the related literature on clustering algorithms and decision trees. In Section 2.1, we discuss constrained clustering approaches. In Section 2.2, we discuss how decision trees have been learned as solutions to the clustering problem. In Section 2.3, we discuss a method for learning decision trees which our work builds upon.

2.1 Constrained Clustering

The problem of constrained clustering without decision tree restriction has been solved using combinatorial optimization before. One prominent example is the framework presented in Dao, Duong, et al. 2017. Their approach supports pairwise links as well as cluster-level constraints including bounds on the size and local density of clusters. Their focus is on the bi-criteria objective of maximum diameter and minimum split and the authors demonstrate that their approach can be embedded in a complete Pareto front generation algorithm. They use Constraint Programming (CP) and global constraints to outperform the two well-known Branch-and-Bound and Graph Coloring baselines, with the added benefit of CP expressiveness enabling the encoding of more complex constraints.

Constrained clustering can also be solved by local search approaches, usually based on k-means. The CVQE algorithm (Davidson and Ravi 2005b) extends k-means with constraint support by adding penalties for violated constraints to the error function. The MPCK-means algorithm (Bilenko et al. 2004) is also based on k-means and integrates pairwise constraints with unsupervised data through learning distance metrics. Other clustering approaches such as hierarchical (Davidson and Ravi 2005a) and spectral (Wang and Davidson 2010) have also been extended with support for user-specified constraints.

2.2 Decision Tree Clustering

Requiring a clustering solution to conform to a decision tree adds interpretability by revealing how each point is assigned to a cluster and which features impacted the decision making. However, the restriction on the solution space also comes with unavoidable cost to the clustering objective. Recent literature has proven lower bounds for the increase in cost and proposed increasingly tight upper bounds via algorithms with guaranteed approximation factors. Moshkovitz et al. 2020 shows that local search approaches can have arbitrarily large cost and proves a $\Omega(\log k)$ lower bound factor for k-means and k-medians objectives. They further present algorithms for $O(k)$-approximation of k-medians and $O(k^2)$-approximation of k-means. Gamlath et al. 2021 later improved the upper bounds to $O(\log^2 k)$ and $O(k \log^2 k)$ for the k-medians and k-means objectives respectively and proved new lower bound of $\Omega(k)$ for k-means. Relaxed variations of decision trees such as oblique ones are not subject to these limitations and can be learned to represent partitions founds by non-tree algorithms such as k-means (Gabidolla and Carreira-Perpiñán 2022). Relaxing the decision tree structure could also be beneficial in enabling more efficient learning algorithms, as is the case with soft decision trees and continuous optimization (Cohen 2023).

The DL8.5 algorithm presented in Aglin et al. 2020 is a decision tree learning approach based on Branch-and-Bound search and Dynamic Programming. DL8.5 can optimize an additive cost function for leaves by recursively exploring all possible splits within a decision tree. In addition to bounding and pruning the search space, caching is used to improve the search performance through exploiting the fact that the same subset of data might appear in multiple subproblems. DL8.5 was originally introduced for the problem of classification but was later shown to also work with an additive cost for clustering (Aglin et al. 2021). The cost being optimized is the sum of Euclidean distance for each point and the centroid of its cluster. DL8.5 for clustering does not support a cluster to be spread across multiple leaves and neither can it naturally support constraints due to its search-based framework.

A combinatorial optimization approach to the decision tree clustering problem without constraints is presented in Bertsimas, Orfanoudaki, et al. 2021. Building upon the authors' previous decision tree classification work (Bertsimas and J. Dunn 2017), the problem is formulated as a Mixed Integer Optimization (MIO) instance that optimizes the Silhouette Metric (Rousseeuw 1987). However, the authors highlight the lack of scalability of their formulation and instead focus on a heuristic local search algorithm. Their iterative coordinate-descent approach is limited to one leaf per cluster and approximates a solution in the Silhouette Metric (Rousseeuw 1987) or the Dunn index (J. C. Dunn 1974) validation criteria without any approximation guarantees.

2.3 SAT Encoding of Decision Trees

The method presented in Shati et al. 2021 and its extension (Shati et al. 2023b) aims to learn decision trees via SAT for the problem of classification. Two variations of the SAT encoding are used for learning minimum depth and maximum accuracy decision trees, two well-known optimization problems for tree classifiers. They employ a direct encoding of numerical and categorical features, addressing the computational cost of binarizing all features in advance. The direct encoding of categorical values further improves the expressiveness of the tree, enabling more compact solutions. Their method is shown to outperform SAT and CP baselines in terms of performance and solution quality.

3 Preliminaries

In this section, we define the problem of constrained clustering with decision trees as solutions. In Section 3.1, we define decision trees and how they can be used to assign data points to clusters. In Section 3.2, we define user-provided pairwise constraints and the bi-criteria objective to subsequently define the two variations of the constrained clustering problem which restrict and do not restrict the solution to be a decision tree, respectively. In Section 3.3, we show that deep enough trees are sufficiently expressive to represent any clustering. In Section 3.4, we describe external evaluation metrics that can be used to assess the quality of clustering solutions. In Section 3.5, we describe the declarative combinatorial optimization problem of MaxSAT.

3.1 Decision Trees

We define decision trees by first defining branchings and tree structures as a basis.

Definition 1 (Branching). *Given a finite set of features F, a branching is a division of data points into two groups. A branching can either operate on a binary feature $f_b \in F$ or a binarization of a numerical feature $f_n \in F$ through pairing with a threshold $a \in \mathbb{R}$. For $x[f_n]$ which represents the value of numerical feature f_n for data point x, being smaller or equal to the threshold ($x[f_n] \leq a$) equates to a binary value of* 1 *while being larger ($x[f_n] > a$) equates to* 0.

Note that a binary feature can be seen as a special case of a numerical one. Henceforth, we will only use numerical features to be concise, unless noted otherwise. Furthermore, we will only use the more general representation of a branching, which is a feature paired with a threshold.

Definition 2 (Tree Structure). *A tree structure $\mathcal{T}$ is a rooted directed tree. Nodes of $\mathcal{T}$ are divided into leaf nodes $\mathcal{T}_L$ and non-leaf nodes that are referred to as branching nodes $\mathcal{T}_B$ and contain the root node $\delta \in \mathcal{T}_B$. Each branching node has exactly two children orienting away from the root represented by the left and right children functions $l, r : \mathcal{T}_B \to (\mathcal{T}_B \cup \mathcal{T}_L)$. The parent function $p : (\mathcal{T}_B \cup \mathcal{T}_L - \{\delta\}) \to \mathcal{T}_B$ represents the edges in reverse $\forall t_p, t_c : p(t_c) = t_p \Leftrightarrow (l(t_p) = t_c \vee r(t_p) = t_c)$.*

Definition 3 (Decision Tree). *Given a set of features F and integer labels $[1..k]$, a decision tree is a tuple $\mathcal{D} = (\mathcal{T}, \beta, \alpha, \theta)$ where $\mathcal{T} = (\mathcal{T}_B, \mathcal{T}_L, \delta, p, l, r)$ is a tree structure, $\beta : \mathcal{T}_B \to F$ is the feature selection function, $\alpha : \mathcal{T}_B \to \mathbb{R}$ is the threshold selection function, and $\theta : \mathcal{T}_L \to [1..k]$ is the leaf labelling function.*

Next, we describe how decision trees assign data point to clusters. This process involves entering the data point at the root and iteratively delivering it to the left or right child based on its values until it reaches a leaf node labelled with a cluster.

Definition 4 (Decision Tree Label Assignment). *Given a set of features F, clusters $[1..k]$, a data point $x \in \mathbb{R}^{|F|}$ (an evaluation of F), and a decision tree $\mathcal{D} = (\mathcal{T}, \beta, \alpha, \theta)$, the value of the function $\Theta_{\mathcal{D}}(\delta, x)$ (simplified as $\Theta_{\mathcal{D}}(x)$) represents the cluster to which $\mathcal{D}$ assigns x. The function $\Theta_{\mathcal{D}} : (\mathcal{T}_B \cup \mathcal{T}_L) \times \mathbb{R}^{|F|} \to [1..k]$ is recursively*

defined as follows:

$$\Theta(t,x) = \begin{cases} \theta(t) & \text{if } t \in \mathcal{T}_L \\ \Theta(l(t),x) & \text{elseif } x[\beta(t)] \leq \alpha(t) \\ \Theta(r(t),x) & \text{else} \end{cases}$$

Every step of the recursion corresponds to a branching. A value of 1 *(*0*) for the branching leads the point to be directed to the left (right) child. Every node* t *invoked with a point* x *is said to contain that point. A leaf node assigns all of the data points that it contains to the cluster corresponding to its label.*

Lastly, we define the depth parameter of a decision tree and the property of being fully balanced.

Definition 5 (Decision Tree Depth). *Given a decision tree* $\mathcal{D} = (\mathcal{T}, \beta, \alpha, \theta)$*, we recursively define the depth of each node* $t \in \mathcal{T}_B \cup \mathcal{T}_L$ *as* $depth(t) = depth(p(t)) + 1$ *with* $depth(\delta) = 0$*. The depth of* $\mathcal{D}$ *is then defined to be the maximum depth among its leaf nodes* $depth(\mathcal{D}) = \max_{t \in \mathcal{T}_L} depth(t)$*.*

Definition 6 (Fully Balanced Decision Tree). *A decision tree* $\mathcal{D} = (\mathcal{T}, \beta, \alpha, \theta)$ *is said to be fully balanced if all of its leaf nodes have the same depth, i.e.,* $\forall t_1, t_2 \in \mathcal{T}_L : depth(t_1) = depth(t_2)$*.*

3.2 Problem Definition

If a decision tree $\mathcal{D}$ is intended as a clustering solution, $\Theta_{\mathcal{D}}(x)$ is interpreted as the cluster in which x is placed. Two points that are assigned to the same cluster are said to be clustered *together* or *co-clustered*, while two points that are assigned to different clusters are said to be clustered *separately*. Furthermore, given a dataset $X \subseteq \mathbb{R}^{|F|}$, a decision tree $\mathcal{D}$ is said to have non-empty clusters if $\forall k \in [1..k] : \sum_{x_i \in X} \mathbb{1}(\Theta_{\mathcal{D}}(x_i) = k) \geq 1$. A decision tree is assumed to have non-empty clusters unless noted otherwise. Consistent with recent work (Frost et al. 2020), each cluster is allowed to spread across multiple leaves to improve expressiveness and the ability to satisfy user-provided constraints.

We define how a clustering solution can satisfy pairwise constraints and introduce the two components of our objective. As our main clustering objective, we consider the well-known bi-criteria objective of minimizing the maximum diameter and maximizing the minimum split (Dao, Duong, et al. 2017). Specifically, we consider a bounded approximation parameterized by ϵ such that $\epsilon = 0$ indicates a Pareto optimal solution for [MD, MS].

Definition 7 (Must-links and Cannot-links). *Given a dataset* $X \subset \mathbb{R}^{|F|}$*, must-link and cannot-link pairwise constraints are each a set of pairs of points* $ML, CL \subseteq X \times X$*. A clustering* Θ *is said to satisfy* ML *and* CL *if it clusters must-links together and cannot-links separately,*

$$\forall (x_1, x_2) \in ML : \Theta(x_1) = \Theta(x_2) \tag{1}$$

$$\forall (x_1, x_2) \in CL : \Theta(x_1) \neq \Theta(x_2). \tag{2}$$

Definition 8 (Maximum Diameter and Minimum Split). *Given a clustering* Θ *and a dataset* $X \subset \mathbb{R}^{|F|}$*, the minimum split (*$\text{MS}_\Theta$*) is the shortest distance between a pair of points that are clustered separately, and the maximum diameter (*MD_Θ*) is the longest distance between a pair of points that are clustered together:*

$$\text{MD}_\Theta = max(\{|x_1 - x_2| \mid x_1, x_2 \in X, \Theta(x_1) = \Theta(x_2)\}) \tag{3}$$

$$\text{MS}_\Theta = min(\{|x_1 - x_2| \mid x_1, x_2 \in X, \Theta(x_1) \neq \Theta(x_2)\}). \tag{4}$$

We assume $|x_i - x_j|$ *to be the Euclidean distance. However the objectives can also be defined for any other metric space and its corresponding distance function. Our approach throughout the rest of the paper remains applicable in any such setting.*

Next, we define the notions of optimality and order when optimizing a bi-criteria objective.

Definition 9 (Pareto Optimality). *A clustering Θ is said to dominate another clustering Θ' if it achieves a better or equal value in one objective and strictly better value in the other.*

$$(\mathrm{MD}_\Theta < \mathrm{MD}_{\Theta'} \wedge \mathrm{MS}_\Theta \geq \mathrm{MS}_{\Theta'}) \vee (\mathrm{MD}_\Theta \leq \mathrm{MD}_{\Theta'} \wedge \mathrm{MS}_\Theta > \mathrm{MS}_{\Theta'})$$

A clustering that is not dominated by any other clustering is said to be Pareto optimal. Two solutions are equivalent if they have the same objective values. A set of Pareto optimal solutions is considered complete if it contains at least one equivalent for every Pareto optimal solution. The set of objective value pairs corresponding to a complete Pareto optimal set is referred to as the Pareto front.

Next, we formally define the approximated decision tree clustering problem which aims to find an approximation of a Pareto optimal solution.

Problem 1 (Decision Tree Clustering). *Given a set of numerical features F, clusters $[1..k]$, dataset $X \subset \mathbb{R}^{|F|}$, must-link and cannot-link constraints $ML, CL \subseteq X \times X$, fixed depth d, and approximation parameter ϵ, find a fully balanced decision tree $\mathcal{D}$ of depth d, such that:*

(1) *$\Theta_\mathcal{D}$ satisfies the constraints ML and CL;*
(2) $\mathrm{MD}_\mathcal{D} \leq \mathrm{MD}_{\mathcal{D}^*} + \epsilon$;
(3) $\mathrm{MS}_\mathcal{D} \geq \mathrm{MS}_{\mathcal{D}^*} - \epsilon$,

where $\mathrm{MD}_\mathcal{D}$ and $\mathrm{MS}_\mathcal{D}$ are shorthands for the maximum diameter and minimum split of the clustering represented by $\mathcal{D}$ and $\mathcal{D}^*$ is an arbitrary Pareto optimal solution to the problem with respect to the bi-criteria of maximizing MS *and minimizing* MD.

The variation of Problem 1 with no restriction for the solution to conform to a decision tree, is simply referred to as the problem of constrained clustering (CC).

Problem 2 (Constrained Clustering). *Given a set of numerical features F, clusters $[1..k]$, dataset $X \subset \mathbb{R}^{|F|}$, must-link and cannot-link constraints $ML, CL \subseteq X \times X$, and approximation parameter ϵ, find a clustering Θ such that:*

(1) *Θ satisfies the constraints ML and CL;*
(2) $\mathrm{MD}_\Theta \leq \mathrm{MD}_{\Theta^*} + \epsilon$;
(3) $\mathrm{MS}_\Theta \geq \mathrm{MS}_{\Theta^*} - \epsilon$,

where Θ^* is an arbitrary Pareto optimal solution to the problem with respect to the bi-criteria of maximizing the minimum split and minimizing the maximum diameter.

By removing the minimum split component from the bi-criteria objective, we obtain variations of Problem 1 and Problem 2 that solely focus on the maximum diameter. To define the MD-only variations, we can simply remove the third condition (respectively $\mathrm{MS}_\mathcal{D} \geq \mathrm{MS}_{\mathcal{D}^*} - \epsilon$ and $\mathrm{MS}_\Theta \geq \mathrm{MS}_{\Theta^*} - \epsilon$) from the definitions of Problem 1 and Problem 2.

3.3 Completeness

Problem 1 is not guaranteed to be feasible due to two potential reasons: (1) the sets of must-links and cannot-links are contradictory (e.g., $ML \cap CL \neq \emptyset$); (2) there exists an arbitrary clustering Θ satisfying the constraints but no such clustering can be represented by a decision tree of a given depth. In case of the latter and consistent with established literature (Shalev-Shwartz and Ben-David 2014), we show that there always exists a sufficiently deep tree that represents any consistent labelling. Note that a given labelling could come from a ground-truth clustering solution or, if the decision tree is intended for classification, supervised labels of a training set.

Definition 10 (Consistent Labelling). *Given a set of features F, integer labels $[1..k]$, and a set of data points $x_i \in X$, a labeling $\gamma : X \rightarrow [1..k]$ of dataset X is consistent if and only if there are no two identical points that are assigned different labels, i.e.,*

$$\neg\exists x_1, x_2 \in X : (\forall f \in F : x_1[f] = x_2[f]) \wedge \gamma(x_1) \neq \gamma(x_2). \tag{5}$$

Theorem 1. *Given a set of features F and integer labels $[1..k]$, a set of data points $x_i \in X$, and a labelling $\gamma : X \rightarrow [1..k]$ such that γ is a consistent labelling of X, there always exists a fully balanced tree $\mathcal{D}$ with sufficiently large $depth(\mathcal{D})$ that completely matches the given labelling $\forall x \in X : \Theta_{\mathcal{D}}(x) = \gamma(x)$.*

Proof. We provide a constructive proof for the theorem. Our proof starts with a single root node δ and iteratively deepens the tree until the labels of the points contained in each leaf node are homogeneous with regard to their cluster as specified by the ground-truth. Finally, each leaf node is labeled with the cluster of the points that it contains. The procedure is formally described below.

(1) Initiate the tree structure with a single root node δ as a leaf and the decision tree with empty α and β functions.[1]
(2) If there is no leaf node t such that $\exists x_1, x_2 \in \Psi(t) : \gamma(x_1) \neq \gamma(x_2)$, go to (7).
(3) For every leaf node t, replace t with a branching node t_p and two leaf nodes t_l and t_r as its left and right children. If $\exists x_1, x_2 \in \Psi(t) : \gamma(x_1) \neq \gamma(x_2)$, go to (4). If not, go to (5).
(4) Due to consistent labeling, we know there exists feature f such that $x_1[f] \neq x_2[f]$. Set $\beta(t_p) = f$ and $\alpha(t_p) = min(x_1[f], x_2[f])$. Go to (6).
(5) Set $\beta(t_p)$ and $\alpha(t_p)$ to any arbitrary valid value.
(6) Go to (2).
(7) For each leaf node t, set $\theta(t) = \gamma(x)$ where $x \in \Psi(t)$. In case the leaf does not contain any points, label the leaf arbitrarily.
(8) Output the constructed decision tree.

