Coevolutionary Algorithm for Building Robust Decision Trees under Minimax Regret

Adam Żychowski[1], **Andrew Perrault**[2], **Jacek Mańdziuk**[1, 3, 4]
[1]Faculty of Mathematics and Information Science, Warsaw University of Technology
[2]Department of Computer Science and Engineering, The Ohio State University
[3]Faculty of Computer Science, AGH University of Krakow
[4]Center of Excellence in Artificial Intelligence, AGH University of Krakow
adam.zychowski@pw.edu.pl, perrault.17@osu.edu, jacek.mandziuk@pw.edu.pl

Abstract

In recent years, there has been growing interest in developing robust machine learning (ML) models that can withstand adversarial attacks, including one of the most widely adopted, efficient, and interpretable ML algorithms—decision trees (DTs). This paper proposes a novel coevolutionary algorithm (CoEvoRDT) designed to create robust DTs capable of handling noisy high-dimensional data in adversarial contexts. Motivated by the limitations of traditional DT algorithms, we leverage adaptive coevolution to allow DTs to evolve and learn from interactions with perturbed input data. CoEvoRDT alternately evolves competing populations of DTs and perturbed features, enabling construction of DTs with desired properties. CoEvoRDT is easily adaptable to various target metrics, allowing the use of tailored robustness criteria such as minimax regret. Furthermore, CoEvoRDT has potential to improve the results of other state-of-the-art methods by incorporating their outcomes (DTs they produce) into the initial population and optimize them in the process of coevolution. Inspired by the game theory, CoEvoRDT utilizes mixed Nash equilibrium to enhance convergence. The method is tested on 20 popular datasets and shows superior performance compared to 4 state-of-the-art algorithms. It outperformed all competing methods on 13 datasets with adversarial accuracy metrics, and on all 20 considered datasets with minimax regret. Strong experimental results and flexibility in choosing the error measure make CoEvoRDT a promising approach for constructing robust DTs in real-world applications.

Introduction

Decision trees (DTs) is a popular, easily interpretable machine learning (ML) algorithm for classification and regression tasks. One of the primary challenges in DT construction is dealing with noisy and high-dimensional data. In particular, it has been shown that ML models (including DTs) are vulnerable to adversarial, perturbed samples that trick the model into misclassifying them (Kantchelian, Tygar, and Joseph 2016; Zhang, Zhang, and Hsieh 2020; Grosse et al. 2017). To address this challenge, researchers have proposed new defensive algorithms for creating *robust* classification models (see, e.g., Chakraborty et al. (2021)). A model is defined to be robust to some perturbation range of its input samples when it assigns the same class to all the samples within that perturbation range, so that small malicious alterations of input objects should not deceive a robust classifier.

The vast majority of defensive algorithms for DTs focus on adversarial accuracy (Kantchelian, Tygar, and Joseph 2016; Chen et al. 2019; Guo et al. 2022; Ranzato and Zanella 2021; Justin et al. 2021). We argue that there are better metrics in principle, and the focus on adversarial accuracy has been driven by computational tractability. Adversarial accuracy is highly sensitive to accuracy on the worst-case perturbation, and when the perturbation range can be large, this can lead to flattening of intuitively good models and bad ones, as the worst-case perturbations can "defeat" all models.

There are other metrics that can better evaluate model robustness, like *max regret* (Savage 1951). Max regret is defined as the maximum difference between the result of the given model and the result of the optimal model for any input data perturbation within a given range. Minimizing max regret is referred to as the minimax regret decision criterion. Adversarial accuracy might provide an overly optimistic or pessimistic view of the model's robustness by focusing only on absolute accuracy value. In contrast, max regret is a more realistic approach since it counts the magnitude of the potential loss by considering the model trained on perturbed data. However, max regret cannot be directly optimized and used as a splitting criterion in the state-of-the-art algorithms.

In recent years, researchers successfully explored the potential of coevolutionary algorithms to various optimization problems (Mahdavi, Shiri, and Rahnamayan 2015) including DTs induction (Aitkenhead 2008). Coevolutionary algorithms consist in simultaneous evolution of multiple populations, each of them representing a different aspect of the problem. By fostering competition between populations, coevolutionary algorithms can guide the search towards the optimal solutions.

Considering the limitations of traditional DT algorithms and the promises of coevolutionary computation, we propose a novel coevolutionary algorithm specifically tailored for creating robust decision trees (RDTs) in adversarial contexts. Our approach leverages the power of adaptive coevolution, allowing to exploit the competitive interactions between populations of decision trees and adversarial perturbations to adapt and converge toward robust and accurate classifications for complex and noisy data. In this process, we

can freely define robustness metrics to optimize (including max regret) which leads to the models better tailored to handle perturbed high-dimensional data. Because of the inherent flexibility of evolutionary methods, we can additionally integrate other objective criteria, such as fairness (Aghaei, Azizi, and Vayanos 2019; Jo et al. 2022).

The main contribution of this paper is proposition of a novel coevolutionary algorithm (CoEvoRDT) capable of creating robust decision trees. CoEvoRDT has the following key properties:

- supremacy over state-of-the-art (SOTA) solution on 13 out of 20 datasets with adversarial accuracy metric and on all 20 datasets with minimax regret,
- predominance over existing evolutionary-based approaches for RDTs construction,
- to the best of our knowledge, it is the first algorithm able to directly optimize *minimax regret* for RDTs,
- it employs novel game theoretic approach for constructing the *Hall of Fame* with *Mixed Nash Equilibrium*,
- the algorithm is easily adaptable to various target metrics,
- by design, CoEvoRDT can be used for potential improvement of results of other SOTA methods by including their resulting DTs in the initial population and optimizing them through coevolution.

