Individually fair optimal decision trees

Using a dynamic programming approach

Chrysanthos Kindynis[1]

Supervisor(s): Emir Demirović[1]**, Koos van der Linden**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2023

Name of the student: Chrysanthos Kindynis
Final project course: CSE3000 Research Project
Thesis committee: Emir Demirović, Koos van der Linden, Burcu Özkan

Abstract

In this paper, we tackle the problem of creating decision trees that are both optimal and individually fair. While decision trees are popular due to their interpretability, achieving optimality can be difficult. Existing approaches either lack scalability or fail to consider individual fairness. To address this, we define individual fairness as a separable optimization task by analyzing the fairness gained and lost within a sub-tree. Using the Streed framework, we implement an algorithm that constructs optimal decision trees with the lowest misclassification score and individual fairness value above a certain threshold. Our algorithm has been tested on various datasets, demonstrating its effectiveness and scalability. This research is a significant step towards creating fair decision trees that are optimal, fair, and scalable.

Keywords: optimal decision trees, individual fairness, dynamic programming, separability.

1 Introduction

A decision tree is a helpful machine learning model that provides classification or numerical predictions on data. It achieves this by repeatedly asking questions about the data characteristics, making it easy for humans to understand its logic. Due to its interpretability, it has become one of the most commonly used techniques for classification problems [3].

Today, machine learning models are increasingly used to assist decision-making processes. This has led to the search for optimal decision trees, i.e. trees that can globally maximize their objective under a specified size limit. However, finding such optimal decision trees is an *NP-hard* problem [10], making it a difficult challenge to overcome.

Common approaches for constructing optimal decision trees, such as Mixed-Integer-Programming (MIP) [18], Boolean Satisfiability (SAT) [11] and Constraint Programming (CP) [17], use general-purpose solvers that try to face the problem as a whole. These appear flexible in incorporating various objectives but do not scale beyond depth three. On the other hand, a Dynamic Programming (DP) approach [6] [12] explores the recursive structure of the tree, which along with other algorithmic enhancements such as caching and use of bounds, allows the performance to be several orders of magnitude better. These approaches, however, lack generalizability in incorporating various objectives and constraints.

The wide application of such models has raised concerns about their fairness properties [2] [13]. Research on "fair" machine learning has introduced various ways to ensure fairness [4], with a focus on group-based fairness [1] [9] [20] [16]. Notably, among these studies, the "Streed" generalized DP framework [16] for constructing optimal decision trees was introduced, allowing for a wider variety of optimization objectives to be included in the dynamic programming approach. The only condition for using this framework lies in the ability to calculate the objective of two subtrees *separately*. By contrast to the focus of the research, Dwork et al. [7] showcase that even if group fairness is established in a model, the outcome can still be unfair for the individuals. Therefore, establishing individual fairness by ensuring similar outcomes for similar individuals [7] is important. Research has explored individual fairness in decision trees using Abstract Interpretation [14], but the developed model does not guarantee optimality.

In this research, we aim to answer the following questions: Can the Streed DP framework [16] be utilized to create optimal classification decision trees that are fair to the individual? If yes, how does its performance scale across different data sets and depth limits? Using the Streed framework ensures that the model is optimal, which differs from the existing individual fairness solution. Additionally, we aim to take advantage of the DP approach's scalability benefits instead of relying on the general purpose solver's scalability capabilities. Furthermore, we address the limitations of group fairness models by incorporating individual fairness constraints.

Our findings include achieving the separability condition by arguing about individual fairness lost and gained within a subtree. The experiment results show the algorithm's scalability across the three parameters of the number of instances, features and similar individuals. We found that the algorithm can solve problems with a depth limit of two within milliseconds, while a depth limit of three can often be solved. In cases with few similar individuals, even depth four can be solved.

Overall, we have three main contributions: 1) Definition of an optimization task which considers the accuracy and individual fairness while meeting the separability conditions; 2) Implementation of its algorithm, utilizing the "Streed" framework, which constructs optimal decision trees with the lowest miss classification score and individual fairness value above a certain threshold 3) Scalability and performance analysis which shows that the depth limit of 3 is well within this algorithm's abilities.

This paper is organized as follows: We begin by reviewing previous research in this field. Next, we provide the necessary background information and notation in the preliminaries section. Our main contributions are presented, followed by the experimental results demonstrating our proposed approach's performance and scalability. Finally, we summarize the key findings, discuss the implications, and suggest areas for future research.