This procedure is guaranteed to halt since the number of pairs of data points that are contained in the same node but have different labels is strictly decreasing at each iteration. Moreover, given how the leaves are labeled, the resulting tree is guaranteed to assign $\gamma(x)$ to every data point x. □

By using Theorem 1 and setting $\gamma = \Theta$, we conclude that Problem 1 is always feasible with deep enough trees if there exists an arbitrary clustering with consistent labelling satisfying the pairwise constraints.

3.4 Clustering Evaluation Metrics

Consistent with previous work on constrained clustering (Berg and Järvisalo 2017; Cohen et al. 2020; Dao, Vrain, et al. 2016; Davidson, Ravi, and Shamis 2010), we employ external evaluation of clustering solutions based on given ground-truth labels. We use two well-known metrics that determine how closely clustering solutions resemble the ground truth. Specifically, we maximize the Adjusted Rand Index (ARI) (Hubert and Arabie 1985) and the Normalized Mutual Information (NMI) (Strehl and Ghosh 2002). These metrics represent the accuracy of a given clustering solution since they quantify the similarity to the ground truth.

Given a ground-truth labeling $\hat{\Theta}$ and a clustering solution Θ, the ARI metric is calculated from the number of pairs that are co-clustered by both ($\sigma^{\hat{\Theta},\Theta}$), by neither ($\sigma$), only by $\hat{\Theta}$ ($\sigma^{\hat{\Theta}}$), and only by Θ (σ^{Θ}).

[1] Note that the root is by definition a branching node (Definition 2). However, in this procedure, we treat it as a leaf node at the start. Since for any non-trivial dataset this procedure is guaranteed to at least iterate once, we are guaranteed to replace the leaf root node with a branching root node.

$$\text{ARI} = \frac{2(\sigma^{\hat{\Theta},\Theta}.\sigma - \sigma^{\Theta}.\sigma^{\hat{\Theta}})}{(\sigma^{\hat{\Theta},\Theta} + \sigma^{\hat{\Theta}})(\sigma^{\hat{\Theta}} + \sigma) + (\sigma^{\hat{\Theta},\Theta} + \sigma^{\Theta})(\sigma^{\Theta} + \sigma)}$$

The NMI metric is the normalized version of the Mutual Information metric. It is calculated using the entropy of the ground-truth labeling ($H(\hat{\Theta})$), the entropy of the clustering ($H(\Theta)$), and the conditional entropy of ground-truth given the clustering ($H(\hat{\Theta}|\Theta)$).

$$\text{NMI} = \frac{2.[H(\hat{\Theta}) - H(\hat{\Theta}|\Theta)]}{[H(\hat{\Theta}) + H(\Theta)]}$$

3.5 MaxSAT

Boolean satisfiability (SAT) was the first problem to be proven to be NP-complete. SAT remains an integral part of proving new problems to be NP-hard. A SAT instance in Conjunctive Normal Form (CNF) is composed of boolean variables, literals that are boolean variables or their negation, and clauses that are disjunction of literals. In order to solve a SAT instance, one would need to find a truth assignment to the variables that satisfies the conjunction of clauses (Min Li and Manya 2009).

Maximum satisfiability (MaxSAT) is the optimization variation of SAT that aims to maximize the number of satisfied clauses. Partial MaxSAT (Z. Fu and Malik 2006) is a generalization of both SAT and MaxSAT. In partial MaxSAT, clauses are divided into hard and soft subgroups. Solving an instance of partial MaxSAT amounts to finding a truth assignment that satisfies all hard clauses and maximizes the number of soft clauses satisfied. Hereafter, the simplified term clause is used to refer to hard clauses, and MaxSAT is used to refer to the partial MaxSAT problem. SAT and MaxSAT are declarative combinatorial problems that can be used to formulate and solve other NP problems by using their highly efficient off-the-shelf solvers.

4 Distance Classes

In this section, we introduce *distance classes*, a grouping of pairs of data points based on their distance. These classes help us approximate the maximum diameter and minimum split values, leading to a simpler optimization of the objectives.

To encode the ϵ-approximation of the two objectives in Problem 1, we divide the set of all pairs of data points based on their distance into μ groups called distance classes $\hat{D} = \{D_1, D_2, ..., D_\mu\}$. Each distance class covers a continuous range of values from a lower to an upper bound, containing a pair if and only if its distance falls into that interval. Distance class intervals are non-overlapping and the smallest and largest distances of pairs within each class are less than ϵ apart. Any method of grouping pairs that satisfies the requirements is sound with regard to the approximation. However, we employ the simplest approach of sorting all pairs based on their distance and greedily adding them one by one, creating new classes as needed to guarantee that the distances in each class are at most ϵ apart.

The purpose of distance classes are to ensure that all pairs in the same class are treated similarly with regard to being clustered together or not. We consider the index λ^+ such that any pair of data points in distance classes $D_1..D_{\lambda^+}$ must be clustered together (Eq. (6)). Similarly, we consider the index λ^- such that any pair in distance classes $D_{\lambda^-+1}..D_\mu$ must be clustered separately (Eq. (7)). The pairs in distance classes in-between the two indices $D_{\lambda^++1}..D_{\lambda^-}$ are free to be clustered together or separately on an individual level.

$$((x_1, x_2) \in D_w, w \leq \lambda^+) \rightarrow \Theta_{\mathcal{D}}(x_1) = \Theta_{\mathcal{D}}(x_2) \tag{6}$$

$$((x_1, x_2) \in D_w, w > \lambda^-) \rightarrow \Theta_{\mathcal{D}}(x_1) \neq \Theta_{\mathcal{D}}(x_2) \tag{7}$$

4.1 ϵ-Pareto Optimality

We now use distance classes to describe how we can approximate the MD and MS values. Note that in any feasible clustering, for all λ^+ and λ^- values that satisfy Eq. (6) and Eq. (7), we have that MS $\geq min(\{|x_1 - x_2| \,|\, (x_1, x_2) \in D_{\lambda^+ + 1}\})$ and MD $\leq max(\{|x_1 - x_2| \,|\, (x_1, x_2) \in D_{\lambda^-}\})$. We therefore maximize λ^+ as a proxy for maximizing the minimum split and minimize λ^- as a proxy for minimizing the maximum diameter in Problem 1. Proposition 1 shows that optimizing λ^+ and λ^- as proxy objectives will lead to guaranteed approximation of [MD, MS].

Proposition 1 (ϵ-approximation of [MD, MS]). *Let $\mathcal{D}$ be a decision tree clustering, if $\mathcal{D}$ is a Pareto optimal solution with regard to minimizing λ^- and maximizing λ^+ then $\mathcal{D}$ is an ϵ-Pareto optimal solution with regard to the bi-criteria* [MD, MS] *objective as defined in Problem 1.*

Note that Pareto optimality in λ^+ and λ^- is defined analogous to Pareto optimality in MS and MD (Definition 9).

Proof. We show that there always exists a solution $\mathcal{D}^*$ that is Pareto optimal in MD and MS and is within the ϵ-neighborhood of $\mathcal{D}$ in the two objectives.

- Let $\text{MS}_{\mathcal{D}}$ ($\text{MD}_{\mathcal{D}}$) be the minimum split (maximum diameter) achieved by the solution $\mathcal{D}$. Moreover, let $\lambda^+_{\mathcal{D}}$ and $\lambda^-_{\mathcal{D}}$ be the optimized values of λ^+ and λ^- corresponding to solution $\mathcal{D}$.
- To prove by contradiction, assume that no such $\mathcal{D}^*$ exists. As a result, there should exist another solution $\mathcal{D}'$ that dominates $\mathcal{D}$ and is not in its ϵ neighborhood.
- Assume, without loss of generality, that $\text{MS}_{\mathcal{D}'} > \text{MS}_{\mathcal{D}} + \epsilon$ and $\text{MD}_{\mathcal{D}'} \leq \text{MD}_{\mathcal{D}}$.
- The pairs corresponding to values $\text{MS}_{\mathcal{D}'}$ and $\text{MS}_{\mathcal{D}}$ cannot be in the same distance class due to being more than ϵ far apart. Thus, $\text{MS}_{\mathcal{D}'} \in D_w$ where $w > \lambda^+_{\mathcal{D}} + 1$ and $\lambda^+_{\mathcal{D}} + 1$ is a valid λ^+ value for $\mathcal{D}'$.
- Since $\text{MD}_{\mathcal{D}'} \leq \text{MD}_{\mathcal{D}}$, $\lambda^-_{\mathcal{D}}$ is also a valid λ^- value for $\mathcal{D}'$.
- Therefore, $\mathcal{D}'$ dominates $\mathcal{D}$ in optimizing λ^+ and λ^- values as well. Leading to contradiction as $\mathcal{D}$ is a Pareto optimal solution with regard to minimizing λ^- and maximizing λ^+.
- We conclude that there exists $\mathcal{D}^*$ that satisfies the conditions described in Problem 1. Therefore, $\mathcal{D}$ is an ϵ-approximation of a Pareto optimal solution in MD and MS.

□

5 SAT-based Encoding for ϵ-optimal Decision Tree Clustering

In this section we present our novel approach to formulate Problem 1 into a SAT instance. As part of our approach, we reuse the direct SAT encoding of numerical branchings which was first presented in Shati et al. 2023b for decision tree classifiers. We assume we are given a number of clusters k and a fully balanced tree structure $\mathcal{T}$ of depth d and learn the values of β, α, and θ functions. The assumption of a fully balanced structure is not limiting as it encapsulates and is more expressive than all unbalanced structures of lower or equal depth. The alternative approach of considering unbalanced structures directly requires a decision to be made in advance or as part of the optimization. In Section 5.1, we introduce the variables used to encode different aspects of a tree clustering solution. In Section 5.2, we present the hard clauses that guarantee the validity of our encoding and the soft clauses that model our objectives.

5.1 Variables and Structure

We use the following set of boolean variables in our encoding to represent the β, α, and θ functions of the decision tree, the λ^+ and λ^- values, and how the data points are clustered. The decision tree and how it operates on data points is encoded similar to Shati et al. 2023b. To represent the cluster assignment to leaves and data points, we use a unary encoding (with variables $g_{t,c}$ and $x_{i,c}$, respectively). More specifically, the assigned cluster is determined by the number of variables set to true along a sequence of length $k-1$. A well-formed unary encoding

should not include the sub-sequence 01 at any point. We use unary encoding to enable symmetry-breaking between equivalent cluster assignments. As an example, if we take all data points from the first cluster in an arbitrary clustering, assign them to the second, and vice-versa, we will end up with an equivalent clustering. Adding symmetry-breaking to the encoding can help avoid consideration of equivalent solutions, and improve the performance as a result. We also use unary encoding b_w^- and b_w^+ variables to model λ^- and λ^+ variables, respectively. Unary encoding helps us easily determine where each distance class is located in the division based on λ^- and λ^+ values.

- $\{a_{t,j} \mid t \in \mathcal{T}_B, j \in F\}$: Represents whether feature j is chosen at branching node t. It is equivalent to $\beta(t) = j$.
- $\{s_{i,t} \mid x_i \in X, t \in \mathcal{T}_B\}$: Represents whether point x_i is directed towards the left child, if it passes through branching node t. It is equivalent to $x_i[\beta(t)] \leq \alpha(t)$.
- $\{z_{i,t} \mid x_i \in X, t \in \mathcal{T}_L\}$: Represents whether point x_i ends up at leaf node t.
- $\{g_{t,c} \mid t \in \mathcal{T}_L, 1 < c \leq k\}$: Represents whether the cluster assigned to leaf t is or comes after c. It is equivalent to $\theta(t) \geq c$.
- $\{x_{i,c} \mid x_i \in X, 1 < c \leq k\}$: Represents whether the cluster assigned to point x_i is or comes after c. It is equivalent to $\Theta_{\mathcal{D}}(x_i) \geq c$.
- $\{b_w^- \mid 1 \leq w \leq \mu\}$: Represents (the negation of) whether the pairs in distance class D_w should be clustered separately. It is equivalent to $w \leq \lambda^-$
- $\{b_w^+ \mid 1 \leq w \leq \mu\}$: Represents whether the pairs in class D_w should be clustered together. It is equivalent to $w \leq \lambda^+$.

5.2 Clauses

We encode the construction of the decision tree (Eq. (8) to Eq. (16)) following the approach first presented in Shati et al. 2023b which learned decision trees as classifiers. The direct support for numerical features in this approach lends itself well to our purposes, since numerical features are prevalent in clustering problems. See the original paper for a detailed description of the clauses in Eq. (8) to Eq. (16).

The following clauses guarantee that exactly one feature is chosen at each branching node (Eq. (8) and Eq.(9)), the points are directed to the left or right child of each branching node based on their value of the chosen feature (Eq. (10) and Eq. (11)), the appearance of points at leaves correctly corresponds to the path that they are directed through within the tree (Eq. (12), Eq. (13), and Eq. (14)), and thresholds are non-trivial (Eq. (15) and Eq. (16)). Note that the set $O_j(X)$ contains all consecutive pairs of points when ordered according to feature j and $O_j^=(X)$ is its subset limiting the members to only those that have equal j values. Moreover, the set $A_l(t)$ ($A_r(t)$) contains all nodes that have t as descendent of their left (right) child. Lastly, $\#_j^1$ and $\#_j^{|X|}$ represent the indices of the first and last points when ordered by j values, respectively.

$$(\neg a_{t,j}, \neg a_{t,j'}) \qquad \forall t \in \mathcal{T}_B, j \neq j' \in F \tag{8}$$

$$(\bigvee_{j \in F} a_{t,j}) \qquad \forall t \in \mathcal{T}_B \tag{9}$$

$$(\neg a_{t,j}, s_{i,t}, \neg s_{i',t}) \qquad \forall t \in \mathcal{T}_B, j \in F, (i,i') \in O_j(X) \tag{10}$$

$$(\neg a_{t,j}, \neg s_{i,t}, s_{i',t}) \qquad \forall t \in \mathcal{T}_B, j \in F, (i,i') \in O_j^{=}(X) \tag{11}$$

$$(\neg z_{i,t}, s_{i,t'}) \qquad \forall t \in \mathcal{T}_L, x_i \in X, t' \in A_l(t) \tag{12}$$

$$(\neg z_{i,t}, \neg s_{i,t'}) \qquad \forall t \in \mathcal{T}_L, x_i \in X, t' \in A_r(t) \tag{13}$$

$$(z_{i,t}, \bigvee_{t' \in A_l(t)} \neg s_{i,t'}, \bigvee_{t' \in A_r(t)} s_{i,t'}) \qquad \forall t \in \mathcal{T}_L, x_i \in X \tag{14}$$

$$(\neg a_{t,j}, s_{\#_j^1,t}) \qquad \forall t \in \mathcal{T}_B, j \in F \tag{15}$$

$$(\neg a_{t,j}, \neg s_{\#_j^{|X|},t}) \qquad \forall t \in \mathcal{T}_B, j \in F \tag{16}$$

The following clauses extend the basic decision tree encoding to support ϵ-optimal clustering trees that satisfy the *ML* and *CL* constraints. The clauses in Eq. (17) guarantee well-formed unary encoding of cluster labels in each leaf.

$$(g_{t,c}, \neg g_{t,c+1}) \qquad \forall t \in \mathcal{T}_L, c \in [2..k-1] \tag{17}$$

The clauses in Eq. (18)) and Eq. (19) guarantee that the label assigned to each data point matches the label of the leaf that contains the point.

$$(\neg z_{i,t}, \neg g_{t,c}, x_{i,c}) \qquad \forall t \in \mathcal{T}_L, x_i \in X, c \in [2..k] \tag{18}$$

$$(\neg z_{i,t}, g_{t,c}, \neg x_{i,c}) \qquad \forall t \in \mathcal{T}_L, x_i \in X, c \in [2..k] \tag{19}$$

The clauses in Eq. (20) and Eq. (21) break the symmetry between the equivalent cluster assignments. We consider two cluster assignments Θ^1 and Θ^2 equivalent if there exists a relabelling, $\Gamma : K \to K$, such that $\Gamma \circ \Theta^1 = \Theta^2$. To break symmetries, we force data points in the ascending order of their index to be assigned to the first empty cluster, if they need a new one. Specifically, Eq. (20) guarantees that the $c-1$-th point can only be assigned to the first $c-1$ clusters and Eq. (21) guarantees that if all previous points are assigned to the first $c-2$ clusters, the current point can only be assigned to the first $c-1$ clusters. These clauses enforce that there are no two feasible clusterings that are relabellings of each other. Note that we do not eliminate viable solutions, since any clustering can be renamed into an equivalent one that respects this property.

$$(\neg x_{c-1,c}) \qquad \forall c \in [2..k] \tag{20}$$

$$(\neg x_{i,c}, \bigvee_{i'<i} x_{i',c-1}) \qquad \forall x_i \in X, c \in [3..k], c < i \tag{21}$$

Assuming $|X| \geq k'$, the clause in Eq. (22), alongside the symmetry-breaking ones (Eq. (20) and Eq. (21)), guarantees that k' clusters are non-empty. Specifically, it enforces that at least one point is not assigned to the first $k'-1$ clusters. Thus, given the ordered assignment of clusters due to symmetry-breaking, there should be at least k' non-empty clusters. While we support any valid value for k', we focus on the setting where the lower bound for the number of non-empty clusters is equal to the upper bound, i.e., $k' = k$, in our experiments.

$$(\bigvee_{i} x_{i,k'}) \tag{22}$$

The clauses in Eq. (23), Eq. (24), and Eq. (25), hereafter referred to as the *unconditional separating clauses*, guarantee that pairs in CL are clustered separately.

$$(x_{i,2}, x_{i',2}) \qquad \forall (i,i') \in CL \tag{23}$$

$$(\neg x_{i,k}, \neg x_{i',k}) \qquad \forall (i,i') \in CL \tag{24}$$

$$(\neg x_{i,c}, \neg x_{i',c}, x_{i,c+1}, x_{i',c+1}) \qquad \forall (i,i') \in CL, c \in [2..k-1] \tag{25}$$

The clauses in Eq. (26) and Eq. (27), hereafter referred to as the *unconditional co-clustering clauses*, guarantee that pairs in ML are clustered together.

$$(\neg x_{i,c}, x_{i',c}) \qquad \forall (i,i') \in ML, c \in [2..k] \tag{26}$$

$$(x_{i,c}, \neg x_{i',c}) \qquad \forall (i,i') \in ML, c \in [2..k] \tag{27}$$

The clauses in Eq. (28), Eq. (29), and Eq. (30), hereafter referred to as the *conditional separating clauses*, guarantee that a b^-_w variable being set to false forces the pairs in distance class w to be clustered separately.

$$(b^-_w, x_{i,2}, x_{i',2}) \qquad \forall D_w \in \hat{D}, (i,i') \in D_w \tag{28}$$

$$(b^-_w, \neg x_{i,k}, \neg x_{i',k}) \qquad \forall D_w \in \hat{D}, (i,i') \in D_w \tag{29}$$

$$(b^-_w, \neg x_{i,c}, \neg x_{i',c}, x_{i,c+1}, x_{i',c+1}) \qquad \forall D_w \in \hat{D}, (i,i') \in D_w, c \in [2..k-1] \tag{30}$$

The clauses in Eq. (31) and Eq. (32), hereafter referred to as the *conditional co-clustering clauses*, guarantee that a b^+_w variable being set to true forces the pairs in distance class w to be clustered together.

$$(\neg b^+_w, \neg x_{i,c}, x_{i',c}) \qquad \forall D_w \in \hat{D}, (i,i') \in D_w, c \in [2..k] \tag{31}$$

$$(\neg b^+_w, x_{i,c}, \neg x_{i',c}) \qquad \forall D_w \in \hat{D}, (i,i') \in D_w, c \in [2..k] \tag{32}$$

Lastly, the clauses in Eq. (33), Eq. (34), and Eq. (35) guarantee that b^+_w and b^-_w variables represent a valid unary encoding and have values that result in well-defined λ^+ and λ^- indices. Specifically, there can exist $\lambda^+ \leq \lambda^-$ values such that $b^+_1..b^+_{\lambda^+}$ and $b^-_1..b^-_{\lambda^-}$ are set to true and the rest of b^+_w and b^-_w variables are set to false.

$$(\neg b^-_w, b^-_{w-1}) \qquad \forall D_w \in \hat{D}, w > 1 \tag{33}$$

$$(\neg b^+_w, b^+_{w-1}) \qquad \forall D_w \in \hat{D}, w > 1 \tag{34}$$

$$(\neg b^+_w, b^-_w) \qquad \forall D_w \in \hat{D} \tag{35}$$

5.2.1 Decoding. Assuming an assignment of boolean values to the variables in Section 5.1 that is satisfying with regard to the clauses in Section 5.2, we can decode a decision tree solution $\mathcal{D} = (\mathcal{T}, \beta, \alpha, \theta)$ by finding values for the β, α, and θ functions. Note that $\mathcal{T}$ is given as input.