Problem Definition

Let $X \subset \mathbb{R}^d$ be a d-dimensional instance space (inputs) and Y be the set of possible classes (outputs). A classical classification task is to find a function (model) $h : X \to Y$, $h(x_i) = y_i$, where y_i is true class of x_i. Classification performance of model h can be measured by accuracy:

$$\mathrm{acc}(h) = \frac{1}{|X|} \sum_{x_i \in X} I[h(x_i) = y_i],$$

where $I[h(x_i) = y_i]$ returns 1 if h predicts the true class of x_i, and 0, otherwise.

Let $\mathcal{N}_\varepsilon(x) = \{z : ||z - x||_\infty \leq \varepsilon\}$ be a ball with center x and radius ε under the L_∞ metric. The **adversarial accuracy** of a model h is accuracy on the perturbation in the perturbation set that produces the lowest accuracy. It is formally defined as

$$\mathrm{acc}_{\mathrm{adv}}(h, \epsilon) = \frac{1}{|X|} \sum_{x_i \in X} \min_{z_i \in \mathcal{N}_\varepsilon(x_i)} I[h(z_i) = y_i].$$

The **max regret** of a model h is the maximum *regret* among all possible perturbations $z \in \mathcal{N}_\varepsilon$. Regret is the difference between the best accuracy possible on a particular perturbation and the accuracy h achieves:

$$\mathrm{regret}(h, \{z_i\}) = \max_{h'} \mathrm{acc}(h', \{z_i\}) - \mathrm{acc}(h, \{z_i\}),$$

where $\mathrm{acc}(h, \{z_i\})$ is the accuracy achieved by h when $\{x_i\}$ is replaced with $\{z_i\}$. Max regret be expressed as:

$$\mathrm{mr}(h) = \max_{z_i \in \mathcal{N}_\varepsilon(x_i)} \mathrm{regret}(h, \{z_i\})$$

.

The problem addressed in this paper is finding a DT trained on X that for a given ε optimizes (maximizes for adversarial accuracy or minimizes for max regret) a given robustness metric (one of the two above-mentioned).

Motivating Example

Consider a financial institution that makes loan acceptance decisions. DTs are well-suited for such high-stakes scenario (Alaradi and Hilal 2020). The dataset of loan applicants in Figure 1 has two features: income I and credit score CS. The system should correctly make a binary credit decision D: accept (1) or reject (0).

For the data in T1, a simple one-node DT can achieve 100% accuracy. One possible decision rule to achieve this is $CS \geq 55$, which we call DT1.

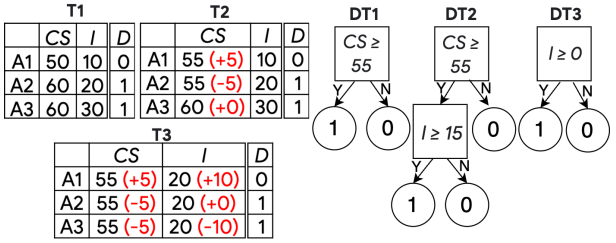

Figure 1: Motivational example – perturbed input data and 3 decision trees.

The features of training data may not be representative of test data due to bugs in the system, inaccurate input data, or distribution shift. Table T2 shows a potential perturbation of the data in T1. In T2, DT1 misclassifies A1 (returning 1 instead of 0). A more robust decision tree, DT2, accurately classifies all applicants in T1 and T2.

T3 is an example with a larger perturbation that affects both income and credit score. In this perturbation, the three applicants have the same features, meaning that no DT can classify them correctly. Thus, the adversarial accuracy against any perturbation set that includes T3 is at most $\frac{2}{3}$. However, achieving such accuracy is easy, any DT that always predicts 1 will do so, including the decision rule $I \geq 0$ (DT3). Consequently, for methods optimizing the adversarial accuracy metric, DT3 is one of the optimal solutions, but it is neither robust nor desired. Maximizing adversarial accuracy myopically focuses on the hardest perturbations in the perturbation set.

From a max regret perspective, DT2 outperforms DT3. Max regret considers not only the worst-case perturbation accuracy, but also the difference between the accuracy of the optimal DT and the robust DT for every perturbation. The regret of DT2 is 0 on all three datasets, resulting in a max regret of 0. DT3 achieves regret of $\frac{1}{3}$ on T1 and T2 and 0 on T3, resulting in a max regret of $\frac{1}{3}$. Thus, under the minimax regret criteria, DT2 would be selected over DT3. Adversarial accuracy loses its ability to distinguish between models as perturbations become large—intuitively good and bad can achieve the same scores.

Related Work

There has been substantial recent work on the construction of robust decision trees. One line of work aims to improve robustness by choosing more appropriate splitting criteria. RIGDT-h (Chen et al. 2019) constructs robust DTs based on

the introduced notion of adversarial Gini impurity, a modification of classical Gini impurity (Breiman 2017) adapted to perturbed input data. This method was further improved in the GROOT algorithm (Vos and Verwer 2021), which mimics the greedy recursive splitting strategy that traditional DTs use and scores splits with the adversarial Gini impurity. The most recent approach, Fast Provably Robust Decision Trees (FPRDT) (Guo et al. 2022) is a greedy recursive approach to a direct minimization of the adversarial loss. It uses the 0/1 loss, rather than traditional surrogate losses such as square loss and softmax loss, as the splitting criterion during the construction of the DT. In experiments, we compare to Guo et al. (2022) and refer the reader to that paper for comparisons with prior methods. Max regret cannot be directly optimized by these methods, which leverage specific properties of adversarial accuracy to design splitting criteria.

Ranzato and Zanella (2021) introduced a genetic adversarial training algorithm (Meta-Silvae) to optimize DT stability, building on a history of using genetic algorithms for DTs Barros et al. (2011), and leveraging the geometric of adversarial accuracy. Our approach utilizes a coevolutionary method and, to the best of our knowledge, it is the first application of this technique to creating robust DTs. At the same time, the effectiveness of coevolutionary algorithms was experimentally proven in many other domains including multi-objective optimization (Meneghini, Guimaraes, and Gaspar-Cunha 2016; Tian et al. 2020), non-cooperative games (Razi, Shahri, and Kian 2007; Żychowski and Mańdziuk 2023), preventing adversarial attacks (O'Reilly and Hemberg 2018), or Generative Adversarial Networks training (Costa, Lourenço, and Machado 2019; Toutouh et al. 2023).

