Optimal Recourse Summaries via Bi-Objective Decision Tree Learning

Ioannis Chatzis
johnco.chatzis@gmail.com

Jason Liartis
jliartis@ails.ece.ntua.gr

Athanasios Voulodimos
thanosv@mail.ntua.gr

Giorgos Stamou
gstam@cs.ntua.gr

Artificial Intelligence and Learning Systems Laboratory
National Technical University of Athens, Greece

Abstract

Actionable Recourse provides individuals with actions they can take to change an unfavorable classifier outcome. While useful at the instance level, it is ill-suited for global auditing and bias detection, since aggregating local actions is costly and often inconsistent. Recourse Summaries address this limitation by partitioning the population and assigning one shared action per subgroup, enabling comparison across subgroups. Designing summaries involves a fundamental trade-off between recourse effectiveness and recourse cost, which existing methods do not adequately address. We introduce Summaries of Optimal and Global Actionable Recourse (SOGAR), which formulates recourse summary learning as an optimal decision tree learning problem and finds the Pareto front – the complete set of solutions where improving one objective necessarily worsens the other. SOGAR enables post-hoc selection of the desired trade-off without retraining. Using shallow axis-parallel decision trees and sparse leaf actions, SOGAR produces stable, low-cost, and effective recourse summaries that outperform existing approaches across effectiveness and cost metrics.

1 Introduction

Automated decision-making systems based on opaque machine learning models are increasingly being applied in critical domains such as credit scoring, hiring, education, and healthcare decisions. In such high-risk settings, high predictive accuracy is insufficient for stakeholders, as the inspection of the system's decisions is not only useful but often a legal requirement as well, e.g., by Article 86 of the EU AI Act.[1] To address this, there are efforts both on the fronts of inherent interpretability [29] and post-hoc interpretability [28, 21].

A very popular post-hoc method is *counterfactual explanations* which specify the changes to the input features that would produce a different classifier output [33]. Counterfactual explanations are very interpretable and often preferred by humans as they mimic their thinking process [11]. Ustun et al. reframe the problem around an individual who needs to take actions to change the unfavorable decision of an automated system, terming this setting *actionable recourse* [31].

Global counterfactual explanations and *recourse summaries* extend these explanations to provide insight for an entire population. Global counterfactual explanations try to identify a small set of

[1] https://artificialintelligenceact.eu/article/86/

Preprint.

low-cost actions that effectively change the classifier's output with respect to a reference population [19, 17]. Recourse summaries further identify population subgroups and assign one action to each subgroup, which is especially helpful for identifying biases by comparing the optimal action for each subgroup [27, 15]. Both approaches try to balance a set of objectives that often include the following: **(i)** Coverage, the portion of the population that is assigned with actions for recourse, **(ii)** Recourse cost, the estimated effort of enacting the assigned recourse, **(iii)** Recourse loss, the portion of the population whose recourse successfully flips the classifier's output. Also referred to as ineffectiveness or incorrectness, **(iv)** Action sparsity, the number of input features affected by the assigned recourse, **(v)** Subgroup count, **(vi)** Subgroup description length, how many features are used to specify each subgroup. Not all of these objectives are sufficiently addressed by existing works. Some provide limited coverage [27], some prioritize loss over cost [17], others scalarize the objectives with arbitrary coefficients [27, 15], and none solve their provided formulation to a *global optimum*.

To address this, we introduce *Summaries of Optimal and Global Actionable Recourse (SOGAR)*, which formulates recourse summary learning as a bi-objective optimal decision tree problem, the objectives being the recourse cost and loss. Each summary is represented as a shallow, axis-parallel decision tree that partitions the affected population, with one action assigned per leaf. We produce a small number of groups with simple descriptions and sparse actions by imposing thresholds on the number of leaves, tree depth, and action sparsity. Under the separable conditions of the STreeD framework [32], this yields solutions that guarantee global optimality over the specified objectives and constraints. Rather than committing to one single scalarization of cost and effectiveness, SOGAR produces the full Pareto front of non-dominated summaries for a fixed maximum depth and number of leaves, exposing the complete trade-off between the objectives. Since finding optimal decision trees is NP-hard [14], it also offers early termination, returning the best solutions found within the allocated time budget.

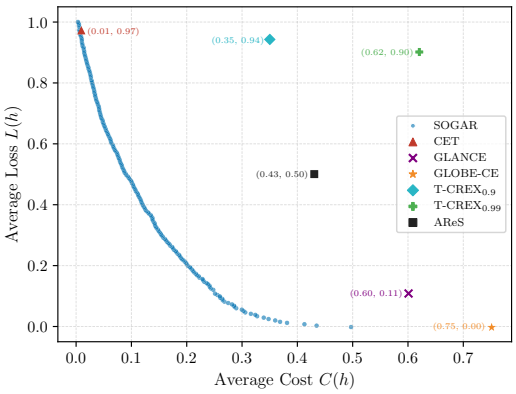


Figure 1: SOGAR vs. baselines on Employee Attrition, LightGBM classifier, fold 1 of the 10-fold cross validation. Closest to $(0, 0)$ is better. Blue dots show the full SOGAR trade-off; other methods produce a single solution each.

Our contributions as summarized are the following:

- We define SOGAR as a bi-objective optimal decision tree learning problem for actionable recourse summaries that jointly optimizes recourse effectiveness and action cost under tree size and action sparsity constraints.
- We cast SOGAR as an optimization task within the dynamic programming framework STreeD, prove it is exactly solvable, compute the full non-dominated Pareto front, and further accelerate the leaf-action evaluation by introducing CPU and GPU parallelization.
- We evaluate SOGAR against recourse-summary and global counterfactual explanation baselines on benchmark tabular datasets and show that the exact optimization recovers additional non-dominated summaries, thereby improving the cost-effectiveness trade-off.
- We validate SOGAR's auditing capabilities by recovering the gender-based bias of the Adult dataset [3] in the Pareto front.

The remainder of the paper is organized as follows. In Section 2 we present the related literature. In Section 3 we provide the preliminaries and formalize recourse summaries and the SOGAR objective. In Section 4 we present the end-to-end formulation of our framework. Finally, in Section 5 we compare our method against 5 other baselines, across 4 datasets of different sizes and discuss the results.

2 Related Work

Counterfactual Explanations and Algorithmic Recourse. Counterfactual Explanations (CE) and Algorithmic Recourse (AR) are two similar explainability techniques, that given a classifier and an input instance, find an alternative instance that changes the classifier's output. CEs minimize a notion of distance between the original and the alternative instances [33, 9]. AR, being more human-centric, frames the problem as providing individuals who have been treated unfavorably by an automated decision-making system with actions they can take to alter the decision, and minimizes the burden of enacting the recourse [31, 16]. AR typically imposes constraints on the proposed actions, such as actionability, feasibility, proximity and sparsity of edits.

Global Counterfactual Explanations and Recourse Summaries. Global Counterfactual Explanations and Recourse Summaries are directly situated in the global setting, identifying a small set of edits or actions that minimize average recourse cost and loss — the frequency with which the assigned recourse fails to flip the classifier's label. The first work along those lines was AReS, which uses itemset mining to find two-level population subgroups and assigns one action to each subgroup while optimizing for recourse coverage, cost, loss, and interpretability [27]. AReS has two major limitations: high running times and inconsistency — some individuals may be assigned multiple actions while others receive none. T-CREx adopts the tree structure to assign counterfactuals to individuals, but prioritizes solution sparsity over effectiveness and does not impose actionability constraints [6]. GLOBE-CE takes a different approach: rather than finding specific actions that flip the classifier's output, it identifies fixed directions that can be added to inputs with variable magnitude [19]. GLANCE begins by clustering the population and generating a set of candidate actions for each cluster [17]. It then merges clusters that are similar in feature or action space and selects the optimal action for each, prioritizing coverage, yielding a small final set of counterfactual actions. CET is the work closest to ours [15]. It partitions the population into subgroups using a decision tree structure, assigning one action to each leaf. This ensures that an individual is assigned exactly one action. CET uses stochastic local search to find good tree splits and minimizes a linear combination of cost and ineffectiveness, configured by a hyperparameter γ, to assign actions to leaves. Like AReS, CET has been criticized for excessive running times, which we confirm in our experiments in Section 5.

SOGAR, our method, shares CET's tree-based format but differs in key respects. Instead of stochastic local search, we find globally optimal solutions using dynamic programming. Additionally, we do not linearize the objective; instead, we identify all Pareto-optimal solutions that improve either cost or ineffectiveness, allowing users to select the solution that best satisfies their needs. This approach also eliminates the need to recompute solutions for different values of γ, if the initial solution proves unsatisfactory. Although finding global optima is NP-hard, the STreeD framework [32] proves much faster in practice through efficient caching and pruning, which we enhance with additional action-cost caches and a GPU implementation.

Optimal Decision Trees. Decision Tree learning is NP-hard [14] and is most often approached with a greedy algorithm such as CART [7] or C4.5 [26]. Modern advancements in hardware and optimization algorithms have made finding the global optimum computationally feasible. The first successful method utilized itemset lattices [25], which was further developed [2] and other approaches emerged using Mathematical Programming [4, 1], Constraint Programming [24], Branch and Bound search [13, 22] and Dynamic Programming [10, 32]. Most works are concerned with optimizing accuracy, but others allow optimization of more complex metrics such as F1-score [20] or other tasks altogether such as survival analysis [35] or clustering [5]. The STreeD framework in particular, introduced in [32], presents a set of necessary and sufficient conditions, which, when satisfied, allow for any task to be solved by their algorithm, which utilizes dynamic programming with caching and pruning. We prove that finding recourse summaries is such a task, and we integrate it into the STreeD framework. To our knowledge, SOGAR is the first work in the field of optimal decision trees to tackle actionable recourse or counterfactual explanations.

