Anytime Approximate Formal Feature Attribution

Jinqiang Yu
Department of Data Science and AI, Monash University, Melbourne, Australia
Australian Research Council OPTIMA ITTC, Australia

Graham Farr
Department of Data Science and AI, Monash University, Melbourne, Australia

Alexey Ignatiev
Department of Data Science and AI, Monash University, Melbourne, Australia

Peter J. Stuckey
Department of Data Science and AI, Monash University, Melbourne, Australia
Australian Research Council OPTIMA ITTC, Australia

Abstract

Widespread use of artificial intelligence (AI) algorithms and machine learning (ML) models on the one hand and a number of crucial issues pertaining to them warrant the need for explainable artificial intelligence (XAI). A key explainability question is: given this decision was made, what are the input features which contributed to the decision? Although a range of XAI approaches exist to tackle this problem, most of them have significant limitations. Heuristic XAI approaches suffer from the lack of quality guarantees, and often try to approximate Shapley values, which is not the same as explaining which features contribute to a decision. A recent alternative is so-called formal feature attribution (FFA), which defines feature importance as the fraction of formal abductive explanations (AXp's) containing the given feature. This measures feature importance from the view of formally reasoning about the model's behavior. Namely, given a system of constraints logically representing the ML model of interest, computing an AXp requires finding a minimal unsatisfiable subset (MUS) of the system. It is challenging to compute FFA using its definition because that involves counting over all AXp's (equivalently, counting over MUSes), although one can approximate it. Based on these results, this paper makes several contributions. First, it gives compelling evidence that computing FFA is intractable, even if the set of contrastive formal explanations (CXp's), which correspond to minimal correction subsets (MCSes) of the logical system, is provided, by proving that the problem is #P-hard. Second, by using the duality between MUSes and MCSes, it proposes an efficient heuristic to switch from MCS enumeration to MUS enumeration on-the-fly resulting in an adaptive explanation enumeration algorithm effectively approximating FFA in an anytime fashion. Finally, experimental results obtained on a range of widely used datasets demonstrate the effectiveness of the proposed FFA approximation approach in terms of the error of FFA approximation as well as the number of explanations computed and their diversity given a fixed time limit.

2012 ACM Subject Classification Theory of computation → Constraint and logic programming; Computing methodologies → Machine learning

Keywords and phrases Explainable AI, Formal Feature Attribution, Minimal Unsatisfiable Subsets, MUS Enumeration

Digital Object Identifier 10.4230/LIPIcs.SAT.2024.28

Supplementary Material *Software (Source Code)*: `https://github.com/jinqiang-yu/Anytime-FFA/`

Acknowledgements This research was partially funded by the Australian Government through the Australian Research Council Industrial Transformation Training Centre in Optimisation Technologies, Integrated Methodologies, and Applications (OPTIMA), Project ID IC200100009.

1 Introduction

The rise of the use of artificial intelligence (AI) and machine learning (ML) methods to help interpret data and make decisions has exposed a keen need for these algorithms to be able to explain their decisions/judgements. Lack of explanation of opaque and complex models leads to lack of trust, and allows the models to encapsulate unfairness, discrimination and other unwanted properties learnt from the data or through training.

For a classification problem, a key explainability question is: "given a decision was made (a class was imputed to some data instance), what are the features that contributed to the decision?". A more complex question is: "given the decision was made, how important was each feature in making that decision?". There are many heuristic approaches to answering this question, mostly based on sampling around the instance [49], and attempting to approximate Shapley values [32]. But there is strong evidence that Shapley values do not really compute the importance of a feature to a decision [16, 35].

By building on techniques for handling over-constrained systems and minimal unsatisfiability [2, 31, 3, 34, 29], *formal* approaches to explainability (formal explainable AI, FXAI) are able to compute formal *abductive explanations* (AXp's) for a decision, that is a minimal set of features which are enough to ensure the same decision will be made [51, 21]. Namely, an abductive explanation can be associated with a minimal unsatisfiable subset (MUS) of a set of clauses logically representing the decision function of an ML classifier [20]. FXAI approaches can also compute formal *contrastive explanations* (CXp's), that is a minimal set of features, which must change in order to change the decision [39, 20]. Similarly to the case of AXps's, these can be associated with minimal correction subsets (MCSes) of a logical representation of the decision function [20]. Hence a wealth of algorithms originating from minimal unsatisfiability and over-constrained systems [2, 31, 3, 45, 28, 26, 34, 29] are directly applicable for the computation and enumeration of AXp's and CXp's [20, 36]. Here, enumeration of formal explanations builds on the use of the minimal hitting set duality between AXp's and CXp's [20] and the application of the well-known MARCO algorithm originally proposed for implicit hitting set based enumeration of MUSes of unsatisfiable CNF formulas [45, 27, 29]. Until recently there was no formal approach to ascribing importance to features.

A recent and attractive approach to formal feature attribution, called FFA [56], is simple. Compute all the abductive explanations for a decision, then the importance of a feature for the decision is simply the proportion of abductive explanations in which it appears. FFA is crisply defined, and easy to understand, but it is challenging to compute, as deciding if a feature has a non-zero attribution is at least as hard as deciding feature relevancy [15, 56].

Yu *et al.* [56] show that FFA can be efficiently computed by making use of the hitting set duality between AXp's and CXp's. By trying to enumerate CXp's, a side effect of the algorithm is to discover AXp's. In fact, the algorithm will usually find many AXp's before finding the first CXp. The AXp's are guaranteed to be diverse, since they need to be broad in scope to ensure that the CXp is large enough to hit all AXp's that apply to the decision.

Using AXp's collected as a side effect of CXp enumeration is effective at the start of the enumeration. But as we find more and more AXp's as side effects we eventually get to a point where many more CXp's are generated than AXp's. Experimentation shows that if we wish to enumerate all AXp's then indeed we should not rely on the side effect behavior, but simply enumerate AXp's directly. This leads to a quandary: to get fast accurate approximations of FFA we wish to enumerate CXp's and generate AXp's as a side effect. But to compute the final correct FFA we wish to compute all AXp's, and we are better off directly enumerating

AXp's.