Given that our base tree encoding follows Shati et al. 2023b, the decoding procedure remains mostly the same. We decode β by setting $\beta(t) = j$ for every $a_{t,j}$ variable that is true. Note that there will be no conflicts since at most one $a_{t,j}$ variable can be true for each node $\forall t \in \mathcal{T}_B : \sum_{j \in F} a_{t,j} = 1$. We decode α by choosing a threshold value that correctly directs the data points in each branching node to left or right based on the corresponding $s_{i,t}$ values. Specifically, we set $\alpha(t) = x_i[j]$ where $\beta(t) = j$ and $(i,i') \in O_j(X)$ are consecutive data points along the feature j that are directed differently ($s_{i,t}$ is true and $s_{i',t}$ is false). Note that the clauses in Eq. (15) and Eq. (16) guarantee the existence of such pair.

Our decoding procedure for the θ function differs from Shati et al. since we employ a unary encoding for leaf labels rather than a one-hot encoding. We set $\theta(t)$ to 1 plus the number of $g_{t,c}$ variables that are true for each leaf node $t \in \mathcal{T}_L$. Note that the rest of variables that are particularly introduced for learning clustering solutions do not directly impact the decision tree and, thus, are not involved in the decoding procedure.

5.2.2 Objective. In Section 4, we established that in order to find an ϵ-approximation of an [MD, MS] Pareto optimal solution, we need to find a solution that is Pareto optimal with regards to maximizing λ^+ and minimizing λ^-. Note that λ^+ and λ^- are the number of b^+_w and b^-_w variables set to true, respectively.

$$\lambda^+ = \sum_w \mathbb{1}(b^+_w = true) \tag{36}$$

$$\lambda^- = \sum_w \mathbb{1}(b^-_w = true) \tag{37}$$

To obtain a Pareto optimal solution, we can optimize any function that is increasing with regard to λ^- and decreasing with regard to λ^+ as objective. We opt for minimizing the simple combination of $\lambda^- - \lambda^+$ and model it using the soft clauses with unit weights (indicated by $\langle\rangle$) in Eq. (38) and Eq. (39).

$$\langle 1 \rangle (\neg b^-_w) \qquad D_w \in \hat{D} \tag{38}$$

$$\langle 1 \rangle (b^+_w) \qquad D_w \in \hat{D} \tag{39}$$

6 Smart Pairs

Our naive encoding of unconditional and conditional co-clustering and separating clauses (Eq. (23) to Eq. (32)) leads to a quadratic number of clauses in the size of the dataset $|X|$. This is significant since all the other parts of the encoding are linear in $|X|$. Due to our objectives, the quadratic size is unavoidable, but there are ways to reduce the number of clauses nonetheless. In this section, we propose pruning techniques that see the set of points as nodes in a graph and exploit connections between pairs.

The proposed pruning framework keeps track of connections while our SAT instance is being constructed. At any point, pairs that are already forced to be clustered together are represented by *positive* edges and pairs that are already forced to be clustered separately by *negative* edges. The positive edges imply a set of connected components of points that will be clustered together while a negative edge between nodes in different components indicates that each of the components are mutually exclusive, i.e., each component will be in a different cluster. We incrementally build the set of positive edges (E^+) and the set of negative edges (E^-), with each new edge being classified as *inner* or *crossing*, based on the previous members.

Definition 11 (Inner and Crossing edges). *Given the sets E^+ and E^-, a new edge is said to be an:*

- *Inner edge, if it connects two nodes within an existing connected component based on E^+.*
- *Crossing edge, if it connects two nodes in two connected components based on E^+ that are mutually exclusive based on E^-.*

We will use edges and the pairs of points that they represent interchangeably onward.

Identifying a new edge as inner or crossing while adding co-clustering or separating clauses can help us detect infeasibility or redundancy. Since E^+ (E^-) represents the set of pairs that are forced to be clustered together (separately), an inner pair is forced to have the same label and a crossing pair to have different labels. Thus, it would be redundant to, and we avoid, adding co-clustering clauses (Eq. (26), Eq. (27), Eq. (31), and Eq. (32)) for inner pairs and separating clauses (Eq. (23), Eq. (24), Eq. (25), Eq. (28), Eq. (29), and Eq. (30)) for crossing pairs. Furthermore, it would be infeasible to enforce co-clustering for crossing pairs and separation for inner pairs. Our treatment of the unconditional clauses (Eq. (23) to Eq. (27)) and of the conditional ones (Eq. (28) to Eq. (32)) differ in the case of infeasibility detection and in the construction of E^+ and E^- sets.

- **Infeasibility Detection:** If the unconditional separation or co-clustering of a pair is infeasible, the problem is infeasible. However, for conditional clauses, detecting infeasibility only leads to fixing the corresponding b^+_w or b^-_w variable to make the conditional clause non-enforcing by satisfying it. Note that nullifying a

conditional clause implies the nullification of all of the subsequent ones. So we can stop adding clauses from a series of conditional ones as soon as one is detected to be infeasible.

- **Construction of E^+ and E^-:** For the unconditional clauses the E^+ and E^- sets are respectively constructed from ML and CL links. For conditional ones however, we also include the pairs that are implied to be co-clustered (separated), due to the order of distance imposed by the MS (MD) objective. Namely, if a pair of longer distance is being co-clustered for optimizing MS, a pair of shorter distance is implied to be already co-clustered. Analogously, if a pair of shorter distance is being separated for optimizing MD, a pair of longer distance is implied to be already separated. Note that implied pairs for the processing of conditional co-clustering clauses are not valid for the processing of conditional separating clauses, and vice-versa.

We construct the sets E^+ and E^- by 1) incrementally adding the members of ML to E^+, corresponding to unconditional co-clustering clauses; 2) incrementally adding the members of CL to E^-, corresponding to unconditional separating clauses; 3) incrementally adding all pairs of the dataset in the ascending order of containing distance classes to E^+, corresponding to conditional co-clustering clauses; 4) resetting E^+ to its state prior to stage 3 and incrementally adding all pairs of the dataset in the descending order of containing distance classes to E^-, corresponding to conditional separating clauses. Note that we reset E^+ to remove all edges corresponding to condition co-clustering clauses, as they are no longer implied to hold when we shift the focus to conditional separating clauses. We add unconditional clauses first since they do not require a reset for E^+ or E^- and can be beneficial in further stages of the algorithm. Moreover, we add unconditional co-clustering clauses before separating ones because constructing E^- while $E^+ = \emptyset$ does not lead to any infeasible or redundant cases. We can use any arbitrary order to add members of ML (CL) to E^+ (E^-), but we opt for ascending distance for co-clustering clauses and descending distance for separating clauses. Algorithm 1 provides a detailed description of the inner-workings of the Smart Pairs algorithm.

7 Soft Pairwise Constraints

Due to noise and error in producing must-links (ML) and cannot-links (CL) or the limitations of fixed-depth (d) decision trees, Problem 1 in its general form is not guaranteed to have a feasible solution. However, there always exists a solution if $ML = CL = \emptyset$ and $2^d \geq k$, as we can simply find a tree with arbitrarily bad objective values that contains all labels. This shows that there is a balance to be found in the amount of constraints to satisfy. We observe through our experiments (Section 10.2) that infeasibility can appear even when adding a moderate number of links, prematurely impeding solution quality enhancement through adding constraints. One approach to potentially tackle infeasibility is to consider incrementally deeper trees. However, simply increasing depth can cause issues in performance. Furthermore, we experimentally observe (Section 10.1 and Section 10.2) the benefits of limiting solutions to shallow trees, and arbitrarily increasing the depth can actively work against such advantages. We instead opt for disregarding a subset of links that cause infeasibility and utilizing the rest. Namely, we optimize the number of links that we can satisfy instead of enforcing the satisfaction of all constraints.

In this section, we introduce *soft constraints* to address the issue of infeasibility caused by adding restrictive clustering constraints. We propose an extension of our encoding that works by considering must-links and cannot-links as soft constraints that are encouraged, rather than required, to be satisfied. Treating must-links and cannot-links as soft constraints makes them objectives to be optimized in addition to our primary [MD, MS] objective. Since the number of satisfied constraints is dissimilar from λ^+ and λ^- in nature, it would be challenging to convincingly combine the two goals and weigh them against one another. Hence, our approach focuses on prioritizing the satisfaction of links over the clustering objective.

We introduce multiple schemes for optimizing the pairwise links and the [MD, MS] objective. In Section 7.1, we propose the 1-stage approach that optimizes link satisfaction simultaneously with the objective. In Section 7.2, we propose the 2-stage scheme that optimizes ML and CL constraints in a pre-processing stage and uses the satisfied

Algorithm 1 Smart Pairs

$E^+ = E^- = \emptyset$
for $(x_i, x_{i'}) \in ML$ sorted in ascending $<^*$ **do**
  if $(x_i, x_{i'})$ is not *inner* **then**
    $E^+ = E^+ \cup \{(x_i, x_{i'})\}$
    add clauses from Eq. (26) and Eq. (27) for $(x_i, x_{i'})$
  end if
end for
for $(x_i, x_{i'}) \in CL$ sorted in descending $<^*$ **do**
  if $(x_i, x_{i'})$ is *inner* **then**
    Return Infeasible
  end if
  if $(x_i, x_{i'})$ is not *crossing* **then**
    $E^- = E^- \cup \{(x_i, x_{i'})\}$
    add clauses from Eq. (23), Eq. (24), and Eq. (25) for $(x_i, x_{i'})$
  end if
end for
$\hat{E}^+ = E^+$
for $(x_i, x_{i'}) \in X \times X$ sorted in ascending $<^*$ **do**
  set w s.t. $(x_i, x_{i'}) \in D_w$
  if $(x_i, x_{i'})$ is *crossing* **then**
    add clause $(\neg b^+_w)$
    Break
  end if
  if $(x_i, x_{i'})$ is not *inner* **then**
    $E^+ = E^+ \cup \{(x_i, x_{i'})\}$
    add clauses from Eq. (31) and Eq. (32) for $(x_i, x_{i'})$ and w
  end if
end for
$E^+ = \hat{E}^+$ {Discard implied pairs for conditional co-clusering}
for $(x_i, x_{i'}) \in X \times X$ sorted in descending $<^*$ **do**
  set w s.t. $(x_i, x_{i'}) \in D_w$
  if $(x_i, x_{i'})$ is *inner* **then**
    add clause (b^-_w)
    Break
  end if
  if $(x_i, x_{i'})$ is not *crossing* **then**
    $E^- = E^- \cup \{(x_i, x_{i'})\}$
    add clauses from Eq. (28), Eq. (29), and Eq. (30) for $(x_i, x_{i'})$ and w
  end if
end for

links as hard constraints when optimizing [MD, MS]. In Section 7.3, we further divide constraint optimization into two stages corresponding to CL and ML sets, respectively. Assuming that all stages are run to completion, there are theoretical connections between the solutions found by the soft constraint schemes and the hard constraint

approach. All schemes are equivalent when the CL and ML sets are fully feasible. Moreover, the 1-stage approach is guaranteed to find a higher or equal objective value in λ^+ and λ^- compared to the 2-stage approach since the clustering objective is optimized simultaneously with link satisfaction. Lastly, the 2-stage approach is guaranteed to satisfy a higher or equal number of links compared to the 3-stage approach since CL and ML satisfaction are optimized simultaneously. In Section 7.4, we discuss how soft ML and CL sets can be integrated into the Smart Pairs algorithm.

7.1 1-stage Approach

The 1-stage approach optimizes the number of satisfied pairwise links and the [MD, MS] objective in the same stage. It completely prioritizes link satisfaction by using a sufficiently high enough weight. The encoding for the 1-stage scheme is slightly different than the encoding presented in Section 5.

We use $\{e^m_{i,i'} \mid (i,i') \in ML\}$ and $\{e^c_{i,i'} \mid (i,i') \in CL\}$ variables in addition to those employed in the original encoding. They, respectively, represent whether the must-link and cannot-link constraint corresponding to the pair (i,i') is satisfied. We remove the unconditional separating and co-clustering clauses (Eq. (23) to Eq. (27)) and add the clauses in Eq. (40) to Eq. (44), that use variables $e^m_{i,i'}$ and $e^c_{i,i'}$ to enforce a must-link or cannot-link only if its corresponding variable is true. We use the soft clauses in Eq. (45) and Eq. (46) to maximize the number of $e^m_{i,i'}$ and $e^c_{i,i'}$ variables set to true and, as a result, the number of satisfied links. We use $2|\hat{D}| + 1$ as the weight for both (indicated in $\langle\rangle$) to guarantee that link satisfaction is completely prioritized over optimizing [MD, MS]. Note that this is true as all of the soft clauses responsible for optimizing [MD, MS] have a total weight of $2|\hat{D}|$ (Eq. (38) and Eq. (39)). We decode the solution as before since the inner workings of the solution are encoded the same. Note that this variation of the encoding allows a solution even it does not satisfy any links. Thus, the instance cannot become infeasible due to clustering constraints, independent of the quantity.

$$(\neg e^c_{i,i'}, x_{i,1}, x_{i',1}) \qquad \forall (i,i') \in CL \tag{40}$$

$$(\neg e^c_{i,i'}, \neg x_{i,k-1}, \neg x_{i',k-1}) \qquad \forall (i,i') \in CL \tag{41}$$

$$(\neg e^c_{i,i'}, \neg x_{i,c}, \neg x_{i',c}, x_{i,c+1}, x_{i',c+1}) \qquad \forall (i,i') \in CL, c \in [2..k-1] \tag{42}$$

$$(\neg e^m_{i,i'}, \neg x_{i,c}, x_{i',c}) \qquad \forall (i,i') \in ML, c \in [2..k] \tag{43}$$

$$(\neg e^m_{i,i'}, x_{i,c}, \neg x_{i',c}) \qquad \forall (i,i') \in ML, c \in [2..k] \tag{44}$$

$$\langle 2|\hat{D}| + 1\rangle (e^c_{i,i'}) \qquad (i,i') \in CL \tag{45}$$

$$\langle 2|\hat{D}| + 1\rangle (e^m_{i,i'}) \qquad (i,i') \in ML \tag{46}$$

7.2 2-stage Approach

To alleviate the computational cost of optimizing the constraints and the objective simultaneously, we can divide them into two stages. The 2-stage approach dedicates the first stage to optimizing the combined number of satisfied must-links and cannot-links as the pre-processing stage and optimizes the [MD, MS] objective in the second stage. Slightly modified versions of the encoding in Section 5 are solved at each stage.

Stage 1. We dispose of all of the components in the encoding regarding the modelling and optimization of [MD, MS] bi-criteria objective. Namely, λ^+ and λ^- indices, b^+_w and b^-_w variables, the conditional co-clustering and separating clauses (Eq. (28) to Eq. (32)), the clauses enforcing the proxy objectives to be well-defined (Eq. (33) to Eq. (35)), and the soft clauses (Eq. (38) and Eq. (39)) are removed. Similar to the 1-stage approach, we introduce variables $\{e^m_{i,i'} \mid (i,i') \in ML\}$ and $\{e^c_{i,i'} \mid (i,i') \in CL\}$, and add clauses Eq. (40) to Eq. (44). Lastly, we maximize the number of $e^m_{i,i'}$ and $e^c_{i,i'}$ variables set to true by adding the soft clauses with unit weights in Eq. (47) and Eq. (48).

$$\langle 1 \rangle (e^{c}_{i,i'}) \qquad (i, i') \in CL \tag{47}$$

$$\langle 1 \rangle (e^{m}_{i,i'}) \qquad (i, i') \in ML \tag{48}$$

Stage 2. We modify the ML and CL sets by removing any member with corresponding $e^{M}_{i,i'}$ or $e^{C}_{i,i'}$ variable set to false, i.e., that is not satisfied in stage 1. We then solve the encoding in Section 5 without any changes using the modified sets, and decode the solution as before. Note that the modified set of pairwise constraints are shown to be satisfied in stage 1. Thus, stage 2 is guaranteed to be feasible by using the same solution.