3 Problem Setting

Here we introduce our notation and formulation of recourse summaries.

Throughout this paper, we assume that we are working in a classification setting, where a fixed classifier f assigns labels from $\{0, 1\}$ to instances drawn from a set $\mathcal{X}$. We assume that 1 is the positive, desired label and 0 is the negative, undesired label. In multiclass settings we can simply designate which labels are desired and map those to 1 and the rest to 0. We also assume that instances are n-dimensional vectors of any combination of real, categorical and binary features. We denote the j-th feature of an instance by x_j. We also assume that we have an action set $\mathcal{A}$ whose elements a transform an instance x to a new instance $a(x) \in \mathcal{X}$. For example, f could be a loan approval model, with "approved" being the desired class and "rejected" being the undesired class. $\mathcal{X}$ could consist of features such as the applicant's age, income and requested loan amount, while an action could be "Increase your income by \$5,000 and reduce the requested loan amount by \$20,000". Finally, for computational purposes, we assume that we have a binarization function $\phi : \mathcal{X} \to \{0, 1\}^{|\mathcal{F}|}$ where $\mathcal{F}$ is a set of binary features derived from thresholds on numerical features and encoding on categorical features.

Individualized Recourse. Let c denote a function $c : \langle \mathcal{X}, \mathcal{A} \rangle \to \mathbb{R}_{\geq 0}$ that assigns a cost to an action a for an instance x. Individualized recourse refers to finding the minimal cost action that maps an instance x to a positively classified instance, i.e.,

$$\arg\min_{a \in \mathcal{A}} c(a, x), \ s.t. \ f(a(x)) = 1 \tag{1}$$

Recourse Summaries. Given a population of instances along with the labels assigned to them by the classifier $D = \{\langle x^{(i)}, f(x^{(i)}) \rangle\}_{i=0}^{N}$, the affected instances is the subset of instances assigned to the undesired class, $D_0 = \{x^{(i)} : \langle x^{(i)}, f(x^{(i)}) \rangle \in D, \ f(x^{(i)}) = 0\}$. Assigning one action to the entire affected population, could lead to arbitrarily high costs when D_0 contains points far from the decision boundaries, as shown in Proposition 1 of [15].

In order to keep recourse costs low, while assigning only a few actions to the entire population, *recourse summaries*, as introduced in [27], define population subgroups $X_k \subseteq \mathcal{X}, k = 1, \ldots, K$ and assign one action to each subgroup. The objective is additionally relaxed by allowing an action to be ineffective for some instances of the affected population, but we measure the count of those instances as its *recourse loss*:

$$l(a, x) = \mathbb{I}[f(a(x)) = 0] \tag{2}$$

$$l(a, X_k) = \sum_{x \in X_k} l(a, x) \tag{3}$$

Where $\mathbb{I}$ is the indicator function.

Using the subgroups X_k and the actions assigned to them a_k we can treat the recourse summary as a mapping h from instances to actions, such that $h(x) = a_k$ if $x \in X_k$. Therefore, we can define the cost and the loss functions at the level of the recourse summary by aggregating over D_0:

$$C(h) = \sum_{x \in D_0} c(h(x), x) \tag{4}$$

$$L(h) = \sum_{x \in D_0} l(h(x), x) \tag{5}$$

Increasing the number of subgroups can only improve C and L, as is shown in Proposition 2 of [15]. Despite this, we need the number of subgroups to be relatively small, otherwise it becomes hard to draw global insights.

For a fixed set of subgroups, there is a fundamental trade-off between the cost and the loss of the actions assigned to each subgroup. To decrease the loss, we need to reduce the number of instances for which the assigned action is ineffective, thus the pool of available actions becomes smaller and the cost of the chosen action can only increase or stay the same. As discussed in Section 2, prior work either solves for a linear combination of the objectives or enforces a threshold on one of them and optimizes the other and do not provide any optimality guarantees for their solutions. SOGAR instead computes the full Pareto front of globally optimal solutions. This avoids arbitrary thresholds and scalarization coefficients on the objectives and allows the end-user to select their ideal trade-off between cost and recourse loss. This also avoids the issue of having to re-run the process

for computing recourse summaries multiple times if the initial selection of objective coefficients and thresholds results in unsatisfactory solutions. SOGAR is the first method to provide this guarantee for recourse summaries.

Recourse Summary Trees. A *recourse summary tree* is a decision tree that defines a mapping $\tau : \mathcal{X} \to \mathcal{A}$. Each internal node v performs a test of the form $[x_{j(v)} \leq b_v]$. If an instance passes the test, it proceeds recursively to the right child of v and to the left child if it fails. These splits induce a partition of $\mathcal{X}$ into a finite collection of disjoint subgroups $\{X_k\}_{k \in \mathcal{L}}$, where $\mathcal{L}$ indexes the leaf nodes, and thus τ serves as a recourse summary. Each leaf is labeled with an action $a_k \in \mathcal{A}$ and the induced decision rule is $\tau(x) = a_k$ if $x \in X_k$. Trees with simple tests on features result in very simple subgroup definitions and by controlling their maximum leaf count and maximum depth we can control the subgroup count and their description length.

Given the pair of minimization objectives (C, L), a solution τ is said to strictly dominate τ' if $C(\tau) \leq C(\tau')$, $L(\tau) \leq L(\tau')$ and at least one of the inequalities is strict, denoted $\tau \succ \tau'$. Given a set of candidate recourse summary trees $\mathcal{T}$, such as all trees with a maximum depth of 3, the Pareto front P of $\mathcal{T}$ is the set of non-dominated solutions:

$$P = \{\tau \in \mathcal{T} \mid \nexists \tau' \in \mathcal{T} \; (\tau' \succ \tau)\} \tag{6}$$

Each point on the Pareto front represents a summary where improving cost would require sacrificing effectiveness, and vice versa.

Proposition 1 (Optimality)**.** *For any maximum depth d, the complete Pareto front of recourse summary trees, all trees that are not dominated by another tree of maximum depth d in both cost C and recourse loss L, can be computed exactly via dynamic programming.*

The proof (Appendix A) shows that recourse summary tree optimization is a *separable* optimization task in the sense of [32], enabling the use of their dynamic programming framework, named STreeD, with depth and leaf count constraints for interpretability, and caching and pruning for efficient computation. STreeD uses the binarized features $\mathcal{F}$ for defining splits, but for accurately computing the objectives, we use the original features $\mathcal{X}$. Thus, our method operates on the binarized dataset $D_B = \{\langle \phi(x), x \rangle : x \in D_0\}$ that contains pairs of (binarized, original) instances.

4 Proposed Method: SOGAR

In this section, we present our method, SOGAR, as an end-to-end pipeline, under the formalization of Section 3 which is visualized in Figure 2.

Action Set. SOGAR defines a single-edit action as one of the following operations. Binary variables: Flipping the value. Categorical variables: Changing from one category to another. Numerical variables: Increasing or decreasing the value by n bins. Single-edit actions can be pooled together to form k-edit actions. The action set $\mathcal{A}$ contains all actions of up to k edits, with the default value of k being 3. The user can also impose actionability constraints by specifying features for which no actions will be generated.

Cost & Loss functions. As cost the function c, SOGAR uses the *Maximum Percentile Shift (MPS)* function, similar to [15, 31]:

$$c_{MPS}(a, x) = \max_j |Q_j(a(x)_j) - Q_j(x_j)|, \tag{7}$$

where Q_j denotes the cumulative distribution function (CDF) of feature j. The cost function was chosen over a simple norm-based distance function because it is scale-invariant [15] and yields a realistic depiction of the effort required to enact the action by quantifying the difficulty of moving within a certain distribution. Recourse failure is captured by the loss function defined in Equation (2).

Cache computation. The action set $\mathcal{A}$ is used to create the cache, where the cost and loss are precomputed for every action applied on every affected instance. Directly evaluating c_{MPS} and the classifier output of every transformed input $a(x)$ inside tree search is computationally prohibitive because of the combinatorial nature of the problem, where the same instance-action pairs are reused across many candidate leaves. Thus, the $O(1)$ look-ups of cached cost-loss pairs preserve efficiency.

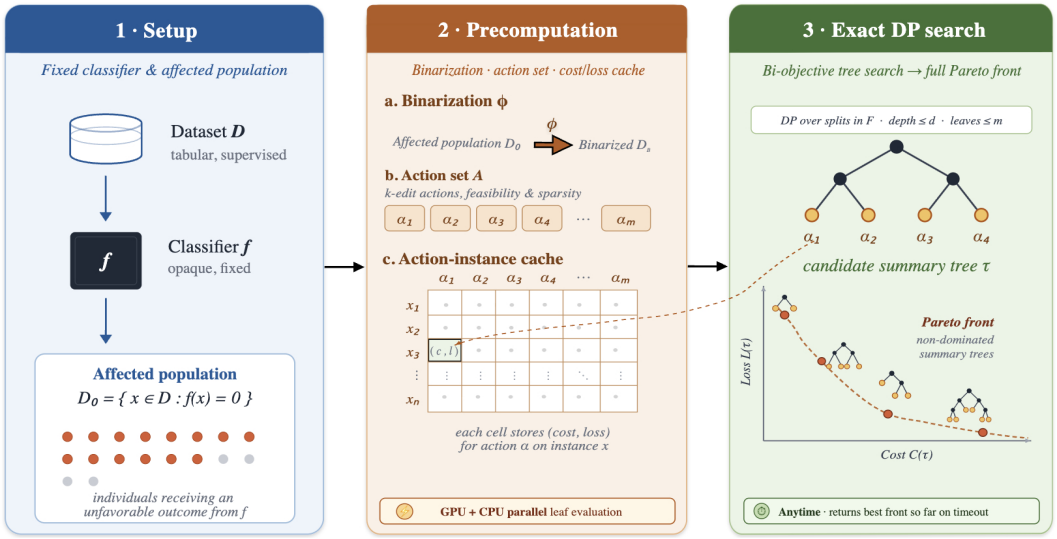


Figure 2: The figure depicts the end-to-end pipeline of SOGAR. From left to right, the process starts by fixing a classifier f, and retrieving the affected population that requires recourse, and immediately proceeding to the binarization process through ϕ. Then, the set of feasible shared actions $\mathcal{A}$ is created across the entire population, which is used to compute the cache of cost and loss, for every action applied on every instance. Finally, the optimization task is solved using the dynamic programming framework of STreeD and the complete Pareto front of solutions that improve either loss or cost is computed.

