Efficient Explanations for Rule Ensembles

Hao Hu
University of Lleida, Lleida, Spain

Alexey Ignatiev
Monash University, Melbourne, Australia

Joao Marques-Silva
ICREA, University of Lleida, Lleida, Spain

Abstract

Tree ensembles (TEs) are among the most widely used machine learning models, yet explaining their predictions remains a computational challenge. Recent work in formal explainable AI (FXAI) has focused on computing abductive explanations for TEs using Boolean satisfiability (SAT) and maximum satisfiability (MaxSAT). However, these methods often fail to scale with the growth of the number and depth of the trees in the ensemble. This paper addresses these scalability limitations by shifting focus to Rule Ensembles (REs), a structurally simpler alternative to TEs. We make three primary contributions. First, we adapt an existing MaxSAT-based explanation framework designed for TEs to function with general REs. Second, we devise a dedicated logic encoding for REs combining SAT solving with pseudo-Boolean (PB) constraints for determining the winning class. Finally, empirical experiments on standard tabular and image datasets demonstrate a significant advantage of the proposed SAT-based approach for REs over the state-of-the-art MaxSAT-based approach for TEs.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming; Computing methodologies → Artificial intelligence

Keywords and phrases Explainable Artifical Intelligence, Rule Ensembles, Boolean Satisfiability

Digital Object Identifier 10.4230/LIPIcs.CP.2026.29

Supplementary Material *Software (Source code)*: `https://github.com/nwpuhh/yreason`
archived at `swh:1:dir:b2545180c7325181a959cca2493c8797254d0734`

Funding This work is supported by Spanish Government the grant PID2023-152814OB-I00.

Acknowledgements The authors express their sincere thanks to all reviewers for their supportive questions and suggestions.

1 Introduction

One main goal of Explainable AI (XAI) is to help human decision makers in understanding the decisions made by systems of AI. Broadly, three main XAI approaches have been studied in the past. (i) the adoption of intrinsically *interpretable ML models* [21, 20]; (ii) *model-agnostic* post-hoc explanations that do not exploit the ML model's computed function and only exploit sampling of the model; and (iii) *model-aware* post-hoc explanations (or *formal* explanations) that exploit's the model's computed function for reasoning purposes, and so formally represent the model's semantics. Among these approaches, so-called interpretable ML models are generally limited in predictive performance and existing results substantiate the need for post-hoc explanations [34, 46]. Moreover, model-agnostic explanations are well-known to produce untrustable results [26, 53, 25]. In contrast, formal explanations are logically rigorous, given the ML model, but exhibit the downside of lack of scalability of automated reasoning. In recent years, formal explanations have been applied to decision trees [34] and tree ensembles [27, 2, 3, 35], among others [42, 43, 15]. Tree ensembles (TEs),

including boosted trees [12, 18, 17] and random forests [10], attract considerable interest because of their solid predictive performance. However, tree ensembles require numerous trees, often with considerable depths, to achieve high degrees of generalization. As a result, for such tree ensembles, finding formal explanations becomes a very challenging computational problem.

One solution studied in this paper is to consider alternative models, that are are tightly related, that perform similarly in practice, but for which finding formal explanations is an easier computational problem in practice. Concretely, the paper considers rule ensembles instead of tree ensembles. In fact, a tree ensemble can be viewed as a rule ensemble of the union of decision rule sets, where each set of rules is generated by iterating over all paths of each tree in the tree ensemble. Therefore, the number of unique decision rules measures the size of a tree/rule ensemble. While it is plain that the worst-case size of a tree ensemble grows exponentially with the increase of tree depth, the maximum size of a rule ensemble is a preset parameter and the length of each rule (the number of conditions) has no influence to the size. The overall size of a rule ensemble serves as a principal reason for the issues of scalability when computing formal explanations.

In this paper, we adopt an existing MaxSAT-based explanation framework designed for tree ensembles [27] to general rule ensembles, for computing subset-minimal explanations. One of the key difficulties in the MaxSAT-based approach is the need of optimality in determining the winning class. Therefore, to further improve the efficiency, we devise a dedicated logic encoding for rule ensembles in the winning class determination by exploiting pseudo-Boolean (PB) constraints. Similar to the MaxSAT-based approach, we use the well-known incremental assumption-based SAT solving [16] to avoid regeneration of propositional formulas during the explanation extraction. The empirical results at first confirm the fact that, compared to tree ensembles, rule ensembles yield significant reduction in model size while attaining competitive prediction quality with reduced floating-point precision. Similarly, the experimental results on standard tabular datasets show the efficiency improvement by lifting tree ensembles to rule ensembles of model sizes smaller than one hundred trees. Furthermore, the results on MNIST image dataset demonstrate the significant advantage of the proposed SAT-based approach for rule ensembles of very large model sizes (around 20 thousand rules). Our results indicate that, on average, the state-of-the-art MaxSAT-based approach for TEs needs more than 3000 seconds to explain an instance. In contrast, the novel proposed SAT-based approach for explaining REs depends on the accuracy assumed, with average running times around 1000 seconds for the more accurate models, and 25 seconds for the less accurate models.

The paper is organized as follows. Section 2 introduces the notation and definitions used throughout the paper. Section 3 details the novel encodings devised for finding formal explanations for rule ensembles. Section 4 analyzes the experimental results, and validates the performance gains achieved by adopting rule ensembles, and using the encodings proposed in this paper. Section 5 concludes the paper.

2 Technical Background

SAT, MaxSAT, PB Constraints

The paper adopts standard terminology [8] for Boolean satisfiability (SAT) [45], maximum satisfiability (MaxSAT) [40, 6], and Pseudo-Boolean (PB) constraints [51]. A *literal* is a Boolean variable or its negation and a *clause* is a disjunction of literals. For convenience, clauses are noted as a set of literals. A propositional formula is said to be a *conjunctive*

normal form (CNF) when it is a conjunction of clauses. A truth assignment α, maps each variable to $\{0, 1\}$. Given a truth assignment, a clause is satisfied if at least one of its literals is assigned value 1, otherwise it is falsified. A CNF is satisfied if all of its clauses are satisfied, otherwise it is falsified. The decision problem for propositional logic (SAT) is to decide whether there exists a truth assignment to satisfies the given CNF formula. If no such truth assignment exists, the CNF formula is *unsatisfiable*.

In this paper, we focus on *partial weighted maximum satisfiability* (MaxSAT). A formula ϕ is a pair of CNF formulas $\phi = (\mathcal{H}, \mathcal{S})$, where $\mathcal{H}$ denotes the *hard clauses* and $\mathcal{S}$ denotes the *soft clauses*, and such that $\mathcal{H} \land \mathcal{S}$ is unsatisfiable. In brief, $\mathcal{H}$ must be satisfied, and $\mathcal{S}$, each clause corresponds to a weight, are not restricted to be satisfied. The partial weighted MaxSAT problem finds a truth assignment that maximizes the sum of weights of the satisfied soft clauses, while satisfying all the hard clauses.

Pseudo-Boolean (PB) constraints restrict a weighted sum of Boolean variables using a linear inequality. A normalized *linear PB constraint* has the following form: $\sum_j a_j l_j \geq b$, where a_j is the integer coefficient for literal l_j and b is an integer called as the degree of the constraint. There are numerous efficient encodings of PB constraints into CNF [7, 1, 47].

Classification Problems, Tree Ensembles, Rule Ensembles

Classification problems are defined on a set of features as $\mathcal{F} = \{1, \ldots, M\}$ and a set of classes $\mathcal{K} = \{c_1, \ldots, c_K\}$. Each feature $j \in \mathcal{F}$ is characterized by a domain D_j. Accordingly, feature space is defined as the cartesian product of the domains of the features, $\mathbb{F} = D_1 \times D_2 \times \cdots \times D_M$. We use $\mathbf{x} = (x_1, \ldots, x_M)$ as a generic point in feature space, where $x_j \in D_j$ indicates the value of feature j. A classifier implements a classification function $\tau : \mathbb{F} \to \mathcal{K}$. The prediction of a classifier for $\mathbf{x}$ is denoted as $\tau(\mathbf{x})$. An instance (or an example) is denoted as a pair $(\mathbf{v}, c_p)$, where $\mathbf{v} = (v_1, \ldots, v_M)$ is a specific point in feature space and $c_p \in \mathcal{K}$ indicates the predicted class by a classifier, i.e $c_p = \tau(\mathbf{v})$.

To improve the predictive performance of a single ML model, ensemble methods train multiple models and then combine them [54]. There are two major categories: Boosting methods (that includes AdaBoost [17], GBDT [18], XGBoost [12], among others) and Bagging methods (that includes Bagging [9], Random Forests [10], among others). This paper focuses on tree ensembles (TEs) trained with the XGBoost algorithm and rule ensembles (REs) trained with the Boomer algorithm [50, 49]. [1] Both XGBoost and Boomer employ *gradient boosting methods* that train multiple models sequentially, where in each stage, new models are generated to compensate for the inaccuracies of the constructed model ensembles. We introduce the two models for multi-classification problem separately.

XGBoost uses *regression decision trees* [11]. A regression (or a decision) tree $\mathcal{T}_i$ is a directed acyclic graph composed of a set of paths $\mathcal{P}_i$. A path $p_r \in \mathcal{P}_i$ is composed of multiple non-terminal nodes, each representing a condition on some feature and a single final *terminal node* t_r tagged with a real-valued weight. This weight is predicted in the conjunction of the conditions in the path are satisfied. For the purposes of tree ensembles, and for a non-terminal node, the condition represented on the value of a feature $j \in \mathcal{F}$ is in the form $x_j < d, d \in D_j$, where d is the *splitting threshold*. A tree computes a function $\mathcal{T}_i : \mathbb{F} \to \mathbb{R}$, that maps an instance $\mathbf{x} \in \mathbb{F}$ to the weight associated to the leaf lead to. For each class $i \in \{1, \ldots, K\}$, the tree ensemble TE contains $N \in \mathbb{N}_{>0}$ trees, which are represented as

[1] Throughout, tree ensembles will be denoted by TE and rule ensembles by RE. A specific instantiation of a TE will be denoted by $\mathfrak{Te}$, and a specific instantiation of a RE will be denoted by $\mathfrak{Re}$.