7.3 3-stage Approach

The 1-stage and 2-stage schemes consider optimization of must-links and cannot-links simultaneously and weigh them equally. However, it is not guaranteed that the two types of pairwise constraints are equally informative and contribute the same to the accuracy of the solution. The difference is further exacerbated when there is a lack of balance in the size of ML and CL sets. For example, if ML is significantly larger than CL, soft constraint optimization has a trivial solution that satisfies almost all of must-links and almost none of cannot-links. The trivial solution has one dominating cluster that contains almost all of the input dataset. The trivial solution can be the optimal solution if it is challenging to satisfy a considerable subset of both ML and CL.

We propose the 3-stage scheme to separate optimizing the satisfaction of soft must-links and cannot-links. The 3-stage approach resembles the 2-stage approach (Section 7.2) in solving modified versions of the main encoding (Section 5), but has two pre-processing stages instead of one. Specifically, it first maximizes cannot-link satisfaction, then must-link satisfaction, and then optimizes the [MD, MS] objective. Note that we do not consider the alternative approach of prioritizing must-links over cannot-links, as that would be prone to the aforementioned trivial solution.

Stage 1. We dispose of all objective optimization components and unconditional separating and co-clustering clauses, as recounted in Section 7.2. We introduce variables $\{e^{c}_{i,i'} \mid (i, i') \in CL\}$ to represent whether the cannot-link constraint corresponding to the pair (i, i') is satisfied. We add the clauses in Eq. (40) to Eq. (42) to ensure that all cannot-links are correctly claimed to be satisfied. Finally, we add the soft clauses with unit weights in Eq. (47) to maximize the number of satisfied cannot-links.

Stage 2. We modify the CL set by removing any member with the corresponding $e^{c}_{i,i'}$ variable set to false. Similar to stage 1, we dispose of all objective optimization components and unconditional co-clustering clauses. However, the unconditional separating clauses are kept as we consider cannot-links to be hard constraints at this stage. We introduce variables $\{e^{m}_{i,i'} \mid (i, i') \in ML\}$ to represent whether the must-link constraint corresponding to the pair (i, i') is satisfied. We add the clauses in Eq. (43) and Eq. (44) to ensure that all must-links are correctly claimed to be satisfied. Finally, we add the soft clauses with unit weights in Eq. (48) to maximize the number of satisfied must-links.

Stage 3. We modify the ML set by removing any member with corresponding $e^{m}_{i,i'}$ variable set to false. We then solve the encoding in Section 5 without any changes using the modified sets from the last two stages, and decode the solution as before. Note that similar to the final stage in Section 7.2, the final stage is guaranteed to have a feasible solution.

7.4 Smart Pairs Integration

In this section, we discuss how transforming pairwise constraints from hard to soft affects their integration into the Smart Pairs algorithm (Section 6). We used the unconditional clauses corresponding to ML and CL sets (Eq. (23) to Eq. (27)) to be the basis of sets E^{+} and E^{-} in the Smart Pairs algorithm. However, soft cannot-links

and must-links are no longer unconditionally required to be satisfied. Thus, the inclusion of ML (CL) members into E^+ (E^-) is rendered unsound in any stage that its satisfied subset is yet to be determined. Specifically, ML and CL both cannot be integrated in the 1-stage approach, stage 1 of the 2-stage approach, and stage 1 of the 3-stage approach, and ML cannot be integrated in stage 2 of the 3-stage approach.

8 Decision Tree Clustering Pareto Front

In Section 5, we proposed an encoding for decision tree clustering whose solution is proven to be an ϵ-approximation of a Pareto optimal solution in [MD, MS]. However, there can exist more than one Pareto optimal solution. Our approximated solution is chosen arbitrarily and is not guaranteed to hold any particular advantage over the other Pareto optimal solutions.

In contrast, exploring the Pareto front as a whole is beneficial as it allows finding higher quality alternatives, producing sets of diverse choices for the solution, and integrating domain expertise as manual solution selection. In this section, we propose an algorithm to solve a generalization of Problem 1 that aims to find approximations of all Pareto optimal solutions in the [MD, MS] objective, rather than an arbitrary one.

Problem 3 (Complete Pareto Optimal Decision Tree Clustering). *Given a set of numerical features F, clusters $[1..k]$, dataset $X \subset \mathbb{R}^{|F|}$, must-link and cannot-link constraints $ML, CL \subseteq X \times X$, fixed depth d, and approximation parameter ϵ, find a set of fully balanced decision trees $\{\mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^P\}$ of depth d, such that:*

(1) *For all $p \in [1..P]$, $\Theta^p_{\mathcal{D}}$ satisfies the constraints ML and CL;*
(2) *For all $\mathcal{D}^* \in \mathbb{D}^*$, $\exists p : (\mathrm{MD}_{\mathcal{D}^p} \leq \mathrm{MD}_{\mathcal{D}^*} + \epsilon) \wedge (\mathrm{MS}_{\mathcal{D}^p} \geq \mathrm{MS}_{\mathcal{D}^*} - \epsilon)$,*

where $\mathbb{D}^*$ is the set of all Pareto optimal solutions to the problem with respect to the bi-criteria of maximizing the minimum split and minimizing the maximum diameter.

The encoding presented in Section 5 has proxy objectives λ^+ and λ^- that approximate MS and MD. We highlighted in Section 5.2 that we combine the two proxies by minimizing $\lambda^- - \lambda^+$ and thus, aim for an arbitrary Pareto optimal solution in λ^+ and λ^-. In order to solve Problem 3 however, we solve a series of instances to find a complete Pareto optimal set and the Pareto front in λ^+ and λ^-. In Proposition 2, we show that the Pareto front in λ^+ and λ^- includes approximations of any Pareto optimal solution [MD, MS].

Proposition 2 (Complete ϵ-approximation of [MD, MS]). *Let $\{\mathcal{D}^1, \mathcal{D}^2, ..., \mathcal{D}^P\}$ be a complete Pareto optimal set of decision tree clusterings in λ^+ and λ^- objectives. For every $\mathcal{D}^*$ that is a Pareto optimal tree clustering in [MD, MS], there exists p with $(\mathrm{MD}_{\mathcal{D}^p} \leq \mathrm{MD}_{\mathcal{D}^*} + \epsilon) \wedge (\mathrm{MS}_{\mathcal{D}^p} \geq \mathrm{MS}_{\mathcal{D}^*} - \epsilon)$.*

Proof. We show that for any arbitrary Pareto optimal solution in minimum split and maximum diameter $\mathcal{D}^*$, there exists a solution $\mathcal{D}^p$ in its ϵ-neighborhood.

- Let $\mathrm{MS}_{\mathcal{D}^*}$ ($\mathrm{MD}_{\mathcal{D}^*}$) be the minimum split (maximum diameter) achieved by the solution $\mathcal{D}^*$. Furthermore, let the pair (x^s_1, x^s_2) ((x^d_1, x^d_2)) correspond to the minimum split (maximum diameter) distance.
- Set $\lambda^+_{\mathcal{D}^*}$ such that $(x^s_1, x^s_2) \in D_{\lambda^+_{\mathcal{D}^*}+1}$ and $\lambda^-_{\mathcal{D}^*}$ such that $(x^d_1, x^d_2) \in D_{\lambda^-_{\mathcal{D}^*}}$.
- Given that the solution $\mathcal{D}^*$ achieves $\lambda^+_{\mathcal{D}^*}$ and $\lambda^-_{\mathcal{D}^*}$ values in λ^+ and λ^-, any complete Pareto optimal set in λ^+ and λ^- should contain a solution with objective values equal or better than $\lambda^+_{\mathcal{D}^*}$ and $\lambda^-_{\mathcal{D}^*}$. Let $\mathcal{D}^p$ be that solution.
- Since $\lambda^+_{\mathcal{D}^p} \leq \lambda^+_{\mathcal{D}^*} \wedge \lambda^-_{\mathcal{D}^p} \geq \lambda^-_{\mathcal{D}^*}$, we conclude that $(\mathrm{MD}_{\mathcal{D}^p} \leq \mathrm{MD}_{\mathcal{D}^*} + \epsilon) \wedge (\mathrm{MS}_{\mathcal{D}^p} \geq \mathrm{MS}_{\mathcal{D}^*} - \epsilon)$, proving the proposition for an arbitrary Pareto optimal solution.

□

To find a complete Pareto optimal set in λ^+ and λ^-, we use a well-known method for generating complete Pareto optimal sets in bi-criteria problems (Dao, Duong, et al. 2017; T'kindt and Billaut 2006). We present the

Algorithm 2 Complete Pareto Optimal Set

$F := \emptyset$
$\mathcal{D} :=$ Minimize(λ^-)
while true **do**
  $\mathcal{D} :=$ Maximize(λ^+, with $\lambda^- \leq \mathcal{D}_{\lambda^-}$)
  $F := F \cup \{\mathcal{D}\}$
  $\mathcal{D} :=$ Minimize(λ^-, with $\lambda^+ > \mathcal{D}_{\lambda^+}$)
  if $\mathcal{D}$ is null **then**
    Break
  end if
end while
Return F

details of using this method for our setting in Algorithm 2. At each step, we need to optimize one objective while the other is being (strictly) bounded by a threshold at each step. We propose a modification to the encoding presented in Section 5 that supports having a lower (upper) bound on λ^+ (λ^-) and optimizing λ^- (λ^+). In order to focus on a single object, we dispose of all of the components that are purely responsible for the optimization of the other one, and in order to enforce a bound on an objective, we add all pairs that are implied by the bound to be co-clustered (separated) as part of the *ML* (*CL*) set. The details of each case is as follows:

- **When we aim to minimize λ^- with $\lambda^+ \geq \lambda^{lb}$:** We dispose of b^+_w variables, the clauses in Eq. (31), Eq. (32), Eq. (34), and Eq. (35), and the soft clauses in Eq. (39). Next, we add the pairs belonging to the first λ^{lb} distance classes to *ML*.
- **When we aim to maximize λ^+ with $\lambda^- \leq \lambda^{ub}$:** We dispose of b^-_w variables, the clauses in Eq. (28) to Eq. (30), Eq. (33) to Eq. (35), and the soft clauses in Eq. (38). Next, we add the pairs belonging to all but the first λ^{ub} distance classes to *CL*.

Since our objectives are both integers, we can simply implement strict inequality by increasing λ^{lb} or decreasing λ^{ub} by 1.

Note that Algorithm 2 is compatible with hard pairwise constraints or soft ones that have been previously optimized in 1-stage (Section 7.1), 2-stage (Section 7.2), or 3-stage (Section 7.3) schemes. We only focus on the combination with hard constraints for our experimental studies due to brevity and because: 1) a non-trivial optimization of soft constraints is likely to shrink the search space leading to a small and mostly homogeneous Pareto front; 2) Pareto front generation allows a domain expert to investigate soft constraint satisfaction and its trade-off against quality without explicitly optimizing it.

9 Experimental Setup

In this section, we describe the details of our implementation, the baselines that we compare against, the method that we use for generating constraints, and how and on which datasets we evaluate our approach.

9.1 Implementation

Our approach is implemented in Python and Java and we use the Loandra solver (Berg, Demirović, et al. 2019) to solve the encodings that we generate. Loandra is an any-time solver that guarantees optimality if run to completion, but can also produce intermediate solutions. We set a time limit of 30 minutes every time a call to the solver is made, a feasible solution is salvaged in case of a timeout if possible. We run experiments on a server with two 12-core Intel E5-2697v2 CPUs and 128G of RAM.

9.2 Datasets

We run our experiments on a varied set of datasets ranging in terms of domain, size, and number of features. Our benchmarks include seven real datasets from the UCI repository (Dua and Graff 2017) and four synthetic datasets from FCPS (Ultsch and Lötsch 2020). All of our datasets come with a given number of clusters that we use as an input for our encoding, and a ground-truth labeling that we use to evaluate our solutions. We consider fully-balanced decision trees where all leaves have the same depth. For each dataset, we fix the depth value to the lowest number that provides more than twice as many leaves as there are clusters. We normalize the values of each feature in all datasets to the range $[0, 100]$ so that features with larger values do not dominate the pairwise distances. The properties of each dataset alongside the chosen depth values are provided in Table 1.

Table 1. Real (top) and synthetic (bottom) datasets and their properties.

Dataset	$\lvert X\rvert$	$\lvert F\rvert$	k	d
Iris	150	4	3	3
Wine	178	13	3	3
Glass	214	9	7	4
Ionosphere	351	34	2	3
Seeds	210	7	3	3
Libras	360	90	15	5
Spam	4601	57	2	3
Lsun	400	2	3	3
Chainlink	1000	3	2	3
Target	770	2	6	4
WingNut	1016	2	2	3

The real datasets are selected with emphasis on their features and how informative and meaningful they are in their corresponding domains. We do so because the interpretability of an ML model is highly reliant on the interpretability of the features that it utilizes. For example, a decision tree with a single branching on the output of a complex black-box model has a simple structure, but is not interpretable in any way.

9.3 Constraint Generation

To understand the effects of constraints on our approach, we synthetically generate pairwise clustering constraint sets that vary relative to the size of the dataset. Specifically, for a given κ value with $0 \leq \kappa \leq \frac{|X|-1}{2}$, we generate a set of $\kappa \cdot |X|$ random pairwise constraints following Wagstaff and Cardie 2000: (1) We generate $\kappa \cdot |X|$ pairs of data points without repetition; (2) We add each pair to the must-links (ML) constraint set if both data points share the same ground-truth label and to the cannot-links (CL) set otherwise. To mitigate the potential bias due to the random generation of constraints, each experiment is run with 20 different seeds with the overall mean being reported.

9.4 Baselines

To evaluate the performance of our approach, we compare its results against its variations with regard to structure and objective, in addition to baselines from the literature.

9.4.1 Constrained Clustering. Constrained clustering (Problem 2), i.e., finding a clustering with approximated Pareto optimality of [MD, MS] without restriction to a decision tree, is our first baseline. To solve constrained

clustering using our approach, we remove the tree-related components of the encoding. Namely, we remove the variables $a_{t,j}$, $s_{i,t}$, $z_{i,t}$, $g_{t,c}$ and the clauses in Eq. (8) to Eq. (19). Instead, we introduce the clauses in Eq. (49) that guarantee a well-formed unary encoding of labels.[2]

$$(x_{i,c}, \neg x_{i,c+1}) \qquad \forall x_i \in X, c \in [2..k-1] \tag{49}$$

While we do not include local search-based approaches for constrained clustering as baselines in our experiments, previous work found that they tend to be outperformed by combinatorial approaches in terms of solution quality and constraint utilization (Dao, Duong, et al. 2017).

9.4.2 Maximum Diameter. Our second and third baselines are the variants of Problem 1 and Problem 2 that only focus on optimizing (an approximation of) the maximum diameter objective. In case of Problem 2 and no approximation (i.e., $\epsilon = 0.0$), the variant is equivalent to a well-known baseline formulated in previous works from the literature (Dao, Duong, et al. 2017; Dao, Vrain, et al. 2016). In other words, it has the same set of (feasible and) optimal solutions as the baseline.

To solve the MD-only variations, we remove all MS-related components of the two encodings for Problem 1 and Problem 2. Namely, we remove b^+_w variables, conditional co-clustering clauses in Eq. (31) and Eq. (32), clauses guaranteeing λ^* to be well-defined in Eq. (34) and Eq. (35), and soft clauses modelling the MS objective in Eq. (39).

9.4.3 Mixed Integer Optimization. To our knowledge, the model in Bertsimas, Orfanoudaki, et al. 2021 is the only approach in the literature that solves the problem of decision tree clustering by formulating it as a well-known combinatorial optimization problem, i.e., Mixed Integer Optimization (MIP). Their model is restricted to one leaf per cluster and optimizes the Silhouette objective given sufficient time. However, Bertsimas, Orfanoudaki, et al. note that their model does not scale well, therefore not releasing their code nor presenting any experimental results. They instead focus on a heuristic procedure that does not have any solution quality guarantees and cannot naturally support clustering constraints.

Due to the relevance to our work and the lack of similar approaches in the literature, we aim to provide a comparison against the MIP model. Thus, we have implemented the model using the Gurobi v10 solver. Note that their model did not originally support clustering constraints and we extend it to incorporate the pairwise constraints that we support. Furthermore, we have detected and addressed a series of issues ranging from minor typographical ones to crucial ones that make it impossible to solve the model. We provide the details of our implemented version of Bertsimas, Orfanoudaki, et al.'s MIP formulation for decision tree clustering in Appendix A.

9.5 Evaluation

We use the ground-truth labels included in our benchmarks to evaluate the quality of clustering solutions obtained by our approach. To determine how closely our learned clustering resembles the ground-truth, we use the Adjusted Rand Index (ARI) and the Normalized Mutual Information (NMI) criteria (Section 3.4).

10 Experimental Results

In this section, we run an extensive set of experiments to evaluate our SAT encoding for decision tree clustering with support for constraints. We compare our approach against baselines (Section 10.1), study infeasibility and the impact of the depth parameter (Section 10.2), investigate the benefits of soft constraints (Section 10.3), explore the Pareto fronts of our objective (Section 10.4), and examine the effects of pruning and approximation on performance (Section 10.5).

[2]Note that these clauses are redundant in the encoding of tree clustering as the condition is already enforced for the leaves.

10.1 Comparison Against Baselines

In our first set of experimental results, presented in Table 2 (real datasets) and Table 3 (synthetic datasets), we solve Problem 1 with $\epsilon = 0.1$ for ML and CL sets generated with various κ values. The results are then compared against three baseline variations achieved through removing the tree-clustering restriction and MS component of the objective, described in Section 9.4. We report average ARI and NMI metrics, average runtimes, and the number of feasible instances across 20 seeds. Recall that a decision tree clustering problem with (consistent) ML and CL sets might be infeasible for a fixed depth. We exclude infeasible and unknown[3] cases from the computed average ARI and NMI values.

Note that the MIP formulation from Bertsimas, Orfanoudaki, et al. 2021, which we revised and implemented, is absent from our comparison. Our experiments with the MIP model found that it is unable to find a feasible solution for any of the datasets and the depths specified in Table 2 and Table 3 across multiple runs, both in settings with and without constraints. This result is consistent with Bertsimas, Orfanoudaki, et al.'s observation on the limited scalability of their MIP formulation and emphasizes the strong performance of our approach.

The results in Table 2 and Table 3 show that our approach can produce high quality interpretable solutions with non-zero ϵ values in the time limit. We observe that finding a tree or non-tree solution that is Pareto optimal in [MD, MS] almost always leads to better ARI and NMI scores compared to only optimizing MD. The cases where the MD-only objective achieves better scores mostly correspond to empty ML and CL sets ($\kappa = 0$).