Binarization. We binarize features to obtain the feature set $\mathcal{F}$, which is required by the STreeD framework to search over all possible variable splits. Categorical features are one-hot encoded and continuous features or integer features are discretized using equal-width bins. After the tree search algorithm terminates, the binary splits can be back-translated into simple, interpretable equality and threshold tests on the original variables.

Optimization. Using the cached costs and losses, we can formulate recourse summary learning as an optimization task in the STreeD framework. The objective function of the optimization task is the cost-loss pair $C(\tau), L(\tau)$ and the dominance operator $\succ$ is element-wise comparison as outlined in Section 3. We also use constraints on the maximum tree depth d, and maximum leaf count m, to keep the summaries small and interpretable. A detailed formulation of the task is also given in Appendix A. STreeD proceeds to solve this optimization problem, returning the full Pareto front of non-dominated solutions, as defined in Equation (6). For a given maximum depth d, STreeD performs bottom-up dynamic programming, recursing on d. It utilizes caching of partial solutions and strong bounds to prune the search space, dramatically reducing the number of subproblems evaluated. We further improve the scalability of the algorithm by implementing multi-thread CPU and GPU acceleration as outlined in the following sections. Even though the nature of the task is combinatorial, the caching and pruning of STreeD, along with our cost-loss cache and parallelization, allows the problem to be solved to global optimality within a reasonable time interval. Finally, a timeout can be imposed, after which STreeD returns all non dominated solutions it has found thus far, without guaranteeing that the are contained in the true Pareto front. Our experiments in Appendix C show that, even with a limited computational budget, SOGAR can recover near-optimal solutions.

Leaf Acceleration. Every time a subproblem is solved, for each leaf, STreeD needs to evaluate each action on the individuals reaching that leaf. Aggregating the cost and loss of these individuals for some action is an embarassingly parallelizable task. We speed up the solver by implementing multi-thread CPU and GPU computation of these aggregates, significantly reducing running times. We provide a comparison of running times with and without GPU acceleration in Appendix E.

5 Experiments

In this section, we present our setup and introduce our comparison metrics. We present experiments across four tabular datasets and compare the performance of the baselines with our method using evaluation scores and runtime efficiency. We also perform an experiment showcasing how SOGAR can be used to audit classifiers for bias. Finally, we present our observations and analyze the Pareto fronts generated, discussing the generalization and usefulness of Pareto set solutions to auditors.

5.1 Experimental Setup

Experiments presented in this section were conducted on a machine with i9-14900KS CPU (24 cores), 128GB RAM, and RTX 4090 GPU (24GB VRAM). The SOGAR optimization task is implemented as a STreeD task in C++17. The remainder of the implementation, including preprocessing, binarization, and cache computation, was all implemented in Python 3.13. Total compute time was 150 h in main experiments (13.6 h SOGAR, 136.5 h rest baselines), and an additional 67.1 h in ablation studies in Appendices C–D.

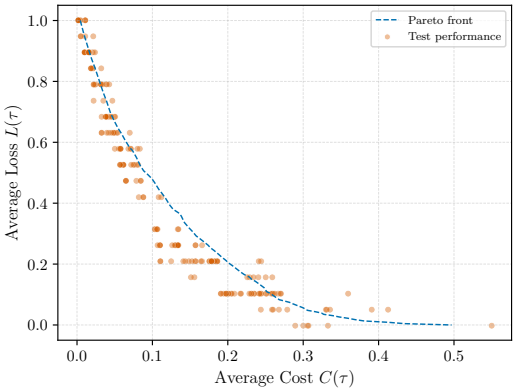


Figure 3: Pareto front on Employee Attrition (LightGBM, depth 3). Blue dashed line: train Pareto front. Orange dots: held-out test metrics. Average Euclidean distance: 0.04 ± 0.02 .

Datasets. We examine four tabular datasets, appropriate for auditing the classifiers. These are the Employee Attrition[2] [30], the German Credit[3] [12], the Bank Marketing[3] [23], and the Adult Income[3] [3] datasets. The Attrition and German Credit datasets are small to medium in size, with 1,400 and 1,000 instances, respectively, while the Bank Marketing[4] and the Adult datasets are large in size with 4,500 and 50,000 instances, respectively. For our method and the ones viable to control the actionability of features, we constrain sensitive features as immutable, such as Gender, Age, or Race of individuals.

Classifiers. The predictive models we used for auditing across all our experiments are two state-of-the-art tree-ensemble classifiers, LightGBM [18] and XGBoost [8], and a simple Deep Neural Network classifier. The parameters were set to default values for every model and dataset, as the objective was model auditing, not classification accuracy. We further discuss the models chosen and the parameters set for each architecture in Appendix B.

Baselines. We compare to 5 prior baselines, these of CET [15], AReS [27] , GLOBE-CE [19], GLANCE [17] and T-CREx [6]. For each baseline, we run the experiments using the default optimal parameters set by the authors, as indicated in each experiment.

Evaluation Metrics. To demonstrate the direct comparison with baseline methods, we report the recourse cost based on the MPS function defined in Equation (7), and evaluate the effectiveness of recourse based on the recourse loss defined in Equation (2). Finally, we adopt the sum of the two metrics as the invalidity score used by [15][5]. The invalidity metric, which equally weighs cost and loss, serves as a tie-breaker for solutions which do not Pareto-dominate one another.

[2]Database Contents License (DbCL) v1.0

[3] CC BY 4.0

[4]'additional' variant obtained by https://archive.ics.uci.edu/dataset/222/bank+marketing

[5]The invalidity score defined by [15] is a scaled sum of cost and loss using a γ coefficient on the loss function. In their experiments, they explicitly set $\gamma = 1$, thus, we do the same in the reported metrics.

Table 1: Evaluating SOGAR against competing methods on 4 datasets and 3 classifiers with metrics averaged over 10 folds. Metrics reported are recourse cost (MPS), recourse loss, invalidity (their sum), and computational time in seconds. Lower is better for all metrics. Out of all solutions produced by SOGAR, metrics are reported for the one with the lowest invalidity. A dash indicates that the algorithm runtime exceeded the 1 hour timeout, and no metrics were collected.

Models	**Algorithms**	**Attrition**				**German**				**Bank**				**Adult**			
		cost ↓	*loss* ↓	*inv.* ↓	*time(s)* ↓	*cost* ↓	*loss* ↓	*inv.* ↓	*time(s)* ↓	*cost* ↓	*loss* ↓	*inv.* ↓	*time(s)* ↓	*cost* ↓	*loss* ↓	*inv.* ↓	*time(s)* ↓
DNN	CET	0.30	0.10	0.40	198.7	0.04	0.44	0.48	255.3	0.27	0.60	0.86	378.9	–	–	–	–
	AReS	0.23	0.23	0.46	139.3	0.17	0.39	0.56	1,845.3	–	–	–	–	–	–	–	–
	GLOBE-CE	0.81	<0.01	0.81	7.6	0.91	0.01	0.92	7.58	0.87	0.04	0.89	23.2	0.98	0.00	0.98	170.5
	GLANCE	0.54	<0.01	0.54	11.66	0.47	<0.01	0.47	8.5	0.65	<0.01	0.66	10.75	0.96	0.00	0.96	47.22
	T-CREx$_{0.9}$	0.35	0.77	1.13	7.0	0.53	0.78	1.31	32.19	–	–	–	–	0.66	0.91	1.58	15.04
	T-CREx$_{0.99}$	0.56	0.65	1.21	176.32	0.60	0.70	1.25	19.63	–	–	–	–	0.64	0.86	1.50	10.4
	SOGAR	**0.05**	**<0.01**	**0.05**	250.0	**0.02**	**0.00**	**0.02**	28.2	**0.17**	0.29	**0.46**	268.9	**0.03**	0.03	**0.06**	1,248.4
XGBoost	CET	0.40	0.17	0.57	342.6	0.12	0.25	0.37	151.2	0.51	0.24	0.75	438.8	–	–	–	–
	AReS	**0.02**	0.97	0.99	11.68	0.41	0.13	0.55	2,362.8	–	–	–	–	–	–	–	–
	GLOBE-CE	0.75	**0.00**	0.75	**5.90**	0.92	0.25	1.16	**5.5**	0.91	**0.01**	0.92	30.91	0.98	**0.00**	0.98	205.2
	GLANCE	0.56	0.02	0.58	11.70	0.50	**<0.01**	0.50	8.4	0.69	0.03	0.72	10.6	0.96	<0.01	0.96	35.1
	T-CREx$_{0.9}$	0.33	0.72	1.10	7.99	0.40	0.75	1.20	8.2	0.32	0.99	1.31	6.87	0.70	0.90	1.46	12.52
	T-CREx$_{0.99}$	0.48	0.64	1.12	14.13	0.56	0.57	1.13	8.03	–	–	–	–	0.35	0.99	1.13	10.23
	SOGAR	0.23	0.02	**0.26**	254.1	**0.09**	0.02	**0.12**	26.4	**0.21**	0.26	**0.47**	250.7	**0.30**	0.34	**0.64**	763.0
LightGBM	CET	**0.26**	0.58	0.84	286.7	0.13	0.31	0.44	173.8	0.60	0.01	0.61	383.9	–	–	–	–
	AReS	0.44	0.44	0.88	307.4	0.19	0.65	0.84	680.4	0.36	0.80	1.15	213.7	–	–	–	–
	GLOBE-CE	0.79	**0.01**	0.80	**7.96**	0.91	0.23	1.14	**6.1**	0.92	**<0.01**	0.92	29.8	0.97	**0.00**	0.97	171.6
	GLANCE	0.62	0.08	0.70	12.17	0.52	**<0.01**	0.53	9.3	0.71	**<0.01**	0.71	**11.0**	0.93	<0.01	0.93	**32.0**
	T-CREx$_{0.9}$	0.37	0.80	1.16	7.08	0.44	0.72	1.16	8.25	0.38	0.66	1.04	6.83	0.28	0.96	1.24	10.88
	T-CREx$_{0.99}$	0.58	0.84	1.42	112.67	0.54	0.61	1.15	10.0	0.47	0.58	1.04	8.71	0.45	0.99	1.44	10.5
	SOGAR	0.30	0.07	**0.37**	356.4	**0.11**	0.03	**0.14**	41.9	**0.19**	0.06	**0.25**	166.2	0.35	0.17	**0.52**	1,235.3