$\mathcal{T}_{Kj+i}, j \in \{0,\ldots,N-1\}$. Each tree contributes a weight, and the overall prediction of a TE is the class with the largest total weight. Therefore, the sum of weights for class i $w_i^t : \mathbb{F} \to \mathbb{R}$, for an instance $\mathbf{x} \in \mathbb{F}$, is defined as follow:

$$w_i^t(\mathbf{x}) = \sum_{j\in\{0,\ldots,N-1\}} \mathcal{T}_{Kj+i}(\mathbf{x}), \quad i \in \{1,\ldots,K\}$$

In contrast to XGBoost, Boomer uses *decision rules* [13, 14]. A rule is composed of a *body*, i.e. a conjunction of literals, and a *head*, representing the rule's outcome when the conjunction of literals holds true. Observe that a decision rule $\mathcal{R}_i$ is analogous to a path in a regression tree, in that it contains multiple conditions and predicts some *head* h_i with multiple real weights (or a single weight). Each condition relates the value of a feature $j \in \mathcal{F}$ with some value and is of the form $x_j \circ d, \circ \in \{\leq, >\}, d \in D_j$. For the case of a rule ensemble (RE), there exists a *default rule* $\mathcal{R}_0$ with a head of K default weights for each class, and $N \in \mathbb{N}_{>0}$ rules for each class $i \in \{1,\ldots,K\}$, which are represented as $\mathcal{R}_{Kj+i}, j \in \{0,\ldots,N-1\}$. Each rule contributes a weight, and the overall decision of a given RE is the class with the largest total weight. We denote w_i^{r0} for the weight of class i in the default rule $\mathcal{R}_0$. Meanwhile, $\mathcal{R}_{Kj+i} : \mathbb{F} \to \mathbb{R}$ is denoted as the function mapping the an instance to the weight for class i, which return the weight attached to the head if the instance satisfies all conditions, or 0 if it fails. Then, the sum of weights for class i $w_i^r : \mathbb{F} \to \mathbb{R}$, for an instance $\mathbf{x} \in \mathbb{F}$, is defined as follow:

$$w_i^r(\mathbf{x}) = w_i^{r0} + \sum_{j\in\{0,\ldots,N-1\}} \mathcal{R}_{Kj+i}(\mathbf{x}), \quad i \in \{1,\ldots,K\}$$

▶ **Example 1.** An example of a TE (resp. RE) for the Iris dataset is shown in Fig 1(a) (resp. Fig 1(b)). There are 4 numeric features and 3 classes, each class has 2 trees (resp. rules) in the TE (resp. RE). Given an instance ((sepal.length=5.1) ∧ (sepal.width=3.5) ∧ (petal.length=1.4) ∧ (petal.width=0.2), c_1), for the TE, the scores of the 3 classes are: $w_1^t = \mathcal{T}_1 + \mathcal{T}_4 = 0.42604 + 0.29323 = 0.71927$, $w_2^t = \mathcal{T}_2 + \mathcal{T}_5 = -0.40254$, $w_3^t = \mathcal{T}_3 + \mathcal{T}_6 = -0.41410$, which predicts c_1 (setosa). For the RE, $w_1^r = w_1^{r0} + \mathcal{R}_1 + \mathcal{R}_4 = -0.31356 + 0.38446 + 0 = 0.07090$, $w_2^r = w_2^{r0} + \mathcal{R}_2 + \mathcal{R}_5 = -0.50248$, $w_3^r = w_3^{r0} + \mathcal{R}_3 + \mathcal{R}_6 = -0.38136$, which predicts c_1 (setosa) too.

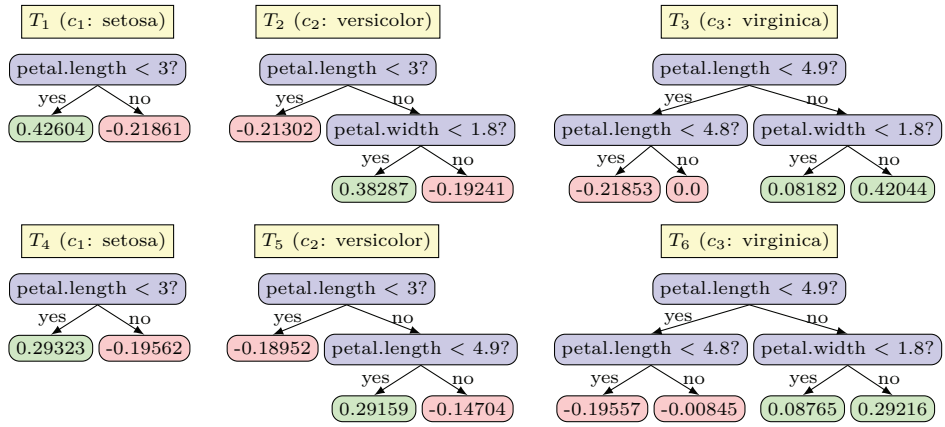
Interpretability, Abductive Explanations

A rigorous definition of interpretability has proven elusive [41]. In this paper, we associate interpretability with the succinctness of explanations obtained from the ML model. Furthermore, we study *abductive explanations* (AXps) [52, 30, 33, 23, 22], i.e. subset-minimal sets of feature-value pairs that are sufficient for the given prediction. Formally, for a given instance $(\mathbf{v}, c_p)$, an AXp is a minimal subset $\mathcal{X} \subseteq \mathcal{F}$ such that:

$$\forall(\mathbf{x} \in \mathcal{F}).\ [\bigwedge_{j\in\mathcal{X}} (x_j = v_j)] \to (\tau(\mathbf{x}) = c_p)$$

AXps are also called *prime implicant* (PI) explanations in [52]. We use the acronym AXp throughout the paper.

▶ **Example 2.** For both the TE and RE of Example 1, and for the instance considered, one can observe that the assignment (petal.length=1.4) suffices for guaranteeing the prediction c_1 (setosa). Therefore, an AXp for both the TE and the RE given this particular instance is $\{petal.length\}$, which represents the literal $\{(petal.length = 1.4)\}$.



(a) A simple TE trained via XGBoost.

R_0 : **DEFAULT** c_1 : setosa = -0.31356, c_2 : versicolor = -0.29661, c_3 : virginica = -0.38136
R_1 : **IF** petal.length <= 2.45 **THEN** c_1 : setosa = 0.38446
R_2 : **IF** petal.length <= 2.45 **THEN** c_2 : versicolor = -0.20587
R_3 : **IF** petal.width > 1.75 **THEN** c_3 : virginica = 0.38542
R_4 : **IF** petal.length > 2.45 **THEN** c_1 : setosa = -0.20329
R_5 : **IF** petal.length > 5.05 **AND** sepal.width > 2.75 **THEN** c_2 : versicolor = -0.20348
R_6 : **IF** petal.width > 1.75 **THEN** c_3 : virginica = 0.27310

(b) A simple RE trained via Boomer.

Figure 1 Illustrative example of a TE and a RE for the well known Iris dataset.

MaxSAT-based Explanations for Tree Ensembles

An often-used algorithm for finding one AXp for a TE [27] is the well-known deletion-based algorithm [30].[2] The operation of the deletion-based algorithm can be summarized as follows. For a given TE, the algorithm computes an explanation $\mathcal{X}$ by iterating through the set of features $\mathcal{F}$. For each feature j, the algorithm checks whether $\mathcal{X} \setminus \{j\}$ entails the predicted class c_p. If it does, then feature j is irrelevant and it is removed. Otherwise, feature j must be kept in $\mathcal{X}$. The correctness of the deletion-based algorithm results from the definition of the predicate WAXp being monotonic with respect to its argument.

A key aspect of the approach proposed in [27] is that it uses $(K-1)$ MaxSAT oracles to decide entailment, i.e. a set of fixed features suffices for the prediction. Each MaxSAT oracle call checks whether an arbitrary class $c_l, l \in \{1, \ldots, K\}, l \neq p$ has larger total weight than the total weight of c_p. If any such oracle call does, then the entailment check fails. Otherwise, the entailment check succeeds.

We briefly introduce some necessary notation, as follows: for a given TE, m_j indicates the number of threshold splits of feature j in all trees in TE, $s_{j,k}$ denotes the k-th split threshold of feature j in the sorted order indicating $x_j < s_{j,k}$. In total, there are m_j+1 disjoint intervals, with the k-th interval being $I_k = [s_{j,k-1}, s_{j,k})$, where $k \in [m_j + 1]$ ($I_1 = [\min(D_j), s_{j,1})$

[2] Other variants have been studied [32], including the QuickXplain [36], dichotomic search [19] and progression [44] algorithms.

and $I_{m_j+1} = [s_{j,m_j}, \max(D_j)]$). Each auxiliary Boolean variable $l_{j,k}, j \in \mathcal{F}, k \in [m_j + 1]$ is associated with the interval I_k, where $l_{j,k} = 1$ iff $x_j \in I_k$. For a given instance $(\mathbf{v}, c)$, a slight abuse of notation is $l[\![x_j = v_j]\!] \equiv l_{j,k}$ iff $v_j \in I_k$. Moreover, the Boolean variables $o_{j,k}, k \in [m_j]$ are associated with the upper bound, i.e. $o_{j,k} = 1$ iff $x_j < s_{j,k}$. We can use literals $o_{j,k}$ ($\neg o_{j,k}$, resp) to enforce the 'yes' (resp. 'no') branch of the node $x_j < s_{j,k}$. Furthermore, Boolean variable t_r indicates the leaf node corresponding to path $p_r \in \mathcal{P}_i$ in tree $\mathcal{T}_i$, is reached ($t_r = 1$) or not ($t_r = 0$), and w_r indicates the corresponding weight when $t_r = 1$. In the case of propositional logic, the total weight of class c_i can be represented as the constraint $\sum_i = \sum_{\mathcal{T}_{Kj+i} \in \mathfrak{T}_\mathfrak{e}} \sum_{p_r \in \mathcal{P}_{Kj+i}} (w_r \cdot t_r)$; the notation $\sum_i$ is used for simplicity.