In this paper, we develop an *anytime* approach to computing approximate FFA, by starting with CXp enumeration, and then dynamically switching to AXp enumeration when the rate of AXp discovery by CXp enumeration drops. In doing so, we are able to quickly get accurate approximations, but also arrive to the full set of AXp's quicker than pure CXp enumeration. As direct CXp enumeration is feasible to do without the need to resort to the hitting set duality [36], one may want to estimate FFA by first enumerating CXp's. The second contribution of this paper is to investigate this alternative approach and to show that even if a(n) (in)complete set of CXp's is given, determining FFA is computationally expensive being #P-hard even if all CXp's are of size two.

The paper is organized as follows. The next section introduces the notation used throughout the paper. The main results of the paper are given in Section 3, which (1) theoretically argues that exact FFA computation is computationally hard and (2) it shows how to efficiently approximate FFA during the entire explanation enumeration process, which is done by switching from CXp enumeration to AXp enumeration on the fly. Section 4 provides experimental evidence that the proposed switching scheme is beneficial in practice as it helps us get to better quality approximations of FFA if compared to the standard setups of MARCO. Finally, Section 5 concludes the paper.

2 Preliminaries

Here we introduce the required propositional satisfiability (SAT) related notation as well as background on formal explainable AI in order to define formal feature attribution (FFA).

2.1 Satisfiability and Minimal Unsatisfiability

We assume standard definitions for propositional satisfiability (SAT) and maximum satisfiability (MaxSAT) solving [5]. A propositional formula is said to be in *conjunctive normal form* (CNF) if it is a conjunction of clauses. A *clause* is a disjunction of literals. A *literal* is either a Boolean variable or its negation. Whenever convenient, clauses are treated as sets of literals while CNF formulas are treated as *sets of clauses*. A truth assignment maps each variable of a formula to a value from $\{0, 1\}$. Given a truth assignment, a clause is said to be satisfied if at least one of its literals is assigned value 1; otherwise, it is falsified by the assignment. A formula is satisfied if all of its clauses are satisfied; otherwise, it is falsified. If there exists no assignment that satisfies a CNF formula, then the formula is *unsatisfiable*.

In the context of unsatisfiable formulas, the maximum satisfiability (MaxSAT) problem is to find a truth assignment that maximizes the number of satisfied clauses. While a number of variants of MaxSAT exist [5, Chapters 23 and 24], hereinafter, we are interested in Partial Unweighted MaxSAT, which can be formulated as follows. A formula ϕ is represented as a conjunction of *hard* clauses $\mathcal{H}$, which must be satisfied, and *soft* clauses $\mathcal{S}$, which represent a preference to satisfy those clauses, i.e. $\phi = \mathcal{H} \land \mathcal{S}$ (or $\phi = \mathcal{H} \cup \mathcal{S}$ in the set theory notation). The Partial Unweighted MaxSAT problem consists in finding an assignment that satisfies all the hard clauses and maximizes the total number of satisfied soft clauses. In the analysis of an unsatisfiable formula ϕ, one is also often interested in identifying minimal unsatisfiable subsets (MUSes) and minimal correction subsets (MCSes) of ϕ, which can be defined as follows[1].

[1] The problems we are tackling with these formalisms in this paper belong to decidable fragments of

▶ **Definition 1** (Minimal Unsatisfiable Subset (MUS)). *Let $\phi = \mathcal{H} \cup \mathcal{S}$ denote an unsatisfiable set of clauses, i.e. $\phi \vDash \bot$. A subset of clauses $\mu \subseteq \mathcal{S}$ is a* Minimal Unsatisfiable Subset *(MUS) iff $\mathcal{H} \cup \mu \vDash \bot$ and $\forall \mu' \subsetneq \mu$ it holds that $\mathcal{H} \cup \mu' \nvDash \bot$.*

Informally, an MUS can be seen as a minimal explanation of unsatisfiability for an unsatisfiable formula ϕ as it provides the minimal information that needs to be added to the hard clauses $\mathcal{H}$ to obtain unsatisfiability. Alternatively, one may be interested in *correcting* the formula by removing some of the clauses in $\mathcal{S}$ to achieve satisfiability.

▶ **Definition 2** (Minimal Correction Subset (MCS)). *Let $\phi = \mathcal{H} \cup \mathcal{S}$ denote an unsatisfiable set of clauses, i.e. $\phi \vDash \bot$. A subset of clauses $\sigma \subseteq \mathcal{S}$ is a* Minimal Correction Subset *(MCS) iff $\mathcal{H} \cup \mathcal{S} \setminus \sigma \nvDash \bot$ and $\forall \sigma' \subsetneq \sigma$ it holds that $\mathcal{H} \cup \mathcal{S} \setminus \sigma' \vDash \bot$.*

Informally, an MCS can be seen as a minimal way to "correct" unsatisfiability of an unsatisfiable formula ϕ. A fundamental result in reasoning about unsatisfiable CNF formulas is the minimal hitting set (MHS) duality relationship between MUSes and MCSes [48, 6]. That is if the sets of all MUSes and MCSes of formula ϕ are denoted as $\mathbb{U}_\phi$ and $\mathbb{C}_\phi$ then $\mathbb{U}_\phi = \mathrm{MHS}(\mathbb{C}_\phi)$ and $\mathbb{C}_\phi = \mathrm{MHS}(\mathbb{U}_\phi)$ where $\mathrm{MHS}(S)$ returns the minimal hitting sets of S, that is the minimal sets that share an element with each subset in S. More formally, $\mathrm{HS}(S) = \{t \subseteq (\cup S) \mid \forall s \in S,\ t \cap s \neq \emptyset\}$ and $\mathrm{mins}(S) = \{s \in S \mid \forall t \subsetneq s,\ t \notin S\}$ returns the subset-minimal elements of a set of sets, and $\mathrm{MHS}(S) = \mathrm{mins}(\mathrm{HS}(S))$. This result has been extensively used in the development of algorithms for MUSes and MCSes [2, 31, 29], and also applied in a number of different settings. Recent years have witnessed the emergence of a large number of novel algorithms for the extraction and enumeration of MUSes and MCSes [38, 1, 29, 37, 46, 13, 43, 4].

2.2 Classification Problems

We assume classification problems classify data instances into classes $\mathcal{K}$ where $|\mathcal{K}| = k \geq 2$. We are given a set of m features $\mathcal{F}$, where the value of feature $i \in \mathcal{F}$ comes from a domain $\mathbb{D}_i$, which may be Boolean, (bounded) integer or (bounded) real. The *complete feature space* is defined by $\mathbb{F} \triangleq \prod_{i=1}^{m} \mathbb{D}_i$.