5.2 Quantitative Comparison

Table 1 presents the comparison of SOGAR to the other baseline methods on the 4 aforementioned datasets. The metrics we report are the recourse cost, recourse loss, invalidity, and computational time. We perform 10-fold cross validation on each dataset and report the averaged metrics. CET, AReS, and SOGAR are computationally heavy methods, so a time limit of 1 hour was imposed, which CET and AReS exceeded in some datasets. SOGAR never reaches this time limit and recovers all Pareto-optimal solutions.

SOGAR always outperforms all other methods on invalidity. It also has the lowest cost in every dataset besides Attrition. The methods that most often outperform SOGAR in terms of loss are GLOBE-CE and GLANCE, but even in these cases SOGAR's loss is often a close 2nd or 3rd.

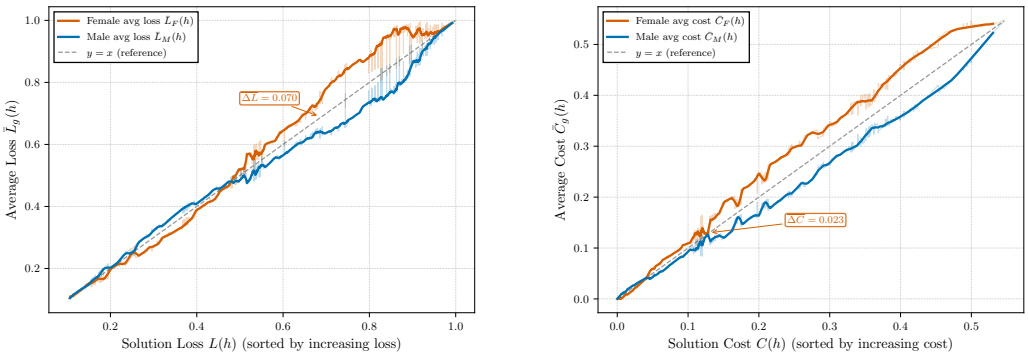
SOGAR's running time is higher than GLOBE-CE and GLANCE, but it produces the entire Pareto front in a single run (e.g., ≈6,000 solutions on Adult at ≈0.2s/solution), eliminating the need for repeated runs under different hyperparameter settings. Additionally, timeouts can be imposed with minimal quality degradation, as shown in Appendix C. Ablation studies on tree depth, action sparsity, binning granularity, and timeout sensitivity are provided in Appendices C–D, and standard deviations across folds are reported in Table 4 of Appendix B.

5.3 Pareto Front and Generalization

In Figure 3, the blue dashed line depicts the entire Pareto front of recourse summary trees of maximum depth 3, as computed by SOGAR on the training dataset. The trade-off between the two metrics becomes explicitly quantified and an end-user can decide which point on the Pareto front best fits their needs. We also plot metrics on a held-out portion of the same dataset to examine whether optimizing the objectives directly leads to over-fitting. Each orange dot represents a SOGAR solution evaluated on this held-out dataset. We can see that the orange dots are located very close to or on the dashed blue line, so there is little to no overfitting, with an average Euclidean distance of 0.04.

5.4 Bias Auditing Experiment

Setup We run SOGAR on the Adult Income dataset [3] (32,561 training instances) using the same LightGBM classifier and hyperparameters as the main experiments in Table 1 (depth 3, MaxNodes 7, MinLeaf 500). Grouping individuals by Sex, the classifier exhibits a Disparate Impact Ratio of $\mathbf{0.64}$, well below the $4/5$ threshold [34] signifying a level of bias: 86.5% of females receive adverse predictions versus 55.1% of males. Of the 19,199 adversely-predicted training instances, 43.8% are female. To further investigate the presence of bias we examine the differences between the recourse that females and males across the Pareto front of optimal recourse summary trees. Per-group metrics



(a) Loss ($\Delta L = 0.070$). (b) Cost ($\Delta C = 0.023$).

Figure 4: Loss and cost disparity across the Pareto front. The cost gap is persistent while the loss gap is more prominent.

are computed by routing every instance through all the Pareto front, assigning each instance the cost and loss of its landed leaf, then averaging per group and across the full front.

Findings. SOGAR generates 4,142 Pareto-optimal solutions. Across the front, **96.4%** of solutions exhibit higher invalidity for females than males on training instances (mean gap $\Delta I = +0.093$; test instances: 95.9%, $+0.081$). Figure 5 shows that this gap persists across the entire invalidity range. The decomposition in Figure 4 reveals that cost is worse for females for almost of regions of the Pareto front, while loss is favorable towards females for low-loss regions. Nevertheless, the loss disparities are wide enough in the high regions to contribute more to the total invalidity gap; $\Delta L = +0.070$, $\Delta C = +0.023$. The invalidity disparity is widest above the mid-range and narrows toward both extremes of the front. Single-output methods such as GLOBE-CE [19] and GLANCE [17] produce solutions in the low-loss and high-cost region where this gap is at its narrowest and would therefore understate the disparity that the full Pareto front exposes. Extended statistics are reported in Appendix F.

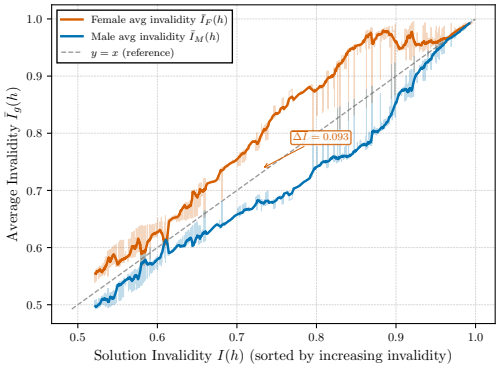


Figure 5: Invalidity ($\Delta I = 0.093$) across the Pareto front. Female group average is consistently above male across the entire front.

6 Conclusion

We introduced SOGAR, a method for computing recourse summaries by formulating the problem as bi-objective optimal decision tree learning. Leveraging the STreeD dynamic programming framework, SOGAR produces the complete Pareto front of recourse summary trees, providing global optimality guarantees over both recourse cost and effectiveness. Our experiments across four benchmark datasets demonstrate that SOGAR consistently outperforms existing recourse summary and global counterfactual explanation methods. On the invalidity metric, which equally weights cost and loss, SOGAR achieves the best performance in all datasets. The shallow, axis-parallel tree structure produces interpretable subgroup definitions, while the sparse actions assigned to each leaf remain actionable for affected individuals. The diversity of the Pareto front solutions allows us to examine how recourse differs per group for different values of total cost or loss, something that single-solution methods might miss.

7 Limitations

We acknowledge several limitations of our approach. First, SOGAR incurs higher computational costs than methods such as GLOBE-CE and GLANCE, though this is partially offset by producing an entire frontier of solutions rather than a single one. For time-sensitive applications, the anytime property of our algorithm allows early termination with the best solutions found thus far, but the optimality guarantees are then lost. Second, scalability to very large and high-dimensional datasets remains challenging. The possible combinations of action-instance pairs get excessively large and the cache our method computes might exceed available hardware resources, and while STreeD's caching and pruning mitigate the combinatorial explosion of the search space, these datasets may require additional constraints or feature selection. Finally, we point out that even though SOGAR can aid in auditing classifiers, it should not serve as the only criterion. Bias and fairness are complex multi-dimensional topics and multiple methods should be employed for a complete audit. Relying on just the results of SOGAR could potentially obscure bias hidden in complex relations between features or conflate representation bias within datasets with classifier bias.

References

[1] Sina Aghaei, Andrés Gómez, and Phebe Vayanos. Strong optimal classification trees. *arXiv preprint arXiv:2103.15965*, 2021.

[2] Gaël Aglin, Siegfried Nijssen, and Pierre Schaus. Learning optimal decision trees using caching branch-and-bound search. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3146–3153, 2020.

[3] Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

[4] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, July 2017.

[5] Dimitris Bertsimas, Agni Orfanoudaki, and Holly Wiberg. Interpretable clustering via optimal trees. *arXiv preprint arXiv:1812.00539*, 2018.

[6] Tom Bewley, Salim I Amoukou, Saumitra Mishra, Daniele Magazzeni, and Manuela Veloso. Counterfactual metarules for local and global recourse. *arXiv preprint arXiv:2405.18875*, 2024.

[7] L Breiman, JH Friedman, R Olshen, and CJ Stone. Classification and regression trees. 1984.