The MaxSAT encoding for c_l consists of two major parts: (1) Hard constraints on the tree structure for trees in TE, containing the constraints encoding the valid domain relationship, constraints relating the path and its corresponding leaf, and constraints enforcing only one leaf selected for each tree. (2) Soft constraints for the $\sum_l - \sum_p$ with current explanation $\mathcal{X}$. From the Corollary 1 in [27], to succeed the entailment check, the optimal value must be negative, i.e. $\sum_l < \sum_p$, for all $l \neq p$. If for some l, there is a truth assignment such that $\sum_l \geq \sum_p$, then entailment is no longer preserved.

3 Efficient AXp Extraction for Boosted Rules

Although the logic encodings developed to TEs [27] can be used for REs, the simpler structure of REs enables devising a novel and simpler logic encoding. This section details the new encodings. Furthermore, the section also details optimizations that have been adopted, which further improve performance.

3.1 Adapting MaxSAT Encoding for Rule Ensembles

Boolean Variables

As noted earlier, by construction each literal in the body of decision rules is of the form $x_j \circ d$, with $\circ \in \{\leq, >\}$. Accordingly, the disjoint interval needs be changed as $I^r_k = (s_{j,k-1}, s_{j,k}]$. The auxiliary Boolean variables $l_{j,k}$ is also used to name these intervals, i.e. $l_{j,k} = 1$ iff $x_j \in I^r_k$. Meanwhile, the Boolean variables $o_{j,k}$ are redefined as $o_{j,k} = 1$ iff $x_j \leq s_{j,k}$, so as to align with the conditions appearing in decision rules. Instead of variables t_r indicating the leaf is reached or not, we propose Boolean variables $h_r, r \in [K \cdot N]$ indicating the head of an arbitrary r-th rule is reached ($h_r = 1$) or not ($h_r = 0$). Note that if multiple rules share the same series of conditions, a single h_r represents all such occurrences. In particular, h_0 is specially for the default rule $\mathcal{R}_0$, which is always assigned as 1 as the head of the default rule is always reached.

Necessary Constraints

We adapt the previously introduced MaxSAT encoding for TEs to REs. For the hard constraints, we retain the constraints capturing the valid domain relationship between intervals made by the RE. Rather than relating the all paths and their corresponding leaves in TE, as in REs, for each rule, we associate the sequence of conditions to the head variable of that rule. For an arbitrary rule $\mathcal{R}_r$, we slightly abuse the notation p_r as the sequence of conditions appearing in $\mathcal{R}_r$, the constraint is described as follow:

$$\Big(\bigwedge_{x_j \leq s_{j,k} \in p_r} o_{j,k} \wedge \bigwedge_{x_j > s_{j,k} \in p_r} \neg o_{j,k}\Big) \leftrightarrow h_r \tag{1}$$

Moreover, the constraints enforcing only one leaf selected for each tree in TEs are unnecessary for REs. For brevity, we denote the set of hard constraints by $\mathcal{H}$.

▶ **Example 3.** For the RE shown in Fig 1(b), there are 2 thresholds for feature petal.length $\{2.45, 5.05\}$ and 2 thresholds for feature patal.width $\{1.7, 2.75\}$. Assuming petal.length is the 3rd feature and petal.width is the 4-th one, we introduce variables $o_{3,k}, k \in [2]$ and $o_{4,k}, k \in [2]$. As an example, $o_{3,2} \leftrightarrow (x_3 \leq 5.05)$, $\neg o_{4,1} \leftrightarrow (x_4 > 1.7)$. Moreover, we note that $\mathcal{R}_1$ and $\mathcal{R}_2$, $\mathcal{R}_3$ and $\mathcal{R}_6$ share the same series of conditions. We introduce variables h_0, h_1, h_3, h_4, h_5, where h_1 is also for the head of $\mathcal{R}_2$ and h_3 is also for the head of $\mathcal{R}_6$. Then the constraints relating the series of conditions to the head of rules are represented as follow:

$$\{h_0,\ o_{3,1} \leftrightarrow h_1,\ \neg o_{4,1} \leftrightarrow h_3,\ \neg o_{3,1} \leftrightarrow h_4,\ \neg o_{3,2} \land o_{4,2} \leftrightarrow h_5\}$$

Similarly, the total weight of class c_i can be expressed as the clauses $\sum_i^r = w_i^{r0} \cdot h_0 + \sum_{j \in \{0,\ldots,N-1\}} (w_{Kj+i} \cdot h_{Kj+i})$. Given an instance $(\mathbf{v}, c_p)$, to ensure that c_p remains the winning class under the current explanation $\mathcal{X}$, for every $c_l, l \in \{1, \ldots, K\}, l \neq p$, we have:

$$\begin{aligned} \max \quad & S_{l,p} = \sum\nolimits_l^r - \sum\nolimits_p^r \\ s.t \quad & \mathcal{H} \land \bigwedge_{j \in \mathcal{X}} l[\![x_j = v_j]\!] \end{aligned} \tag{2}$$

where $S_{l,p}$ is set as soft clauses, and the optimal value is negative. To avoid introducing unnecessary hard clauses, we consider the hard clauses in $\mathcal{H}$ that are associated with $S_{l,p}$. That is, the hard clauses in $\mathcal{H}$ concerning to class c_l and class c_p.

▶ **Example 4.** For our running example, we can represent the total weight of the three classes as follow. $\sum_1^r = -0.31356 \cdot h_0 + 0.38446 \cdot h_1 - 0.20329 \cdot h_4$, $\sum_2^r = -0.29661 \cdot h_0 - 0.20587 \cdot h_1 - 0.20348 \cdot h_5$, $\sum_3^r = -0.38136 \cdot h_0 + 0.38542 \cdot h_3 + 0.27310 \cdot h_3 = -0.38136 \cdot h_0 + 0.65852 \cdot h_3$. Then, there are 2 MaxSAT problems, for the soft clauses, the one is $S_{2,1} = \{(0.01695, h_0), (0.59033, \neg h_1), (0.20329, \neg h_4), (0.20348, \neg h_5)\}$, the other is $S_{3,1} = \{(0.06780, \neg h_0), (0.38446, \neg h_1), (0.65852, h_3), (0.20329, h_4)\}$. To handle soft clauses with negative weight, the common approach is to set the literal into negation with the absolute weight.

3.2 Determining the Winning Class with PB Constraints

In recent years, the best-performing solvers in recent MaxSAT competitions have relied on iterative SAT solver calls within the MaxSAT solving process [6]. This fact motivates us to explain REs by combining SAT solving with PB constraints to determine the winning class, rather than relying directly on MaxSAT solving. An equivalent formulation of Equation 2 is described as follows:

$$\begin{aligned} \forall \alpha, \quad & (\sum\nolimits_l^r - \sum\nolimits_p^r) < 0 \\ s.t \quad & \mathcal{H} \land \bigwedge_{j \in \mathcal{X}} l[\![x_j = v_j]\!] \end{aligned} \tag{3}$$

where α is a truth assignment over all the boolean variables used in the encoding. Therefore, the negation of Equation (3), indicating that c_p is no longer the winning class, can be expressed as follows:

$$\begin{aligned} \exists \alpha, \quad & (\sum\nolimits_l^r - \sum\nolimits_p^r) \geq 0 \\ s.t \quad & \mathcal{H} \land \bigwedge_{j \in \mathcal{X}} l[\![x_j = v_j]\!] \end{aligned} \tag{4}$$

Algorithm 1 Entailment check with SAT encoding.

Function `EntCheckSAT`$(\mathcal{H}, \mathcal{H}^{pred}, (\mathbf{v}, c_p), \mathcal{X})$**:**
  Input: $\mathcal{H}$: Hard clauses, $\mathcal{H}^{pred}$: PB constraints, $(\mathbf{v}, c_p)$: Input instance, $\mathcal{X}$: Candidate explanation
  Output: *true* or *false*
1 **for** $\mathcal{H}^{pred}_{l,p} \in \mathcal{H}^{pred}$ **do**
    // $\mathcal{H}^{pred}_{l,p}$ `is the PB constraint for` $(\sum_l - \sum_p) \geq 0$
2   **if** SAT$(\mathcal{H}^{pred}_{l,p} \wedge \mathcal{H} \wedge \bigwedge_{j \in \mathcal{X}} l[\![x_j = v_j]\!])$ **then**
3   **return false**
4   **end**
5 **end**
6 **return true**

From Equation 4, the determination of the winning class is transformed from $(K-1)$ MaxSAT optimization problems into $(K-1)$ SAT decision problems augmented with PB constraints. Let $\mathcal{H}^{pred}_{l,p}$ denotes the PB constraint encoding $(\sum_l^r - \sum_p^r) \geq 0$, and $\mathcal{H}^{pred}$ be the set of PB constraints for $l \in \{1, \ldots, K\}, l \neq p$. Algorithm 1 describes how the entailment check is performed using SAT oracles. That is, if any SAT oracle satisfies the CNF $(\mathcal{H}^{pred}_{l,p} \wedge \mathcal{H} \wedge \bigwedge_{j \in \mathcal{X}} l[\![x_j = v_j]\!])$, then c_p is no longer the winning class and the entailment check fails. Otherwise, if all SAT oracles return unsatisfiable, the entailment check succeeds.