An alternative, exact, approach to robust DTs proposed by Justin et al. (2021) uses a mixed-integer optimization formulation. However, at present, its present applicability is limited to small datasets (approximately less than 3200 samples and/or up to 36 features).

Although, to the best of our knowledge, max regret has not been specifically studied in DTs, it was applied to other domains, e.g., neural network training (Alaiz-Rodrıguez, Guerrero-Curieses, and Cid-Sueiro 2007), reinforcement learning (Azar, Osband, and Munos 2017; Xu et al. 2021), robust planning in uncertain Markov decision processes (Rigter, Lacerda, and Hawes 2021), Security Games (Nguyen et al. 2014), and computing randomized Nash equilibrium (Gilbert and Spanjaard 2017).

CoEvoRDT Algorithm

A general overview of the *Coevolutionary method for Robust Decision Trees* (CoEvoRDT) is presented in Algorithm 1. CoEvoRDT maintains two populations: one contains encoded DTs, and the other contains input data perturbations. Both populations are initialized with random elements and then developed alternately. First, the DT population is modified by evolutionary operators (crossover, mutation, and selection) through l_c generations. Then, the perturbation population is evolved through the same number of l_c generations. The above loop is repeated until the stop condition is satisfied.

Algorithm 1: CoEvoRDT pseudocode.

```
P_T ← InitializeDecisionTreesPopulation()
P_P ← InitializePerturbationsPopulation()
HoF_T = HoF_P = ∅ // HoF - Hall of Fame
N_top = 20

while stop condition not satisfied do
    for 1..l_c do
        P_T ← P_T ∪ Crossover(P_T)
        P_T ← P_T ∪ Mutate(P_T)
        P_T ← Evaluate(P_T, P_P, HoF_P)
        P*_T ← GetElite(P_T)
        while |P*_T| < N_T do
            P*_T ← P*_T ∪ BinaryTournament(P_T)
        end while
        P_T ← P*_T
        T, P ← MixedNashEquilibrium(P_T, P_P)
        HoF_T ← HoF_T ∪ T
        HoF_P ← HoF_P ∪ P
    end for

    for 1..l_c do
        P_P ← P_P ∪ Crossover(P_P)
        P_P ← P_P ∪ Mutate(P_P)
        P_P ← Evaluate(P_P, P_T, HoF_T, N_top)
        P*_P ← GetElite(P_P)
        while |P*_P| < N_P do
            P*_P ← P*_P ∪ BinaryTournament(P_P)
        end while
        P_P ← P*_P
        T, P ← MixedNashEquilibrium(P_T, P_P)
        HoF_T ← HoF_T ∪ T
        HoF_P ← HoF_P ∪ P
    end for
end while

return arg max_{t∈P_T} ξ(t)
```

Decision Tree Population

The DT population contains N_T individuals. Each individual represents one candidate solution (DT) which is encoded as a list of nodes. Each node is represented by a 7-tuple: $node = \{t, c, P, L, R, o, v, a\}$, where t is a node number ($t = 0$ is the root node), c is a class label of a terminal node (meaningful only for terminal nodes), P is a pointer to the parent node, L and R are pointers to the left and right children, respectively (null in a terminal node), o indicates which operator is to be used ($<, >, =$) and v is a real number that indicates the value to be tested on attribute a.

The **initial population** consists of trees generated by randomly choosing attributes and split values, and halting the growth of each DT when the tree reaches a depth randomly selected from an interval $[2, 10]$.

Each individual from the population is selected for **crossover** with probability p_c. Selected individuals are paired randomly, and the crossover operator selects random nodes in two individuals and exchanges the entire subtrees corresponding to each selected node, generating two offspring individuals which are added to the current population.

The **mutation** operator introduces random changes to the

individuals. Each individual is mutated with probability p_m. The mutation operator applies randomly one of the three following actions: (i) replacing a subtree with a randomly generated one, (ii) changing the information in a randomly selected node (setting a new random splitting value v or operator o), (iii) prune a randomly selected subtree. For each mutated individual the mutation is applied 10 times and the highest-fitness individual (among these 10) is added to the current population.

The **evaluation** procedure is performed against the perturbation population (described next). For each individual (a candidate DT) the metric being optimized is computed against all perturbations from the adversarial population. Since the number of perturbations is relatively small any arbitrary chosen metric can be effectively calculated and assigned as an individual's fitness value.

Perturbation Population

The perturbation population consists of N_P individuals $P \subset \mathbb{R}^d$. Each of them represents a perturbed input set X (one perturbation per instance), i.e., $\forall_{x \in X} \exists!_{p_x \in P} : p_x \in \mathcal{N}_\varepsilon(x)$.

The **initial population** contains random perturbations generated by drawing uniformly each perturbation element from the set of possible ones (according to ε criteria).

The **crossover** procedure selects a random subset of individuals (each individual is taken with probability p_c) and pairs them randomly. Then, for each pair, perturbed input instances from both individuals are mixed randomly, i.e., given two crossed parents $c^0 = (x_1^0, \ldots, x_n^0)$ and $c^1 = (x_1^1, \ldots, x_n^1)$ the offspring is $\bar{c}^0 = (x_1^{i_1}, \ldots, x_n^{i_n})$ and $\bar{c}^1 = (x_1^{1-i_1}, \ldots, x_n^{1-i_n})$, where $i_1, \ldots, i_n \in \{0, 1\}$.

Mutation is applied with probability p_m independently to each individual. If a chromosome is selected for mutation, for each input instance and each attribute with probability 0.5 its encoded value is randomly perturbed, i.e. a new random feasible (according to ε constraint) value is assigned.