2 Related Work

At the beginning of the search for high-quality decision trees, heuristic methods like Cart [5] and C4.5 [15] were used due to the difficulty of finding optimal ones. These methods are built by optimizing a local objective function, such as information gain or entropy. Although the resulting decision trees are of high quality and calculated quickly, they do not guarantee optimality in their objective.

On the other hand, *Mixed Integer Programming* is a standard method for constructing optimal decision trees. The problem is formulated as a mixed integer program, where linear functions model the objectives and constraints. Off-

the-shelf software is then utilized to find the optimal decision tree by calculating the values of the variables that satisfy all constraints and maximize the objective. Separating these two steps allows this method to optimize for various objectives and constraints.

In this field of study, my research relates to Aghaei et al.'s work [1], where they are developing fair models that minimize discrimination against people of a protected category. A protected characteristic (e.g. race) is described as a socially sensitive characteristic based on which discrimination is forbidden by the law [8]. The decision trees developed by Aghaei et al. [1] minimize a linear loss function, including a discrimination regularizer responsible for twisting the trade-off between accuracy and fairness. Note that none of the two fairness metrics corresponds to individual fairness. They instead refer to direct and indirect discrimination based on protected features. Notably, while MIP guarantees the creation of a decision tree that minimizes the defined loss function, the analysis of its runtime performance [1] could be in much more detail. For example, analyzing the training runtime for various tree depths and data set sizes would be helpful since this would enable us to discuss scalability and usability within available resources and compare alternative approaches.

Moreover, *dynamic programming* is an alternative approach to constructing optimal decision trees. By contrast, dynamic programming does not attempt to solve the problem as a monolith. However, it exploits the recursive structure of decision trees by looking at sub-trees as individual sub-problems. It also enhances its performance by pruning based on estimated lower and upper bounds and caching sub-solutions. This approach requires that all the features in the data are binary. If not, a binarisation method is applied as a pre-processing step, transforming the features into binary.

Additionally, Demirovic et al. [6] implemented the DP approach we just introduced. The *MurTree* algorithm they implemented can generate an optimally accurate decision tree for a classification task, with a much faster (and therefore scalable) performance compared to the MIP algorithms implemented so far. However, this algorithm is unsuitable for achieving individual fairness as it focuses solely on accuracy and does not accommodate different objectives and constraints.

Furthermore, the research by Linden et al. [16] attempts to push the limits of dynamic programming by generalizing the *MurTree* algorithm to optimize for any *separable* objective. The paper introduces necessary and sufficient conditions for the separability of an objective or constraint and suggests that any task with such objectives can be implemented using dynamic programming. Their algorithm called *"Streed"* (Separable Trees with Dynamic programming), highly influenced my research since it offers a general DP framework and has promising results on scalability.

Additionally, the ethical aspect of the algorithm includes individual fairness, a principle introduced by Dwork et al. [5], who also noted that a distance metric is necessary to define similarity between individuals. This distance metric (strictly) only requires the following properties: $d(x, y) \geq 0$, $d(x, y) = d(y, x)$, and $d(x, x) = 0$. One way to differentiate between group and individual fairness is that group fairness involves maintaining statistical parity, which means that the demographics of those who receive positive or negative classifications are the same as those of the entire population. On the other hand, individual fairness does not consider this aspect. Additionally, Dwork et al. [5] argue that even if group fairness is achieved, individuals may still perceive the outcome as unfair, highlighting the significance of individual fairness. This research exclusively focuses on ensuring individual fairness in the developed models since group fairness has already been resolved through a dynamic programming approach in [16].

Ranzato et al. [14] used Abstract Interpretation to research individual fairness in decision trees. They proposed the individual fairness metric of the proportion of fair data points, with a data point being considered fair if it has the same label as every other data point close to it. Additionally, they consider similarity based on distance in the input space, not distance in the data set (e.g. k-nearest). Moreover, they optimize for a linear loss function, a weighted sum of the 0.9 miss-classification score and 0.1 individual fairness metric. Although their approach may generate models that are fairer than state-of-the-art methods, it is still possible for them to produce sub-optimal loss values. Therefore, optimality cannot be guaranteed.

3 Preliminaries

This section introduces necessary decision tree notation, defines the optimization task problem, introduces the requirement of separability and provides the dynamic programming framework.