However, PB constraints accept only integer coefficients for all literals, which requires mapping all the real-valued rule weights to integers. Unfortunately, pseudo-boolean encodings of very large integers results in very large encodings, even when efficient encodings are used. One possible alternative approach is to restrict the original real-valued weights of rules in REs to a smaller number of decimal places (i.e., no more than w decimal places, for some pre-fixed). Conversion to integer values is then attained by multiplying by 10^w. To further reduce the size of the resulting PB constraints, the *greatest common divisor* (gcd) of the coefficients can be used to normalize them.

▶ **Example 5.** Two reduced versions of RE shown in Fig 1(b) are shown in Fig 2. Fig 2(a) (Fig 2(b), resp) shows the reduced version of preserving 2 (1, resp) decimal places for all weights. Considering again the instance in Example 1, using the the original RE, $\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2$ are satisfied for both reduced versions. Both reduced REs also result in the correct prediction c_1 as the original RE does. However, the PB constraints generated for the two reduced REs are different. Regarding Example 4, for Boomer-W2, the PB constraints generated are as follow: $\mathcal{H}^{pred}_{2,1}$: $h_0 + 59\neg h_1 + 20\neg h_4 + 20\neg h_5 \geq 0$, $\mathcal{H}^{pred}_{3,1}$: $7\neg h_0 + 38\neg h_1 + 66h_3 + 20h_4 \geq 0$. For Boomer-W1, the PB constraints are: $\mathcal{H}^{pred}_{2,1}$: $6\neg h_1 + 2\neg h_4 + 2\neg h_5 \geq 0$, after the gcd minimization, the final $\mathcal{H}^{pred}_{2,1} : 3\neg h_1 + \neg h_4 + \neg h_5 \geq 0$. And $\mathcal{H}^{pred}_{3,1} : \neg h_0 + 4\neg h_1 + 7h_3 + 2h_4 \geq 0$. Clearly, the PB constraints for Boomer-W1 is more compact than the ones for Boomer-W2.

Using the entailment check with SAT encoding described in Algorithm 1, the overall explanation procedure is depicted in Algorithm 2, which follows the deletion-based AXp extraction method in [30]. The algorithm iterates through the feature $j \in \mathcal{X}$ (initially set as $\mathcal{F}$), and tests whether $\mathcal{X} \setminus \{j\}$ entails the prediction c_p by invoking Algorithm 1. If it does, then feature j is irrelevant and is removed. Otherwise, it must be kept in $\mathcal{X}$. Once all features are examined, the $\mathcal{X}$ updated is returned as an AXp.

R_0^2 : **DEFAULT** $c_1 = -0.31,\ c_2 = -0.30,\ c_3 = -0.38$
R_1^2 : **IF** petal.length <= 2.45 **THEN** $c_1 = 0.38$
R_2^2 : **IF** petal.length <= 2.45 **THEN** $c_2 = -0.21$
R_3^2 : **IF** petal.width > 1.75 **THEN** $c_3 = 0.39$
R_4^2 : **IF** petal.length > 2.45 **THEN** $c_1 = -0.20$
R_5^2 : **IF** petal.length > 5.05 **AND** sepal.width > 2.75 **THEN** $c_2 = -0.20$
R_6^2 : **IF** petal.width > 1.75 **THEN** $c_3 = 0.27$

(a) The reduced RE preserving 2 decimal places of weights (denoted as Boomer-W2).

R_0^1 : **DEFAULT** $c_1 = -0.3,\ c_2 = -0.3,\ c_3 = -0.4$
R_1^1 : **IF** petal.length <= 2.45 **THEN** $c_1 = 0.4$
R_2^1 : **IF** petal.length <= 2.45 **THEN** $c_2 = -0.2$
R_3^1 : **IF** petal.width > 1.75 **THEN** $c_3 = 0.4$
R_4^1 : **IF** petal.length > 2.45 **THEN** $c_1 = -0.2$
R_5^1 : **IF** petal.length > 5.05 **AND** sepal.width > 2.75 **THEN** $c_2 = -0.2$
R_6^1 : **IF** petal.width > 1.75 **THEN** $c_3 = 0.3$

(b) The reduced RE preserving 1 decimal places of weights (denoted as Boomer-W1).

Figure 2 Two reduced versions of RE shown in Fig 1(b).

Algorithm 2 Deletion-based AXp Extraction for Rule Ensemble.

Function `ExtractAXp`$(\mathfrak{Re}, (\mathbf{v}, c_p))$**:**
  Input: $\mathfrak{Re}$: RE computing the $\tau(\mathbf{v})$, $(\mathbf{v}, c_p)$: Input instance
  Output: $\mathcal{X}$: abductive explanation
  `// The explanation` $\mathcal{X}$ `is over-estimation`
1 $\mathcal{X} \leftarrow \mathcal{F}$
  `// SAT encoding for hard clauses and PB constraints`
2 $\langle \mathcal{H}, \mathcal{H}^{pred} \rangle \leftarrow \mathsf{Encode}(\mathfrak{Re})$
3 **for** $j \in \mathcal{X}$ **do**
    `// Entailment check for the need of` j.
4   **if** $\mathsf{EntCheckSAT}(\mathcal{H}, \mathcal{H}^{pred}, (\mathbf{v}, c_p), \mathcal{X} \setminus \{j\})$ **then**
5      $\mathcal{X} \leftarrow \mathcal{X} \setminus \{j\}$
6   **end**
7 **end**
  `//` $\mathcal{X}$ `is an AXp`
8 **return** $\mathcal{X}$

4 Experimental Results

This section reports on the evaluation of the methods proposed in this paper. The experiments are twofold. First, we select 21 publicly available tabular datasets from UCI Machine Learning Repository [37] and Penn Machine Learning Benchmarks [38]. Second, the MNIST image dataset [39] is used for testing the scalability limits of the considered approaches.

4.1 Experimental Setup

The experiments are performed on a MacBook Pro with Apple M4 CPU and 24 GB RAM. The implementation is prototyped in the form of a Python script, which makes heavy use of the PySAT toolkit [28, 31] and PBLib [48]. The MaxSAT solver used is RC2 [29], a modern core-guided solver that can handle real weights. The SAT solver in use is Glucose 3 [5]. PB constraints are encoded using the *best* encoding automatically selected by PBLib.

4.2 Results on Tabular Datasets

Ensemble models to explain include both TEs and REs trained on the 21 selected tabular datasets with the use of XGBoost [12] and Boomer [50], respectively. For each dataset, 80% of the samples are used for training, the rest are used for testing. Also, to select a tree or rule ensemble model for a dataset, we apply grid search targeting the minimization of the number of unique paths in trees (or rules for REs) in the model while ensuring competitive predictive performance, i.e. test accuracy ≥ 0.95. If no such model reaches the 0.95 test accuracy, then we select the one with the best test accuracy. Furthermore, in light of the proposed SAT encoding, we also consider two *reduced* versions of the RE models preserving 1 and 2 decimal places for each weight in the model. The choice of 1 and 2 decimal places is motivated by the scalability considerations.

Note that the prediction quality of the trained models is detailed in Table 3 in Appendix. We summarize briefly the results as follow. For TEs, the average number of unique paths is 400.9, with the average training (test, resp.) accuracy being 0.989 (0.933, resp.). For REs, the average number of unique rules is 129.7, with the average training (test, resp.) accuracy of 0.979 (0.931, resp.). For precision-reduced REs with 2 decimal places preserved, the average training and test accuracy goes down to 0.978 and 0.930. For precision-reduced REs with 1 decimal place preserved, the average training and test accuracy gets to 0.977 and 0.923, respectively. Observe that the accuracy of the precision-reduced REs is almost not affected by precision reduction. Also, RE model size in general is significantly smaller than that of TEs.

For the models trained, the following methods of computing AXps are evaluated.

- **XGBoost-MaxSAT (SOTA)**: The MaxSAT approach to AXp extraction for TEs [27].[3]
- **Boomer-MaxSAT**: The proposed MaxSAT approach for computing AXps for REs.
- **Boomer-SAT-W***: The proposed SAT approach for computing AXps for precision-reduced REs. The "W*" indicates the number of decimal places preserved in rule weights, i.e. $* \in \{1, 2\}$.

For each dataset, we randomly pick 200 instances to explain; if the dataset has less than 200 instances overall, all instances are considered. All competitors are set to compute a single AXp for per instance selected. Since the AXp computation for tabular datasets is lightweight, the results reported here impose *no* time or memory limit.

Experimental results for tabular datasets are summarized in the scatter plots of Figure 3. Here, a point with coordinates (x, y) corresponds to the execution time x and y in seconds spent by two methods (shown in the X and Y axes labels) on computing an AXp for a single instance. The runtime of the SOTA method XGBoost-MaxSAT varies from 0.001 to

[3] While another formal explainer for tree ensembles, PyXAI [4], is available, it is excluded from our analysis here due to documented implementation defects [24].

13.58 seconds, with the average being 0.72 seconds. Boomer-MaxSAT outperforms the SOTA approach by about *an order of magnitude* with the runtime per instance ranging from 0.001 to 1.18 seconds and the average time of 0.09 seconds. For Boomer-SAT-W2, the runtime for per instance goes further down to 0.001–0.18 seconds, with the mean of 0.02 seconds. The best performance is demonstrated by Boomer-SAT-W1, where the time for per instance varies from 0.001 to 0.04 seconds, with the average being 0.01 seconds. The performance comparison of Boomer-MaxSAT (Boomer-SAT-W2, resp.) and XGBoost-MaxSAT is detailed in Figure 3a (Figure 3b, resp.). The performance of precision-reduced Boomer-SAT-W2 and Boomer-SAT-W1 is contrasted to that of Boomer-MaxSAT in Figure 3d and Figure 3c, respectively.