Interestingly, we observe that when the tree clustering problem is feasible, it manages to produce higher quality solutions compared to its non-tree counterpart, constrained clustering (CC), in both objectives. The exceptions are mostly limited to the Chainlink dataset with the [MD, MS] objective and the $\kappa = 0$ case with the MD objective. It seems unintuitive for tree clustering to produce higher quality solutions as it is strictly less expressive than CC. We conjecture that both clustering objectives tend to perform better with tree clustering because of its inherently restricted yet structured solution space, resulting in potentially worse objective values but higher ARI scores.

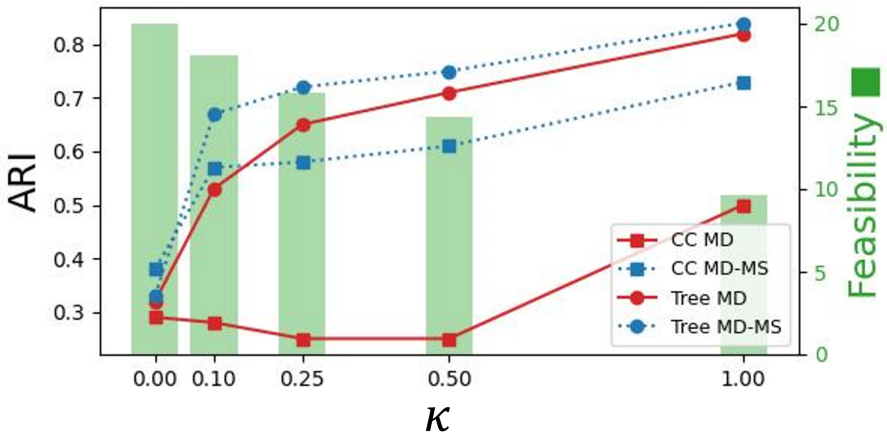

Fig. 1. ARI and feasibility results from Table 2 and Table 3 averaged over 11 datasets.

Figure 1 summarizes the results from Table 2 and Table 3 and highlights the average benefits of tree clustering (○), the bi-criteria objective (blue and dotted), and constraints across all datasets. In the presence of no constraints,

[3]Instances where the solver timed out or crashed without finding a feasible solution are considered "unknown".

Table 2. Interpretable tree clustering with constraints for $\epsilon = 0.1$ averaged over 20 runs and compared against three baselines (real datasets). Gen-Time indicates clauses generation time, and Time indicates total time.

Data	κ	Tree [MD, MS]					Tree MD			CC [MD, MS]		CC MD	
		ARI	NMI	Gen-Time(s)	Time(s)	Feas.	ARI	NMI	Feas.	ARI	NMI	ARI	NMI
Iris	0	0.60	0.67	0.49	0.85	20	0.60	0.67	20	0.60	0.67	**0.70**	**0.72**
	0.1	**0.83**	**0.82**	0.48	0.76	20	0.70	0.71	20	**0.82**	0.82	0.57	0.61
	0.25	**0.85**	**0.84**	0.47	0.68	20	0.77	0.77	20	0.79	0.79	0.60	0.62
	0.5	**0.91**	**0.89**	0.45	0.68	20	0.87	0.85	20	0.80	0.79	0.58	0.61
	1	**0.95**	**0.93**	0.44	0.70	13	0.94	0.92	13	0.91	0.89	0.68	0.68
Wine	0	0.00	0.02	0.64	1.79	20	0.14	**0.20**	20	0.00	0.02	**0.17**	**0.20**
	0.1	**0.71**	**0.70**	0.59	1.23	20	0.39	0.42	20	0.51	0.50	0.37	0.38
	0.25	**0.80**	**0.78**	0.58	2.19	20	0.60	0.60	20	0.52	0.51	0.22	0.25
	0.5	**0.84**	**0.81**	0.55	4.10	20	0.75	0.71	20	0.54	0.53	0.18	0.20
	1	**0.94**	**0.92**	0.56	4.00	20	0.90	0.87	20	0.71	0.67	0.20	0.20
Glass	0	0.23	0.38	1.19	7.91	20	0.22	0.34	20	0.23	**0.39**	**0.24**	0.37
	0.1	**0.22**	**0.33**	1.01	10.63	20	0.19	0.30	20	0.21	0.32	**0.22**	0.32
	0.25	**0.22**	**0.31**	0.96	75.82	20	0.18	0.29	20	0.16	0.26	0.16	0.24
	0.5	**0.28**	**0.35**	0.94	1098.45	20	0.26	**0.35**	20	0.15	0.23	0.13	0.21
	1	-	-	0.93	301.54	0	-	-	0	**0.14**	**0.20**	0.08	0.15
Ionosphere	0	0.01	0.02	1.43	11.65	20	0.06	**0.06**	20	0.01	0.02	**0.09**	**0.06**
	0.1	**0.31**	**0.25**	1.22	32.77	20	0.11	0.07	20	0.14	0.12	0.04	0.03
	0.25	**0.51**	**0.40**	1.19	664.44	13	0.49	0.38	14	0.21	0.16	0.07	0.04
	0.5	-	-	1.21	57.53	0	-	-	0	0.39	**0.32**	0.10	0.06
	1	-	-	1.21	8.38	0	-	-	0	**0.78**	**0.70**	0.70	0.62
Seeds	0	**0.71**	0.67	0.67	1.47	20	0.59	0.58	20	0.68	0.65	**0.71**	**0.69**
	0.1	**0.68**	**0.66**	0.62	0.98	20	0.64	0.62	20	0.63	0.62	0.53	0.56
	0.25	**0.73**	**0.71**	0.59	1.34	20	0.72	0.69	20	0.61	0.61	0.51	0.55
	0.5	**0.79**	**0.76**	0.58	1.89	18	0.77	0.74	18	0.62	0.61	0.55	0.58
	1	-	-	0.56	1.18	0	-	-	0	0.78	0.76	0.71	0.70
Libras	0	0.18	0.44	11.2	1811.31	20	0.18	0.44	20	**0.19**	**0.45**	0.14	0.38
	0.1	**0.18**	**0.44**	10.56	1404.89	20	0.15	0.41	20	0.17	0.43	0.14	0.37
	0.25	**0.17**	**0.42**	10.25	1139.53	20	0.14	0.38	20	0.14	0.38	0.11	0.34
	0.5	**0.16**	**0.41**	9.92	787.08	20	0.14	0.37	20	0.12	0.34	0.09	0.30
	1	**0.16**	0.40	9.91	1673.42	13	0.14	0.37	12	0.11	0.32	0.08	0.27
Spam	0	0.00	0.00	80.36	309.24	20	-0.01	0.01	20	0.00	0.00	**0.13**	**0.09**
	0.1	-	-	56.86	292.93	0	-	-	0	**0.06**	**0.04**	0.04	0.03
	0.25	-	-	56.69	194.23	0	-	-	0	0.04	0.03	0.05	**0.04**
	0.5	-	-	56.36	136.92	0	-	-	0	**0.08**	**0.05**	0.06	0.03
	1	-	-	56.71	105.56	0	-	-	0	0.61	0.55	**0.62**	**0.56**

the ARI scores of the four approaches are fairly close. However, adding a moderate number of constraints causes the CC approach with the MD objective to fall behind significantly, highlighting the importance of using either the bi-criteria objective or tree clustering. Furthermore, all approaches employing tree clustering or the bi-criteria

Table 3. Interpretable tree clustering with constraints for $\epsilon = 0.1$ averaged over 20 runs and compared against three baselines (synthesized datasets). Gen-Time indicates clauses generation time, and Time indicates total time.

Data	κ	Tree [MD, MS]					Tree MD			CC [MD, MS]		CC MD	
		ARI	NMI	Gen-Time(s)	Time(s)	Feas.	ARI	NMI	Feas.	ARI	NMI	ARI	NMI
Lsun	0	**0.40**	**0.51**	1.29	2.04	20	**0.40**	0.50	20	**0.40**	**0.51**	0.38	0.49
	0.1	**0.90**	**0.91**	0.95	1.82	20	0.68	0.69	20	0.74	0.77	0.31	0.33
	0.25	**1.00**	**1.00**	0.92	1.49	20	0.88	0.86	20	0.90	0.90	0.32	0.31
	0.5	**1.00**	**1.00**	0.89	1.45	20	0.95	0.93	20	**1.00**	**1.00**	0.31	0.29
	1	**1.00**	**1.00**	0.87	1.22	20	0.98	0.97	20	**1.00**	**1.00**	0.49	0.39
Chainlink	0	0.24	0.19	4.06	6.70	20	0.00	0.00	20	**1.00**	**1.00**	0.06	0.05
	0.1	0.87	0.80	2.8	6.20	19	0.83	0.75	19	**1.00**	**1.00**	0.06	0.06
	0.25	0.93	0.87	2.78	3.84	1	0.90	0.83	1	**1.00**	**1.00**	0.03	0.03
	0.5	-	-	2.78	3.33	0	-	-	0	**1.00**	**1.00**	0.03	0.03
	1	-	-	2.76	3.01	0	-	-	0	**1.00**	**1.00**	0.63	0.56
Target	0	0.33	0.39	5.51	14.16	20	0.31	0.34	20	0.06	0.22	**0.38**	**0.42**
	0.1	**1.00**	**1.00**	3.23	6.39	20	0.64	0.60	20	**1.00**	**1.00**	0.57	0.54
	0.25	**1.00**	**1.00**	3.17	6.79	20	0.88	0.81	20	**1.00**	**1.00**	0.40	0.37
	0.5	**1.00**	**1.00**	3.18	8.99	20	0.95	0.91	20	**1.00**	**1.00**	0.23	0.24
	1	**1.00**	**1.00**	3.16	7.46	20	0.98	0.96	20	**1.00**	**1.00**	0.44	0.38
Wingnut	0	**1.00**	**1.00**	4.46	6.50	20	**1.00**	**1.00**	20	**1.00**	**1.00**	0.16	0.16
	0.1	**1.00**	**1.00**	3.18	3.54	20	**1.00**	**1.00**	20	**1.00**	**1.00**	0.21	0.18
	0.25	**1.00**	**1.00**	3.13	3.39	20	**1.00**	**1.00**	20	**1.00**	**1.00**	0.29	0.25
	0.5	**1.00**	**1.00**	3.14	3.37	20	**1.00**	**1.00**	20	**1.00**	**1.00**	0.49	0.42
	1	**1.00**	**1.00**	3.11	3.29	20	**1.00**	**1.00**	20	**1.00**	**1.00**	0.82	0.75

objective always achieve better ARI scores when increasing κ. The tree clustering with bi-criteria approach always outperforms the baselines when constraints are present, with the difference most notable with a low number of constraints. The downside to using the tree clustering approaches, however, is evident in the number of feasible solutions that is decreasing as κ increases, exposing the trade-off between interpretability via decision trees and satisfaction of user-provided clustering constraints. Note that the number of feasible solutions corresponds to the tree clustering with the [MD, MS] objective, but the results for the MD objective are the same in all but two cases.

Figure 2 shows an example decision tree of depth 2 found as a Pareto optimal ($\epsilon = 0$) clustering solution for the synthetic Lsun dataset without pairwise constraints ($\kappa = 0$). We see that the solution fails to find the intended clusters in the data and instead focuses on a clustering with a significantly smaller maximum diameter. Figure 3 shows the solution that is found in a similar setting with just 4 pairwise constraints added ($\kappa = 0.01$). We see that the intended clusters are found and perfect accuracy (ARI of 1.0) is achieved. This example demonstrates how a minimal number of constraints combined with the structure of decision trees can significantly improve the accuracy of an interpretable clustering solution.

10.2 Depth and Infeasibility

Our next set of experimental results investigates the effects of the depth parameter on quality and feasibility of decision tree solutions. We solve Problem 1 with three depth values for each dataset, centered around the values specified in Table 2 and Table 3. The results presented in Table 4 show that increasing depth can help solve the

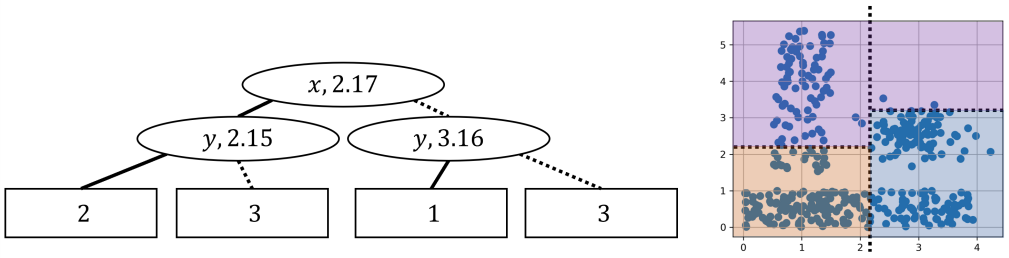

Fig. 2. A Pareto optimal (in [MD, MS], $\epsilon = 0$) decision tree and its corresponding clustering solution for the Lsun dataset.

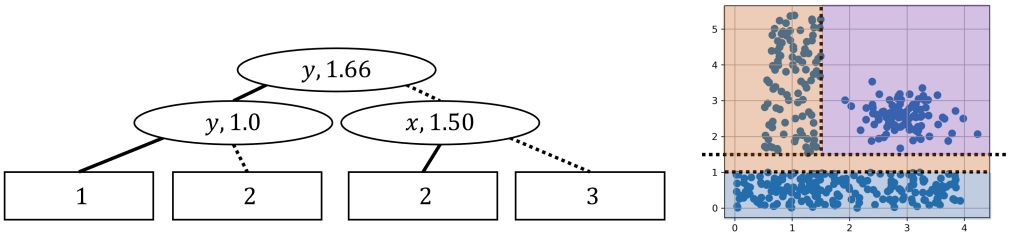

Fig. 3. A Pareto optimal (in [MD, MS], $\epsilon = 0$) decision tree with $\kappa = 0.01$ and its corresponding clustering solution for the Lsun dataset.

infeasibility issue in almost all of the datasets, but it is not guaranteed to do as it fails to produce any feasible solutions for the Spam dataset even at a depth of four. Furthermore, the results show that increasing the depth beyond the point of feasibility will, in all but one case, lead to lower or equal average ARI and NMI scores. This observation is in line with our claim based on Table 2 and Table 3 that the limited structure of trees improves the clustering quality, since shallow trees are more restricted than deeper ones.

10.3 Soft Constraints

Results from Table 4 have underlined the challenge in balancing solution feasibility and accuracy when adding constraints to a fixed-depth decision tree. One way to approach this issue is to address infeasibility through the use of soft constraints that were introduced in Section 7. For our next experiment, we solve Problem 1 with the 1-stage (Section 7.1), 2-stage (Section 7.2), and 3-stage (Section 7.3) schemes to soft constraint optimization. We use higher values of κ for this experiment, as the 2-stage approach is equipped to deal with the infeasibility due to a higher number of constraints. We report the time spent by the solver in each stage independently to distinguish which optimization is the most challenging in multi-stage schemes.

Table 5 and Table 6 presents the results for the 1-stage scheme for the the real and synthetic datasets, respectively. Compared to the results in Table 2 and Table 3, the 1-stage approach achieves slightly worse solutions in cases where all randomization seeds are feasible but also manages to produce solutions in infeasible instances that are more constrained. By maintaining feasibility, it allows us to continually improve ARI and NMI scores by adding constraints without the risk of infeasibility. The results for the 2-stage scheme are presented in Table 7

Table 4. Tree clustering with the [MD, MS] objective for different tree depths ($\kappa = 0.5$, $\epsilon = 0.1$) averaged over 20 runs.

Dataset	Depth	ARI	NMI	Feas.	Time (s)
Iris	2	-	-	0	0.55
	3	0.90	0.88	20	0.69
	4	0.86	0.85	20	0.93
Wine	2	0.90	0.87	1	0.71
	3	0.85	0.82	20	4.42
	4	0.77	0.75	20	5.25
Glass	3	-	-	0	4.61
	4	0.29	0.36	20	1166.98
	5	0.25	0.31	20	46.48
Ionosphere	2	-	-	0	1.83
	3	-	-	0	59.87
	4	0.67	0.55	13	1250.52
Seeds	2	-	-	0	0.74
	3	0.81	0.77	18	1.91
	4	0.76	0.73	20	2.20
Libras	4	0.17	0.40	2	1805.05
	5	0.16	0.41	20	814.88
	6	0.15	0.39	20	496.58
Spam	2	-	-	0	73.70
	3	-	-	0	140.83
	4	-	-	0	629.25
Lsun	2	1.00	1.00	20	1.40
	3	1.00	1.00	20	1.49
	4	1.00	1.00	20	1.68
Chainlink	2	-	-	0	2.78
	3	-	-	0	3.32
	4	1.00	1.00	20	5.04
Target	3	-	-	0	7.19
	4	1.00	1.00	20	8.84
	5	1.00	1.00	20	9.60
Wingnut	2	1.00	1.00	20	2.91
	3	1.00	1.00	20	3.35
	4	1.00	1.00	20	4.03

and Table 8. We observe that the 2-stage scheme is also similarly enabling a higher utilization of clustering constraints to improve accuracy. When comparing against the 1-stage in feasibility and accuracy, we see that each approach is superior in some cases. The incomparability could be due to the 1-stage approach being able to achieve higher objective values in theory but being more computationally demanding in practice. Table 9 and Table 10 present the results for the 3-stage scheme. Similar to before, we see a higher utilization of constraints that lead to higher quality solutions and an inconclusive comparison against the 1-stage and 2-stage schemes.

Table 5. Tree clustering ([MD, MS] objective with $\epsilon = 0.1$) using 1-stage soft constraints method, averaged over 20 runs (real datasets).