[8] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[9] Yu-Liang Chou, Catarina Moreira, Peter Bruza, Chun Ouyang, and Joaquim Jorge. Counterfactuals and causability in explainable artificial intelligence: Theory, algorithms, and applications. *Information Fusion*, 81:59–83, May 2022.

[10] Emir Demirović, Anna Lukina, Emmanuel Hebrard, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Peter J Stuckey. Murtree: Optimal decision trees via dynamic programming and search. *Journal of Machine Learning Research*, 23(26):1–47, 2022.

[11] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[12] Hans Hofmann. Statlog - german credit data. UCI Machine Learning Repository, 1994. DOI: https://doi.org/10.24432/C5NC77.

[13] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. *Advances in neural information processing systems*, 32, 2019.

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

[15] Kentaro Kanamori, Takuya Takagi, Ken Kobayashi, and Yuichi Ike. Counterfactual explanation trees: Transparent and consistent actionable recourse with decision trees. In *International Conference on Artificial Intelligence and Statistics*, pages 1846–1870. PMLR, 2022.

[16] Amir-Hossein Karimi, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse: from counterfactual explanations to interventions. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 353–362, New York, NY, USA, March 2021. Association for Computing Machinery.

[17] Loukas Kavouras, Eleni Psaroudaki, Konstantinos Tsopelas, Dimitrios Rontogiannis, Nikolaos Theologitis, Dimitris Sacharidis, Giorgos Giannopoulos, Dimitrios Tomaras, Kleopatra Markou, Dimitrios Gunopulos, Dimitris Fotakis, and Ioannis Emiris. Glance: Global actions in a nutshell for counterfactual explainability, 2025.

[18] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[19] Dan Ley, Saumitra Mishra, and Daniele Magazzeni. Globe-ce: A translation-based approach for global counterfactual explanations. (arXiv:2305.17021), December 2023. arXiv:2305.17021 [cs].

[20] Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and scalable optimal sparse decision trees. (arXiv:2006.08690), November 2022. arXiv:2006.08690 [cs].

[21] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. (arXiv:1705.07874), November 2017. arXiv:1705.07874 [cs].

[22] Hayden McTavish, Chudi Zhong, Reto Achermann, Ilias Karimalis, Jacques Chen, Cynthia Rudin, and Margo Seltzer. Fast sparse decision tree optimization via reference ensembles. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 9604–9613, 2022.

[23] S. Moro, P. Rita, and P. Cortez. Bank marketing. UCI Machine Learning Repository, 2014. DOI: https://doi.org/10.24432/C5K306.

[24] Nina Narodytska, Alexey Ignatiev, Filipe Pereira, and Joao Marques-Silva. Learning optimal decision trees with sat. In *International Joint Conference on Artificial Intelligence 2018*, pages 1362–1368. Association for the Advancement of Artificial Intelligence (AAAI), 2018.

[25] Siegfried Nijssen and Elisa Fromont. Mining optimal decision trees from itemset lattices. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 530–539, 2007.

[26] J Ross Quinlan. *C4. 5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[27] Kaivalya Rawal and Himabindu Lakkaraju. Beyond individualized recourse: Interpretable and interactive summaries of actionable recourses. *Advances in Neural Information Processing Systems*, 33:12187–12198, 2020.

[28] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[29] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

[30] Pavan Subhash. Ibm hr analytics employee attrition & performance dataset. `https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset`, 2017. Accessed: 2025-11-04.

[31] Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, page 10–19, New York, NY, USA, January 2019. Association for Computing Machinery.

[32] Jacobus van der Linden, Mathijs de Weerdt, and Emir Demirović. Necessary and sufficient conditions for optimal decision trees using dynamic programming. *Advances in Neural Information Processing Systems*, 36:9173–9212, 2023.

[33] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr, 2018.

[34] Elizabeth Anne Watkins and Jiahao Chen. The four-fifths rule is not disparate impact: a woeful tale of epistemic trespassing in algorithmic fairness. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, pages 764–775, 2024.

[35] Rui Zhang, Rui Xin, Margo Seltzer, and Cynthia Rudin. Optimal sparse survival trees. *Proceedings of machine learning research*, 238:352, 2024.

A Proof of Proposition 1

We prove Proposition 1 by showing that recourse summary tree optimization is a *separable* optimization task in the framework of [32]. By their Theorem 4.7 this guarantees that dynamic programming computes the exact Pareto front.

A.1 Defining the Optimization Task

We define the optimization task $o = \langle g, t, \succ, \oplus, c, s_0 \rangle$ as follows.

State Space A state is a pair $s = \langle \mathcal{D}_s, \mathcal{F}_s \rangle$. $\mathcal{D}_s \subseteq \mathcal{D}_B$ is a subset of the (original, binarized) instance pairs $\mathcal{D}_s \subseteq \mathcal{D}_B$. $\mathcal{F}_s \subseteq \mathcal{F}$ is the set of binary features used in branching decisions of ancestor nodes.

Solution Space A solution is a pair $v = (v_C, v_L) \in \mathbb{R}_{\geq 0} \times \mathbb{Z}$. v_C is the action cost aggregated over the instances reaching that leaf and v_L is the recourse loss aggregated in the same way.

Initial State $s_0 = \langle \mathcal{D}_B, \varnothing \rangle$

Cost Function For a state $s = \langle \mathcal{D}_s, \mathcal{F}_s \rangle$, the cost function g is 0 when branching on features:

$$g(s, f) = (0, 0), \quad f \in \mathcal{F} \tag{8}$$

It is equal to the aggregated costs when assigning actions to leaves:

$$g(s, a) = \left(\sum_{(\phi(x), x) \in \mathcal{D}_s} c(a, x), \sum_{(\phi(x), x) \in \mathcal{D}_s} l(a, x) \right), \quad a \in \mathcal{A} \tag{9}$$

Transition Function For a state $s = \langle \mathcal{D}_s, \mathcal{F}_s \rangle$, when branching on feature $f \in \mathcal{F} \setminus \mathcal{F}_s$ the transition function t is:

$$t(s, f) = \langle \{(\phi(x), x) \in \mathcal{D}_s : \phi(x)_f = 0\}, \mathcal{F}_s \cup \{f\} \rangle \tag{10}$$

$$t(s, \bar{f}) = \langle \{(\phi(x), x) \in \mathcal{D}_s : \phi(x)_f = 1\}, \mathcal{F}_s \cup \{f\} \rangle \tag{11}$$

Comparison Operator (Pareto Dominance) The comparison operator $\succ$ is the strict Pareto dominance for element-wise comparison:

$$(v_C, v_L) \succ (v'_C, v'_l) \Leftrightarrow (v_C \leq v'_C) \wedge (v_L \leq v'_L) \wedge ((v_C, v_L) \neq (v'_C, v'_l)) \tag{12}$$

In other words, at least one of the inequalities is not strict. Note that $\succ$ "translates" to $\leq$ (flipped order), because $v \succ v'$ means "v is preferred to v'" and therefore at least one of the minimization objectives is lower.

Combining Operator The combining operator $\oplus$ is element-wise addition:

$$(v_C, v_L) \oplus (v'_C, v'_L) = (v_C + v'_C, v_L + v'_L) \tag{13}$$

Constraint We consider the unconstrained case, $c(v, s) = 1,\ \forall v, s$.

In our implementation, we also use maximum leaf count and minimum leaf size as constraints, but these can be incorporated without affecting separability, as demonstrated in [32]. Maximum leaf count is especially useful, since it imposes a bound on the total number of population subgroups which improves the interpretability of the recourse summary tree.

A.2 Verification of Separability Conditions

By Theorem 4.6 of [32], an optimization task is separable if and only if its cost function g and transition function t are Markovian, its combining operator $\oplus$ preserves order over its comparison operator $\succ$ and the constraint c is anti-monotonic.

The cost function g and transition function t, as defined in equations (8), (9), (10) and (11), are indeed Markovian as by their definition they only depend on the current state and the selected action/feature. The trivial constraint $c(v, s) = 1$ is also trivially anti-monotonic.

Lemma 1 (Order Preserving Combining Operator)**.** *The combining operator $\oplus$, as defined in Equation* (13)*, preserves the order of the comparison operator $\succ$ as defined in Equation* (12)*.*

Proof. Let $v = (v_C, v_L)$, $v' = (v'_C, v'_L)$, and $u = (u_C, u_L)$ be solution values with $v \succ v'$. We must show that $v \oplus u \succ v' \oplus u$. By the definitions of the operators and the properties of element-wise addition, we have:

- $v_C \leq v'_C$ and $v_L \leq v'_L$, with at least one strict.
- $v_C + u_C \leq v'_C + u_C$ and $v_L + u_L \leq v'_L + u_L$, with at least one strict.
- Therefore $v \oplus u \succ v' \oplus u$

Since $\oplus$ is commutative, it also holds that $u \oplus v \succ u \oplus v'$. □

We have verified all necessary condition of Theorem 4.6, therefore recourse summary tree computation is separable. By Theorem 4.7 of [32], their dynamic programming formulation, termed *STreeD*, computes the complete Pareto front of recourse summary trees:

$$T(s, d) = \begin{cases} \text{opt}\left(\bigcup_{a \in \mathcal{A}} \{g(s, a)\}, s\right) & \text{if } d = 0 \\ \text{opt}\left(\bigcup_{f \in \mathcal{F}} \text{merge}\left(T(t(s, f), d-1), T(t(s, \bar{f}), d-1), s, f\right), s\right) & \text{if } d > 0 \end{cases} \tag{14}$$

Where the definitions of opt and merge are taken directly from [32] and are the following:

$$\text{merge}(\Theta_1, \Theta_2, s) = \{v_1 \oplus v_2 \oplus g(s, f) \mid v_1 \in \Theta_1, v_2 \in \Theta_2\} \tag{15}$$

$$\text{opt}(\Theta, s) = \text{nondom}(\text{feas}(\Theta, s)) \tag{16}$$

$$\text{nondom}(\Theta) = \{v \in \Theta \mid \nexists v' \in \Theta (u' \succ v)\} \tag{17}$$

$$\text{feas}(\Theta, s) = \{v \in \Theta \mid c(v, s) = 1\} \tag{18}$$

This concludes the proof of Proposition 1. □

B Dataset Preprocessing, Model Parameters & Analytical Quantitative Results

B.1 Datasets.