Overall, the proposed methods for computing AXps for REs are significantly more efficient on tabular data than the SOTA method for TEs. Moreover, the proposed SAT-based method for precision-reduced REs outperforms the MaxSAT-based method for REs by about an order of magnitude in execution time, further improving the efficiency of AXp computation and making the proposed approach a viable alternative to the state of the art.

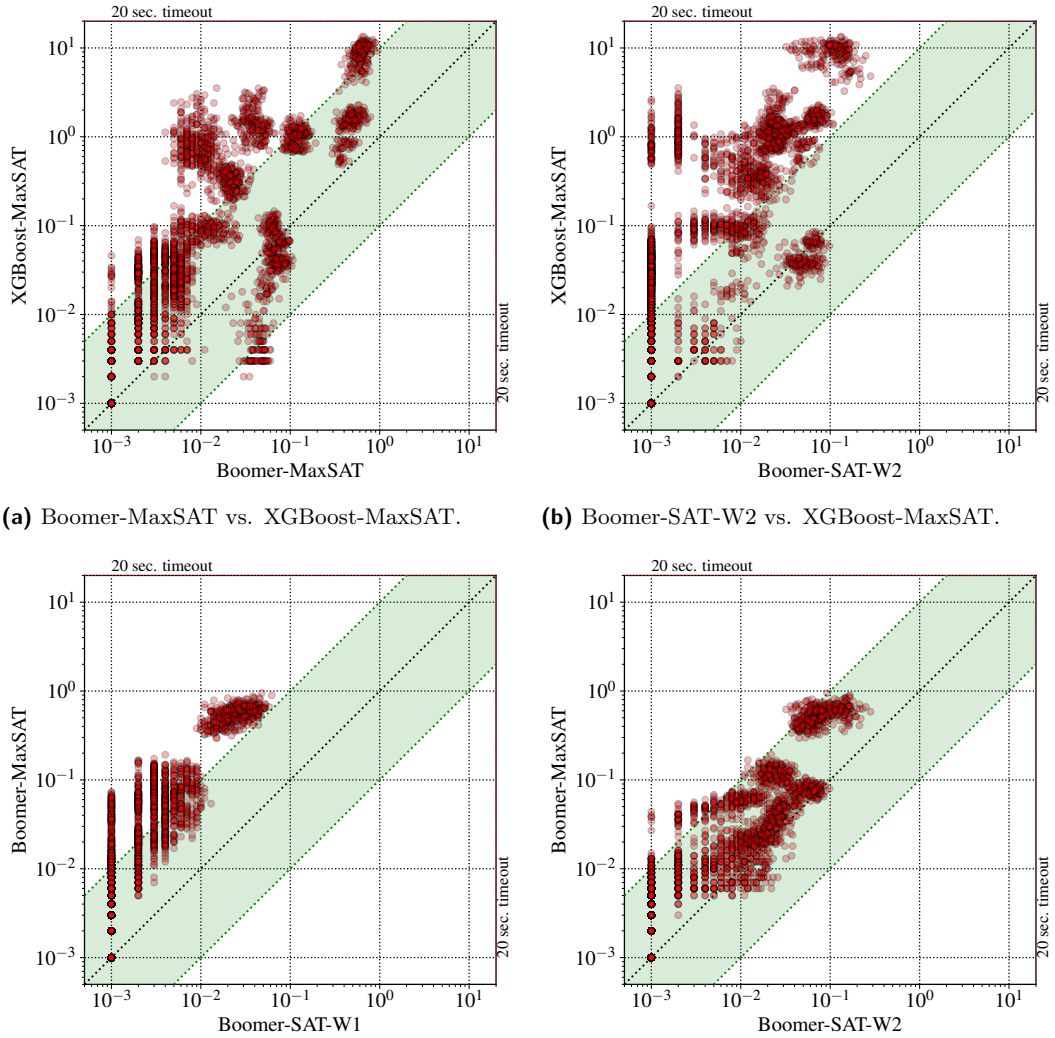
4.3 Results on MNIST Dataset

Motivated by the success of the proposed approach to RE explainability observed on tabular datasets, this section further evaluates its scalability on the MNIST image dataset.

Ensemble models to explain include both tree and rule ensemble models trained on the MNIST dataset with the use of XGBoost and Boomer, respectively. Similar to the above, we apply the standard 80%-20% splitting of the MNIST dataset used for training and testing. The models are selected based on the grid search targeting the minimization of the number of unique rules (paths in trees, resp.) in the model while ensuring test accuracy ≥ 0.95. Note that given the scalability limitations of the SOTA method for TEs, the number of unique paths in the selected TEs is limited to 20K; for a fair comparison, the same upper bound is used for RE models. Also and similar to the above, for the selected rule ensembles, we consider two precision-reduced versions preserving 1 and 2 decimal places for the weights, for the proposed SAT encoding. Finally, the two reduced versions preserving 1 and 2 decimal places for the weights are also applied to selected tree ensembles, for the proposed SAT encoding.

Table 1 reports model accuracy for the selected TEs and REs sorted by the number of rules in ascending order. The column *Model-(md, n)* indicates ensemble type, maximum depth (conditions, resp.) for each model, and the number of trees (rules, resp.) per class. The column # indicates the number of unique rules (or paths in TEs) for the selected model, and the columns *train* and *test* indicate the training and test accuracy. The columns *Model-W1* and *Model-W2* report the training and test accuracy of the precision-reduced variants of the REs and TEs; the values in these columns show the differences between the corresponding original REs/TEs and their reduced versions. Observe that the precision-reduced variant Model-W2 has almost no lose in prediction quality. Also, while the accuracy of Boomer-W1 is lower than that of the original model XGBoost-W1 exhibits no loss in precision quality.

Here, we evaluate the same three approaches to the computation of a single AXp in the case of the MNIST dataset. To clearly distinguish the models used, we specify additional information on the maximum depth (number of conditions, resp.) and the number of trees (rules, resp.) for TEs (REs, resp.). For example, *XGBoost-MaxSAT(4,70)* indicates the use of the *XGBoost-MaxSAT* method for explaining the TE *XGBoost-(4,70)* in Table 1. Note that due to the increased difficulty of the explanation task, we randomly pick 100 instances



(a) Boomer-MaxSAT vs. XGBoost-MaxSAT.

(b) Boomer-SAT-W2 vs. XGBoost-MaxSAT.

(c) Boomer-SAT-W1 vs. Boomer-MaxSAT.

(d) Boomer-SAT-W2 vs. Boomer-MaxSAT.

Figure 3 Detailed runtime comparisons between different methods for the selected TEs and REs on the considered datasets. Both x-axis and y-axis indicate the AXp computation runtime in seconds.

from the dataset to explain. As above, all candidate methods are set to compute a single AXp for per instance selected. Furthermore, as the ensemble models here are significantly more complex (than those for tabular datasets), which negatively impacts the scalability of the considered methods, we set the time limit of 1800 seconds for the computation of each AXp, with no memory limit imposed.

Experimental results are summarized in Table 2. The column *Model-Method(md, n)* lists the target models together with the explanation approaches used. The column *Encoding Size* shows the encoding size, where sub-columns $\#V$ and $\#C$ represent the number of Boolean variables and the number of clauses. *Execution Time* details the runtime for AXp extraction in seconds, where *max*, *min*, and *avg* indicate the maximum, minimum, and average execution time, while $\#F$ reports the number of instances explained within the

Table 1 Prediction performance of selected TEs and REs in MNIST dataset, in ascending order of the number of unique rules. For columns "Model-W1" and "Model-W2", the values in *train* and *test* indicate the differences between the corresponding original TEs and REs.

Model-(md,n)	#	Orig		Model-W1		Model-W2	
		train	test	train	test	train	test
XGBoost-(4, 70)	10145	0.993	0.966	0	0.001	0	0
XGBoost-(6, 20)	10444	0.987	0.957	0	0	0	0
XGBoost-(5, 40)	10984	0.993	0.965	0	0	0	0.001
XGBoost-(4, 80)	11501	0.996	0.968	0	-0.001	0	0
XGBoost-(4, 90)	12852	0.998	0.969	-0.001	-0.001	0	0
XGBoost-(4, 100)	14199	0.999	0.970	0	-0.001	0	0
XGBoost-(6, 30)	14587	0.996	0.965	0	0	0	0
Boomer-(6, 1500)	14823	0.990	0.956	-0.018	-0.023	0	0
Boomer-(4, 1500)	14843	0.979	0.950	-0.046	-0.047	0	0
Boomer-(5, 1500)	14860	0.985	0.954	-0.029	-0.030	0	0
Boomer-(6, 2000)	19784	0.994	0.958	-0.018	-0.023	0	0.001
Boomer-(4, 2000)	19793	0.984	0.953	-0.047	-0.049	0.001	0
Boomer-(5, 2000)	19827	0.989	0.956	-0.028	-0.031	0	0.001

given time limit. To ensure the reliability, the observations on minimum, maximum and average time of methods are made for results solving more than 60 instances ($\#F \geq 60$). The methods solving less than 60 instances are marked in color blue. The following observations can be made. Unsurprisingly, the best performer is Boomer-SAT-W1, being able to tackle all the instances within the time limit, with the runtime for per instance ranging from 10.36 to 94.72 seconds and the average being 24.30 seconds. XGBoost-SAT-W1 comes next solving nearly all instances within the time limit, with the runtime for per instance varying from 12.44 to 4768.34 seconds and the average being 144.03 seconds. Boomer-SAT-W2 solves most of instances within the time limit, with the runtime varying from 146.23 to 7009.46 seconds and mean time being 1063.93 seconds. Interestingly, as can be observed in Table 2, except XGBoost-SAT-W2(6, 20), the best one of XGBoost-SAT-W2, manages to solve 68 instances under the time restriction, the others cannot keep up with its RE counterpart demonstrating rather poor performance. MaxSAT-based approaches for both TEs and REs are significantly less performant than the proposed SAT-based approaches, with the runtime per instance being in the order of thousands of seconds. For Boomer-MaxSAT, the runtime per instance varies from 1593.22 to 7147.84 seconds, with the average being 3286.08 seconds. For XGBoost-MaxSAT, the runtime for per instance varies from 2017.12 to 7198.28 seconds, with the average being 3772.73 seconds. Figure 4 depicts the cactus plots presenting the raw performance of the representative tested competitors. The Appendix contains Figure 5, which provides details of the raw performance of all tested competitors, where Figure 5a (Figure 5b, resp) shows the performance of all SAT-based (MaxSAT-based, resp) approaches for TEs and REs. The methods are ranked by efficiency from best to worst.