Evaluation of the perturbation individuals is not an obvious task. On the one hand, assigning the average accuracy versus all DTs in the DT population as a fitness value may be a weak approach. Observe that a given perturbation may be powerful only against a specific though relevant subset of the DTs, and as such should be preserved, but averaging across all DTs will decrease its fitness, posing a risk of omitting it in the selection process. On the other hand, if the perturbation fitness value was computed only against the best DT from the DT population, it would lead to an oscillation of the perturbation population. All perturbations would tend to be efficient for a particular DT, becoming vulnerable to other DTs and losing diversity. Thus, we use $N_{\text{top}} = 20$ highest-fitness DTs (merged with all DTs from HoF) to evaluate each perturbation and perform a targeted optimization with $N_{\text{top}} = 1$ only if we would otherwise terminate. See supplementary material (SM) (Żychowski, Perrault, and Mańdziuk 2023) for experimental justification of the above choice.

Hall of Fame

Hall of Fame (HoF) is a mechanism used to retain and store the best-performing individuals (solutions) that have been encountered during the evolutionary process. By preserving them, the HoF prevents the loss of valuable information and ensures that the best-performing solutions are not discarded during the evolution. The most common approach is to add one of the highest-fitness individuals from each generation (Michalewicz 1996). We find this approach to suboptimal with respect to diversity. Although the HoF stores the best solutions, it can also be used to maintain a diverse set of high-performing individuals. Diversity is essential in evolutionary algorithms to avoid premature convergence, when the algorithm gets stuck in a local optimum and fails to explore better solutions. The HoF can promote diversity by storing solutions that represent different regions of the solution space.

In our coevolutionary approach, HoF is used to assess solutions more accurately. Namely, instead of calculating the fitness function only against individuals from the adversarial population, it is calculated against a merged set of HoF and population individuals.

Instead of adding the highest-fitness individual to the HoF, in CoEvoRDT, we use a game-theoretic approach. Decision trees and perturbation populations can be treated as sets of strategies of two players in a non-cooperative zero-sum game. Then, it is possible to calculate mixed Nash equilibrium. The result is the pair of mixed strategies, i.e., a subset of DTs from the population with assigned probabilities $\mathcal{T} = \{(T_1, p_{T_1}), \ldots, (T_n, p_{T_n})\}$ and a similar subset of perturbations with probabilities $\mathcal{P} = \{(P_1, p_{P_1}), \ldots, (P_m, p_{P_m})\}$. Formally, a mixed Nash equilibrium is a pair $(\mathcal{T}, \mathcal{P})$ such as $\forall_{\mathcal{T}' \neq \mathcal{T}} \xi_T(\mathcal{T}', \mathcal{P}) \preceq \xi_T(\mathcal{T}, \mathcal{P})$ and $\forall_{\mathcal{P}' \neq \mathcal{P}} \xi_P(\mathcal{T}, \mathcal{P}') \preceq \xi_P(\mathcal{T}, \mathcal{P})$ where $\xi_{T|P}(\mathcal{T}, \mathcal{P})$ denotes some objective robustness metrics calculated for a "mixed" decision tree $\mathcal{T}$ and "mixed" perturbation $\mathcal{P}$ (either adversarial accuracy or max regret, in our experiments). Note that this is zero-sum because, in robust optimization, the adversary aims to minimize the DT payoff (i.e., objective function): $\xi_T(\mathcal{T}, \mathcal{P}) = -\xi_P(\mathcal{T}, \mathcal{P})$. We add mixed strategies from Nash equilibria to both HoFs, and they are used in the evaluation process as described above. To evaluate a metric against a mixed object (tree or perturbation), we calculate the expected metric value—first computing the metric for each pair of pure strategies and then taking a weighted average according to the Nash equilibrium probabilities. We limit HoF size by the lowest-fitness element when a fixed maximum size is exceeded.

A similar approach was previously proposed in (Ficici and Pollack 2003) but instead of storing in HoF pure strategies from mixed Nash equilibrium we add mixed strategies. The intuition is that a mixed tree is more robust to diverse perturbations, which has a positive impact on the evolution of perturbations. Similarly, mixed perturbations force the DT population to create more robust DTs that are resistant to a wide spectrum of perturbed data. We demonstrate this in the experiments section.

Selection

The selection process decides which individuals from the current population will be promoted to the next generation. In the beginning, e individuals with the highest fitness value

are unconditionally transferred to the next generation. They are called *elite* and preserve the highest-fitness solutions. Then, a *binary tournament* is repeatedly executed until the next generation population is filled with N individuals. In each tournament, two individuals are sampled (with replacement) from the current population (including those affected by crossover and/or mutation). The higher-fitness chromosome (the winner) is promoted to the next generation with probability p_s (so-called selection pressure parameter). Otherwise, the lower-fitness one is promoted.

Stop Condition

The algorithm ends when at least one of the following conditions is satisfied: (a) CoEvoRDT attains the maximum number of generations (l_g), (b) no improvement of the best-found solution (DT) is observed in consecutive l_c generations. If condition (b) is satisfied, an additional local perturbation improvement subroutine is performed. This procedure is part of the stopping condition and aims to find a better perturbation for the best-fitness decision trees (DTs). Specifically, for each DT with the highest fitness value, the perturbation population evolves using the same process as outlined in lines 21–33 of the Algorithm 1, but with $N_{\text{top}} = 1$ (see line 24). This means that the evaluation for each perturbation is conducted against the DT with current highest fitness. If, for all of those DTs, this routine discovers a perturbation that decreases the fitness of the DT, the counter l_c is reset to zero, and the algorithm execution continues (with $N_{\text{top}} = 20$ and the population's state before local perturbation improvement subroutine execution supplemented with the newly found better perturbation).

To verify conditions (a) and (b) only generations of the DT population are considered. The highest-fitness DT is returned as a CoEvoRDT result.