A *data point* in feature space is denoted $\mathbf{v} = (v_1, \ldots, v_m)$ where $v_i \in \mathbb{D}_i, 1 \leq i \leq m$. An *instance* of the classification problem is a pair of feature vector and its corresponding class, i.e. $(\mathbf{v}, c)$, where $\mathbf{v} \in \mathbb{F}$ and $c \in \mathcal{K}$.

We use the notation $\mathbf{x} = (x_1, \ldots, x_m)$ to represent an arbitrary point in feature space, where each x_i will take a value from $\mathbb{D}_i$.

A *classifier* is a total function from feature space to class: $\kappa : \mathbb{F} \to \mathcal{K}$. Many approaches exist to define classifiers including decision sets [9, 25], decision lists [50], decision trees [18], random forests [11], boosted trees [8], and neural nets [42, 17].

▶ **Example 3.** Figure 1 shows a boosted tree (BT) model trained with the use of XGBoost [8] for a simplified version of the *adult* dataset [23]. BT models comprise an ensemble of decision trees; given an instance to classify, each decision tree in a BT model contributes a numeric weight to a particular class and the class with the largest total weight is selected as the model's prediction. For a data instance $\mathbf{v} = \{\text{Education} = \text{Bachelors, Status} = \text{Separated, Occupation} = \text{Sales, Relationship} = \text{Not-in-family}, \text{Sex} = \text{Male, Hours/w} \leq 40\}$, the model

first-order logic. While the definitions provided here are given for the propositional case, their extension to the first-order case is straightforward.

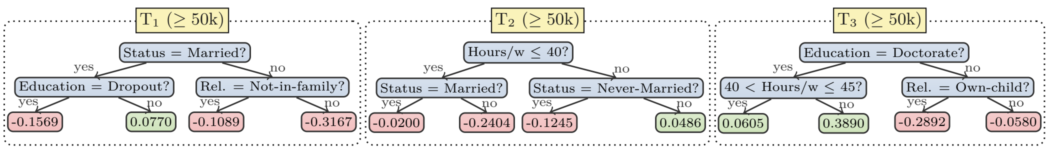


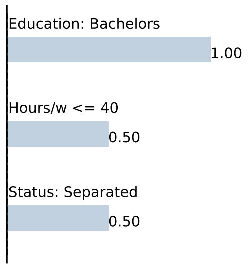
Figure 1 Example boosted tree model [8] trained on the well-known *adult* classification dataset.

$\mathcal{X}_1 = \{$ **Education, Hours/w** $\}$	
IF	Education = Bachelors
AND	Hours/w $\leq$ 40
THEN	Target $<50k$

(a) AXp's $\mathcal{X}_1$

$\mathcal{X}_2 = \{$ **Education, Status** $\}$	
IF	Education = Bachelors
AND	Status = Separated
THEN	Target $<50k$

(b) AXp's $\mathcal{X}_2$



(c) FFA

Figure 2 Examples of both AXp's (no more AXp's exist) followed by FFA for the instance $\mathbf{v}$ shown in Example 3 as well as formal feature attribution (FFA).

predicts $<50k$ because the sum of the weights in the 3 trees for this instance equals $-0.4073 = (-0.1089 - 0.2404 - 0.0580) < 0$.

2.3 Formal Explainability

Given a data point $\mathbf{v}$, classifier κ classifies it as class $\kappa(\mathbf{v})$. A *post-hoc explanation* of the behavior of κ on data point $\mathbf{v}$ tries to explain the behavior of κ on this instance. We consider two forms of formal explanation answering *why* and *why not* (or *how*) questions.

An *abductive explanation* (AXp) is a minimal set of features $\mathcal{X}$ such that any data point sharing the same feature values with $\mathbf{v}$ on these features is guaranteed to be assigned the same class $c = \kappa(\mathbf{v})$ [51, 21]. Formally, $\mathcal{X}$ is a subset-minimal set of features such that:

$$\forall(\mathbf{x} \in \mathbb{F}).\ \left[\bigwedge\nolimits_{i \in \mathcal{X}} (x_i = v_i)\right] \rightarrow (\kappa(\mathbf{x}) = c) \tag{1}$$

▶ **Example 4.** In the context of Figure 1, the two AXp's for the instance $\mathbf{v}$ are shown in Figure 2a and Figure 2b. AXp $\mathcal{X}_1$ indicates that specifying *Education* = *Bachelors* and *Hours/w* ≤ 40 guarantees that any compatible instance is classified as < 50k independent of the values of other features, e.g. *Status* and *Relationship*, since the maximal sum of weights is $0.0770 - 0.0200 - 0.0580 = -0.0010 < 0$ as long as the feature values above are used. Observe that another AXp $\mathcal{X}_1$ for $\mathbf{v}$ is $\{Education, Status\}$, i.e. the model is guaranteed to predict < 50k for any instance in the feature space where features *Education* and *Status* have values *Bachelors* and *Separated*, respectively. Note that no more AXp's exist for instance $\mathbf{v}$. Since both of the two AXp's for $\mathbf{v}$ consist of two features, it is difficult to judge which one is better without a formal feature importance assessment.

▶ **Example 5.** Consider again the ensemble shown in Figure 1. It contains only features *Status*, *Education*, *Relationship*, and *Hours/w*, which can be denoted by integer variables s, e, r, and h, respectively. Note that all the other features of this dataset do not take part in the classification process and can be ignored. Let us map *Status* values *married* and *never-married* to value 1 and 2 of s while value 0 represents all other values. Similarly, we can assign values *dropout*, *doctorate* and *any other value* of feature *Education* to values 1, 2, and 0 of variable e; values *not-in-family*, *own-child*, and *any other value* of feature *Relationship* to values 1, 2, and 0 of variable r. This way $\mathbb{D}_s = \mathbb{D}_e = \mathbb{D}_r = \{0, 1, 2\}$. Finally, according to the tree ensemble, $\mathbb{D}_h = \mathbb{Z}$. As a result and assuming the values assigned by the trees are represented by variables $t_i \in \mathbb{R}$, the classification process for instance $\mathbf{v}$ in Example 3 (predicted as $< 50k$) can be expressed as the following set of *hard* constraints, which are simple to represent in clausal form:

$$\mathcal{H} = \begin{cases} t_1 = -0.1569 & \leftrightarrow \quad (s = 1 \wedge e = 1) \\ t_1 = -0.0770 & \leftrightarrow \quad (s = 1 \wedge (e = 0 \vee e = 2)) \\ t_1 = -0.1089 & \leftrightarrow \quad ((s = 0 \vee s = 2) \wedge r = 1) \\ \quad \dots & \\ t_2 = -0.2404 & \leftrightarrow \quad (h \leq 40 \wedge (s = 0 \vee s = 2)) \\ \quad \dots & \\ t_3 = -0.2892 & \leftrightarrow \quad ((e = 0 \vee e = 1) \wedge r = 2) \\ t_3 = -0.0580 & \leftrightarrow \quad ((e = 0 \vee e = 1) \wedge (r = 0 \vee r = 1)) \\ t_1 + t_2 + t_3 < 0 & \end{cases}$$