3.1 Decision tree notation

We use F to represent the set of features and K to represent the set of labels that describe a data instance. Let D, be a data set consisting of data instances, each described by a feature vector $x \in \{0, 1\}^{|F|}$ a label $k \in K$. We use f (and $\overline{f}$) to denote that a feature is present (or not) in a data instance. By present, we refer to the feature value 1. Similarly, let D_f describe the set of instances in D that satisfy feature f, and $D_{\overline{f}}$ the set of instances that do not. For each data point $dp \in D$, we can access its actual label and predicted label through $dp.k$, $dp.\hat{k}$ respectively. Additionally, the decision tree's maximum depth is represented by d.

3.2 Optimization task

We adopt the definition of an optimization task from [16], which involves a state space **S** and a solution space **V**, and consists of six components:

1. A cost function $g : S \times (F \cup K) \rightarrow V$ that returns the cost of action $a \in F \cup K$ in state $s \in S$, where the action is either branching on feature $f \in F$ or assigning label $k \in K$.
2. A transition function $t : S \times F \times 0, 1 \rightarrow S$ that provides the next state after branching left or right on feature f, denoted by f or $\overline{f} \in F \times 0, 1$.
3. A comparison operator $\succ$: $V \times V \rightarrow 0, 1$ that determines Pareto dominance.

4. A combining operator $\oplus : V \times V \rightarrow V$ that combines solution values into one value.
5. A constraint $c : V \times S \rightarrow 0, 1$ that determines the feasibility of a given solution v and state s.
6. An initial state $s_0 \in S$.

In both our optimization task and the ones presented in [16], the state s can be defined by $\langle D, F \rangle$, with F the branching decisions made in parent nodes and the D dataset containing the relevant data. The transition function is defined as $t(\langle D, F \rangle, f, 1) = \langle D_f, F \cup f \rangle$ and $t(\langle D, F \rangle, f, 0) = \langle D_{\overline{f}}, F \cup f \rangle$, ensuring that the data transfer to the next state adheres to the current branching feature. The initial state s_0 is defined as $\langle D, \emptyset \rangle$.

3.3 Separability

An optimization task is deemed separable when the optimal solutions to subtrees can be computed independently of other subtrees. For example, in the tree in Figure 1, the optimal label L1 should be independent of the branching decision F3 and labels L3 and L4. Feature F2 should be independent of F3.

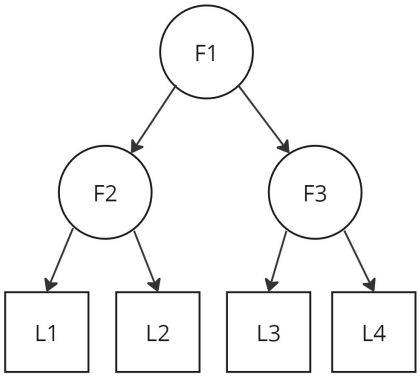

Figure 1: Example of a decision tree of depth two, with three branching nodes (F1, F2, F3), and four leaf nodes (L1, L2, L3, L4)

The formal definition of a "separable" task found in [16] is the following: "An optimization task is separable if and only if the optimal solution to any subtree can be determined independently of any of the decision variables that are not part of that subtree or the parent nodes' branching decisions".

For example, the optimization task of accuracy in a decision tree can be considered separable. We can calculate the misclassification cost in a leaf node by checking which data points end up in that leaf and counting the amount of misclassified data points. In a branching node, we can combine two solutions by adding their misclassification scores, representing the amount of misclassification in their subtrees. Furthermore, we can compare two solutions and argue about dominance by numerically comparing the two misclassification scores. Since none of these operations depends on information about other sub-trees, the optimization task of accuracy can be considered separable.

3.4 Streed: Dynamic Programming Framework

The general DP framework under the name "Streed" (Separable Trees with Dynamic Programming), introduced in [16] and used for calculating the cost of a decision tree given the state s and the depth limit d, is presented here.

$$T(s,d) = \text{opt}\left(\bigcup_{\hat{k} \in K} \{g(s,\hat{k})\}, s\right) \qquad \text{if } d = 0$$

$$T(s,d) = \text{opt}\,(\bigcup_{f \in F} \text{merge}\,(T(t(s,f,1), d-1), T(t(s,f,0), d-1), s, f\ ), s\ ) \qquad \text{if } d > 0$$