Convergence

The alternating optimization can be understood as improving candidate DTs while progressively tightening a bound on the robust objective, as any set of perturbation provides a upper (resp., lower bound) on adversarial accuracy (resp., max regret). When $N_{\text{top}} = 1$ (e.g., at convergence), the bound of the objective has been tightened as much as possible for a candidate DT.

Theorem 1. *If both the decision tree and the perturbation population contain an individual that maximizes their fitness against the opposing population, the decision tree in the current population with the highest fitness optimizes the robust objective.*

Proof. Suppose the stop condition is met and fitness is maximized by the decision tree and perturbation populations. We claim that the highest-fitness decision tree h maximizes the robust objective. Suppose that this is not the case. Then, either (i) there exists a perturbation z that would lower h's fitness and is missing from the adversarial population or (ii) there is a decision tree h' that would have higher fitness than h but is missing. (ii) cannot occur because we assume that h maximizes fitness. (i) cannot happen because $N_{\text{top}} = 1$ when the stop condition is met. □

Results and Discussion

Experimental setup. The proposed method was tested on 20 widely used classification benchmark problems of various characteristics – the number of instances, features, and perturbation coefficient. All selected datasets were used in previous studies mentioned in the Related work section, and they are publicly available at https://www.openml.org. Table 1 summarizes their basic parameters.

dataset	ε	Instances	Features	Classes
ionos	0.2	351	34	2
breast	0.3	683	9	2
diabetes	0.05	768	8	2
bank	0.1	1372	4	2
Japan:3v4	0.1	3087	14	2
spam	0.05	4601	57	2
GesDvP	0.01	4838	32	2
har1v2	0.1	3266	561	2
wine	0.1	6497	11	2
collision-det	0.1	33000	6	2
mnist:1v5	0.3	13866	784	2
mnist:2v6	0.3	13866	784	2
mnist	0.3	70000	784	10
f-mnist:2v5	0.2	14000	784	2
f-mnist:3v4	0.2	14000	784	2
f-mnist:7v9	0.2	14000	784	2
f-mnist	0.2	70000	784	10
cifar10:0v5	0.1	12000	3072	2
cifar10:0v6	0.1	12000	3072	2
cifar10:4v8	0.1	12000	3072	2

Table 1: Basic parameters of the benchmark datasets.

The CoEvoRDT parameter values used in the experiments and their selection process is described in detail in the SM. Since there is no straightforward method to calculate the exact values of adversarial accuracy and minimax regret (due to the presence of infinitely many possible perturbations), the results presented below are computed based on a sample of 10^5 random perturbations. The reasoning behind choosing this particular sample size is explained in the SM.

We adopted the Lemke-Howson algorithm (Lemke and Howson 1964) for calculating Mixed Nash Equilibrium from Nashpy Python library (Knight and Campbell 2018). The CART (Breiman 2017) method was used for computing the reference tree for minimax regret (i.e., highest accuracy trees for a particular perturbation). Statistical significance was checked according to the paired t-test with p-value ≤ 0.05. All tests were run on Intel Xeon Silver 4116 @ 2.10GHz. CoEvoRDT source code is available online at https://github.com/zychowskia/CoEvoRDT.

Robustness. CoEvoRDT was trained separately for max regret and adversarial accuracy, and compared with 4 SOTA methods (discussed in the related work section). The results are shown in Tables 2 and 3, respectively. They were also compared with the CART algorithm (Breiman 2017), a popular method for creating DTs for non-perturbed training data (not designed for the RDT scenario). In both tables, the last column (CoEvoRDT+FPRDT) presents the results of adding the FPRDT output DT to the CoEvoRDT initial population, and running CoEvoRDT afterwards. On the max regret metric, CoEvoRDT clearly outperforms all other competitors on all datasets. Adding the FPRDT tree only narrowly improves its outcome. The results support our claim that SOTA methods, which cannot directly minimize max regret, perform

dataset	CART	Meta Silvae	RIGDT-h	GROOT	FPRDT	CoEvoRDT	**CoEvoRDT+FPRDT**
ionos	.094±.000	.075±.007	.071±.006	.061±.005	.061±.006	**.052±.004**	.052±.005
breast	.103±.000	.056±.006	.069±.006	.059±.005	.057±.005	**.049±.004**	.049±.005
diabetes	.202±.000	.126±.008	.132±.009	.124±.009	.117±.007	**.096±.006**	.094±.007
bank	.186±.000	.102±.007	.108±.008	.090±.006	.089±.007	**.076±.006**	.076±.006
Japan3v4	.107±.000	.090±.006	.083±.006	.067±.006	.066±.004	**.062±.006**	.061±.006
spam	.097±.000	.079±.006	.083±.006	.074±.006	.074±.006	**.070±.005**	.069±.005
GesDvP	.152±.000	.129±.008	.133±.010	.129±.008	.131±.009	**.114±.007**	.114±.007
har1v2	.105±.000	.074±.006	.084±.007	.068±.006	.068±.006	**.064±.005**	.064±.005
wine	.140±.000	.125±.008	.127±.009	.111±.009	.109±.008	**.090±.006**	.090±.007
collision-det	.142±.000	.099±.007	.093±.007	.088±.006	.091±.007	**.061±.006**	.059±.006
mnist:1v5	.249±.000	.078±.007	.076±.006	.071±.006	.067±.005	**.055±.006**	.055±.005
mnist:2v6	.268±.000	.083±.007	.087±.006	.072±.005	.069±.005	**.055±.004**	.054±.004
mnist	.395±.000	.143±.009	.139±.009	.125±.007	.124±.009	**.113±.008**	.112±.008
f-mnist2v5	.273±.000	.254±.015	.249±.015	.223±.013	.238±.014	**.196±.011**	.196±.011
f-mnist3v4	.290±.000	.259±.014	.254±.015	.246±.014	.232±.013	**.202±.011**	.199±.011
f-mnist7v9	.283±.000	.255±.014	.251±.015	.237±.014	.240±.014	**.208±.013**	.207±.012
f-mnist	.427±.000	.345±.020	.337±.018	.292±.017	.286±.016	**.238±.014**	.237±.015
cifar10:0v5	.419±.000	.351±.019	.379±.021	.347±.019	.314±.018	**.241±.015**	.236±.013
cifar10:0v6	.403±.000	.362±.021	.368±.020	.342±.018	.341±.019	**.289±.016**	.289±.016
cifar10:4v8	.408±.000	.357±.019	.360±.021	.339±.018	.331±.019	**.283±.016**	.281±.017