Observe that instance $\mathbf{v}$ can be specified as a set of *soft* unit clauses $\mathcal{S} = \{(e = 0), (s = 0), (r = 1), (h \leq 40)\}$. Observe that formula $\mathcal{H} \wedge \mathcal{S}$ is unsatisfiable having two MUSes $\{(e = 0), (h \leq 40)\}$ and $\{(e = 0), (s = 0)\}$, which correspond to the two AXp's shown in Example 4.

A dual concept of *contrastive explanations* (CXp's) helps us understand *how* to reach another prediction [39, 20, 36]. A *contrastive explanation* (CXp) for the classification of data point $\mathbf{v}$ with class $c = \kappa(\mathbf{v})$ is a minimal set of features that must change so that κ can return a different class. Formally, a CXp is a subset minimal set of features $\mathcal{Y}$ such that

$$\exists(\mathbf{x} \in \mathbb{F}). \left[\bigwedge\nolimits_{i \notin \mathcal{Y}} (x_i = v_i)\right] \wedge (\kappa(\mathbf{x}) \neq c) \tag{2}$$

It is known [20] that formal abductive and contrastive explanations for ML predictions are related with the concepts of MUSes and MCSes (defined earlier) of an *unsatisfiable* formula encoding the ML classification process $\kappa(\mathbf{v}) = c$, namely if one represents $[\kappa(\mathbf{x}) \neq c]$ as hard clauses and $[\bigwedge_{i=1}^{m} (x_i = v_i)]$ as soft clauses. For this reason, the set $\mathbb{A}$ of all AXp's $\mathcal{X}$ explaining classification $\kappa(\mathbf{v}) = c$ and the set $\mathbb{C}$ of all CXp's $\mathcal{Y}$ explaining the same classification enjoy a *minimal hitting set duality* [20], similarly to MUSes and MCSes. That is $\mathbb{A} = \text{MHS}(\mathbb{C})$ and is $\mathbb{C} = \text{MHS}(\mathbb{A})$. This property can be made use of when computing or enumerating AXp's and/or CXp's [20, 33, 36].

▶ Remark 6. Thanks to the relation between AXp's (resp., CXp's) for a given ML prediction on the one hand and MUSes (resp., MCSes) of formula encoding the decision process on the other hand, all the ideas and algorithms considered in this paper can be directly applied in any context where MUSes and MCSes are of use.

▶ **Example 7.** Consider the BT model and instance $\mathbf{v}$ in Example 2. Observe that $\mathcal{Y} = \{\textit{Education}\}$ is a CXp for instance $\mathbf{v}$ since the prediction for this instance can be changed if feature *Education* is allowed to take another value, e.g. changing the value of feature *Education* to *Doctorate* triggers that the sum of the weights in the 3 trees becomes $-0.1089 - 0.2404 + 0.3890 = 0.0397 > 0$. By further examining the model and $\mathbf{v}$, more subsets of features can be identified as CXp's for $\mathbf{v}$. The complete set of CXp's for this instance is $\{\{Education\}, \{Status, Hours/w\}\}$, which minimally hits the set of AXp's shown in Example 4. Also observe that the set of CXp's corresponds to the set of MCSes of formula $\mathcal{H} \wedge \mathcal{S}$ shown in Example 5: $\{(e = 0)\}$ and $\{(h \leq 40), (s = 0)\}$.

2.4 Formal Feature Attribution

Given the definition of AXp's above, we can now illustrate the *formal feature attribution* (FFA) function by Yu *et al* [56]. Denoted as $\text{ffa}_\kappa(i, (\mathbf{v}, c))$, it returns for a classification $\kappa(\mathbf{v}) = c$ how important feature $i \in \mathcal{F}$ is in making this classification, defined as the proportion of AXp's for the classification $\mathbb{A}_\kappa(\mathbf{v}, c)$, which include feature i, i.e.

$$\text{ffa}_\kappa(i, (\mathbf{v}, c)) = \frac{|\{\mathcal{X} \mid \mathcal{X} \in \mathbb{A}_\kappa(\mathbf{v}, c), i \in \mathcal{X}\}|}{|\mathbb{A}_\kappa(\mathbf{v}, c)|} \tag{3}$$

▶ **Example 8.** Recall Example 4. As there are 2 AXp's for instance $\mathbf{v}$, namely $\{Education, Status\}$ and $\{Education, Hours/w\}$, the prediction can be attributed to the 3 features with non-zero FFA shown in Figure 2c. Namely, features *Education*, *Status*, and *Hours/w* have the attribution values of 1, 0.5, and 0.5, respectively.

2.5 Computing FFA

Inspired by the implicit hitting set [7] based algorithm eMUS/MARCO [45, 26, 19] for enumerating MUSes and MCSes of an unsatisfiable CNF formula, Yu *et al* [56] define an anytime algorithm for computing FFA shown in Algorithm 1. The algorithm collects AXp's $\mathbb{A}$ and CXp's $\mathbb{C}$. They are initialized to empty. While we still have resources, we generate a minimal hitting set $\mathcal{Y} \in \text{MHS}(\mathbb{A})$ of the current known AXp's $\mathbb{A}$ and not already in $\mathbb{C}$ with the call MinimalHS($\mathbb{A}, \mathbb{C}$). If no (new) hitting set exists then we are finished and exit the loop. Otherwise we check if (2) holds in which case we add the candidate to the set of CXp's $\mathbb{C}$. Otherwise, we know that $\mathcal{F} \setminus \mathcal{Y}$ is a correct (non-minimal) abductive explanation, i.e. it satisfies (1). We use the call ExtractAXp to minimize the resulting explanation, returning an AXp $\mathcal{X}$ which is added to the collection of AXp's $\mathbb{A}$. ExtractAXp tries to remove features j from $\mathcal{F} \setminus \mathcal{Y}$ one by one while still satisfying (1). When resources are exhausted, the loop exits and we return the set of AXp's and CXp's currently discovered.