In our experiments we constrained each dataset to provide practical actions, suitable for real-world deployment, by prohibiting the actionability of sensitive features (e.g., Gender, Race, Age) and features that cannot be proposed to be altered, in specific settings for affected individuals, such as the Type or Level of Education, or the Distance of an employee's home from their work place in the Employee Attrition Setting. We also set directionality of actions, which respects features such as time-dependent features or financial status features, where an action of lowering one's income would

be unfavorable. Below, we state the actionability of each dataset based on the percentage of features made immutable, the directionality constraints, and the number of bins allowed to be changed per feature.

Table 2: Actionability configuration per dataset. Immutability % is computed after one-hot encoding of categorical features.

Dataset	**Immutable (%)**	**Actions (>)**
Attrition	26%	120,000
German	20%	90,000
Bank	65%	5,000
Adult	50%	100,000

B.2 Classifiers & SOGAR parameters

Classifiers. We used two state-of-the-art classifiers, LightGBM and XGBoost, and a DNN classifier, all of which employ a black-box architecture. The parameters used to train the models were standard across all datasets, and methods run in our experiments. Specifically, the LightGBM Classifier, was set to `n_estimators` $= 100$ and `n_leaves`$= 16$, the XGBoost was set to `n_estimators` $= 100$ and `max_depth`$= 6$, and the DNN was set to have `depth` $= 5$, `width` $= 50$ and `dropout`$= 0.1$. The rest of the parameters were set to default values.

SOGAR default settings. SOGAR's key hyperparameters are the maximum tree depth d, maximum number of leaf nodes m, minimum leaf size ℓ, and action sparsity k (maximum number of feature changes per action). All main-paper experiments use the following defaults depicted in Table 3. The larger minimum leaf size for Adult (500 vs. 50) is necessary to control the search space on a dataset with over 30,000 training instances and 161 binarised features. Ablations on depth, sparsity, and bins are reported in Appendix D.

Table 3: SOGAR default hyperparameters for all main experiments.

Parameter	**Value**
Max depth d	3
Max nodes m	7
Min leaf size ℓ	50 (500 for Adult)
Action sparsity k	3
Feature bins	10–50 (distribution-adaptive)
Timeout	None (full Pareto front)

B.3 Extended Quantitative Results of Table 1

In this part of the Appendix, we present the extended variant of Table 1 that presents the results of the 10-fold cross validation across the four datasets of Employee Attrition, German Credit, Bank Marketing, and Adult Income, and is used to compare SOGAR with baselines. For the integrity of our quantitative experiments, the extended variants include the standard deviation for all four metrics recorded in Table 1, average *cost, loss, invalidity,* and the *computation time* with the standard deviation.

C Timeout Ablation

Because SOGAR is an *anytime* algorithm, a wall-clock timeout can be imposed to enforce a hard budget, with the solver returning the best non-dominated solutions found so far. Figure 6 illustrates this behaviour on the Bank dataset with LightGBM, where the full Pareto front requires 166 s: even at 10 s, recovered solutions exist below $(0.2, 0.4)$ in cost–loss space, outperforming the second-best competing method CET (invalidity 0.61 on Bank, Table 4b). Table 5 quantifies this across all datasets, reporting degradation as the relative increase in invalidity over the no-timeout baseline:

$$\text{Degradation} = \frac{\text{inv}_\tau - \text{inv}_\infty}{\text{inv}_\infty} \times 100\%.$$

Table 4: Evaluating SOGAR against competing methods on 4 datasets and 3 classifiers, with metrics averaged over 10 folds (mean±std). Lower is better for all metrics. SOGAR reports the solution with lowest invalidity. A dash indicates timeout exceeded.

(a) Attrition and German Credit

Models	Algorithms	Attrition				German			
		cost↓	*loss*↓	*inv.*↓	*time(s)*↓	*cost*↓	*loss*↓	*inv.*↓	*time(s)*↓
DNN	CET	0.30±0.12	0.10±0.07	0.40±0.16	198.7±18.6	0.04±0.02	0.44±0.13	0.48±0.12	255.3±68.2
	AReS	0.23±0.04	0.23±0.12	0.46±0.13	139.3±6.2	0.17±0.30	0.39±0.07	0.56±0.30	1,845.3 ± 210
	GLOBE-CE	0.81±0.02	<0.01±0.02	0.81±0.02	7.6±0.25	0.91±0.003	0.01±0.02	0.92±0.02	7.58±0.32
	GLANCE	0.54±0.06	<0.01±0.002	0.54±0.06	11.66±0.22	0.47±0.16	<0.01±0.004	0.47±0.16	8.5±0.4
	T-CREx$_{0.9}$	0.35±0.11	0.77±0.12	1.13±0.21	7.0±0.33	0.53±0.12	0.78±0.08	1.31±0.13	32.19±73.8
	T-CREx$_{0.99}$	0.56±0.11	0.65±0.15	1.21±0.16	176.32±230	0.60±0.08	0.70±0.15	1.25±0.15	19.63±16
	SOGAR	**0.05±0.002**	**<0.01±0.001**	**0.05±0.003**	250.0±6.2	**0.02±0.004**	**0.00±0.00**	**0.02±0.003**	28.2±0.9
XGBoost	CET	0.40±0.10	0.17±0.14	0.57±0.11	342.6±29.3	0.12±0.03	0.25±0.05	0.37±0.05	151.2±35.7
	AReS	**0.02±0.02**	0.97±0.04	0.99±0.04	11.68±2	0.41±0.06	0.13±0.05	0.55±0.06	2,362.8 ± 123.3
	GLOBE-CE	0.75±0.03	**0.00±0.00**	0.75±0.03	**5.90±0.21**	0.92±0.003	0.25±0.17	1.16±0.17	**5.5±0.0**
	GLANCE	0.56±0.12	0.02±0.03	0.58±0.11	11.70±0.11	0.50±0.15	**<0.01±0.003**	0.50±0.15	8.4±0.07
	T-CREx$_{0.9}$	0.33±0.08	0.72±0.18	1.10±0.25	7.99±1.14	0.40±0.15	0.75±0.18	1.20±0.20	8.2±1.40
	T-CREx$_{0.99}$	0.48±0.17	0.64±0.26	1.12±0.32	14.13±2.65	0.56±0.07	0.57±0.15	1.13±0.22	8.03±28
	SOGAR	0.23±0.02	0.02±0.02	**0.26±0.01**	254.1±13.6	**0.09±0.009**	0.02±0.007	**0.12±0.01**	26.4±1.4
LightGBM	CET	**0.26±0.18**	0.58±0.24	0.84±0.09	286.7±20.1	0.13±0.04	0.31±0.11	0.44±0.10	173.8±13.1
	AReS	0.44±0.05	0.44±0.10	0.88±0.06	307.4±22.7	0.19±0.08	0.65±0.13	0.84±12	680.4±44.6
	GLOBE-CE	0.79±0.02	**0.01±0.008**	0.80±0.03	**7.96±0.28**	0.91±0.0	0.23±0.30	1.14±0.30	**6.1±0.2**
	GLANCE	0.62±0.02	0.08±0.04	0.70±0.04	12.17±0.5	0.52±0.14	**<0.01±0.003**	0.53±0.14	9.3±0.29
	T-CREx$_{0.9}$	0.37±0.12	0.80±0.13	1.16±0.23	7.08±0.29	0.44±0.09	0.72±0.09	1.16±0.10	8.25±1.95
	T-CREx$_{0.99}$	0.58±0.06	0.84±0.12	1.42±0.17	112.67±114.9	0.54±0.04	0.61±0.30	1.15±0.32	10.0±1.4
	SOGAR	0.30±0.02	0.07±0.02	**0.37±0.02**	356.4±9.4	**0.11±0.005**	0.03±0.007	**0.14±0.009**	41.9±2.5