Dataset	κ	ARI	NMI	ML%	CL%	Solver Time (s) S1	Feas.
Iris	0.1	0.83	0.83	100.00	100.00	0.72	20
	0.25	0.87	0.85	100.00	100.00	1.27	20
	0.5	0.90	0.88	100.00	100.00	1.30	20
	1	0.95	0.93	99.58	99.85	12.59	20
	1.5	0.97	0.96	99.44	99.77	42.14	20
	2	0.97	0.96	99.47	99.67	159.57	20
Wine	0.1	0.69	0.68	100.00	100.00	1.54	20
	0.25	0.78	0.76	100.00	100.00	4.03	20
	0.5	0.85	0.82	100.00	100.00	8.84	20
	1	0.94	0.91	100.00	100.00	7.87	20
	1.5	0.97	0.95	100.00	100.00	4.30	20
	2	0.98	0.97	100.00	100.00	3.39	20
Glass	0.1	0.22	0.34	100.00	100.00	19.77	20
	0.25	0.22	0.31	100.00	100.00	179.09	20
	0.5	0.27	0.34	100.00	100.00	1323.76	20
	1	0.29	0.34	85.15	97.23	1800.03	20
	1.5	0.31	0.37	73.04	95.09	1800.03	20
	2	0.33	0.39	68.17	94.34	1800.03	20
Ionosphere	0.1	0.29	0.24	100.00	100.00	53.73	20
	0.25	0.49	0.37	99.54	98.96	1221.32	20
	0.5	0.66	0.54	96.40	92.22	1800.03	20
	1	0.68	0.57	93.69	86.18	1800.03	20
	1.5	0.71	0.61	92.87	86.56	1800.03	20
	2	0.71	0.60	91.50	84.84	1800.04	20
Seeds	0.1	0.68	0.66	100.00	100.00	1.42	20
	0.25	0.73	0.70	100.00	100.00	2.11	20
	0.5	0.81	0.78	99.89	99.93	13.23	20
	1	0.91	0.88	98.70	99.12	1246.82	20
	1.5	0.91	0.87	98.07	98.88	1316.63	13
	2	0.94	0.90	97.59	98.55	1317.04	12
Libras	0.1	0.17	0.43	100.00	100.00	1605.63	20
	0.25	0.16	0.41	100.00	100.00	1250.90	20
	0.5	0.15	0.39	100.00	100.00	998.27	20
	1	0.15	0.38	100.00	100.00	1584.48	20
	1.5	0.16	0.39	98.81	99.99	1774.70	20
	2	0.15	0.38	89.63	99.87	1800.12	20
Spam	0.1	0.41	0.32	81.95	72.61	1800.55	20
	0.25	0.41	0.32	79.79	66.21	1800.57	20
	0.5	0.35	0.27	78.26	57.80	1800.58	20
	1	0.18	0.14	84.83	34.79	1800.63	20
	1.5	0.23	0.18	74.44	48.66	1800.73	20
	2	0.20	0.17	84.22	35.58	1800.89	20

Note that any comparison between different schemes should be done in caution as they employ different number of calls to the solver and, as a result, vary in the total time limit that they are given.

The number of cases in which less than 20 feasible solutions are 4 in 1-stage, 4 in 2-stage, and 5 in 3-stage schemes. We also report the percentage of must-links and cannot-links satisfied by each approach. We see that

Table 6. Tree clustering ([MD, MS] objective with $\epsilon = 0.1$) using 1-stage soft constraints method, averaged over 20 runs (synthesized datasets).

Dataset	κ	ARI	NMI	ML%	CL%	Solver Time (s) S1	Feas.
Lsun	0.1	0.90	0.91	100.00	100.00	2.09	20
	0.25	1.00	1.00	100.00	100.00	2.13	20
	0.5	1.00	1.00	100.00	100.00	2.66	20
	1	1.00	1.00	100.00	100.00	2.76	20
	1.5	1.00	1.00	100.00	100.00	2.11	20
	2	1.00	1.00	100.00	100.00	1.78	20
Chainlink	0.1	0.87	0.80	99.92	100.00	15.59	20
	0.25	0.92	0.86	99.12	99.18	641.31	20
	0.5	0.93	0.88	98.26	99.00	1246.58	10
	1	0.95	0.90	98.48	98.16	1796.83	19
	1.5	0.84	0.80	92.41	68.24	1800.06	20
	2	0.74	0.71	88.71	56.34	1800.07	20
Target	0.1	1.00	1.00	100.00	100.00	11.35	20
	0.25	1.00	1.00	100.00	100.00	13.88	20
	0.5	1.00	1.00	100.00	100.00	12.06	20
	1	1.00	1.00	100.00	100.00	13.61	20
	1.5	1.00	1.00	100.00	100.00	16.18	20
	2	1.00	1.00	100.00	100.00	10.60	20
Wingnut	0.1	1.00	1.00	100.00	100.00	2.18	20
	0.25	1.00	1.00	100.00	100.00	2.16	20
	0.5	1.00	1.00	100.00	100.00	2.15	20
	1	1.00	1.00	100.00	100.00	2.96	20
	1.5	1.00	1.00	100.00	100.00	2.80	20
	2	1.00	1.00	100.00	100.00	2.58	20

most schemes maintain an above 90% satisfaction for both types of pairwise constraints. The exceptions are 21 cases in 1-stage, 20 in 2-stage, and 24 in 3-stage.

In Figure 4 and Figure 5, we summarize and combine the ARI scores from Tables 2–10, for real and synthetic datasets, respectively. We compare in an ascending order of κ values the multiple soft constraint schemes against the original approach of treating links as hard constraints. We report the number of feasible solutions within the figure if any of the instances fail to produce a solution, with a cross mark indicating a failure across the board. The trends visible in the figure matches our observations from the tables. Namely, all soft constraint approaches can improve solution quality by incorporating a higher number of constraints without infeasibility.

10.4 Pareto Front Exploration

For our next set of experimental results, we focus on solving Problem 3. We inquire whether exploring a complete Pareto optimal front leads to significantly higher clustering accuracy compared to retrieving an arbitrary Pareto optimal solution. To provide a fair comparison and avoid putting the solutions of Problem 1 at a disadvantage, we consider a global timeout limit (30 minutes) for generating the Pareto front. A global timeout limit spans over multiple calls to the solver. Specifically, it limits every call to the solver to the remaining time, i.e., the global limit minus the overall time spent. Note that Algorithm 2 requires every optimization step to be optimal. Thus, we stop the algorithm whenever the solver times out and produces a potentially non-optimal solution or no solution at all. An incomplete front would then consist of all of the optimal solutions produced up to that point and the

Table 7. Tree clustering ([MD, MS] objective with $\epsilon = 0.1$) using 2-stage soft constraints method, averaged over 20 runs (real datasets).

Dataset	κ	ARI	NMI	ML%	CL%	Time (s)		Feas.
						S1	S2	
Iris	0.1	0.85	0.84	100.00	100.00	0.24	0.71	20
	0.25	0.85	0.84	100.00	100.00	0.23	0.65	20
	0.5	0.91	0.89	100.00	100.00	0.27	0.65	20
	1	0.95	0.93	99.56	99.85	0.39	0.68	20
	1.5	0.97	0.96	99.40	99.80	0.54	0.57	20
	2	0.98	0.97	99.43	99.70	1.18	0.54	20
Wine	0.1	0.71	0.70	100.00	100.00	0.36	1.16	20
	0.25	0.80	0.78	100.00	100.00	0.53	2.12	20
	0.5	0.84	0.81	100.00	100.00	0.91	4.05	20
	1	0.94	0.92	100.00	100.00	1.08	3.91	20
	1.5	0.97	0.95	100.00	100.00	0.71	1.18	20
	2	0.98	0.97	100.00	100.00	0.68	1.02	20
Glass	0.1	0.22	0.33	100.00	100.00	1.74	10.67	20
	0.25	0.23	0.32	100.00	100.00	2.24	64.42	20
	0.5	0.28	0.36	100.00	100.00	36.77	1060.35	20
	1	0.29	0.36	83.52	98.65	1800.61	1559.42	19
	1.5	0.36	0.42	74.14	96.41	1800.61	490.84	20
	2	0.40	0.43	70.03	95.24	1800.61	206.34	20
Ionosphere	0.1	0.29	0.23	100.00	100.00	5.73	39.72	20
	0.25	0.51	0.40	99.91	98.89	575.71	468.46	20
	0.5	0.63	0.52	95.83	91.32	1800.72	85.88	20
	1	0.70	0.58	92.70	88.36	1800.72	24.26	20
	1.5	0.72	0.61	92.25	87.81	1800.72	13.85	20
	2	0.73	0.62	91.45	86.42	1800.72	10.67	20
Seeds	0.1	0.68	0.66	100.00	100.00	0.30	0.90	20
	0.25	0.73	0.71	100.00	100.00	0.36	1.27	20
	0.5	0.80	0.77	99.89	99.93	1.79	1.84	20
	1	0.91	0.87	98.65	99.16	32.85	1.32	20
	1.5	0.92	0.89	97.92	99.03	135.83	0.95	20
	2	0.94	0.90	97.61	98.77	356.80	0.79	20
Libras	0.1	0.18	0.43	100.00	100.00	37.05	1482.43	20
	0.25	0.18	0.43	100.00	100.00	45.52	1189.50	20
	0.5	0.15	0.40	100.00	100.00	75.18	762.91	20
	1	0.16	0.40	100.00	100.00	170.06	1733.30	12
	1.5	0.28	0.51	97.61	99.96	1126.86	1806.85	1
	2	-	-	-	-	1662.52	1806.84	0
Spam	0.1	0.46	0.36	84.02	75.70	1842.85	753.56	20
	0.25	0.46	0.36	81.06	70.75	1842.61	487.23	20
	0.5	0.41	0.33	79.90	62.88	1843.10	357.49	20
	1	0.33	0.27	76.73	58.29	1843.21	200.68	20
	1.5	0.23	0.19	79.70	44.01	1843.12	210.11	20
	2	0.30	0.24	76.89	52.49	1842.89	175.21	20

potentially non-optimal last one. We use lower values for κ since a highly constrained instance tends to converge and have very few solutions in its complete Pareto front.

In Table 11, we report the size, maximum ARI score, and maximum NMI score of the Pareto optimal fronts averaged across seeds. Furthermore, we highlight the number of cases where a complete Pareto front is successfully

Table 8. Tree clustering ([MD, MS] objective with $\epsilon = 0.1$) using 2-stage soft constraints method, averaged over 20 runs (synthesized datasets).

Dataset	κ	ARI	NMI	ML%	CL%	Time (s) S1	Time (s) S2	Feas.
Lsun	0.1	0.90	0.91	100.00	100.00	1.38	1.68	20
	0.25	1.00	1.00	100.00	100.00	1.38	1.34	20
	0.5	1.00	1.00	100.00	100.00	0.58	1.29	20
	1	1.00	1.00	100.00	100.00	0.62	1.07	20
	1.5	1.00	1.00	100.00	100.00	0.64	0.89	20
	2	1.00	1.00	100.00	100.00	0.72	0.88	20
Chainlink	0.1	0.87	0.79	100.00	99.87	2.52	5.16	20
	0.25	0.91	0.85	99.13	99.19	15.67	3.60	20
	0.5	0.94	0.89	98.73	98.70	68.41	2.56	20
	1	0.95	0.91	98.53	98.32	347.40	1.95	20
	1.5	0.95	0.91	98.29	98.33	1036.76	1.88	20
	2	0.96	0.91	98.23	98.23	1440.14	1.85	20
Target	0.1	1.00	1.00	100.00	100.00	6.55	5.30	20
	0.25	1.00	1.00	100.00	100.00	7.55	5.90	20
	0.5	1.00	1.00	100.00	100.00	5.90	7.88	20
	1	1.00	1.00	100.00	100.00	5.05	6.26	20
	1.5	1.00	1.00	100.00	100.00	4.39	5.61	20
	2	1.00	1.00	100.00	100.00	4.74	6.39	20
Wingnut	0.1	1.00	1.00	100.00	100.00	2.04	1.98	20
	0.25	1.00	1.00	100.00	100.00	2.01	1.80	20
	0.5	1.00	1.00	100.00	100.00	1.99	1.80	20
	1	1.00	1.00	100.00	100.00	1.98	1.70	20
	1.5	1.00	1.00	100.00	100.00	1.99	1.69	20
	2	1.00	1.00	100.00	100.00	1.99	1.66	20

generated (Comp.), where an incomplete front is generated (Inc.), and when no feasible solution is found (Inf.). Lastly, we report average ARI and NMI scores from Table 2 and Table 3 again to compare against maximum scores across each front. We observe that finding a complete Pareto optimal front and selecting the best solution leads to comparable quality when a moderate amount of constraints are present. In non-constrained ($\kappa = 0$) cases however, the best solution of the front is able to achieve significantly higher scores in 5 datasets. The similar impact of Pareto front generation and constraints is expected since increasing the constraints forces the front to converge on a few high-quality solutions. We observe the number of solutions in the Pareto front to almost monotonically decrease as more constraints are added, with a few minor increases from $\kappa = 0.1$ to $\kappa = 0.25$ due to high-quality solutions becoming infeasible and being replaced with more than one dominated ones. Lastly, we see feasible solutions produced in as many instances as the singular Pareto optimal solution approach (Table 2 and Table 3), with the only minor improvement occurring for Ionosphere at $\kappa = 0.25$.

10.5 Ablation

In our last experiment, we study the impact of the Smart Pairs pruning algorithm (Section 6) and objective approximation via distance classes (Section 4) on the performance of our approach. We solve Problem 1 in three settings of pruning and non-zero ϵ, non-zero ϵ with no pruning, and no pruning with zero ϵ. Note that $\epsilon = 0$ corresponds to not utilizing distance classes and approximation. We omit presenting ablation results for soft constraints (Section 7) since: 1) their final objective optimization is not as challenging due to maximal constraint satisfaction confining the search space; 2) there cannot be a direct comparison in number of clauses and runtime

Table 9. Tree clustering ([MD, MS] objective with $\epsilon = 0.1$) using 3-stage soft constraints approach, averaged over 20 runs (real datasets).

Dataset	κ	ARI	NMI	ML%	CL%	Time (s) S1	Time (s) S2	Time (s) S3	Feas.
Iris	0.1	0.83	0.82	100.00	100.00	0.25	0.17	0.73	20
	0.25	0.85	0.84	100.00	100.00	0.24	0.17	0.65	20
	0.5	0.91	0.89	100.00	100.00	0.25	0.20	0.67	20
	1	0.94	0.93	98.98	100.00	0.32	0.36	0.72	20
	1.5	0.96	0.95	98.75	99.94	0.42	0.39	0.62	20
	2	0.97	0.95	98.48	99.77	0.91	0.42	0.57	20
Wine	0.1	0.71	0.70	100.00	100.00	0.40	0.30	1.20	20
	0.25	0.80	0.78	100.00	100.00	0.39	0.47	2.14	20
	0.5	0.84	0.81	100.00	100.00	0.67	0.96	4.08	20
	1	0.94	0.92	100.00	100.00	1.46	1.67	3.95	20
	1.5	0.97	0.95	100.00	100.00	2.05	1.49	1.22	20
	2	0.98	0.97	100.00	100.00	2.48	1.56	1.04	20
Glass	0.1	0.22	0.33	100.00	100.00	1.86	1.86	10.51	20
	0.25	0.22	0.31	100.00	100.00	1.66	2.18	76.23	20
	0.5	0.28	0.35	100.00	100.00	1.62	39.16	1097.62	20
	1	0.33	0.38	79.40	100.00	3.05	1800.45	1490.18	16
	1.5	0.29	0.36	65.00	100.00	8.91	1800.47	498.11	20
	2	0.36	0.41	58.28	100.00	34.43	1800.47	207.79	20
Ionosphere	0.1	0.30	0.24	100.00	100.00	4.59	7.69	32.02	20
	0.25	0.50	0.38	99.13	100.00	8.56	617.02	676.74	19
	0.5	0.51	0.40	82.42	98.50	794.13	1306.95	133.52	20
	1	0.60	0.47	82.62	91.53	1800.72	186.86	18.14	20
	1.5	0.66	0.54	84.67	90.32	1800.72	30.15	13.31	20
	2	0.59	0.47	80.42	85.58	1800.72	24.68	15.61	20
Seeds	0.1	0.68	0.66	100.00	100.00	0.33	0.25	0.93	20
	0.25	0.73	0.71	100.00	100.00	0.34	0.32	1.30	20
	0.5	0.80	0.76	99.54	100.00	0.49	1.45	1.97	20
	1	0.87	0.83	94.88	99.87	4.59	6.10	1.69	20
	1.5	0.88	0.85	93.66	99.42	152.91	5.91	1.11	20
	2	0.90	0.86	93.65	99.26	318.36	4.05	0.95	20
Libras	0.1	0.18	0.44	100.00	100.00	36.51	33.87	1402.16	20
	0.25	0.17	0.42	100.00	100.00	40.08	45.87	1138.11	20
	0.5	0.16	0.41	100.00	100.00	54.42	85.63	782.67	20
	1	0.16	0.40	100.00	100.00	70.06	232.78	1670.91	14
	1.5	-	-	-	-	76.25	1002.12	1806.71	0
	2	-	-	-	-	86.58	1748.72	1806.74	0
Spam	0.1	0.36	0.27	69.25	82.52	1843.08	1053.66	698.94	20
	0.25	0.46	0.36	73.64	78.17	1843.09	594.76	440.64	20
	0.5	0.35	0.60	68.57	70.03	1843.13	542.63	376.67	20
	1	0.35	0.26	68.05	67.99	1843.11	374.44	172.61	20
	1.5	0.36	0.28	68.50	68.50	1842.94	283.34	157.66	20
	2	0.34	0.26	67.96	66.78	1843.19	324.20	147.31	20

in a multi-stage approach, as the result of each stage can affect the following ones. In addition to validation criteria and runtime, we report the average number of clauses alongside how many seeds lead to: 1) feasible solutions (Feas.); 2) solver declaring that no solution exists (Inf.); 3) solver terminating without finding a solution or claiming infeasibility (Unk.); and 4) an out-of-memory error (Mem.).

Table 10. Tree clustering ([MD, MS] objective with $\epsilon = 0.1$) using 3-stage soft constraints approach, averaged over 20 runs (synthesized datasets).