2.6 Graph-Related Notation

The paper uses some (undirected) graph-theoretic concepts. A graph is defined as a tuple, $G = (V, E)$, where V is a finite set of vertices and E is a finite set of unordered pairs of vertices. For simplicity, uv denotes an edge $\{u, v\}$ of E. Given a graph $G = (V, E)$, a *vertex cover* $X \subseteq V$ is such that for each $uv \in E$, $\{u, v\} \cap X \neq \emptyset$. A *minimal* vertex cover is a vertex cover that is minimal wrt. set inclusion.

2.7 The Complexity of Counting

The class #P consists of functions that count accepting computations of polynomial-time non-deterministic Turing machines [53]. A problem is *#P-hard* if every problem in #P is polynomial-time Turing reducible to it; if it also belongs to #P then it is *#P-complete.*

#P-hardness is usually regarded as stronger evidence of intractability than NP-hardness or indeed hardness for any level of the Polynomial Hierarchy.

Algorithm 1 Anytime Explanation Enumeration as defined by Yu *et al* [56].

```
 1: procedure XpEnum($\kappa$, $\mathbf{v}$, $c$)
 2:     $(\mathbb{A}, \mathbb{C}) \leftarrow (\emptyset, \emptyset)$
 3:     while resources available do
 4:         $\mathcal{Y} \leftarrow$ MinimalHS$(\mathbb{A}, \mathbb{C})$
 5:         if $\mathcal{Y} = \bot$ then break
 6:         if $\exists(\mathbf{x} \in \mathbb{F}). \left[\bigwedge_{i \notin \mathcal{Y}} (x_i = v_i)\right] \wedge (\kappa(\mathbf{x}) \neq c)$ then
 7:             $\mathbb{C} \leftarrow \mathbb{C} \cup \{\mathcal{Y}\}$
 8:         else
 9:             $\mathcal{X} \leftarrow$ ExtractAXp$(\mathcal{F} \setminus \mathcal{Y}, \kappa, \mathbf{v}, c)$
10:             $\mathbb{A} \leftarrow \mathbb{A} \cup \{\mathcal{X}\}$
        return $\mathbb{A}$, $\mathbb{C}$
11: procedure ExtractAXp($\mathcal{X}$, $\kappa$, $\mathbf{v}$, $c$)
12:     for $j \in \mathcal{X}$ do
13:         if $\forall(\mathbf{x} \in \mathbb{F}). \left[\bigwedge_{i \in \mathcal{X} \setminus \{j\}} (x_i = v_i)\right] \rightarrow (\kappa(\mathbf{x}) = c)$ then
14:             $\mathcal{X} \leftarrow \mathcal{X} \setminus \{j\}$
        return $\mathcal{X}$
```

3 Approximate Formal Feature Attribution

Facing the need to compute (exact or approximate) FFA values, one may think of a possibility to first enumerate CXp's and then apply the minimal hitting set duality between AXp's and CXp's to determine FFA, without explicitly computing $\mathbb{A} = \text{MHS}(\mathbb{C})$. This looks plausible given that CXp enumeration can be done directly, without the need to enumerate AXp's [20]. However, as Section 3.1 argues, computing FFA given a set of CXp's turns out to be computationally difficult, being (roughly) at least as hard as counting the minimal hitting sets $\text{MHS}(\mathbb{C})$. Hence, Section 3.2 approaches the problem from a different angle by efficient exploitation of the eMUS- or MARCO-like setup [45, 27, 29, 20] and making the algorithm *switch* from CXp enumeration to AXp enumeration on the fly.

3.1 Duality-Based Computation is Hard

We show that determining $\text{ffa}_\kappa(i, (\mathbf{v}, c))$ from $\mathbb{C}$ is #P-hard even when all CXp's have size two. In that special case, the CXp's may be treated as the edges of a graph, which we denote by $G(\mathcal{F}, \kappa, \mathbf{v}, c)$, with vertex set $\mathcal{F}$. The minimal hitting set duality between the CXp's and AXp's then implies that the AXp's $\mathcal{X} \in \text{MHS}(\mathbb{C})$ are precisely the minimal vertex covers of $G(\mathcal{F}, \kappa, \mathbf{v}, c)$. It is known that determining the number of minimal vertex covers in a graph is #P-complete (even for bipartite graphs); this is implicit in [47], as noted for example in [52, p. 400].

When all CXp's have size 2, the formal feature attribution $\text{ffa}_\kappa(i, (\mathbf{v}, c))$ is just the proportion of minimal vertex covers of $G(\mathcal{F}, \kappa, \mathbf{v}, c)$ that contain the vertex i, i.e. the vertex of $G(\mathcal{F}, \kappa, \mathbf{v}, c)$ that represents the feature $i \in \mathcal{F}$. To help express this in graph-theoretic language, write $\#\text{mvc}(G)$ for the number of minimal vertex covers of G. Write $\#\text{mvc}(G, v)$ and $\#\text{mvc}(G, \neg v)$ for the numbers of minimal vertex covers of G that *do* and *do not* contain vertex $v \in V(G)$, respectively. Define

$$\text{ffa}(G, v) := \frac{\#\text{mvc}(G, v)}{\#\text{mvc}(G)}. \tag{4}$$

Then

$$\text{ffa}_\kappa(i, (\mathbf{v}, c)) = \text{ffa}(G(\mathcal{F}, \kappa, \mathbf{v}, c), i).$$

Observe that $\#\text{mvc}(G) = \#\text{mvc}(G, v) + \#\text{mvc}(G, \neg v)$. Then we may rewrite (4) as

$$\text{ffa}(G, v) = \frac{\#\text{mvc}(G, v)}{\#\text{mvc}(G, v) + \#\text{mvc}(G, \neg v)}. \tag{5}$$

▶ **Theorem 9.** *Determining* ffa(G, v) *is #P-hard.*

Proof. We give a polynomial-time Turing reduction from the #P-complete problem of counting minimal vertex covers to the problem of determining ffa for a node in a graph.