Overall, our experimental results on the MNIST dataset confirm the significant advantage of the proposed SAT-based approach for explaining REs of reasonably sized models, which shows great potential in terms of practicality of the proposed approach. In particular, Boomer-SAT-W1 outperforms all methods by a few orders of magnitude. Also, while Boomer-SAT-W2 are somewhat harder to explain than Boomer-SAT-W1, it is still substantially more advantageous in comparison to the MaxSAT-based explainers. Importantly, Boomer-SAT-W2

Table 2 Summarized performance of computing AXps of different methods for different REs/TEs. The columns "#V" and "#C" indicate the number of Boolean variables and clauses in the encoding. The columns "max", "min" and "avg" indicate the maximum, minimum and average execution time. The column "#F" indicates the number of finished instances within the timeout of 7200 seconds.

Model-Method(md,n)	Encoding Size		Execution Time			
	#V	#C	max	min	avg	#F
XGBoost-MaxSAT(4, 70)	33941	358587	3721.41	2017.12	2853.16	100
XGBoost-MaxSAT(6, 20)	36215	405250	4383.44	2033.39	3024.68	100
XGBoost-MaxSAT(5, 40)	37784	413525	4851.79	2097.75	3430.68	100
XGBoost-MaxSAT(4, 80)	38298	403697	5065.75	2255.16	3646.91	100
XGBoost-MaxSAT(4, 90)	42684	449550	6965.47	2387.22	4447.63	100
XGBoost-MaxSAT(4, 100)	46887	491823	7190.94	2322.43	4939.74	92
XGBoost-MaxSAT(6, 30)	50317	562254	7198.28	2752.08	5182.94	87
Boomer-MaxSAT(6, 1500)	114564	199388	5837.06	2444.27	4261.96	100
Boomer-MaxSAT(4, 1500)	86944	144677	2987.60	1593.22	2267.61	100
Boomer-MaxSAT(5, 1500)	102157	174636	3286.52	2083.80	2592.16	100
Boomer-MaxSAT(6, 2000)	144242	251375	7147.84	3341.64	5168.32	67
Boomer-MaxSAT(4, 2000)	110604	184218	5424.56	2514.81	3791.14	100
Boomer-MaxSAT(5, 2000)	129304	221244	7071.81	3405.74	5208.60	65
XGBoost-SAT-W2(4, 70)	674567	729415	6756.56	1353.68	3713.39	21
XGBoost-SAT-W2(6, 20)	769593	833869	7054.14	156.77	2278.38	68
XGBoost-SAT-W2(5, 40)	789467	853006	6869.94	407.46	4020.03	16
XGBoost-SAT-W2(4, 80)	759441	819906	4294.66	790.22	3092.15	3
XGBoost-SAT-W2(4, 90)	838989	904613	2715.27	2166.51	2440.89	2
XGBoost-SAT-W2(4, 100)	928420	1002201	5598.37	3588.53	4593.45	2
XGBoost-SAT-W2(6, 30)	1071281	1155642	6832.89	289.56	4012.49	19
Boomer-SAT-W2(6, 1500)	881917	1020526	1459.31	193.71	568.36	100
Boomer-SAT-W2(4, 1500)	788052	897014	5189.31	146.23	1349.32	96
Boomer-SAT-W2(5, 1500)	829926	954185	2466.73	173.24	593.09	100
Boomer-SAT-W2(6, 2000)	1294371	1477035	3381.00	352.36	1096.78	99
Boomer-SAT-W2(4, 2000)	1095659	1235715	6862.20	239.63	1600.97	91
Boomer-SAT-W2(5, 2000)	1200913	1364368	7009.46	262.91	1398.09	99
XGBoost-SAT-W1(4, 70)	92197	113632	992.48	12.44	70.19	100
XGBoost-SAT-W1(6, 20)	115515	146479	263.85	15.26	38.76	100
XGBoost-SAT-W1(5, 40)	113753	140638	280.29	16.83	62.34	100
XGBoost-SAT-W1(4, 80)	106577	127982	918.70	13.51	110.57	100
XGBoost-SAT-W1(4, 90)	119091	142735	700.41	16.18	140.06	99
XGBoost-SAT-W1(4, 100)	130832	158045	4768.34	17.93	353.81	100
XGBoost-SAT-W1(6, 30)	275655	310569	1887.60	27.64	233.87	100
Boomer-SAT-W1(6, 1500)	161160	261751	29.87	20.80	26.03	100
Boomer-SAT-W1(4, 1500)	119086	185681	24.57	10.36	12.90	100
Boomer-SAT-W1(5, 1500)	141976	227182	24.84	12.05	16.74	100
Boomer-SAT-W1(6, 2000)	208288	330899	46.48	22.50	32.63	100
Boomer-SAT-W1(4, 2000)	151637	237520	94.72	19.71	25.54	100
Boomer-SAT-W1(5, 2000)	182873	289334	45.03	21.52	31.96	100

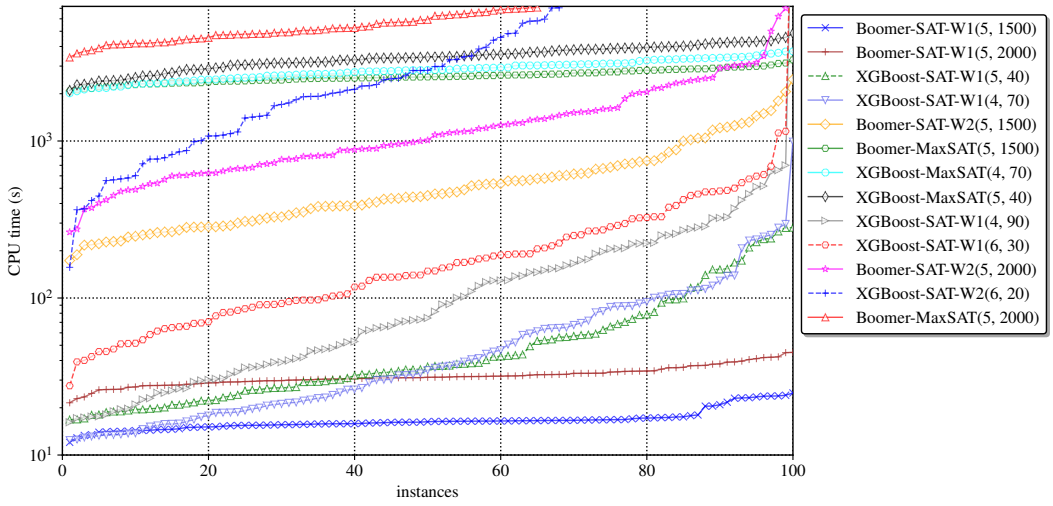


Figure 4 The raw performance on scalability of representative methods computing AXps for different REs/TEs on the MNIST datasets. The timeout is 7200 seconds.

exhibits *no loss* in prediction quality compared to the original REs. Given these results, a natural future research direction is to find a balance between precision reduction and the efficiency of explanation extraction that may improve the practicality of ensemble models when both accuracy and explainability are of concern.

5 Conclusion

This paper adopts an existing MaxSAT-based explanation approach for tree ensembles to general rule ensembles, and introduces a dedicated SAT-based approach to explain rule ensembles with PB constraints to determine the winning class. The lift from tree ensembles to rule ensembles reduces greatly the ensemble model size, thereby directly accelerates the explanation efficiency. Compared to the state-of-the-art method, the SAT-based approach to explain REs significantly enhances scalability, as demonstrated by the results in explaining the MNIST images within a reasonable time.

Several directions for future work emerge from this study. Currently, the proposed SAT-based approach is still intractable when a finer approximation of the original rule ensembles is required, i.e. preserving more decimal places in weights. The primary bottleneck lies in the growth of the PB constraint size. Consequently, investigating more compact PB encodings, or developing dedicated encodings tailored to rule ensemble explanation, is a promising direction. Moreover, the success of *soft cardinality constraints* in modern core-guided MaxSAT solver may inspire new mechanisms for SAT solvers to handle large PB constraints.

References

1 Ignasi Abío, Robert Nieuwenhuis, Albert Oliveras, Enric Rodríguez-Carbonell, and Valentin Mayer-Eichberger. A new look at bdds for pseudo-boolean constraints. *J. Artif. Intell. Res.*, 45:443–480, 2012. doi:10.1613/JAIR.3653.

2 Gilles Audemard, Steve Bellart, Jean-Marie Lagniez, and Pierre Marquis. Computing abductive explanations for boosted regression trees. In *IJCAI*, pages 3432–3441. ijcai.org, 2023. doi:10.24963/IJCAI.2023/382.