Let θ describe a set of possible solutions; we show the link between the notation of the framework and the notation introduced in the optimization task 3.2 by attaching the following formulas from [16]:

$$\text{feas}(\theta, \text{s}) = \{v \in \theta \,|\, c(v,s) = 1\}\,,$$

$$\text{nondom}(\theta) = \{v \in \theta \,|\, \nexists\, v' \in \theta (v' \succ v)\}\,,$$

$$\text{opt}(\theta, \text{s}) = \{nondom(feas(\theta,\ s))\}\,,$$

$$\text{merge}(\theta_1, \theta_2, \text{s}, \text{f}) = \{v1 \oplus v2 \oplus g(s,f) \,|\, v_1 \in \theta_1, v_2 \in \theta_2\}\,.$$

We explain the framework using the decision tree example in Figure 1. This model determines the cost of a decision tree ($T(s,d)$) based on the current depth limit d and the state s of the tree. The state s of the tree is determined by $\langle D, F \rangle$, with D the data set reaching current node and F the branching features of the parent nodes (e.g. F1 and F3 for L4). When we reach a leaf node ($d = 0$), e.g. L4, we calculate the cost $g(s,\hat{k})$ of assigning each possible label to the leaf. We add the cost of each to our set of solutions, and finally, we use our comparison operator $\succ$ to select the optimal solution(s).

While we are in a branching node ($d > 0$), for every possible feature, we recursively calculate the cost of the sub-trees $T(t(s,f), d-1), T(t(s,\overline{f}), d-1)$ that result from branching on the current feature. We merge the 2 solutions in one (using our combining operator $\oplus$) and we add it ($\cup$) to the set of solutions. We then pick the optimal (according to our comparison operator $\succ$) solution(s). Depending on whether or not the comparison operator can provide strict dominance for any pair of solutions, the Pareto Front calculated by the algorithm contains either a single or multiple optimal solutions.

The use of this framework has been proven [16] to find the Pareto front of optimal solutions for any optimization task that is separable.

4 Problem Definition

This research paper focuses on classification decision trees that minimize misclassifications in the training set while adhering to the principle of individual fairness.

We introduce a measurement for individual fairness, based on the definition proposed by Dwork et al. [7], as the *proportion of similar data pairs classified identically*. Hence, the individual fairness metric takes values in the [0, 1] range, and

high values reflect the equitable treatment of similar individuals. We determine the similarity between data points using the Hamming distance metric, which fulfils the conditions of non-negativity, symmetry, and reflexivity. Our similarity assessment is also based on the input space, much like other research on individual fairness [14]. We present:

$$\text{sim}(dp_1, dp_2) = \begin{cases} 1 & \text{if hamming distance} < 1 \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

$$\text{sc}(dp_1, dp_2) = \begin{cases} 1 & \text{if } dp_1.\hat{k} == dp_2.\hat{k} \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Equation 2 checks for the same classification of two data points. The individual fairness index is thus defined as follows:

$$IF = \frac{\sum_{i=1}^{|D|}\sum_{j=i+1}^{|D|}(sim(dp_i, dp_j) * \text{sc}(dp_1, dp_2))}{\sum_{i=1}^{|D|}\sum_{j=i+1}^{|D|}(sim(dp_i, dp_j))} \quad (IF_{index})$$

Given the "Streed" dynamic programming framework (3.4), IF_{index} and an $IF\_threshold$, the task is to define and implement a separable (3.3) optimization task $\langle g, t, \succ, \oplus, c, s_0\rangle$ (3.2) that globally minimizes misclassification score while having an individual fairness value above the $IF\_threshold$.

5 Main Contributions

Our approach uses Equation IF_{index}, and considers the individual fairness gained or lost within each subtree to create a separable strategy.

To calculate the individual fairness gained in a subtree of state $\langle D, F\rangle$, we divide the number of close pairs in D that are classified the same by the total number of close pairs.

$$IF_{gained}(D) = \frac{\sum_{i=1}^{|D|}\sum_{j=i+1}^{|D|}(sim(dp_i, dp_j) * \text{sc}(dp_1, dp_2))}{\sum_{i=1}^{|D|}\sum_{j=i+1}^{|D|}(\text{sim}(dp_i, dp_j))} \quad (IF_{gained})$$

On the other hand, to calculate the individual fairness lost in a subtree of state $\langle D, F\rangle$, we divide the number of close pairs in D that are classified differently by the total number of close pairs.