Suppose we have an oracle that, when given a graph and a vertex, returns the ffa of the vertex in one time-step.

Let G be a graph for which we want to count the minimal vertex covers. Let v be a non-isolated vertex of G. (If none exists, the problem is trivial.) Put

$$\begin{aligned} x &= \#\text{mvc}(G, \neg v), \\ y &= \#\text{mvc}(G, v), \end{aligned}$$

so that $\#\text{mvc}(G) = x + y$. It is routine to show that $x, y > 0$. Initially, x and y are unknown. Our reduction will use an ffa-oracle to gain enough information to determine x and y. We will then obtain $\#\text{mvc}(G) = x + y$.

First, query the ffa-oracle with G and vertex v. It returns

$$p := \frac{y}{x + y},$$

by (5). We can then recover the ratio $x/y = p^{-1} - 1$.

Then we construct a new graph $G_v^{[2]}$ from G as follows. Take two disjoint copies G_1 and G_2 of G. Let v_1 be the copy of vertex v in G_1. For every $w \in V(G_2)$, add an edge $v_1 w$ between v_1 and w. We query the ffa-oracle with $G_v^{[2]}$ and vertex v_1. Let $q = \text{ffa}(G_v^{[2]}, v_1)$ be the value it returns.

If a minimal vertex cover X of $G_v^{[2]}$ contains v_1 then all the edges from v_1 to G_2 are covered. The restriction of X to G_1 must be a minimal vertex cover of G_1 that contains v_1, and the number of these is just $\#\text{mvc}(G, v) = y$. The restriction of X to G_2 must just be a vertex cover of G_2, without any further restriction, and the number of these is just $\#\text{mvc}(G) = x + y$. These two restrictions of X can be chosen independently to give all possibilities for X. So

$$\#\text{mvc}(G_v^{[2]}, v_1) = y(x + y).$$

If a minimal vertex cover X of $G_v^{[2]}$ does not contain v_1 then the edges $v_1 w$, $w \in V(G_2)$, are not covered by v_1. So each $w \in V(G_2)$ must be in X, which serves to cover not only those edges but also all edges in G_2. The restriction of X to G_1 must just be a vertex cover of G_1 that does not contain v_1, and there are $\#\text{mvc}(G, \neg v) = x$ of these. Again, the two restrictions of X are independent. So

$$\#\text{mvc}(G_v^{[2]}, \neg v_1) = x.$$

Therefore

$$q = \frac{y(x + y)}{x + y(x + y)},$$

by (5) (applied this time to $G_v^{[2]}$), so

$$x + y = \frac{x/y}{q^{-1} - 1} = \frac{p^{-1} - 1}{q^{-1} - 1}.$$

We can therefore compute $x + y$ from the values p and q returned by our two oracle calls. We therefore obtain $\#\text{mvc}(G)$. The entire computation is polynomial-time. ◻

▶ **Corollary 10.** *Determining* $\text{ffa}_\kappa(i, (\mathbf{v}, c))$ *from the set of CXp's is #P-hard, even if all CXp's have size 2.*

Unfortunately, Theorem 9 and Corollary 10 suggest that relying solely on the *direct* enumeration of CXp's in the fashion of the first phase of CAMUS-like algorithms [30, 31] when computing formal feature attribution does not make the problem simple. One will still need one to implicitly or explicitly enumerate AXp's to be able to compute FFA.

3.2 Switching from CXp to AXp Enumeration

As discussed in Section 2, [56] proposed to apply implicit hitting set enumeration for approximating FFA thanks to the duality between AXp's and CXp's. The approach builds on the use of the MARCO algorithm [45, 27, 29] in the anytime fashion, i.e. collects the sets of AXp's and CXp's and stops upon reaching a given resource limit. As MARCO can be set to target enumerating either AXp's or CXp's depending on user's preferences, the dual explanations will be collected by the algorithm as a *side effect.* Quite surprisingly, the findings of [56] show that for the purposes of *practical* FFA approximation it is beneficial to target CXp enumeration and get AXp's by duality. An explanation offered for this by [56] is that MARCO has to collect a large number of dual explanations before the minimal hitting sets it gets may realistically be the target explanations.

Our practical observations confirm the above. Also note that the AXp's enumerated by MARCO need to be *diverse* if we want to quickly get good FFA approximations. Due to the *incremental* operation of the minimal hitting set enumeration algorithms, this is hard to achieve if we *target* AXp enumeration. But if we aim for CXp's then we can extract diverse AXp's by duality, which helps us get reasonable FFA approximations quickly converging to the exact FFA values.

Nevertheless, our experiments with the setup of [56] suggest that AXp enumeration in fact tends to finish much earlier than CXp enumeration despite "losing" at the beginning. This makes one wonder what to opt for if good and quickly converging FFA approximation is required: AXp enumeration or CXp enumeration. On the one hand, the latter quickly gives a large set of diverse AXp's and good initial FFA approximations. On the other hand, complete AXp enumeration finishes much faster, i.e. exact FFA is better to compute by targeting AXp's.

Motivated by this, we propose to set up MARCO in a way that it starts with CXp enumeration and then seamlessly switches to AXp enumeration using two simple heuristic criteria. It should be first noted that to make efficient switching in the target explanations, we employ pure SAT-based hitting set enumerator, where an incremental SAT solver is called multiple times aiming for minimal or maximal models [12], depending on the phase. This allows us to keep all the explanations found so far without ever restarting the hitting set enumerator.

As we observe that AXp's are normally larger than CXp's, both criteria for switching the target build on the use of the average *size* of the last w AXp's and the last w CXp's enumerated in the most recent iterations of the MARCO algorithm. (Recall that our MARCO setup aims for subset-minimal explanations rather than cardinality-minimal explanations, i.e. neither target nor dual explanations being enumerated are cardinality-minimal.) This can be seen as inspecting “sliding windows” of size w for both AXp’s and CXp’s. In particular, assume that the sets of the last w AXp’s and CXp’s are denoted as $\mathbb{A}^w$ and $\mathbb{C}^w$, respectively.