Dataset	κ	ARI	NMI	ML%	CL%	Time (s)			Feas.
						S1	S2	S3	
Lsun	0.1	0.90	0.91	100.00	100.00	1.31	1.20	1.70	20
	0.25	1.00	1.00	100.00	100.00	0.77	1.04	1.34	20
	0.5	1.00	1.00	100.00	100.00	0.67	0.55	1.31	20
	1	1.00	1.00	100.00	100.00	0.62	0.72	1.09	20
	1.5	1.00	1.00	100.00	100.00	0.62	0.55	0.92	20
	2	1.00	1.00	100.00	100.00	0.62	0.58	0.91	20
Chainlink	0.1	0.86	0.79	99.92	100.00	1.82	1.05	5.17	20
	0.25	0.91	0.84	97.77	99.78	3.80	2.78	3.63	20
	0.5	0.93	0.87	96.97	99.26	11.12	1.80	2.51	20
	1	0.94	0.90	97.21	98.76	42.97	1.21	1.94	20
	1.5	0.95	0.91	97.71	98.52	131.85	0.81	1.90	20
	2	0.95	0.91	97.69	98.41	262.17	0.75	1.88	20
Target	0.1	1.00	1.00	100.00	100.00	6.49	6.66	5.42	20
	0.25	1.00	1.00	100.00	100.00	8.55	5.55	5.58	20
	0.5	1.00	1.00	100.00	100.00	7.47	5.78	7.81	20
	1	1.00	1.00	100.00	100.00	5.78	4.21	6.26	20
	1.5	1.00	1.00	100.00	100.00	4.82	3.20	5.53	20
	2	1.00	1.00	100.00	100.00	4.67	3.14	6.37	20
Wingnut	0.1	1.00	1.00	100.00	100.00	2.02	0.46	1.96	20
	0.25	1.00	1.00	100.00	100.00	2.03	0.46	1.82	20
	0.5	1.00	1.00	100.00	100.00	2.02	0.47	1.81	20
	1	1.00	1.00	100.00	100.00	2.01	0.46	1.72	20
	1.5	1.00	1.00	100.00	100.00	2.00	0.46	1.70	20
	2	1.00	1.00	100.00	100.00	1.99	0.47	1.66	20

The results in Table 12 show that employing pruning and approximation allows us to significantly reduce the runtime and the number of clauses. The improvement in performance translates to finding higher quality solutions in all cases except for Iris and Wine, easy instances that enjoy a slight benefit in accuracy when approximation of the objective is not in effect ($\epsilon = 0$). For Libras, Spam, and Lsun, the improved performance can help us find more feasible solutions and prove infeasibility for more cases. Note that unlike ϵ-approximation, the Smart Pairs algorithm is guaranteed not to change the set of feasible and optimal solutions. However, since we optimize $\lambda^- - \lambda^+$ and there can be multiple optimal solutions, the different encoding may still lead to a different solution with slightly different ARI and NMI scores, even if the timeout limit is not reached.

11 Concluding Remarks

We present a MaxSAT encoding of decision tree clustering with pairwise constraints as the first combinatorial optimization approach to interpretable constrained clustering. Our approach builds on the direct non-binary branchings first presented in Shati et al. 2023b and equips it with a clustering objective and support for hard and soft constraints. Specifically, it optimizes the well-known bi-criteria objective of maximum diameter and minimum split with guaranteed ϵ-approximation of a Pareto optimal solution while satisfying must-links and cannot-links. We also detail how our encoding can be employed in an iterative algorithm that provides approximations of all, rather than one, of Pareto optimal solutions. The complete approximation of the Pareto front allows the user to choose a viable solution based on domain-specific expertise or heuristics.

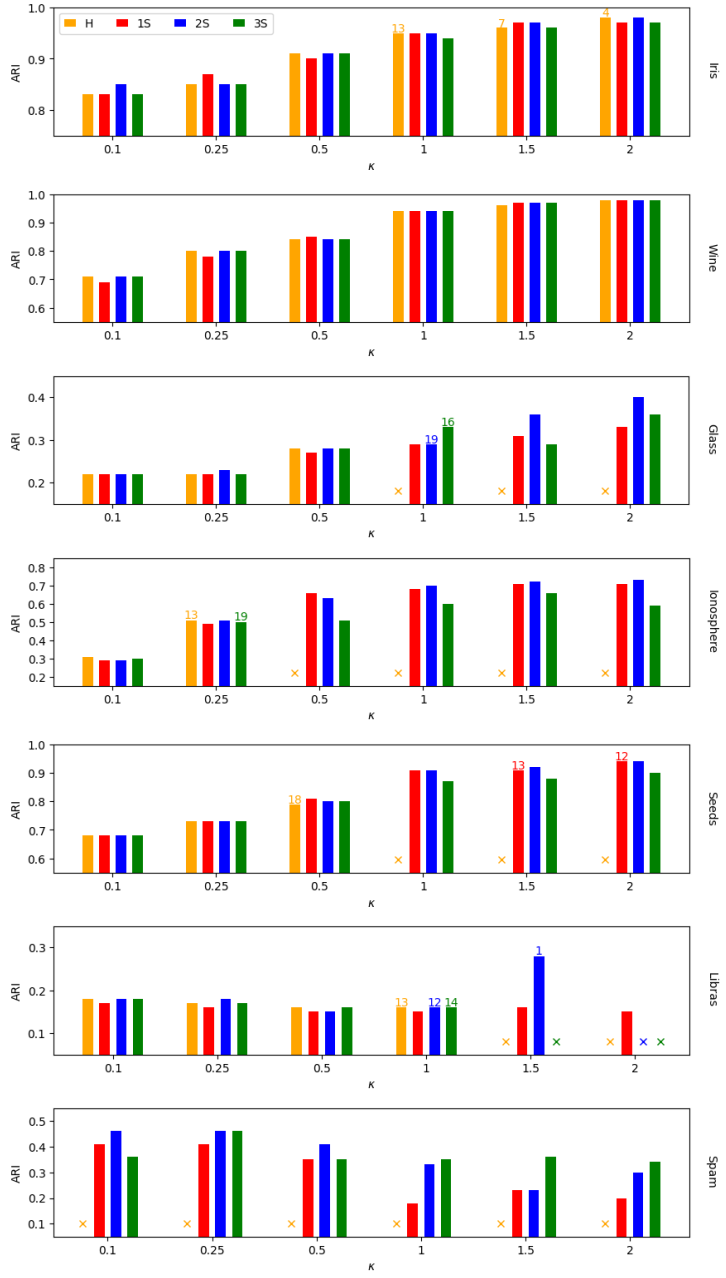

Fig. 4. Tree clustering ([MD, MS] objective with $\epsilon = 0.1$) using hard constraints (H) and 1-stage (1S), 2-stage (2S), and 3-stage (3S) soft constraint approaches for real datasets. In-figure numbers indicate the number of feasible solutions (with 20 invisible and 0 as ×).

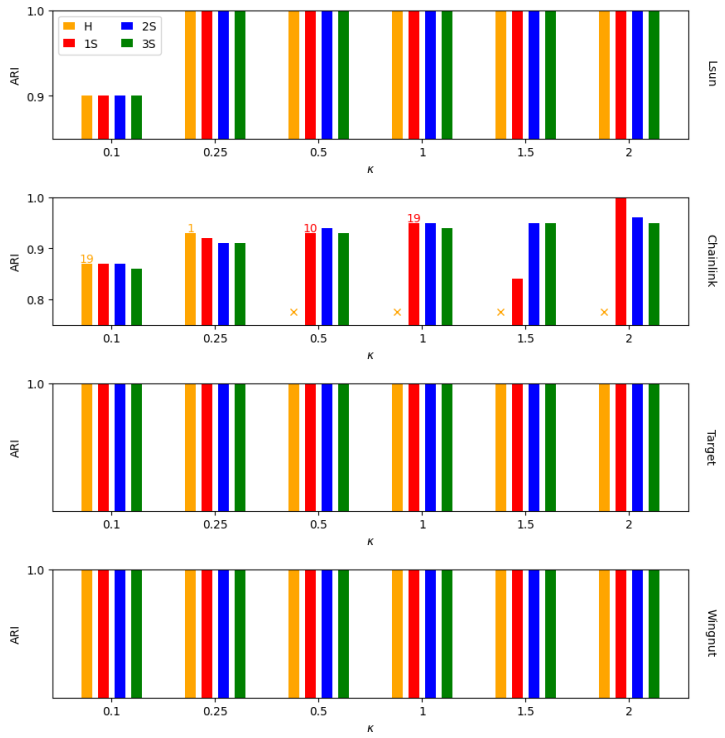

Fig. 5. Tree clustering ([MD, MS] objective with $\epsilon = 0.1$) using hard constraints (H) and 1-stage (1S), 2-stage (2S), and 3-stage (3S) soft constraint approaches for synthetic datasets. In-figure numbers indicate the number of feasible solutions (with 20 invisible and 0 as ×).

Our solutions are interpretable as the decision for placing a data point into a cluster is made by a decision tree. They are also constrained since must-links and cannot-links can be used to represent domain-specific knowledge that should be adhered to, ranging from instance-level constraints to higher-level constraints regarding clusters that can be translated to pairwise ones. Using the formulation approach with the declarative combinatorial optimization problem of MaxSAT allows us to have guarantees on solution quality, flexibility in specifying constraints, and the ability to spread a cluster among multiple leaf nodes. All areas where most of the current body of work on decision tree clustering, which primarily relies on local search and heuristics, fails.

In the Smart Pairs algorithm, we integrate must-links and cannot-links with our objective to provide a pruning framework that detects redundant and infeasible clauses at the time of encoding. As a result, not every clause that we describe in the paper will end up in the instances that we solve, even though there is an equivalency in feasible solutions and the objective value. Smart Pairs algorithm demonstrates that formulation in a combinatorial optimization problem can be pruned in an automated way to improve performance, as long as equivalency with the original formulation is proven.

We further add support for soft constraints in 1-stage, 2-stage, and 3-stage approaches to address the potential issue of infeasibility and pave the way for utilizing a larger amount of semi-supervision through pairwise constraints. The soft approach to constrained ML lacks guaranteed enforcement, but is still clear in how constraint satisfaction is optimized and which constraints are enforced in the end. Our approach can be easily modified to

Table 11. Tree clustering with the [MD, MS] objective ($\epsilon = 0.1$), best of Pareto front (F) compared against single Pareto optimal solution (S), averaged over 20 runs.

Dataset	κ	ARI (F)	NMI (F)	Size	Time (s)	Comp./Inc./Inf.	ARI (S)	NMI (S)
Iris	0	0.89	0.87	7	7.71	20 / 0 / 0	0.6	0.67
	0.1	0.87	0.85	1.75	1.98	20 / 0 / 0	0.83	0.82
	0.25	0.87	0.86	2.1	2.41	20 / 0 / 0	0.85	0.84
Wine	0	0.43	0.55	3	4.85	20 / 0 / 0	0	0.02
	0.1	0.71	0.70	1.4	3.14	20 / 0 / 0	0.71	0.7
	0.25	0.77	0.74	1.8	11.22	20 / 0 / 0	0.8	0.78
Glass	0	0.23	0.37	11	39.43	20 / 0 / 0	0.23	0.38
	0.1	0.23	0.35	3.05	47.11	20 / 0 / 0	0.22	0.33
	0.25	0.27	0.35	3.65	533.59	18 / 2 / 0	0.22	0.31
Ionosphere	0	0.10	0.14	3	63.52	20 / 0 / 0	0.01	0.02
	0.1	0.38	0.29	4.3	292.01	20 / 0 / 0	0.31	0.25
	0.25	0.51	0.40	0.7	1356.61	5 / 10 / 5	0.51	0.4
Seeds	0	0.69	0.66	5	8.31	20 / 0 / 0	0.71	0.67
	0.1	0.71	0.69	2.65	4.44	20 / 0 / 0	0.68	0.66
	0.25	0.77	0.74	2.9	7.81	20 / 0 / 0	0.73	0.71
Libras	0	0.17	0.42	0	1808.08	0 / 20 / 0	0.18	0.44
	0.1	0.17	0.43	0.5	1777.20	2 / 18 / 0	0.18	0.44
	0.25	0.16	0.41	0.6	1663.99	4 / 16 / 0	0.17	0.42
Spam	0	0.00	0.01	6	719.16	20 / 0 / 0	0	0
	0.1	-	-	0	234.59	0 / 0 / 20	-	-
	0.25	-	-	0	168.26	0 / 0 / 20	-	-
Lsun	0	1.00	1.00	3	7.52	20 / 0 / 0	0.4	0.51
	0.1	1.00	1.00	4.95	10.42	20 / 0 / 0	0.9	0.91
	0.25	1.00	1.00	3.8	7.89	20 / 0 / 0	1	1
Chainlink	0	0.23	0.18	9	38.37	20 / 0 / 0	0.24	0.19
	0.1	0.88	0.81	1.3	11.65	19 / 0 / 1	0.87	0.8
	0.25	0.93	0.87	0.05	2.45	1 / 0 / 19	0.93	0.87
Target	0	1.00	1.00	9	125.56	20 / 0 / 0	0.33	0.39
	0.1	1.00	1.00	7.65	98.13	20 / 0 / 0	1	1
	0.25	1.00	1.00	4.1	50.14	20 / 0 / 0	1	1
Wingnut	0	1.00	1.00	1	9.02	20 / 0 / 0	1	1
	0.1	1.00	1.00	1	3.65	20 / 0 / 0	1	1
	0.25	1.00	1.00	1	3.47	20 / 0 / 0	1	1

support hard and soft pairwise constraints simultaneously, with the hard ones representing strict user-provided guidelines and the soft ones mostly utilized for accuracy.

We experimentally show that we can find interpretable solutions of high quality with modest run times. Notably, the first exact approach to do so. We further show that decision trees, the bi-criteria objective, and pairwise constraints can improve solution quality on their own and by complementing each other. This is an important result, demonstrating that restrictions to interpretable solutions, surprisingly, improve accuracy. This

Table 12. Ablation study ([MD, MS] objective with $\kappa = 0.5$) averaged over 20 runs.

Dataset	ϵ	Pruning	ARI	NMI	Time (s)	# Clauses	Feas./Inf./Unk./Mem.
Iris	0.1	SP	0.90	0.88	0.70	3.49E+04	20 / 0 / 0 / 0
	0.1	-	0.90	0.88	2.13	1.02E+05	20 / 0 / 0 / 0
	0	-	0.91	0.89	199.24	1.52E+05	20 / 0 / 0 / 0
Wine	0.1	SP	0.85	0.82	4.42	4.81E+04	20 / 0 / 0 / 0
	0.1	-	0.85	0.82	6.27	1.53E+05	20 / 0 / 0 / 0
	0	-	0.86	0.83	149.16	2.25E+05	20 / 0 / 0 / 0
Glass	0.1	SP	0.29	0.36	1166.21	1.25E+05	20 / 0 / 0 / 0
	0.1	-	0.26	0.32	1290.41	5.45E+05	20 / 0 / 0 / 0
	0	-	0.24	0.32	1414.21	6.52E+05	20 / 0 / 0 / 0
Ionosphere	0.1	SP	-	-	59.19	1.51E+05	0 / 20 / 0 / 0
	0.1	-	-	-	63.59	3.95E+05	0 / 20 / 0 / 0
	0	-	-	-	312.83	6.84E+05	0 / 20 / 0 / 0
Seeds	0.1	SP	0.81	0.77	1.91	5.13E+04	18 / 2 / 0 / 0
	0.1	-	0.80	0.77	3.86	1.88E+05	18 / 2 / 0 / 0
	0	-	0.80	0.77	300.76	2.89E+05	18 / 2 / 0 / 0
Libras	0.1	SP	0.16	0.41	825.30	2.09E+06	20 / 0 / 0 / 0
	0.1	-	0.15	0.39	945.47	4.77E+06	20 / 0 / 0 / 0
	0	-	0.10	0.32	1460.35	5.07E+06	9 / 0 / 0 / 11
Spam	0.1	SP	-	-	143.34	3.83E+06	0 / 20 / 0 / 0
	0.1	-	-	-	343.13	4.61E+07	0 / 20 / 0 / 0
	0	-	-	-	2698.46	9.90E+07	0 / 5 / 15 / 0
Lsun	0.1	SP	1.00	1.00	1.48	5.08E+04	20 / 0 / 0 / 0
	0.1	-	1.00	1.00	6.13	5.96E+05	20 / 0 / 0 / 0
	0	-	0.96	0.93	1686.21	9.90E+05	14 / 0 / 0 / 6
Chainlink	0.1	SP	-	-	3.26	8.50E+04	0 / 20 / 0 / 0
	0.1	-	-	-	17.13	2.08E+06	0 / 20 / 0 / 0
	0	-	-	-	92.84	4.57E+06	0 / 20 / 0 / 0
Target	0.1	SP	1.00	1.00	8.86	2.61E+05	20 / 0 / 0 / 0
	0.1	-	1.00	1.00	61.87	4.96E+06	20 / 0 / 0 / 0
	0	-	0.96	0.93	1843.70	6.43E+06	20 / 0 / 0 / 0
Wingnut	0.1	SP	1.00	1.00	3.32	9.41E+04	20 / 0 / 0 / 0
	0.1	-	1.00	1.00	18.84	2.14E+06	20 / 0 / 0 / 0
	0	-	1.00	1.00	1679.13	4.71E+06	20 / 0 / 0 / 0

result expands upon one of the central claims in the interpretable ML literature (Rudin 2019), extending it to the problem of clustering. The improvement can hypothetically be explained by observing that the natural structure and simulatability in interpretable solutions aptly complements the clustering objective in aspects that it does not encompass. Moreover, our results show that clustering constraints improve solution accuracy in the context of decision tree solutions, similar to how they do so for non-tree solutions (Wagstaff and Cardie 2000).

Our experiments demonstrate the trade-off between solution quality and feasibility. We show how the support for soft constraints in our proposed schemes allows us to consider a larger set of constraints without infeasibility

and further improve accuracy. Moreover, we run the complete Pareto front generation algorithm and show that solutions of higher quality can be found when there is an absence of constraints. Picking a solution from the front can be seen as another form of user engagement that can replace the utilization of constraints in improving accuracy. Lastly, we perform ablation studies and highlight the impact of ϵ-approximation and Smart Pairs algorithm on the size of the produced encodings and, as a result, performance.

Our work can be extended in multiple interesting directions in the future. Multiple solutions of varying accuracy can have the same objective values of MD and MS. A secondary criterion, such as within cluster sum of squares (Brusco and Steinley 2007), can be considered and optimized as a secondary stage to distinguish such solutions. With regard to soft constraints, other optimization strategies can be investigated. One such alternative is to chain link satisfaction and only consider a link when all the previous ones, according to an arbitrary or distance-based order, are satisfied. Our preliminary results on chained link satisfaction suggests that it might lead to feasibility or quality issues. Further evaluation is needed to make a conclusive statement. Additionally, experiments can be performed to study the simultaneous optimization of the bi-criteria objective and soft pairwise constraint satisfaction. Examples include weighing one objective against the other or producing all Pareto optimal solutions. In any case where multiple objectives are being optimized, alternative strategies and tools, such as multi-objective solvers (Jabs et al. 2022), can be investigated and empirically evaluated for exploring the space of all Pareto optimal solutions more efficiently. Another direction to enhance performance is a more thorough pruning. The Smart Pairs algorithm can be equipped with more sophisticated algorithms from graph theory to detect more infeasible and redundant cases. For example, the current version permits the infeasible case in which there are two clusters and three data points with cannot-links between them. It also does not detect redundancy when there are two clusters and a must-link is being added between two points that both are connected to a third point with cannot-links. Lastly, the decision tree encoding can be extended to new objectives and new constraints with potential integration into the Smart Pairs algorithm in interesting ways.