$$IF_{lost}(D) = \frac{\sum_{i=1}^{|D|}\sum_{j=i+1}^{|D|}(sim(dp_i,dp_j)*(1-\text{sc}(dp_1,dp_2)))}{\sum_{i=1}^{|D|}\sum_{j=i+1}^{|D|}(sim(dp_i,dp_j))} \quad (IF_{lost})$$

Remember that each tree is described by state $s : \langle D, F\rangle$, with D being the data set that reaches the tree's root. The *solution object* $v \in V$ of any (sub)tree in our task includes the following attributes:

- mc: number of miss-classifications in D
- lb: lower bound on individual fairness $\rightarrow IF_{gained}$
- ub: upper bound on individual fairness $\rightarrow$ (1 - IF_{lost})

To calculate the lower and upper bounds at branch nodes, it is efficient [1] to use the already calculated bounds of the solutions of the two sub-trees. To accurately estimate the current bounds, we need to enhance the bounds of the two sub-trees with information about IF_{index} that the two sub-trees were unaware of. This is captured by similar data points that are being merged in this branching node. Figure 1 is used to illustrate this concept. Note that L1 and L2 can calculate the IF_{lost} and IF_{gained} for every close pair present in the data of the leaf. We can get an almost accurate estimation by numerically adding the IF_{lost} and IF_{gained} of L1 and L2 for F2's bounds. However, the only information missing comes from close pairs, with one data instance in L1 and the other in L2. Therefore, to obtain precise bounds for a branching node, we need to enhance the result of the numerical addition of the sub-tree bounds, with IF_{lost} and IF_{gained} in the close pairs being merged in the current node [2]. However, to determine whether the identification of just merged close pairs is a gain or loss of individual fairness, we need to know their classification. We have added the following field to the solution object to address this.

- $labels$: contains the classification of data points in D close to those not present in the current sub-tree.

We now go through the six components $\langle g, t, \succ, \oplus, c, s_0\rangle$ of the optimization task (3.2) and define them all in a separable (3.3) way. The transition function and initial state components are simply adopted from [16] as introduced in 3.2.

Firstly, we present the *cost function* g with $g(\langle D, F\rangle, \hat{k}) \rightarrow V$ returning the cost of assigning label $\hat{k}$ in a leaf node. We denote $D_{\hat{k}}$ as the data set of the given state D, concatenated with the predicted label $\hat{k}$. The lower bound is computed as $v.lb = IF_{gained}(D_{\hat{k}})$. Individual fairness is never lost in a leaf node since any similar pair present in the leaf will never get a different label; hence the upper bound $v.ub$ is fixed at 1. The misclassification score $v.ms$ counts the number of data points in $D_{\hat{k}}$ whose label differs from $\hat{k}$. The $v.labels$ contains the classification of the data points that have been split with a close pair in a parent node.

Secondly, the *comparison operator* $sol_1 \succ sol_2$ determines Pareto dominance. It is important to note that we cannot guarantee that a solution will eventually satisfy the fairness threshold unless the lower bound is already above it. Therefore, we cannot determine dominance solely based on a higher missclassification score.

$$sol_1 \succ sol_2 \quad \text{if and only if:}$$
$$((sol_1.lb > sol_2.ub \text{ OR } sol_1.lb > \text{IF\_threshold})$$
$$\text{AND } sol_1.ms < sol_2.ms)\,.$$

Notably, this strict dominance criterion ensures that optimal solutions are never excluded from our search space, enabling the identification of the global optimal Pareto front of solutions.

[1] faster than computing IF_{gained} and IF_{lost} using the complete dataset D of the subtree, which comes with $O(|D|^2)$ time complexity.

[2] explicit notation for is provided in the explanation of combining operator $\oplus$.

Thirdly, the *combining operator* $\oplus$ takes two solutions, sol_1 and $sol_2 \in V$, which describe the cost of their respective sub-trees, and combines these costs into a single solution sol. The two misclassification scores are added as $sol.mc = sol_1.ms + sol_2.mc$. We define

$$D_m = sol_1.labels \cup sol_2.labels,$$

containing information about the classification of "split and close" data points of the two sub-trees. The lower bound of the new solution is calculated as

$$new\_sol.lb = sol\_1.lb + sol\_2.lb + IF_{gained}(D_m),$$

while the upper bound is given by

$$new\_sol.ub = 1-(1-sol\_1.ub)(1-sol\_2.ub)-IF_{lost}(D_m).$$

These equations update the bounds based on the number of close pairs merged in the current node. The $sol.labels$ subset of D_m only includes labels with close data points in other sub-trees.