First, switching can be done as soon as we observe that CXp’s on average are *much* smaller than AXp’s, i.e. when

$$\frac{\sum_{\mathcal{X}\in\mathbb{A}^w}|\mathcal{X}|}{\sum_{\mathcal{Y}\in\mathbb{C}^w}|\mathcal{Y}|}\geq\alpha, \tag{6}$$

where $\alpha\in\mathbb{R}$ is a predefined numeric parameter. The rationale for this heuristic is as follows. Recall that extraction of a subset-minimal *dual* explanation is done by ExtractDualXp() by means of deciding the validity of the corresponding predicate, either (1) or (2), while iteratively removing features from the feature set completementary to the candicate set, i.e. $\mathcal{F}\setminus\mu$ (see Section 2.5). As such, if the vast majority of CXp’s are much smaller than their AXp counterparts, which implies that $|\mathcal{F}\setminus\mu|\gg|\mu|$, then extracting these dual AXp’s from $\mathcal{F}\setminus\mu$ may be expensive as it leads to a large number of SAT oracle calls (namely $|\mathcal{F}\setminus\mu|$ calls) per dual AXp to extract. Hence, we prefer to switch the enumerator to the opposite phase such that ExtractDualXp() deals with a smaller number of decision oracle calls on average. Note that having small dual CXp’s will also result in the lion’s share of these oracle calls being *satisfiable*, i.e. potentially cheap.

Second, we can switch when the average CXp size “stabilizes”. Here, let us denote a new CXp being just computed as $\mathcal{Y}_{\text{new}}$. Then the second criterion can be examined by checking if the following holds:

$$\left||\mathcal{Y}_{\text{new}}|-\frac{\sum_{\mathcal{Y}\in\mathbb{C}^w}|\mathcal{Y}|}{w}\right|\leq\varepsilon, \tag{7}$$

with $\varepsilon\in\mathbb{R}$ being another numeric parameter. This condition is meant to signify the point when the set of dual AXp’s we have already accumulated is diverse enough for all the CXp’s to be of roughly equal size, which is crucial for good FFA approximations. Once we have reached a high level of FFA approximation, it makes sense to switch the target phase to AXp as it normally finishes exhaustive explanation enumeration earlier. Overall, the switching can be performed when either of the two conditions (6)–(7) is satisfied.

Algorithm 2 shows the pseudo-code of the adaptive explanation enumeration algorithm. Additionally to the classifier’s representation κ, instance $\mathbf{v}$ to explain and its class c, it receives 3 numeric parameters: window size $w\in\mathbb{N}$ and switching-related constants $\alpha,\varepsilon\in\mathbb{R}$. The set of CXp’s (resp. AXp’s) is represented by $\mathbb{E}_0$ (resp. $\mathbb{E}_1$) while the target phase of the hitting set enumerator is denoted by $\rho\in\{0,1\}$, i.e. at each iteration Algorithm 2 aims for $\mathbb{E}_\rho$ explanations. As initially $\rho=0$, the algorithm targets CXp enumeration. Each of its iterations starts by computing a minimal hitting set μ of the set $\mathbb{E}_{1-\rho}$ subject to $\mathbb{E}_\rho$ (see line 5), i.e. we want μ to be a hitting set of $\mathbb{E}_{1-\rho}$ different from all the target explanations in $\mathbb{E}_\rho$ found so far. If no hitting set exists, the process stops as we have enumerated all target explanations. Otherwise, each new μ is checked for being a target explanation, which is done by invoking a reasoning oracle to decide the validity either of (1) if we target AXp’s, or of (2) if we target CXp’s. If the test is positive, the algorithm records the new explanation μ in $\mathbb{E}_\rho$. Otherwise, using the standard apparatus of formal explanations, it extracts a subset-minimal dual explanation ν from the complementary set $\mathcal{F}\setminus\mu$, which is then recorded in $\mathbb{E}_{1-\rho}$. Each iteration is additionally augmented with a check whether we should switch to the opposite phase $1-\rho$ of the enumeration. This is done in line 12 by testing whether at least one of the conditions (6)–(7) is satisfied.

Algorithm 2 Adaptive Explanation Enumeration

```
 1: procedure AdaptiveXpEnum($\kappa$, $\mathbf{v}$, $c$, $w$, $\alpha$, $\varepsilon$)
 2:     $(\mathbb{E}_0, \mathbb{E}_1) \leftarrow (\emptyset, \emptyset)$                ▷ CXp's and AXp's to collect
 3:     $\rho \leftarrow 0$                                  ▷ Target phase of enumerator, initially CXp
 4:     while true do
 5:         $\mu \leftarrow$ MinimalHS$(\mathbb{E}_{1-\rho}, \mathbb{E}_\rho, \rho)$
 6:         if $\mu = \bot$ then break
 7:         if IsTargetXp$(\mu, \kappa, \mathbf{v}, c)$ then
 8:             $\mathbb{E}_\rho \leftarrow \mathbb{E}_\rho \cup \{\mu\}$          ▷ Collect target explanation $\mu$
 9:         else
10:             $\nu \leftarrow$ ExtractDualXp$(\mathcal{F} \setminus \mu, \kappa, \mathbf{v}, c)$
11:             $\mathbb{E}_{1-\rho} \leftarrow \mathbb{E}_{1-\rho} \cup \{\nu\}$      ▷ Collect dual explanation $\nu$
12:         if IsSwitchNeeded$(\mathbb{E}_\rho, \mathbb{E}_{1-\rho}, w, \alpha, \varepsilon)$ then
13:             $\rho \leftarrow 1 - \rho$                     ▷ Flip phase of MinimalHS
        return $\mathbb{E}_1$, $\mathbb{E}_0$                     ▷ Result AXp's and CXp's
```

▶ Remark 11. Flipping enumeration phase ρ can be seamlessly done because we apply pure SAT-based hitting enumeration [12] where both $\mathbb{E}_\rho$ and $\mathbb{E}_{1-\rho}$ are represented as sets of *negative* and *positive* blocking clauses, respectively. As such, by instructing the SAT solver to opt for minimal or maximal models,[2] we can flip from computing hitting sets of $\mathbb{E}_{1-\rho}$ subject to $\mathbb{E}_\rho$ to computing hitting sets of $\mathbb{E}_\rho$ subject to $\mathbb{E}_{1-\rho}$, and vice versa. Importantly, this can be done while incrementally keeping the internal state of the SAT solver, i.e. no learnt information gets lost after the phase switch. Also, note that although the algorithm allows us to apply phase switching multiple times, our practical implementation switches *once* because AXp enumeration normally gets done much earlier than CXp enumeration, i.e. there is no point in switching back.