Table 2: Max regrets (mean ± std error). CoEvoRDT+FPRDT obtained the best results for all datasets. The best results are bolded. Gray background indicates that a given method is statistically significantly better than all other methods.

dataset	CART	Meta Silvae	RIGDT-h	GROOT	FPRDT	CoEvoRDT	**CoEvoRDT+FPRDT**
ionos	.310±.000	.695±.039	.701±.045	.783±.047	**.795±.047**	.791±.044	.795±.049
breast	.250±.000	.797±.047	.838±.052	.874±.047	.876±.055	**.885±.054**	.889±.056
diabetes	.542±.000	.554±.035	.569±.033	.623±.043	**.648±.039**	.617±.038	.648±.037
bank	.633±.000	.510±.031	.468±.033	.541±.036	**.658±.040**	.657±.043	.663±.037
Japan3v4	.576±.000	.566±.035	.564±.037	.584±.035	**.667±.039**	.665±.037	.668±.037
spam	.302±.000	.637±.036	.467±.028	.723±.045	.746±.049	**.751±.049**	.753±.045
GesDvP	.478±.000	.637±.039	.548±.033	.716±.045	.735±.040	**.740±.046**	.741±.044
har1v2	.232±.000	.706±.045	.707±.047	.806±.048	.804±.049	**.818±.054**	.820±.052
wine	.620±.000	.637±.039	.474±.027	.637±.036	.674±.037	**.688±.046**	.692±.047
collision-det	.743±.000	.772±.047	.764±.044	.784±.052	.792±.051	**.798±.053**	.803±.049
mnist:1v5	.921±.000	.952±.056	.957±.054	.954±.056	**.966±.058**	.964±.059	.969±.061
mnist:2v6	.862±.000	.906±.054	.919±.050	.917±.052	**.922±.049**	.917±.053	.922±.051
mnist	.673±.000	.702±.041	.704±.042	.743±.048	.742±.049	**.745±.043**	.754±.046
f-mnist2v5	.675±.000	.951±.053	.945±.060	.971±.057	.978±.055	**.982±.055**	.982±.059
f-mnist3v4	.632±.000	.808±.049	.793±.044	.819±.048	.865±.050	**.869±.056**	.870±.054
f-mnist7v9	.642±.000	.824±.045	.81±.052	.829±.052	**.876±.050**	.868±.054	.880±.047
f-mnist	.464±.000	.492±.033	.525±.033	.536±.035	.531±.033	**.544±.036**	.546±.040
cifar10:0v5	.296±.000	.502±.033	.347±.026	.485±.036	.678±.046	**.685±.039**	.693±.039
cifar10:0v6	.587±.000	.540±.038	.477±.029	.556±.037	.688±.040	**.692±.046**	.697±.043
cifar10:4v8	.256±.000	.514±.032	.488±.033	.473±.032	.661±.042	**.663±.045**	.664±.037

Table 3: Adversarial accuracies (mean ± std error). CoEvoRDT+FPRDT obtained the best results for all datasets. Box denotes that CoEvoRDT+FPRDT is statistically significantly better than all other methods. The best results are bolded. Gray background indicates that a given method is statistically significantly better than all other methods (except CoEvoRDT+FPRDT).

	minimax regret				adversarial accuracy				computation time [s]			
N	N FPRDT	N CoEvoRDT	CoEvoRDT + N FPRDT	N CoEvoRDT + N FPRDT	N FPRDT	N CoEvoRDT	CoEvoRDT + N FPRDT	N CoEvoRDT + N FPRDT	N FPRDT	N CoEvoRDT	CoEvoRDT + N FPRDT	N CoEvoRDT + N FPRDT
1	.304	.238	.237	.237	.531	.544	.546	.546	19	79	97	97
2	.302	.237	.236	.236	.535	.546	.548	.548	40	161	114	185
3	.301	.237	.236	.236	.539	.548	.549	.550	60	240	134	272
4	.300	.236	.236	.235	.545	.550	.552	.553	80	321	165	362
5	.299	.236	.235	.235	.548	.552	.554	.557	99	406	183	496
10	.293	.234	.233	.233	.552	.557	.559	.562	195	774	264	939
20	.284	.230	.230	.229	.558	.564	.566	.568	391	1553	454	1869
50	.282	.229	.229	.227	.563	.568	.568	.569	956	3863	999	4564
100	.282	.228	.229	.227	.566	.568	.568	.569	1921	7981	1990	9026

Table 4: Best results of repeated N algorithms' runs for fashion-mnist dataset. In CoEvoRDT + N FPRDT output DTs from N FPRDT independent runs were incorporated into CoEvoRDT initial population.

	minimax regret					adversarial accuracy					computation time [s]				
HoF size	Nash mixed tree	Top K as mixed tree	Nash single trees	Top K	Best	Nash mixed tree	Top K as mixed tree	Nash single trees	Top K	Best	Nash mixed tree	Top K as mixed tree	Nash single trees	Top K	Best
0	.261	.261	.261	.261	.261	.533	.533	.533	.533	.533	47	47	47	47	47
10	.242	.248	.247	.251	.259	.535	.535	.535	.534	.533	50	50	50	50	50
20	.240	.246	.245	.249	.256	.536	.536	.536	.536	.534	55	54	55	56	51
50	.241	.244	.245	.249	.254	.536	.536	.536	.536	.534	61	58	59	62	54
100	.239	.243	.243	.247	.253	.538	.538	.537	.537	.535	68	63	66	65	56
200	.238	.242	.242	.244	.250	.543	.539	.540	.539	.535	77	70	76	77	59
500	.237	.241	.241	.243	.248	.545	.540	.540	.540	.536	86	79	91	90	60
∞	.237	.239	.240	.240	.248	.545	.540	.541	.540	.536	86	77	85	85	61