As we move up the decision tree, we learn about the classification of more similar data point pairs. This leads to tighter bounds on the individual fairness value and brings us closer to its global value. At the top of the tree, we have complete knowledge of the classifications of similar data points, which allows us to calculate the exact value of individual fairness.

The current optimization task does not penalize branching on features, therefore the *cost function* g for assigning features $g(\langle D, F\rangle, f) \rightarrow V$ is initialized with the numb values (ms: 0, lb:0, ub: 1, labels: $\{\}$) that do not affect the result when combined ($\oplus$) with another solution.

Last but not least, the *constraint* c determines a solution $v \in V$ as feasible if and only if: $v.ub \geq IF\_threshold$, with $IF\_threshold$ being the minimum value the individual fairness metric value of the constructed decision tree is allowed to have. In contrast with the related research [14], we decided to include individual fairness as a hard constraint to have more control over the individual fairness value we want our model to have.

Overall, we defined and implemented individual fairness as a separable optimization task, and to make it efficient, we only incorporated the minimum necessary computations.

6 Experimental Setup and Results

In this section, the goals of the experiments, their set-up and results are presented.

6.1 Goals

Our goal for the experiments is to investigate the scalability of the algorithm. We found that the data set's number of instances, features, and close pairs are important factors for its runtime performance. To analyze their effects, we plan to vary the values of each parameter while keeping the other two fixed. This way, we can determine the impact of each factor on the algorithm's scalability. Notably, such information can shine a light on the effectiveness of the dynamic programming approach in constructing optimal decision trees, even for objectives that do not appear separable at first glance.

To the best of our knowledge, this is the only research on optimal decision trees that incorporate individual fairness. Therefore, we do not immediately compare our scalability results to an alternative approach. However, we use our knowledge of the scalability limits of common approaches to argue about the significance of our approach.

6.2 Set Up

We now present the setup of the experiments conducted to evaluate the scalability of the algorithm. The experiments aimed to examine the impact of three parameters: the number of *instances*, the number of *features*, and the number of *close pairs* in synthetic datasets, on the runtime performance of the algorithm. When testing for a specific parameter, the other two parameters were set to their default values, as follows:

- Default number of instances: 150.
- Default number of features: 20.
- Default number of close pairs: 50.

To facilitate these experiments, we generated random datasets with varying characteristics, specifically:

- Number of instances $\in$ [50, 100, 250, 500, 750, 1000, 1500, 2000, 2500].
- Number of binary features $\in$ [10, 20, 35, 50, 75, 100, 125].
- Number of close pairs $\in$ [0, 10, 20, 40, 60, 80, 100, 150, 200].

Please remember that using the Hamming distance as the similarity metric provides flexibility in combining these parameters. For example, constructing a dataset containing 2000 instances and only 50 close pairs is possible.

For this scalability study, synthetic datasets were used because they offer better flexibility and control over the data's characteristics. The default parameters can be found in real datasets while minimizing added workload. This approach allows us to analyze the scaling effect of our testing parameters.

All experiments were conducted on a single-threaded AMD Ryzen 7 4800H 2.90 GHz CPU, 16 GB of RAM, and a timeout limit of 900 seconds. We constructed five datasets for each data set configuration and ran the algorithm considering the depth limits: 1, 2, 3 and 4. For each configuration, we present both the average and the confidence interval.

6.3 Results

The performance of the algorithm was evaluated based on the runtime required to solve the decision tree problem for different parameter configurations. Runtime performance above 900 seconds is not present in the graphs since that corresponds to the timeout limit of our experiments.

Number of instances

By looking at our algorithm's performance across different instances (Figure 2), we notice an efficient, in the magnitude of milliseconds, performance in the depth limits of 1 and 2. It is observed that when the depth limit was set at 3, the optimal solutions were discovered within the time limit of 15 minutes for instances below 500. By looking at the curves of the

figure, we also verify the linear correlation between runtime performance and the number of instances.