(b) Bank Marketing and Adult Income

Models	Algorithms	Bank				Adult			
		cost↓	*loss*↓	*inv.*↓	*time(s)*↓	*cost*↓	*loss*↓	*inv.*↓	*time(s)*↓
DNN	CET	0.27±0.16	0.60±0.22	0.86±0.09	378.9±84.1	–	–	–	–
	AReS	–	–	–	–	–	–	–	–
	GLOBE-CE	0.87±0.30	0.04±0.20	0.89±0.30	23.2±0.1	0.98±0.10	0.00±0.00	0.98±0.10	170.5±7
	GLANCE	0.65±0.06	<0.01±0.006	0.66±0.06	10.75±0.33	0.96±0.003	0.00±0.00	0.96±0.003	47.22±5.1
	T-CREx$_{0.9}$	–	–	–	–	0.66±0.30	0.91±0.08	1.58±0.20	15.04±5.09
	T-CREx$_{0.99}$	–	–	–	–	0.64±0.35	0.86±0.16	1.50±0.31	10.4±0.34
	SOGAR	**0.17±0.01**	0.29±0.04	**0.46±0.04**	268.9±6.7	**0.03±0.01**	0.03±0.03	**0.06±0.03**	1,248.4 ± 67.4
XGBoost	CET	0.51±0.11	0.24±0.20	0.75±0.06	438.8±65.7	–	–	–	–
	AReS	–	–	–	–	–	–	–	–
	GLOBE-CE	0.91±0.20	**0.01±0.0**	0.92±0.20	30.91±0.0	0.98±0.01	**0.00±0.00**	0.98±0.10	205.2±6
	GLANCE	0.69±0.12	0.03±0.03	0.72±0.11	10.6±0.21	0.96±0.01	<0.01±0.001	0.96±0.01	35.1±1.53
	T-CREx$_{0.9}$	0.32±0.16	0.99±0.01	1.31±0.15	6.87±0.33	0.70±0.17	0.90±0.13	1.46±0.25	12.52±2.48
	T-CREx$_{0.99}$	–	–	–	–	0.35±0.03	0.99±0.02	1.13±0.20	10.23±0.4
	SOGAR	**0.21±0.03**	0.26±0.06	**0.47±0.04**	250.7±32.3	**0.30±0.04**	0.34±0.06	**0.64±0.04**	763.0±62.8
LightGBM	CET	0.60±0.02	0.01±0.01	0.61±0.01	383.9±119	–	–	–	–
	AReS	0.36±0.02	0.80±0.08	1.15±0.09	213.7±29.2	–	–	–	–
	GLOBE-CE	0.92±0.30	**<0.01±0.0**	0.92±0.30	29.8±0.5	0.97±0.01	**0.00±0.00**	0.97±0.01	171.6±8.9
	GLANCE	0.71±0.09	**<0.01±0.00**	0.71±0.09	**11.0±0.42**	0.93±0.01	<0.01±0.00	0.93±0.01	**32.0±0.89**
	T-CREx$_{0.9}$	0.38±0.20	0.66±0.26	1.04±0.21	6.83±0.55	0.28±0.15	0.96±0.04	1.24±0.16	10.88±2.52
	T-CREx$_{0.99}$	0.47±0.14	0.58±0.28	1.04±0.23	8.71±15.79	0.45±0.17	0.99±0.01	1.44±0.17	10.5±0.85
	SOGAR	**0.19±0.03**	0.06±0.02	**0.25±0.03**	166.2±4.3	0.35±0.03	0.17±0.02	**0.52±0.03**	1,235.3 ± 75.3

For Attrition, German, and Bank, even the 10-second budget produces degradation of only 8–16%, while 60-second solutions are near-optimal. The Adult dataset requires at least 200 s to produce any valid solution, consistent with its higher baseline runtime (1,235 s, Table 4b) and the combinatorial complexity of its large, high-dimensional feature space. Even under the 200-second constraint, Adult degradation is only 5.8%. At 200 s, SOGAR's invalidity of 0.55 remains below all baselines that return solution on the Adult setting, with the runners-up being the solution of GLANCE at 0.93 average invalidity.

D Ablation Studies

D.1 Tree Depth

SOGAR's recourse summary tree has a maximum depth hyperparameter d, which directly controls the number of subgroups exposed to practitioners (up to 2^d leaf nodes). We evaluate $d \in \{2, 3, 4\}$. Depth 3 is the default used in all main-paper experiments (Tables 1); depth 2 and depth 4 isolate the effect of tree expressiveness on solution quality and runtime.

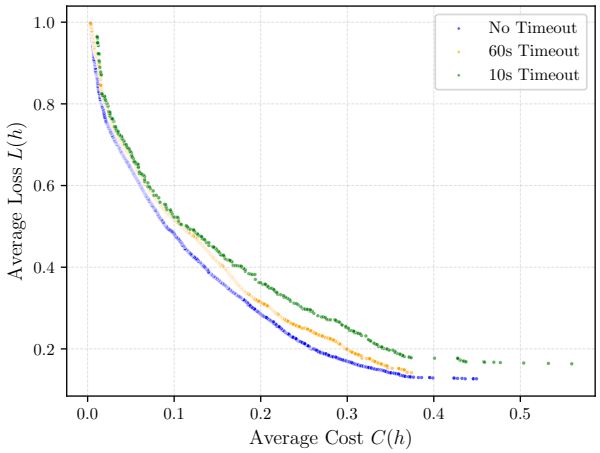


Figure 6: Solutions recovered by SOGAR on Bank (LightGBM) under different timeout values. Blue: full Pareto front (166 s). Orange: best solutions within 60 s. Green: best solutions within 10 s.

Table 5: Timeout ablation (LightGBM, depth 3, 10-fold CV). Degradation is the relative invalidity increase over the no-timeout baseline. Mean±std reported across folds.

Dataset	**Timeout**	*cost*↓	*loss*↓	*inv.*↓	**Deg. (%)**
Attrition	10 s	0.33±0.02	0.10±0.03	0.43±0.03	+16.2
	60 s	0.31±0.02	0.08±0.02	0.39±0.02	+5.4
	No timeout	0.30±0.02	0.07±0.02	0.37±0.02	—
German	10 s	0.12±0.00	0.04±0.01	0.16±0.01	+14.3
	60 s	0.11±0.01	0.03±0.01	0.14±0.01	0.0
	No timeout	0.11±0.01	0.03±0.01	0.14±0.01	—
Bank	10 s	0.19±0.05	0.08±0.04	0.28±0.04	+12.0
	60 s	0.19±0.04	0.08±0.03	0.27±0.04	+8.0
	No timeout	0.19±0.03	0.06±0.02	0.25±0.03	—
Adult	10 s	*no solution found*			—
	60 s	*no solution found*			—
	200 s	0.35±0.01	0.20±0.04	0.55±0.04	+5.8
	No timeout	0.35±0.03	0.17±0.02	0.52±0.03	—

Depth 2. Table 6 reports results for depth $d = 2$. Across all datasets and classifiers, depth-2 trees run substantially faster — often by an order of magnitude — while the increase in invalidity relative to depth 3 is modest. For example, on Adult with LightGBM, invalidity rises from 0.52 (depth 3) to 0.57 (depth 2) while runtime drops from $1{,}235$ s to 51 s, a $24\times$ speed-up for a 9.6% quality degradation. Depth 2 still outperforms all baselines on the invalidity metric across every dataset and classifier combination, making it a practical default when compute is constrained. The solutions are also maximally interpretable, producing exactly four subgroups with simple branching rules.

Depth 3 (default). Depth-3 results with standard deviations across 10 folds are reported in Tables 1. This depth offers the best balance between solution quality and computational cost: it consistently achieves the lowest invalidity among all baselines while remaining tractable on all four datasets without timeouts.

Depth 4. Table 7 reports results for $d = 4$ with the LightGBM classifier. At this depth, the search space grows substantially, as the solver must explore up to 16 leaf nodes, and the number of candidate subproblems increases combinatorially. On the Adult dataset, depth-4 solutions require approximately $15{,}937$ s (≈4.4 hours), a $12.9\times$ increase over depth 3. The invalidity improvement is marginal ($0.52 \rightarrow 0.50$), yielding diminishing returns relative to the computational investment. On smaller datasets where depth-4 remains tractable, some additional quality is recoverable: invalidity improves by 11% on Attrition ($0.37 \rightarrow 0.33$), 7% on German ($0.14 \rightarrow 0.13$), and 12% on Bank

Table 6: Depth-2 results across datasets and classifiers (10-fold CV, mean±std). Lower is better for all metrics.

Dataset	**Model**	*cost*↓	*loss*↓	*inv.*↓	*time(s)*↓
Attrition	DNN	0.06 ± 0.00	$<0.01 \pm 0.00$	0.06 ± 0.00	9 ± 1
	LightGBM	0.31 ± 0.03	0.12 ± 0.02	0.43 ± 0.03	12 ± 1
	XGBoost	0.25 ± 0.01	0.06 ± 0.01	0.31 ± 0.01	10 ± 0
German	DNN	0.02 ± 0.01	$<0.01 \pm 0.00$	0.03 ± 0.01	3 ± 0
	LightGBM	0.11 ± 0.01	0.05 ± 0.01	0.16 ± 0.01	3 ± 1
	XGBoost	0.11 ± 0.01	0.04 ± 0.02	0.15 ± 0.02	3 ± 0
Bank	DNN	0.17 ± 0.01	0.33 ± 0.03	0.49 ± 0.04	8 ± 0
	LightGBM	0.19 ± 0.03	0.08 ± 0.03	0.27 ± 0.03	6 ± 1
	XGBoost	0.22 ± 0.04	0.29 ± 0.07	0.51 ± 0.04	8 ± 1
Adult	DNN	0.03 ± 0.01	0.04 ± 0.03	0.07 ± 0.04	47 ± 1
	LightGBM	0.37 ± 0.02	0.20 ± 0.04	0.57 ± 0.03	51 ± 3
	XGBoost	0.32 ± 0.08	0.37 ± 0.11	0.69 ± 0.04	63 ± 3

Table 7: Depth-4 results with LightGBM across datasets (10-fold CV, mean±std). Lower is better for all metrics.

Dataset	*cost*↓	*loss*↓	*inv.*↓	*time(s)*↓
Attrition	0.29 ± 0.02	0.03 ± 0.01	0.33 ± 0.02	$2{,}710 \pm 97$s
German	0.11 ± 0.01	0.02 ± 0.01	0.13 ± 0.01	582.5 ± 51s
Bank	0.17 ± 0.04	0.05 ± 0.02	0.22 ± 0.03	$3{,}112 \pm 478$s
Adult	0.33 ± 0.02	0.17 ± 0.02	0.50 ± 0.03	$15{,}937 \pm 1,410$s

($0.25 \rightarrow 0.22$), though depth-3 already outperforms all baselines in these settings, in a significantly lower computational cost for a complete Pareto front of solutions. This confirms that depth 3 represents the practical sweet spot for the datasets considered: deeper trees offer only incremental quality gains while incurring substantially higher costs, consistent with the NP-hard nature of the underlying optimization problem.

Table 8 consolidates the depth trade-off for the Adult dataset with LightGBM, which is the most computationally demanding configuration.