Table 5: Results with respect of HoF size for fashion-mnist dataset. ∞ means that there was no limit on HoF size.

significantly worse than CoEvoRDT in terms of this metric. From an adversarial accuracy perspective, for 13 out of 20 datasets, CoEvoRDT yielded the best mean results (5 of them statistically significant). For the remaining 7 datasets FPRDT method was superior (with statistical significance in 3 cases). For this metrics, adding FPRDT tree to the initial CoEvoRDT population (CoEvoRDT+FPRDT) led to clear advantage versus baseline CoEvoRDT and FPRDT alone.

For a more detailed analysis, fashion-mnist dataset is selected as one of the largest. Detailed results for all other datasets are presented in the SM.

Runtime comparison. In general, CoEvoRDT runtime varies from a few seconds to a couple of minutes for the largest datasets. The average computation time of a single run of the strongest competitor, FPRDT is 2 to 8 times lower than CoEvoRDT. Thus, the natural question which can arise is what if we run FPRDT multiple times to have an equal computation budget and choose the best result. This approach is addressed in Table 4, which presents computation time and the best results in terms of minimax regret and adversarial accuracy for multiple runs of FPRDT, CoEvoRDT, and CoEvoRDT initialized with multiple FPRDT outcomes. More runs can notably improve results for all methods. For max regret, even within 100 repeats, FPRDT was not able to find a solution close to the single CoEvoRDT outcome. For adversarial accuracy multiple FPRDT runs outperformed CoEvoRDT, but for greater numbers of repeats (above 20) both methods seem to converge to similar results. Given some constrained computation budget the best option is to run CoEvoRDT + N FPRDT.

HoF size and construction. Table 5 presents the results for various HoF sizes. For each size, 5 variants of constructing HoF are considered: adding one *mixed tree* from mixed Nash equilibrium after each generation (which is the baseline used in CoEvoRDT), adding all *single trees* from mixed Nash equilibrium (i.e., all the pure strategies with positive probability), adding only one highest-fitness tree from the population, adding top K highest-fitness individuals from the current population (where K is the number of trees from Nash mixed equilibrium) and adding one mixed tree composed of top K highest-fitness trees with equal probabilities. Firstly, it is clear that even small HoF significantly improves results for all variants. Moreover, adding a Nash mixed tree seems to be the best option, while adding only one highest-fitness tree is the worst approach. The advantage of Nash mixed tree and top K as a mixed tree with equal probabilities over Nash single trees and top K trees shows the benefit of using mixed trees in the HoF. It may stem from the fact that mixed tree is *more robust* to various perturbations, and consequently the perturbation population is forced to find better perturbations to *outplay* those mixed trees. As a result, the DT population is forced to create even more robust trees. At the same time, the straightforward approach of creating a mixed tree of highest-fitness individuals is less powerful than a mixed tree from mixed Nash equilibrium.

The **generation limit** of CoEvoRDT was set to 1000, but in practice, it was rarely reached and the other stop condition (no improvement of best-found solution) was fulfilled first. The average number of generations across all datasets was 385. The lowest average number was observed for the diabetes (152), and the highest for cifar10:0v5 (865). The depth of DTs generated by CoEvoRDT varies from 6 to 23 which is not much different than DTs created by other methods.

CoEvoRDT **memory consumption** is low and does not exceed 150 MB for the largest datasets.

Conclusions

In this paper, we present CoEvoRDT, a novel coevolutionary algorithm designed to construct robust decision trees capable of handling perturbed high-dimensional data. Our motivation stems from the vulnerability of traditional DT algorithms to adversarial perturbations and the limitations of existing defensive algorithms in optimizing specific metrics like max regret. The flexibility of CoEvoRDT in accommodating various target metrics makes it adaptable to a wide range of applications and domains, including when robustness is mixed with other objectives such as fairness. We propose a novel game-theoretic approach to constructing the Hall of Fame with Mixed Nash Equilibrium, which significantly contributes to the DTs robustness and convergence speed. CoEvoRDT can additionally integrate results from another strong and fast method into the initial population, if one is available, to further improve performance.

CoEvoRDT was comprehensively tested on 20 popular benchmark datasets and compared with 4 SOTA algorithms, presenting on par performance to the best competitive method in adversarial accuracy metrics, and outperforming all competitors in terms of minimax regret.

Our future work focuses on investigating the potential of implementing CoEvoRDT as a multi-population algorithm, such as the island model (Skolicki 2005), to speed up convergence and potentially further boost its performance.

Acknowledgements

We gratefully acknowledge the funding support by program "Excellence initiative—research university" for the AGH University of Krakow as well as the ARTIQ project: UMO-2021/01/2/ST6/00004 and ARTIQ/0004/2021.

Adam Żychowski was funded by the Warsaw University of Technology within the Excellence Initiative: Research University (IDUB) programme.

References

Aghaei, S.; Azizi, M. J.; and Vayanos, P. 2019. Learning optimal and fair decision trees for non-discriminative decision-making. In *AAAI Conference on Artificial Intelligence*, volume 33, 1418–1426.

Aitkenhead, M. J. 2008. A co-evolving decision tree classification method. *Expert Systems with Applications*, 34(1): 18–25.

Alaiz-Rodrıguez, R.; Guerrero-Curieses, A.; and Cid-Sueiro, J. 2007. Minimax regret classifier for imprecise class distributions. *Journal of Machine Learning Research*, 8: 103–130.

Alaradi, M.; and Hilal, S. 2020. Tree-based methods for loan approval. In *2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI)*, 1–6. IEEE.