Acknowledgments

This research was supported by grant number DSI-CGY3R1P18 from the Data Sciences Institute at the University of Toronto. The authors also gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Canada CIFAR AI Chairs Program. Resources used in preparing this research were provided, in part, by the Province of Ontario, the Government of Canada through CIFAR, and companies sponsoring the Vector Institute for Artificial Intelligence (www.vectorinstitute.ai/partners). The final author would also like to thank the Schwartz Reisman Institute for Technology and Society for providing a rich multidisciplinary research environment.

References

S. Aghaei, M. J. Azizi, and P. Vayanos. 2019. "Learning optimal and fair decision trees for non-discriminative decision-making." In: *AAAI Conference on Artificial Intelligence (AAAI)*, 1418–1426.

G. Aglin, S. Nijssen, and P. Schaus. 2020. "Learning optimal decision trees using caching branch-and-bound search." In: *AAAI Conference on Artificial Intelligence (AAAI)*, 3146–3153.

G. Aglin, S. Nijssen, and P. Schaus. 2021. "Pydl8. 5: a library for learning optimal decision trees." In: *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 5222–5224.

F. Avellaneda. 2020. "Efficient inference of optimal decision trees." In: *AAAI Conference on Artificial Intelligence (AAAI)*, 3195–3202.

B. Babaki, T. Guns, and S. Nijssen. 2014. "Constrained clustering using column generation." In: *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 438–454.

J. Berg, E. Demirović, and P. J. Stuckey. 2019. "Core-boosted linear search for incomplete MaxSAT." In: *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*. Springer, 39–56.

J. Berg and M. Järvisalo. 2017. "Cost-optimal constrained correlation clustering via weighted partial maximum satisfiability." *Artificial Intelligence*, 244, 110–142.

D. Bertsimas and J. Dunn. 2017. "Optimal classification trees." *Machine Learning*, 106, 7, 1039–1082.

D. Bertsimas, A. Orfanoudaki, and H. Wiberg. 2021. "Interpretable clustering: an optimization approach." *Machine Learning*, 110, 89–138.
C. Bessiere, E. Hebrard, and B. O'Sullivan. 2009. "Minimising decision tree size as combinatorial optimisation." In: *International Conference on Principles and Practice of Constraint Programming (CP)*. Springer, 173–187.
M. Bilenko, S. Basu, and R. J. Mooney. 2004. "Integrating constraints and metric learning in semi-supervised clustering." In: *Proceedings of the twenty-first international conference on Machine learning*, 11.
L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and regression trees*. Wadsworth & Brooks/Cole Advanced Books & Software.
M. J. Brusco and D. Steinley. 2007. "A comparison of heuristic procedures for minimum within-cluster sums of squares partitioning." *Psychometrika*, 72, 583–600.
E. Cohen. 2023. "Interpretable Clustering via Soft Clustering Trees." In: *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 281–298.
E. Cohen, A. Senderovich, and J. C. Beck. 2020. "An Ising framework for constrained clustering on special purpose hardware." In: *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 130–147.
V. G. Costa and C. E. Pedreira. 2023. "Recent advances in decision trees: An updated survey." *Artificial Intelligence Review*, 56, 5, 4765–4800.
T.-B.-H. Dao, K.-C. Duong, and C. Vrain. 2017. "Constrained clustering by constraint programming." *Artificial Intelligence*, 244, 70–94.
T.-B.-H. Dao, C. Vrain, K.-C. Duong, and I. Davidson. 2016. "A framework for actionable clustering using constraint programming." In: *Proceedings of the Twenty-second European Conference on Artificial Intelligence*, 453–461.
I. Davidson and S. Ravi. 2005a. "Agglomerative hierarchical clustering with constraints: Theoretical and empirical results." In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 59–70.
I. Davidson and S. Ravi. 2005b. "Clustering with constraints: Feasibility issues and the k-means algorithm." In: *Proceedings of the 2005 SIAM international conference on data mining*. SIAM, 138–149.
I. Davidson, S. Ravi, and L. Shamis. 2010. "A SAT-based framework for efficient constrained clustering." In: *Proceedings of the 2010 SIAM international conference on data mining*. SIAM, 94–105.
E. Demirović, A. Lukina, E. Hebrard, J. Chan, J. Bailey, C. Leckie, K. Ramamohanarao, and P. J. Stuckey. 2022. "Murtree: Optimal decision trees via dynamic programming and search." *Journal of Machine Learning Research*, 23, 26, 1–47.
D. Dinler and M. K. Tural. 2016. "A survey of constrained clustering." In: *Unsupervised learning algorithms*. Springer, 207–235.
D. Dua and C. Graff. 2017. *UCI Machine Learning Repository*. (2017). http://archive.ics.uci.edu/ml.
J. C. Dunn. 1974. "Well-separated clusters and optimal fuzzy partitions." *Journal of cybernetics*, 4, 1, 95–104.
N. Frost, M. Moshkovitz, and C. Rashtchian. 2020. *ExKMC: Expanding Explainable k-Means Clustering*. (2020).
Z. Fu and S. Malik. 2006. "On solving the partial MAX-SAT problem." In: *International Conference on Theory and Applications of Satisfiability Testing (SAT)*. Springer, 252–265.
M. Gabidolla and M. Á. Carreira-Perpiñán. 2022. "Optimal interpretable clustering using oblique decision trees." In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 400–410.
B. Gamlath, X. Jia, A. Polak, and O. Svensson. 2021. "Nearly-tight and oblivious algorithms for explainable clustering." *Advances in Neural Information Processing Systems*, 34, 28929–28939.
H. Hu, M. Siala, E. Hébrard, and M.-J. Huguet. 2020. "Learning optimal decision trees with MaxSAT and its integration in AdaBoost." In: *International Joint Conference on Artificial Intelligence and Pacific Rim International Conference on Artificial Intelligence (IJCAI-PRICAI)*.
L. Hubert and P. Arabie. 1985. "Comparing partitions." *Journal of Classification*, 2, 1, 193–218.
A. Ignatiev, J. Marques-Silva, N. Narodytska, and P. J. Stuckey. 2021. "Reasoning-Based Learning of Interpretable ML Models." In: *International Joint Conference on Artificial Intelligence (IJCAI)*, in press.
C. Jabs, J. Berg, A. Niskanen, and M. Järvisalo. 2022. "MaxSAT-Based Bi-Objective Boolean Optimization." In: *25th International Conference on Theory and Applications of Satisfiability Testing (SAT 2022)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.
P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis. 2020. "Explainable ai: A review of machine learning interpretability methods." *Entropy*, 23, 1, 18.
H. Liu and Y. Fu. 2015. "Clustering with partition level side information." In: *2015 IEEE international conference on data mining*. IEEE, 877–882.
H. Liu, Z. Tao, and Y. Fu. 2017. "Partition level constrained clustering." *IEEE transactions on pattern analysis and machine intelligence*, 40, 10, 2469–2483.
C. Min Li and F. Manya. 2009. "MaxSAT, Hard and Soft Constraints." In: *Handbook of Satisfiability*. Vol. 185. Ed. by A. Biere, M. Heule, and H. van Maaren. IOS Press.
M. Moshkovitz, S. Dasgupta, C. Rashtchian, and N. Frost. 2020. "Explainable k-means and k-medians clustering." In: *International Conference on Machine Learning*. PMLR, 7055–7065.
W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. 2019. "Definitions, methods, and applications in interpretable machine learning." *Proceedings of the National Academy of Sciences*, 116, 44, 22071–22080.
N. Narodytska, A. Ignatiev, F. Pereira, J. Marques-Silva, and I. RAS. 2018. "Learning Optimal Decision Trees with SAT." In: *International Joint Conference on Artificial Intelligence (IJCAI)*, 1362–1368.

D. Pelleg and D. Baras. 2007. "K-means with large and noisy constraint sets." In: *European Conference on Machine Learning*. Springer, 674–682.
J. R. Quinlan. 2014. *C4. 5: programs for machine learning*. Elsevier.
J. R. Quinlan. 1986. "Induction of decision trees." *Machine learning*, 1, 1, 81–106.
P. J. Rousseeuw. 1987. "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis." *Journal of computational and applied mathematics*, 20, 53–65.
C. Rudin. 2019. *Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat Mach Intell 1: 206–215*. (2019).
S. Shalev-Shwartz and S. Ben-David. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.
P. Shati, E. Cohen, and S. McIlraith. 2023a. "Optimal decision trees for interpretable clustering with constraints." In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, 2022–2030.
P. Shati, E. Cohen, and S. McIlraith. 2021. "SAT-Based Approach for Learning Optimal Decision Trees with Non-Binary Features." In: *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.
P. Shati, E. Cohen, and S. McIlraith. 2023b. "SAT-based optimal classification trees for non-binary data." *Constraints*, 28, 2, 166–202.
A. Strehl and J. Ghosh. 2002. "Cluster ensembles—a knowledge reuse framework for combining multiple partitions." *Journal of Machine Learning Research*, 3, 583–617.
V. T'kindt and J.-C. Billaut. 2006. *Multicriteria scheduling: theory, models and algorithms*. Springer Science & Business Media.
A. Thalamuthu, I. Mukhopadhyay, X. Zheng, and G. C. Tseng. 2006. "Evaluation and comparison of gene clustering methods in microarray analysis." *Bioinformatics*, 22, 19, 2405–2412.
A. Ultsch and J. Lötsch. 2020. "The fundamental clustering and projection suite (fcps): A dataset collection to test the performance of clustering and data projection algorithms." *Data*, 5, 1, 13.
H. Verhaeghe, S. Nijssen, G. Pesant, C.-G. Quimper, and P. Schaus. 2020. "Learning optimal decision trees using constraint programming." *Constraints*, 25, 3, 226–250.
S. Verwer and Y. Zhang. 2019. "Learning optimal classification trees using a binary linear program formulation." In: *AAAI Conference on Artificial Intelligence (AAAI)*, 1625–1632.
K. Wagstaff and C. Cardie. 2000. "Clustering with Instance-level Constraints." In: *Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer, 1103–1110.
K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. 2001. "Constrained K-means Clustering with Background Knowledge." In: *Proceedings of the Eighteenth International Conference on Machine Learning*, 577–584.
X. Wang and I. Davidson. 2010. "Flexible constrained spectral clustering." In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 563–572.
R.-S. Wu and P.-H. Chou. 2011. "Customer segmentation of multiple category data in e-commerce using a soft-clustering approach." *Electronic Commerce Research and Applications*, 10, 3, 331–341.
H. Yang, L. Jiao, and Q. Pan. 2021. "A survey on interpretable clustering." In: *2021 40th Chinese Control Conference (CCC)*. IEEE, 7384–7388.

A Revised MIP Encoding of Decision Tree Clustering Baseline

In this section, we present a revised version of the decision tree clustering MIP formulation presented in Bertsimas, Orfanoudaki, et al. 2021 to be used as a baseline against our approach. We change the original model to maintain notational consistency, fix typographical mistakes, address various logical issues that are crucial for one's ability to solve the model, and extend the model with support for pairwise constraints. The variables used to represent different aspects of the model are as follows:

- s_i: The $s(i)$ variables in the Silhouette objective.
- q_i: The $b(i)$ variables in the Silhouette objective, namely the average distance between points and their second closest cluster.
- r_i: The $a(i)$ variables in the Silhouette objective, namely the average distance between points and their cluster.
- m_i: $max(b(i), a(i))$ in the Silhouette objective.
- γ_{it} [Binary]: Cluster (leaf) t is the 2nd closest cluster to x_i.
- c_{it}: The distance between x_i and cluster (leaf) t.
- z_{it} [Binary]: x_i belongs to cluster (leaf) t.
- K_t: The size of cluster (leaf) t.
- a_{jt} [Binary]: Feature j is chosen at branching node t.

- d_t [Binary]: Branching node t is activated.
- b_t: Threshold at branching node t.
- l_t [Binary]: Leaf node t is activated.

The following set of linear and quadratic constraints model the objective (Eq. (50)), the relation between Silhouette values (Eq. (51) to Eq. (53)), the selection and distance to the second closest cluster (Eq. (54) to Eq. (56)), the distance to each cluster (Eq. (57) and Eq. (58)), the size of each cluster (Eq. (59) and Eq. (60)), the activation of branching nodes (Eq. (61) to Eq. (63)), the activation of leaf nodes (Eq. (64) and Eq. (65)), and the branchings leading up to appearance at leaves (Eq. (66)-(68)). Note that N_{min} specifies the minimum number of points appearing at each active leaf (minimum support). See Bertsimas, Orfanoudaki, et al. 2021 for a detailed discussion on the role of each constraint. The clauses in Eq. (69) and Eq. (70) are new to the model and guarantee the satisfaction of must-links and cannot-links, respectively. The range of integer and real variables are specified in Eq. (71) to Eq. (74).

$$\text{minimize} \quad -\frac{1}{|X|}\sum_{x_i \in X} s_i \tag{50}$$

$$\text{subject to} \quad s_i m_i = q_i - r_i, \qquad \forall x_i \in X \tag{51}$$

$$m_i \geq q_i, \qquad \forall x_i \in X \tag{52}$$

$$m_i \geq r_i, \qquad \forall x_i \in X \tag{53}$$

$$q_i = \sum_{t \in \mathcal{T}_L} \gamma_{it} c_{it}, \qquad \forall x_i \in X \tag{54}$$

$$\sum_{t \in \mathcal{T}_L} \gamma_{it} = 1, \qquad \forall x_i \in X \tag{55}$$

$$\gamma_{it} \leq (1 - z_{it}), \qquad \forall x_i \in X, t \in \mathcal{T}_L \tag{56}$$

$$r_i = \sum_{t \in \mathcal{T}_L} c_{it} z_{it}, \qquad \forall x_i \in X, t \in \mathcal{T}_L \tag{57}$$

$$c_{it} K_t = \sum_{x_j \in X} z_{jt} |x_i - x_j|, \qquad \forall x_i \in X, t \in \mathcal{T}_L \tag{58}$$

$$\gamma_{it} \leq K_t, \qquad \forall x_i \in X, t \in \mathcal{T}_L \tag{59}$$

$$K_t = \sum_{x_i \in X} z_{it} \qquad \forall t \in \mathcal{T}_L \tag{60}$$

$$\sum_{j \in F} a_{jt} = d_t, \qquad \forall t \in \mathcal{T}_B \tag{61}$$

$$0 \leq b_t \leq 100 d_t, \qquad \forall t \in \mathcal{T}_B \tag{62}$$

$$d_t \leq d_{p(t)}, \qquad \forall t \in \mathcal{T}_B - \{1\} \tag{63}$$

$$z_{it} \leq l_t, \qquad \forall t \in \mathcal{T}_L \tag{64}$$

$$\sum_{x_i \in X} z_{it} \geq N_{min} l_t, \qquad \forall t \in \mathcal{T}_L \tag{65}$$

$$\sum_{t \in \mathcal{T}_L} z_{it} = 1, \qquad \forall x_i \in X \tag{66}$$

$$a_m^T x_i \geq b_m - 100(1 - z_{it}), \qquad \forall x_i \in X, t \in \mathcal{T}_L, m \in A_R(t) \tag{67}$$

$$a_m^T x_i + \epsilon_s \leq b_m + (100 + \epsilon_s)(1 - z_{it}), \qquad \forall x_i \in X, t \in \mathcal{T}_L, m \in A_L(t) \tag{68}$$

$$z_{it} = z_{jt}, \qquad \forall t \in \mathcal{T}_L, (i, j) \in ML \tag{69}$$

$$\sum_{t \in \mathcal{T}_L} (z_{it} - z_{jt})(z_{it} - z_{jt}) \geq 1, \qquad \forall (i, j) \in CL \tag{70}$$

$$a_{jt}, d_t \in \{0, 1\}, \qquad \forall j \in F, t \in \mathcal{T}_B \tag{71}$$

$$z_{it}, l_t \in \{0, 1\}, \qquad \forall x_i \in X, t \in \mathcal{T}_L \tag{72}$$

$$\gamma_{it} \in \{0, 1\}, \qquad \forall x_i \in X, t \in \mathcal{T}_L \tag{73}$$

$$b_t \leq 100, \qquad \forall t \in \mathcal{T}_B \tag{74}$$

The model presented above is different from the original one in various ways. We have categorized our modifications into three groups based on their nature and significance:

- Changes in the notation:
  - Iterating over points: $i = 1, ..., n$ to $x_i \in X$
  - Iterating over features: $j = 1, ..., p$ to $f \in F$
  - Distance between two points: d_{ij} to $|x_i - x_j|$
  - The epsilon value used in modelling branchings (to avoid confusion with the epsilon used for approximating objective in our method): ϵ to ϵ_s

- Typographical errors:
  - The first dimension of the z_{it} in Eq. (72): p to n
  - Sum index in Eq. (57): $\forall t \in T_L$ to $t \in T_L$
  - Branching nodes in Eq. (67) and Eq. (68): T_B to T_L
  - Index of b variables in Eq. (67) and Eq. (68): b_t to b_m
- Logical issues of the model:
  - The big M constant in Eq. (56) serves no purpose: removed M.
  - Gurobi does not allow division to be used in Eq. (51) and Eq. (58): replaced divisions with their equivalent multiplication. Note that the divided-by-zero cases should be monitored to maintain the correct semantics.
  - Eq. (54) allowing q_i to be arbitrarily large will trivialize the objective: changed $\geq$ to $=$.
  - Not having Eq. (59) will allow empty clusters to be selected as the second closest cluster: added Eq. (59).
  - Since ϵ_s is multiplied by a_m^T on the left hand side of Eq. (68), it is possible for points to be directed to both left and right children, if a branching node is disabled: changed $a_m^T(x_i + \epsilon_s)$ to $a_m^T x_i + \epsilon_s$.
  - Not having a constant upper bound for b_t variables causes runtime issues: added Eq. (74).
- Modifications to match our model:
  - The original model does not support must-links and cannot-links: added Eqs. (69) and (70).
  - Constraints Eq. (62), Eq. (67), and Eq. (68) need to be modified for a feature normalization to $[0, 100]$ rather than $[0, 1]$: changed d_t to $100d_t$, $(1 - z_{it})$ to $100(1 - z_{it})$, and $(1 + \epsilon_s)$ to $(100 + \epsilon_s)$, respectively.

Received 19 January 2025; revised 18 June 2025; accepted 19 June 2025