3 Gilles Audemard, Jean-Marie Lagniez, Pierre Marquis, and Nicolas Szczepanski. Computing abductive explanations for boosted trees. In *AISTATS*, Proceedings of Machine Learning Research, pages 4699–4711. PMLR, 2023. URL: https://proceedings.mlr.press/v206/audemard23a.html.

4 Gilles Audemard, Jean-Marie Lagniez, Pierre Marquis, and Nicolas Szczepanski. PyXAI: An XAI library for tree-based models. In *IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pages 8601–8605, 2024. URL: https://www.ijcai.org/proceedings/2024/989.

5 Gilles Audemard, Jean-Marie Lagniez, and Laurent Simon. Improving glucose for incremental SAT solving with assumptions: Application to MUS extraction. In *SAT*, Lecture Notes in Computer Science, pages 309–317. Springer, 2013. doi:10.1007/978-3-642-39071-5_23.

6 Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. Maximum satisfiability. In *Handbook of Satisfiability*, Frontiers in Artificial Intelligence and Applications, pages 929–991. IOS Press, 2021. doi:10.3233/FAIA201008.

7 Olivier Bailleux, Yacine Boufkhad, and Olivier Roussel. New encodings of pseudo-boolean constraints into CNF. In *SAT*, Lecture Notes in Computer Science, pages 181–194. Springer, 2009. doi:10.1007/978-3-642-02777-2_19.

8 Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2021. doi:10.3233/FAIA336.

9 Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, 1996. doi:10.1007/BF00058655.

10 Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001. doi:10.1023/A:1010933404324.

11 Leo Breiman, J. H. Friedman, Richard A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

12 Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *KDD*, pages 785–794. ACM, 2016. doi:10.1145/2939672.2939785.

13 William W. Cohen. Fast effective rule induction. In *ICML*, pages 115–123. Morgan Kaufmann, 1995. doi:10.1016/B978-1-55860-377-6.50023-2.

14 William W. Cohen and Yoram Singer. A simple, fast, and effictive rule learner. In *AAAI/IAAI*, pages 335–342. AAAI Press / The MIT Press, 1999. URL: http://www.aaai.org/Library/AAAI/1999/aaai99-049.php.

15 Adnan Darwiche. Logic for explainable AI. In *LICS*, pages 1–11, 2023. doi:10.1109/LICS56636.2023.10175757.

16 Niklas Eén and Niklas Sörensson. Temporal induction by incremental SAT solving. In *BMC@CAV*, number 4 in Electronic Notes in Theoretical Computer Science, pages 543–560. Elsevier, 2003. doi:10.1016/S1571-0661(05)82542-3.

17 Yoav Freund. Boosting a weak learning algorithm by majority. *Inf. Comput.*, 121(2):256–285, 1995. doi:10.1006/INCO.1995.1136.

18 Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

19 Fred Hemery, Christophe Lecoutre, Lakhdar Sais, and Frédéric Boussemart. Extracting MUCs from constraint networks. In *ECAI*, pages 113–117, 2006. URL: http://www.booksonline.iospress.nl/Content/View.aspx?piid=1657.

20 Hao Hu, Marie-José Huguet, and Mohamed Siala. Optimizing binary decision diagrams with maxsat for classification. In *AAAI*, pages 3767–3775. AAAI Press, 2022. doi:10.1609/AAAI.V36I4.20291.

21 Hao Hu, Mohamed Siala, Emmanuel Hebrard, and Marie-José Huguet. Learning optimal decision trees with maxsat and its integration in adaboost. In *IJCAI*, pages 1170–1176. ijcai.org, 2020. doi:10.24963/IJCAI.2020/163.

22 Xuanxiang Huang, Yacine Izza, Alexey Ignatiev, Martin C. Cooper, Nicholas Asher, and João Marques-Silva. Tractable explanations for d-dnnf classifiers. In *AAAI*, pages 5719–5728. AAAI Press, 2022. doi:10.1609/AAAI.V36I5.20514.

23 Xuanxiang Huang, Yacine Izza, Alexey Ignatiev, and João Marques-Silva. On efficiently explaining graph-based classifiers. In *KR*, pages 356–367, 2021. doi:10.24963/KR.2021/34.

24 Xuanxiang Huang, Yacine Izza, Alexey Ignatiev, and Joao Marques-Silva. Uncovering bugs in formal explainers: A case study with PyXAI. *CoRR*, abs/2511.03169, 2025. doi:10.48550/arXiv.2511.03169.

25 Xuanxiang Huang and João Marques-Silva. On the failings of Shapley values for explainability. *Int. J. Approx. Reason.*, 171:109112, 2024. doi:10.1016/J.IJAR.2023.109112.

26 Alexey Ignatiev. Towards trustable explainable AI. In *IJCAI*, pages 5154–5158, 2020. doi:10.24963/IJCAI.2020/726.

27 Alexey Ignatiev, Yacine Izza, Peter J. Stuckey, and João Marques-Silva. Using maxsat for efficient explanations of tree ensembles. In *AAAI*, pages 3776–3785. AAAI Press, 2022. doi:10.1609/AAAI.V36I4.20292.

28 Alexey Ignatiev, António Morgado, and João Marques-Silva. Pysat: A python toolkit for prototyping with SAT oracles. In *SAT*, Lecture Notes in Computer Science, pages 428–437. Springer, 2018. doi:10.1007/978-3-319-94144-8_26.

29 Alexey Ignatiev, António Morgado, and João Marques-Silva. RC2: an efficient maxsat solver. *J. Satisf. Boolean Model. Comput.*, 11(1):53–64, 2019. doi:10.3233/SAT190116.

30 Alexey Ignatiev, Nina Narodytska, and João Marques-Silva. Abduction-based explanations for machine learning models. In *AAAI*, pages 1511–1519. AAAI Press, 2019. doi:10.1609/AAAI.V33I01.33011511.

31 Alexey Ignatiev, Zi Li Tan, and Christos Karamanos. Towards universally accessible SAT technology. In *SAT*, volume 305 of *LIPIcs*, pages 16:1–16:11. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPIcs.SAT.2024.16.

32 Yacine Izza, Xuanxiang Huang, António Morgado, Jordi Planes, Alexey Ignatiev, and João Marques-Silva. Distance-restricted explanations: Theoretical underpinnings & efficient implementation. In Pierre Marquis, Magdalena Ortiz, and Maurice Pagnucco, editors, *KR*, 2024. doi:10.24963/KR.2024/45.

33 Yacine Izza, Alexey Ignatiev, and João Marques-Silva. On explaining decision trees. *CoRR*, abs/2010.11034, 2020. arXiv:2010.11034.

34 Yacine Izza, Alexey Ignatiev, and João Marques-Silva. On tackling explanation redundancy in decision trees. *J. Artif. Intell. Res.*, 75:261–321, 2022. doi:10.1613/JAIR.1.13575.

35 Yacine Izza and João Marques-Silva. On explaining random forests with SAT. In *IJCAI*, pages 2584–2591. ijcai.org, 2021. doi:10.24963/IJCAI.2021/356.

36 Ulrich Junker. QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems. In *AAAI*, pages 167–172, 2004. URL: http://www.aaai.org/Library/AAAI/2004/aaai04-027.php.

37 Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. UCI machine learning repository. http://archive.ics.uci.edu/ml, 2013. Accessed: 2021-08-01.

38 Trang T. Le, William G. La Cava, Joseph D. Romano, John T. Gregg, Daniel J. Goldberg, Praneel Chakraborty, Natasha L. Ray, Daniel S. Himmelstein, Weixuan Fu, and Jason H. Moore. PMLB v1.0: an open source dataset collection for benchmarking machine learning methods. *CoRR*, abs/2012.00058, 2020. arXiv:2012.00058.

39 Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi:10.1109/5.726791.

40 Chu Min Li and Felip Manyà. Maxsat, hard and soft constraints. In *Handbook of Satisfiability*, Frontiers in Artificial Intelligence and Applications, pages 903–927. IOS Press, 2021. doi:10.3233/FAIA201007.

41 Zachary C. Lipton. The mythos of model interpretability. *Commun. ACM*, 61(10):36–43, 2018. doi:10.1145/3233231.

42 João Marques-Silva. Logic-based explainability in machine learning. In *Reasoning Web*, pages 24–104, 2022. doi:10.1007/978-3-031-31414-8_2.

43 João Marques-Silva and Alexey Ignatiev. Delivering trustworthy AI through formal XAI. In *AAAI*, pages 12342–12350, 2022. doi:10.1609/AAAI.V36I11.21499.

44 João Marques-Silva, Mikolás Janota, and Anton Belov. Minimal sets over monotone predicates in boolean formulae. In *CAV*, pages 592–607, 2013. doi:10.1007/978-3-642-39799-8_39.

45 João Marques-Silva, Inês Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. In *Handbook of Satisfiability*, Frontiers in Artificial Intelligence and Applications, pages 133–182. IOS Press, 2021. doi:10.3233/FAIA200987.

46 Hayden McTavish, Zachery Boner, Jon Donnelly, Margo I. Seltzer, and Cynthia Rudin. Leveraging predictive equivalence in decision trees. In *ICML*, 2025. URL: https://proceedings.mlr.press/v267/mctavish25a.html.

47 Tobias Paxian, Sven Reimer, and Bernd Becker. Dynamic polynomial watchdog encoding for solving weighted maxsat. In *SAT*, Lecture Notes in Computer Science, pages 37–53. Springer, 2018. doi:10.1007/978-3-319-94144-8_3.

48 Tobias Philipp and Peter Steinke. Pblib - A library for encoding pseudo-boolean constraints into CNF. In *SAT*, Lecture Notes in Computer Science, pages 9–16. Springer, 2015. doi:10.1007/978-3-319-24318-4_2.

49 Michael Rapp. BOOMER - an algorithm for learning gradient boosted multi-label classification rules. *Softw. Impacts*, 10:100137, 2021. doi:10.1016/J.SIMPA.2021.100137.