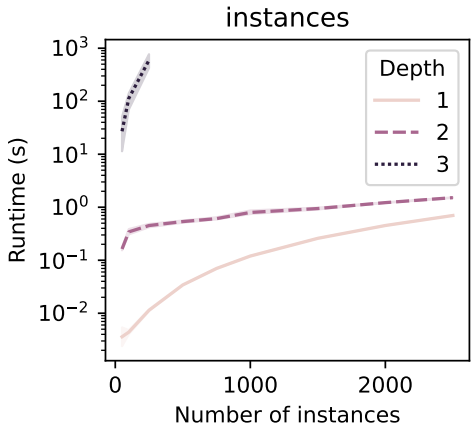

Figure 2: Runtime performance of the optimal decision tree solver for different number of data instances across different depth limits

Number of features

In addition, we analyzed the runtime performance of our algorithm for various feature values. As shown in Figure 3, the algorithm for depth limit of 1 and 2 performed efficiently, completing in the magnitude of seconds, when provided with data sets of any amount of binary features in the given range. We also found that a depth limit of 3 is reachable for many features, as the algorithm found the optimal decision tree within the time-out limit for up to 75 binary features. Again, we verify the positive correlation between the number of features and the runtime performance of our algorithm.

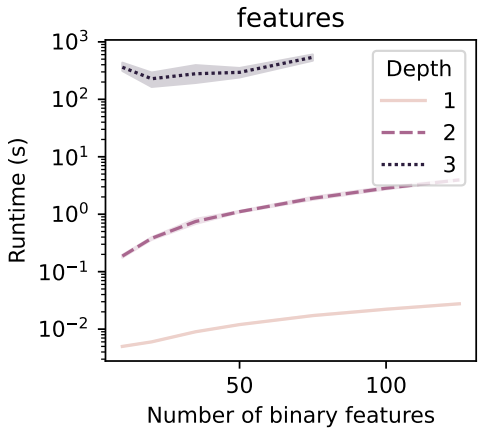

Figure 3: Runtime performance of the optimal decision tree solver for different number of binary features in the data across different depth limits

Number of close pairs

We gained valuable insights by analyzing the runtime performance of our algorithm with varying numbers of close pairs in the dataset (Figure 4). Our algorithm performs efficiently in milliseconds for depth limits up to 2, regardless of the number of close pairs. For a depth limit of 3, our algorithm can construct the optimal decision tree for up to 65 close pairs. Notably, we achieved a depth limit of 4 within the time-out limit for close pair values of 0 and 10 for the first time. We attribute this success to the efficient implementation of the "split and close" labels, which allows the algorithm to combine solutions without iterating through all data points, but only through the limited amount of data points with a close data point in another sub-tree. Therefore, this performance enhancement greatly benefits sparse, in terms of similarity, data sets.

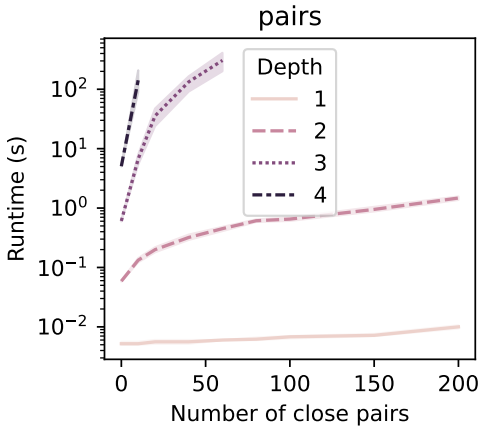

Figure 4: Runtime performance of the optimal decision tree solver for different number of similar pairs in the data across different depth limits

Remarks

To the best of our knowledge, this is the only research that has developed decision trees that are both optimal and individually fair, so we cannot make a direct comparison to alternative approaches. However, it is observed [16] that general-purpose solvers, such as MIP, have difficulty scaling when faced with datasets containing more than a few thousand instances or depth limits beyond 3. On the other hand, our algorithm successfully found optimal solutions for depth limits of 3 and even 4 in some cases, within the not-so-long time-out of 15 minutes. Therefore, our dynamic programming approach offers a promising and competent performance against possible alternative approaches.

7 Responsible Research

The experiments conducted in the previous section adhere to the FAIR Guiding principles for scientific data management, as outlined in Wilkinson et al. [19]. These principles aim to enhance the Findability, Accessibility, Interoperability, and Reusability of digital assets. We address each of these principles below to demonstrate our compliance with them.

- Findable: In order to make our research data easily searchable, we have made the synthetic datasets acces-

sible on the same repository as the code. This will allow for the replication of the experiments presented in this paper, as researchers will be able to obtain and utilize the datasets.