Azar, M. G.; Osband, I.; and Munos, R. 2017. Minimax regret bounds for reinforcement learning. In *International Conference on Machine Learning*, 263–272. PMLR.

Barros, R. C.; Basgalupp, M. P.; De Carvalho, A. C.; and Freitas, A. A. 2011. A survey of evolutionary algorithms for decision-tree induction. *IEEE Transactions on Systems, Man, and Cybernetics*, 42(3): 291–312.

Breiman, L. 2017. *Classification and regression trees*. Routledge.

Chakraborty, A.; Alam, M.; Dey, V.; Chattopadhyay, A.; and Mukhopadhyay, D. 2021. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, 6(1): 25–45.

Chen, H.; Zhang, H.; Boning, D.; and Hsieh, C.-J. 2019. Robust decision trees against adversarial examples. In *International Conference on Machine Learning*, 1122–1131. PMLR.

Costa, V.; Lourenço, N.; and Machado, P. 2019. Coevolution of generative adversarial networks. In *Applications of Evolutionary Computation*, 473–487. Springer.

Ficici, S. G.; and Pollack, J. B. 2003. A game-theoretic memory mechanism for coevolution. In *Genetic and Evolutionary Computation Conference*, 286–297. Springer.

Gilbert, H.; and Spanjaard, O. 2017. A double oracle approach to minmax regret optimization problems with interval data. *European Journal of Operational Research*, 262(3): 929–943.

Grosse, K.; Manoharan, P.; Papernot, N.; Backes, M.; and McDaniel, P. 2017. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*.

Guo, J.-Q.; Teng, M.-Z.; Gao, W.; and Zhou, Z.-H. 2022. Fast Provably Robust Decision Trees and Boosting. In *International Conference on Machine Learning*, 8127–8144. PMLR.

Jo, N.; Aghaei, S.; Benson, J.; Gómez, A.; and Vayanos, P. 2022. Learning Optimal Fair Classification Trees. *CoRR*, abs/2201.09932.

Justin, N.; Aghaei, S.; Gomez, A.; and Vayanos, P. 2021. Optimal robust classification trees. In *The AAAI-22 Workshop on Adversarial Machine Learning and Beyond*.

Kantchelian, A.; Tygar, J. D.; and Joseph, A. 2016. Evasion and hardening of tree ensemble classifiers. In *International Conference on Machine Learning*, 2387–2396. PMLR.

Knight, V.; and Campbell, J. 2018. Nashpy: A Python library for the computation of Nash equilibria. *Journal of Open Source Software*, 3(30): 904.

Lemke, C. E.; and Howson, J. T., Jr. 1964. Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics*, 12(2): 413–423.

Mahdavi, S.; Shiri, M. E.; and Rahnamayan, S. 2015. Metaheuristics in large-scale global continues optimization: A survey. *Information Sciences*, 295: 407–428.

Meneghini, I. R.; Guimaraes, F. G.; and Gaspar-Cunha, A. 2016. Competitive coevolutionary algorithm for robust multi-objective optimization: The worst case minimization. In *IEEE Congress on Evolutionary Computation (CEC)*, 586–593. IEEE.

Michalewicz, Z. 1996. *GAs: Why Do They Work?* Springer.

Nguyen, T.; Yadav, A.; An, B.; Tambe, M.; and Boutilier, C. 2014. Regret-based optimization and preference elicitation for Stackelberg security games with uncertainty. In *AAAI Conference on Artificial Intelligence*, volume 28.

O'Reilly, U.-M.; and Hemberg, E. 2018. An Artificial Coevolutionary Framework for Adversarial AI. In *AAAI Fall Symposium: ALEC*, 50–55.

Ranzato, F.; and Zanella, M. 2021. Genetic adversarial training of decision trees. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 358–367.

Razi, K.; Shahri, S. H.; and Kian, A. R. 2007. Finding Nash equilibrium point of nonlinear non-cooperative games using coevolutionary strategies. In *International Conference on Intelligent Systems Design and Applications*, 875–882. IEEE.

Rigter, M.; Lacerda, B.; and Hawes, N. 2021. Minimax regret optimisation for robust planning in uncertain Markov decision processes. In *AAAI Conference on Artificial Intelligence*, volume 35, 11930–11938.

Savage, L. J. 1951. The theory of statistical decision. *Journal of the American Statistical association*, 46(253): 55–67.

Skolicki, Z. 2005. An analysis of island models in evolutionary computation. In *Annual Workshop on Genetic and Evolutionary Computation*, 386–389.

Tian, Y.; Zhang, T.; Xiao, J.; Zhang, X.; and Jin, Y. 2020. A coevolutionary framework for constrained multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 25(1): 102–116.

Toutouh, J.; Nalluru, S.; Hemberg, E.; and O'Reilly, U.-M. 2023. Semi-Supervised Learning with Coevolutionary Generative Adversarial Networks. In *Genetic and Evolutionary Computation Conference*, 568–576.

Vos, D.; and Verwer, S. 2021. Efficient training of robust decision trees against adversarial examples. In *International Conference on Machine Learning*, 10586–10595. PMLR.

Xu, L.; Perrault, A.; Fang, F.; Chen, H.; and Tambe, M. 2021. Robust reinforcement learning under minimax regret for green security. In *Uncertainty in Artificial Intelligence*, 257–267. PMLR.

Zhang, C.; Zhang, H.; and Hsieh, C.-J. 2020. An efficient adversarial attack for tree ensembles. *Advances in Neural Information Processing Systems*, 33: 16165–16176.

Żychowski, A.; and Mańdziuk, J. 2023. Coevolution of players strategies in security games. *Journal of Computational Science*, 68: 101980.

Żychowski, A.; Perrault, A.; and Mańdziuk, J. 2023. Coevolutionary Algorithm for Building Robust Decision Trees under Minimax Regret. arXiv:2312.09078.