50 Michael Rapp, Eneldo Loza Mencía, Johannes Fürnkranz, Vu-Linh Nguyen, and Eyke Hüllermeier. Learning gradient boosted multi-label classification rules. In *ECML/PKDD (3)*, Lecture Notes in Computer Science, pages 124–140. Springer, 2020. doi:10.1007/978-3-030-67664-3_8.

51 Olivier Roussel and Vasco Manquinho. Pseudo-boolean and cardinality constraints. In *Handbook of Satisfiability*, Frontiers in Artificial Intelligence and Applications, pages 1087–1129. IOS Press, 2021. doi:10.3233/FAIA201012.

52 Andy Shih, Arthur Choi, and Adnan Darwiche. A symbolic approach to explaining bayesian network classifiers. In *IJCAI*, pages 5103–5111. ijcai.org, 2018. doi:10.24963/IJCAI.2018/708.

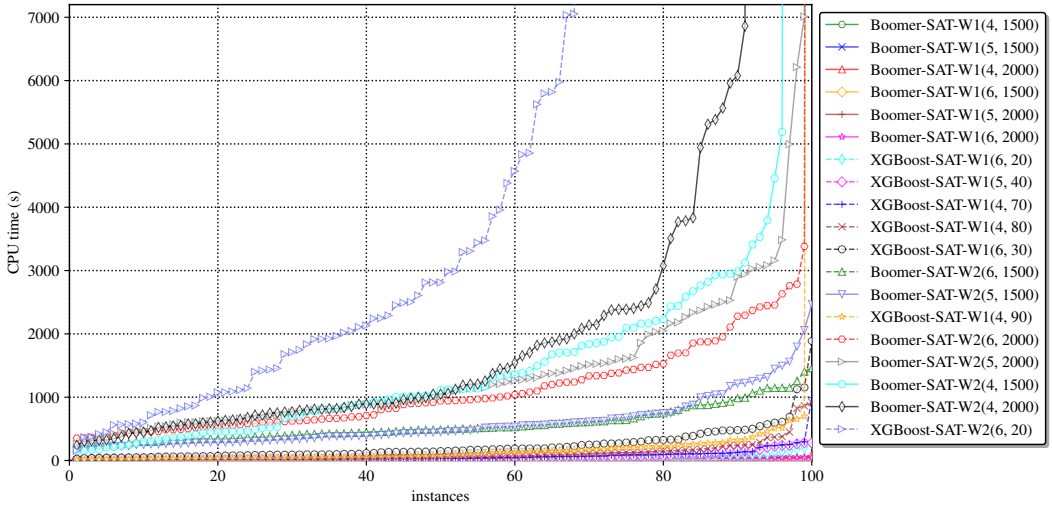
53 Jinqiang Yu, Alexey Ignatiev, Peter J. Stuckey, Nina Narodytska, and Joao Marques-Silva. Eliminating the impossible, whatever remains must be true: On extracting and applying background knowledge in the context of formal explanations. In *AAAI*, 2023.

54 Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. Chapman & Hall/CRC, 1st edition, 2012.

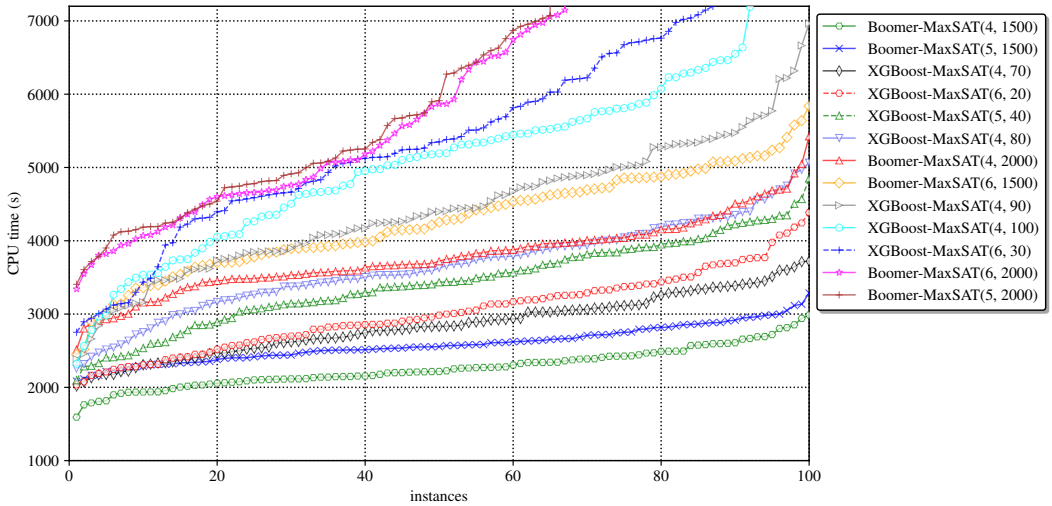
A Detailed Experimental Results

Table 3 Selected TEs and REs for each dataset, with the smallest number of unique rules of $test \geq 0.95$. (Select the TE and RE with the best *test* if all fail to arrive at 0.95) The column "*Datasets*" contains the number of features, classes, and instances. The column "(md, n)" indicates the maximum depth/condition for XGBoost/Boomer, and the number of trees/rules for XGBoost/Boomer. The column "#" indicates the number of unique rules/paths for trees in Boomer/XGBoost per class. "*Boomer-W**" indicates the REs of preserving $(*)$ decimal places in weights, where the *train* and *test* indicate the difference between the corresponding original RE.

Datasets	Boomer				|Boomer-W1|		|Boomer-W2|		XGBoost			
	(md,n)	#	train	test	train	test	Train	Test	(md,n)	#	train	test
ann-thyroid (21-3-7129)	(4, 10)	**29**	0.987	0.987	0.000	0.000	0.000	0.000	(4, 10)	158	**0.998**	**0.997**
appendicitis (7-2-106)	(4, 10)	**11**	0.940	**0.955**	-0.011	0.000	-0.011	0.000	(4, 10)	44	**0.964**	0.909
biodegradation (41-2-1052)	(6, 40)	**41**	0.905	0.886	-0.006	-0.009	0.000	0.000	(4, 10)	126	**0.920**	**0.891**
divorce (54-2-150)	(4, 20)	**18**	**1.000**	**1.000**	0.000	0.000	0.000	0.000	(4, 10)	26	**1.000**	**1.000**
ecoli (7-5-327)	(5, 80)	327	**0.989**	**0.909**	0.000	-0.030	0.000	0.000	(4, 10)	**257**	0.969	0.848
glass2 (9-2-162)	(5, 30)	**30**	0.984	**0.909**	0.000	0.000	0.000	0.000	(6, 20)	128	**1.000**	0.879
ionosphere (34-2-350)	(4, 20)	**21**	0.961	**0.957**	-0.004	0.000	0.000	0.000	(6, 10)	96	**0.996**	0.943
pendigits (16-10-10992)	(4, 30)	**300**	0.958	0.950	-0.003	-0.006	0.000	-0.001	(4, 10)	1120	**0.983**	**0.968**
promoters (58-2-106)	(4, 10)	11	0.964	0.955	0.000	0.000	0.000	0.000	(4, 10)	**2**	**1.000**	**1.000**
segmentation (19-7-210)	(6, 30)	**189**	**1.000**	0.952	0.000	-0.023	0.000	0.000	(4, 10)	190	**1.000**	**0.976**
shuttle (9-7-58000)	(5, 10)	**64**	0.998	0.998	-0.001	-0.001	0.000	0.000	(4, 10)	143	**1.000**	**1.000**
sonar (60-2-208)	(6, 50)	**51**	**1.000**	0.833	0.000	-0.047	0.000	-0.023	(4, 30)	192	**1.000**	**0.857**
spambase (57-2-4210)	(4, 80)	**81**	0.949	0.938	-0.007	-0.008	0.000	-0.001	(4, 40)	402	**0.973**	**0.954**
texture (40-11-5473)	(6, 40)	**440**	0.980	0.955	-0.002	-0.001	0.000	0.001	(4, 10)	909	**0.986**	**0.958**
threeOf9 (9-2-512)	(5, 10)	7	**1.000**	**1.000**	0.000	0.000	0.000	0.000	(4, 10)	**2**	**1.000**	**1.000**
twonorm (20-2-7400)	(6, 30)	**31**	0.970	**0.953**	-0.002	-0.004	0.000	0.000	(4, 20)	304	**0.991**	**0.953**
vowel (13-11-990)	(5, 80)	**841**	0.999	0.914	0.000	-0.020	0.000	0.000	(4, 70)	3816	**1.000**	**0.939**
wdbc (30-2-569)	(6, 20)	**21**	0.982	**0.956**	0.000	0.000	0.000	0.000	(4, 10)	83	**0.996**	**0.956**
wine-recognition (13-3-178)	(4, 10)	**28**	**1.000**	**0.972**	0.000	-0.028	0.000	0.000	(4, 10)	87	**1.000**	**0.972**
wpbc (33-2-194)	(5, 50)	**49**	0.987	0.744	-0.006	0.000	0.000	0.000	(4, 60)	280	**1.000**	**0.769**
zoo (16-7-59)	(5, 30)	134	**1.000**	**0.833**	0.000	0.000	0.000	0.000	(4, 10)	**53**	**1.000**	**0.833**
#Wins	/	17	7	10	/	/	/	/	/	4	20	17
#Avgs	/	129.7	0.979	0.931	0.977	0.923	0.978	0.930	/	400.9	0.989	0.933



(a) All SAT-based appraoches for TEs and REs.



(b) All MaxSAT-based approaches for TEs and REs.

Figure 5 The raw performance on scalability of different methods computing AXps for different REs/TEs on the MNIST datasets.