- Accessible: Once the code repository proposed by Linden et al. [16] becomes public, everyone will be able to access both the code and the synthetic datasets without authentication.
- Interoperable: The data is stored in a common CSV format, with descriptive file names, making it interoperable.
- Reusable: Each data file is also structured in the same way, making it reusable. Each row represents one data point, and starts with the actual label of a data point followed by binary features.

Our goal is to improve the usability of the data collected for our research by following these four standards. By doing so, we can facilitate further experiments in the research field and enhance our understanding of performance related to optimal decision trees.

Notably, the algorithm is implemented to incorporate different distance metrics for defining similarity between individuals. Upon applying this algorithm to real-world cases, we suggest getting advice from an expert about the appropriate distance metric and values to incorporate into the application.

8 Conclusions and Future Work

In summary, we have successfully defined *individual fairness* as a separable optimization task that globally minimizes miss-classification scores while having an individual fairness value above a certain threshold. We implement the algorithm that constructs optimal decision trees for this optimization task through the "Streed" framework. Our scalability analysis has shown promising results, with optimal decision trees constructed up to a depth limit of 3 and, in some cases, 4.

We recommend exploring alternative approaches for individual fairness in optimal decision trees, as their significance seems unnoticed. We also suggest comparing the scalability and performance of our method to any upcoming alternatives. We hope this research can contribute valuable insights to the pursuit of fair and optimal decision trees.

References

[1] Sina Aghaei, Mohammad Javad Azizi, and Phebe Vayanos. Learning optimal and fair decision trees for non-discriminative decision-making. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1418–1426, 2019.

[2] Solon Barocas and Andrew D Selbst. Big data's disparate impact. *California law review*, pages 671–732, 2016.

[3] Dimitris Bertsimas, Jack Dunn, Emma Gibson, and Agni Orfanoudaki. Optimal survival trees. *Machine Learning*, 111(8):2951–3023, 2022.

[4] Reuben Binns. On the apparent conflict between individual and group fairness, 2019.

[5] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Taylor & Francis, 1984.

[6] Emir Demirović, Anna Lukina, Emmanuel Hebrard, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Peter J Stuckey. Murtree: optimal classification trees via dynamic programming and search. *arXiv preprint arXiv:2007.12652*, 2020.

[7] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 214–226, New York, NY, USA, 2012.

[8] Equality and Human Rights Commission. Protected characteristics, 2021.

[9] Vincent Grari, Boris Ruf, Sylvain Lamprier, and Marcin Detyniecki. Achieving fairness with decision trees: An adversarial approach. *Data Science and Engineering*, 5, 06 2020.

[10] Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is np-complete. *Information processing letters*, 5(1):15–17, 1976.

[11] Mikoláš Janota and António Morgado. Sat-based encodings for optimal decision trees with explicit paths. In *Theory and Applications of Satisfiability Testing – SAT 2020: 23rd International Conference, Alghero, Italy, July 3–10, 2020, Proceedings*, page 501–518, Berlin, Heidelberg, 2020. Springer-Verlag.

[12] Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and scalable optimal sparse decision trees. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6150–6160. PMLR, 13–18 Jul 2020.

[13] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Comput. Surv.*, 54(6), jul 2021.

[14] Francesco Ranzato, Caterina Urban, and Marco Zanella. Fairness-aware training of decision trees by abstract interpretation. In *Proceedings of the 30th ACM International Conference on Information Knowledge Management*, CIKM '21, page 1508–1517, New York, NY, USA, 2021. Association for Computing Machinery.

[15] Steven L. Salzberg. C4.5: Programs for machine learning by j. ross quinlan. *Machine Learning*, 16(3), 1994.

[16] Jacobus G. M. van der Linden, Mathijs M. de Weerdt, and Emir Demirović. Optimal decision trees for separable objectives: Pushing the limits of dynamic programming, 2023.

[17] Hélène Verhaeghe, Siegfried Nijssen, Gilles Pesant, Claude-Guy Quimper, and Pierre Schaus. Learning optimal decision trees using constraint programming (extended abstract). In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4765–4769. International Joint Conferences on Artificial Intelligence Organization, 7 2020. Sister Conferences Best Papers.

[18] Sicco Verwer and Yingqian Zhang. Learning optimal classification trees using a binary linear program formulation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):1625–1632, Jul. 2019.

[19] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3(1):1–9, 2016.

[20] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P. Gummadi. Fairness beyond disparate treatment &amp disparate impact. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, apr 2017.