D.2 Action Sparsity

In the default configuration SOGAR allows actions that modify up to three features simultaneously (action sparsity $k = 3$). We assess a sparser setting ($k = 2$), evaluating the trade-off between solution quality, runtime, and memory usage (Table 9). Experiments use the LightGBM classifier at depth 3.

Restricting actions to two features yields approximately a 50% reduction in runtime (e.g. Attrition: $356 \rightarrow 163$ s; Adult: $1{,}235 \rightarrow 534$ s) and a proportional reduction in action-cache memory, since the cache scales with the number of distinct actions. The cost metric is largely unaffected, but recourse loss increases noticeably — particularly on Adult ($0.17 \rightarrow 0.44$) and Attrition ($0.07 \rightarrow 0.17$) — because sparser actions can no longer simultaneously address all necessary feature changes for harder subgroups. On Adult, the invalidity rises from 0.52 to 0.84, which is still below T-CREx (1.24), and GLANCE (0.93), illustrating that sparsity constraints interact strongly with dataset complexity. We therefore recommend $k = 2$ only when memory or time is severely constrained, and note that even under this restriction SOGAR remains competitive with baselines on the smaller datasets.

D.3 Number of Discretisation Bins

Continuous features are discretised into bins before the tree search. In the default configuration the number of bins varies per feature according to its empirical distribution (ranging from 10 to 50 bins). We evaluate two reduced-bin settings (5 and 10 bins applied uniformly to all numerical features) on the Attrition dataset with the LightGBM classifier (Table 10). Degradation is computed relative to the default-bin baseline.

Table 8: Depth ablation summary on Adult Income (LightGBM, 10-fold CV). Degradation is relative invalidity change vs. depth 3.

Depth	*inv.*↓	*time(s)*↓	**Deg. (%)**	**Speed-up**
2	0.57 ± 0.03	51 ± 3	$+9.6\%$	$24.2\times$
3	0.52 ± 0.03	$1{,}235 \pm 75$	—	$1.0\times$
4	0.50 ± 0.03	$15{,}937 \pm 1{,}410$	-3.8%	$0.08\times$

Table 9: Action-sparsity ablation: $k = 2$ vs. default $k = 3$ (LightGBM, depth 3, 10-fold CV, mean±std).

Dataset	*cost*↓	*loss*↓	*inv.*↓	*time(s)*↓
Attrition	0.32 ± 0.02	0.17 ± 0.03	0.50 ± 0.03	162.6 ± 6.8
German	0.12 ± 0.00	0.08 ± 0.01	0.19 ± 0.01	23.4 ± 1.7
Bank	0.14 ± 0.03	0.22 ± 0.03	0.36 ± 0.04	116.7 ± 2.4
Adult	0.40 ± 0.16	0.44 ± 0.18	0.84 ± 0.04	533.9 ± 25.8

Reducing the bin count from the default distribution-adaptive scheme to a uniform 10-bin or 5-bin encoding roughly halves runtime (to 163 s and 127 s respectively) with only 3–5% degradation in invalidity. On this configuration, SOGAR with 5 uniform bins still outperforms all baselines on the invalidity metric, suggesting that reduced binning is a viable strategy when runtime or memory is constrained. The runtime reduction stems from two sources: fewer bins produce fewer binarized features, shrinking the split search space, and they also reduce the number of distinct actions in the cache. We note that very coarse binning risks conflating meaningfully different feature values within a single bin. A practical lower bound are 5 bins below which solution quality may degrade more sharply for features with heavy-tailed distributions.

E Leaf solution acceleration

We have extended STreeD to include multi-threaded CPU and GPU acceleration in our optimization task. As mentioned in Section 4, when assigning an action to a leaf, the solver needs to try all actions and for each of them aggregate the cost-loss pairs for that action of the individuals reaching that leaf. This is an embarrassingly parallelizable task which we implement using the TBB framework for CPU parallelization and the CUDA framework for GPU parallelization. Users can make use whichever acceleration their hardware supports. Table 11 presents the runtimes all datasets using the single-threaded implementation and the GPU implementation. We can see that the GPU incurs significant speed-ups, eliminating time-outs and in the case of the German Credit dataset the running time is reduced by a factor of over 23.

F Bias Auditing Experiment — Extended Analysis

Setup

SOGAR was ran on the Adult Income dataset [3] using the default setting (32,561 training instances; 3,257 held-out, LightGBM, depth 3, MaxNodes 7, MinLeaf 500), the results of which are depicted in Table 1. Per-group metrics are computed by routing all 19,199 unfavorably-predicted training instances through every Pareto-front solution tree, assigning each instance the cost and loss of its landed leaf, averaging per group, then plotting across the full front.

Classifier Fairness

The LightGBM classifier exhibits measurable bias at the prediction level under two standard criteria, as reported in Table 12. Under **demographic parity**, the classifier assigns adverse predictions ($\hat{Y} = 0$) to 86.5% of females versus 55.1% of males on train (85.6% vs. 55.5% on test), yielding a Disparate Impact Ratio of 0.64 (test: 0.65) — well below the $4/5$ threshold [34]. Under **equality of opportunity**, among individuals who truly earn >50K ($Y = 1$), the classifier correctly identifies 90.4% of males as favorable but only 79.2% of females, which depicts a true positive rate (TPR)

Table 10: Binning ablation (LightGBM, Attrition, depth 3, 10-fold CV, mean±std). Degradation is relative invalidity increase vs. default bins.

Bins	*cost*↓	*loss*↓	*inv.*↓	*time(s)*↓	**Deg. (%)**
Default (10–50)	0.30 ± 0.02	0.07 ± 0.02	0.37 ± 0.02	356.4 ± 9.4	—
10 bins (uniform)	0.31 ± 0.01	0.08 ± 0.02	0.38 ± 0.01	163 ± 4	$+3.2\%$
5 bins (uniform)	0.32 ± 0.01	0.07 ± 0.01	0.39 ± 0.01	127 ± 4	$+4.7\%$

Table 11: Comparison of SOGAR's running times with and without GPU acceleration on the all datasets. The timeout is set to 1 hour.

Dataset	**Single-Threaded**	**GPU Acceleration**
Attrition	708.6s	250.0s
German	668.2s	28.2s
Bank	>1h	268.9s
Adult	>1h	1248.4s

Table 12: Classifier fairness metrics (LightGBM, Adult Income, fold 1).

Metric	**Train**	**Test**
Accuracy	0.841	0.832
Demographic parity		
Female n / adverse ($\hat{Y}=0$)	9,730 / 8,417	1,041 / 891
Male n / adverse ($\hat{Y}=0$)	19,574 / 10,782	2,216 / 1,229
Female adverse rate ($\hat{Y}=0$)	86.5%	85.6%
Male adverse rate ($\hat{Y}=0$)	55.1%	55.5%
Disparate Impact Ratio	**0.64**	**0.65**
Equality of opportunity $P(\hat{Y}=1 \mid Y=1)$		
Female n ($Y=1$)	1,054	125
Male n ($Y=1$)	6,002	660
Female true positive rate (TPR)	0.792	0.776
Male true positive rate (TPR)	0.904	0.894
Gap (Male − Female)	$+0.112$	$+0.118$

gap of $+0.112$ on train ($+0.118$ on test). A female who deserves a favorable prediction faces a substantially higher probability of being wrongly denied it than an equivalent male. Both violations are consistent across train and test splits, confirming they reflect the model's learned structure.

Per-Group Recourse Quality & Disparity

Across all 4,142 Pareto-front solutions, females face consistently worse recourse than males on every metric, yet the nature of the disadvantage is asymmetric. The invalidity gap — $+0.093$ in train data and $+0.081$ in test data — is driven mostly by loss ($+0.070$) rather than cost ($+0.023$). Note also that the scale of the plots is different as loss reaches 1.0 at the rightmost, while cost reaches a bit above 0.5, so the loss gap can appear less exaggerated than it actually is. The full per-group breakdown is reported in Table 13, with train–test consistency confirming the disparity reflects the classifier's learned structure rather than properties of the recourse solutions.

The two components also behave differently along the front, as shown in Figure 4. The invalidity gap opens wide early, compresses sharply around $I \approx 0.58$–0.62 where the female and male curves briefly converge, then widens significantly as the invalidity increases very rapidly for females. Eventually, invalidity rises for males as well and meets the female invalidity, depicting that in both extremes invalidity marginally converges. However, it is this late divergence in high cost region of $C \approx 0.62$–0.9 that sustains the near-universal disparity across the front.

Pareto Front

SOGAR audits bias as a by-product of its core output, as it returns a Pareto front of recourse summary trees. An auditor can route all instances through every solution and compare per-group cost, loss, and invalidity at each operating point. A disparity that holds across the entire front, as the 96.4% worse

Table 13: Per-group recourse quality across the Pareto front (4,142 solutions). Standard deviation is across solutions. Gap = Female − Male per solution, averaged across the Pareto front.

Metric	**Group / Statistic**	**Train**	**Test**
Avg. cost	Female	0.116 ± 0.143	0.116 ± 0.145
	Male	0.094 ± 0.112	0.097 ± 0.117
Avg. loss	Female	0.672 ± 0.285	0.676 ± 0.286
	Male	0.602 ± 0.231	0.614 ± 0.239
Avg. invalidity	Female	0.788 ± 0.151	0.792 ± 0.151
	Male	0.696 ± 0.133	0.711 ± 0.134
Mean gap (F−M)	Cost	$+0.023$	$+0.019$
	Loss	$+0.070$	$+0.062$
	Invalidity	$+0.093$	$+0.081$

invalidity for females shows in Table 13, cannot be attributed to a particular cost-loss trade-off choice and instead reflects the classifier's decision boundary itself. This front-wide diagnostic is unavailable to methods that return a single recourse summary or a fixed set of counterfactuals.