4 Experimental Results

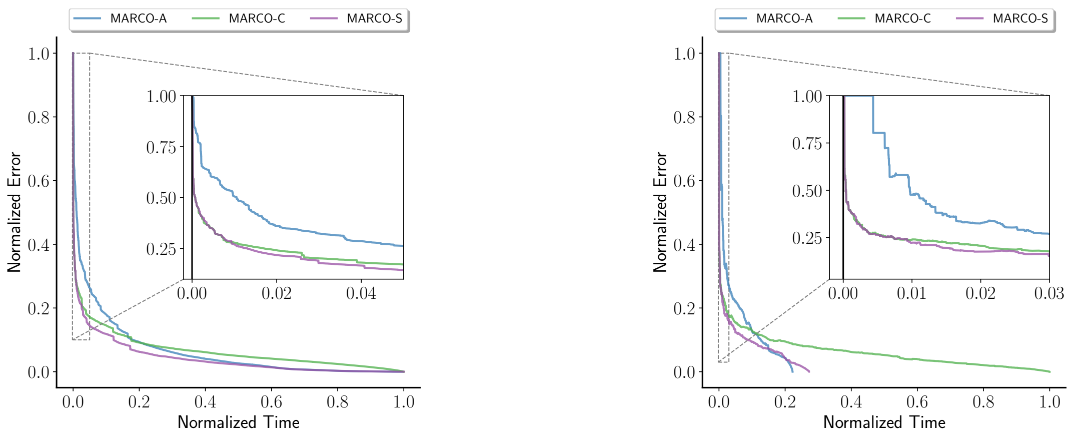
This section evaluates the proposed approach to anytime FFA approximation for the gradient boosted tree (BT) ML models on various publicly available data using a range of metrics. Here we report the results integrating all the experimental data averaged across all data instances in Section 4.2. The results for individual datasets can be found in Section 4.3.

4.1 Experimental Setup

The experiments were performed on an Intel Xeon 8260 CPU running Ubuntu 20.04.2 LTS, with the 8GByte memory limit.

Prototype Implementation. The proposed approach was prototyped as a set of Python scripts, building on the approach of [56]. The proposed approach is referred to as MARCO-S, where the MARCO algorithm switches from CXp to AXp enumeration based on the conditions (6)–(7). The policy we use is to switch if either condition holds as we found

[2] Recall that in SAT solving, a *minimal* model is a satisfying assignment that respects subset-minimality wrt. the set of positive literals, i.e. where none of the 1's can be replaced by a 0 such that the result is still a satisfying assignment [5]. *Maximal* models can be defined similarly wrt. subset-minimality of negative literals.



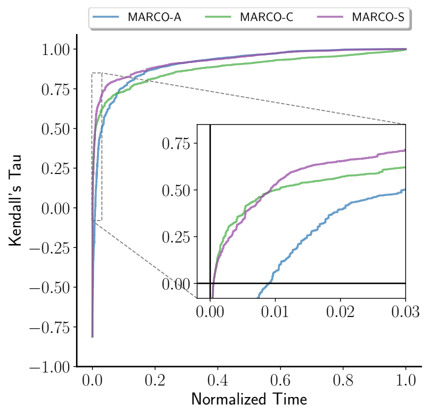
(a) Mean **(b)** Median

Figure 3 FFA approximation error over time.

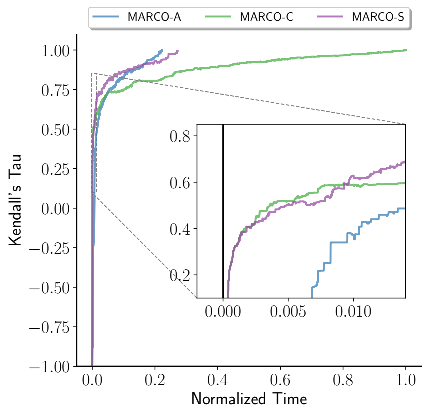
examples where each individual criterion was poor. For this, “sliding windows" of size $w = 50$ are used. Parameter α is set as $\alpha = 2$ in (6) to signify the extent by which the size of AXp’s should be larger than the size of CXp’s while parameter $\varepsilon = 1$ in (7) denoting the stability of the average CXp size.

Datasets and Machine Learning Models. The experiments include five well-known image and text datasets. We use the widely known *MNIST* [10, 44] dataset, which features handwritten digits from 0 to 9, with two concrete binary classification problems created: 1 vs. 3 and 1 vs. 7. Note that we treat MNIST “1vs3” and MNIST “1vs7” as two different datasets. Also, we consider the image dataset *PneumoniaMNIST* [55] differentiating normal X-ray cases from the cases of pneumonia. Since extracting *exact* FFA values for aforementioned image datasets turns out to be hard [56], we perform a size reduction, downscaling these images from $28 \times 28 \times 1$ to $10 \times 10 \times 1$. Additionally, 2 text datasets are considered in the experiments: *Sarcasm* [40, 41] and *Disaster* [14]. The *Sarcasm* dataset contains news headlines collected from websites, along with binary labels indicating whether each headline is sarcastic or non-sarcastic. The *Disaster* dataset consists of the contents of tweets with labels about whether a user announces a real disaster or not. The five considered datasets are randomly divided into 80% training and 20% test data. To evaluate the performance of the proposed approach, 15 test instances in each test set are randomly selected. Therefore, the total number of instances used in the experiments is 75. In the experiments, gradient boosted trees (BTs) are trained by XGBoost [8], where each BT consists of 25 to 40 trees of depth 3 to 5 per class. Test accuracy for *MNIST* (both “1vs3” and “1vs7”), *PneumoniaMNIST*, *Sarcasm*, and *Disaster* datasets is 0.99, 0.83, 0.69, and 0.74, respectively.

Competitors and Metrics. We compare the proposed approach (MARCO-S) against the original MARCO algorithms targeting AXp (MARCO-A) or CXp (MARCO-C) enumeration. We evaluate the FFA generated by these approaches by comparing it to the *exact* FFA through a variety of metrics, including errors, Kendall’s Tau [22], rank-biased overlap (RBO) [54], and Kullback–Leibler (KL) divergence [24]. The *error* is quantified using Manhattan distance, i.e. the sum of absolute differences across all features in an instance. The comparison of feature ranking is assessed by Kendall’s Tau and RBO, while feature distributions are compared

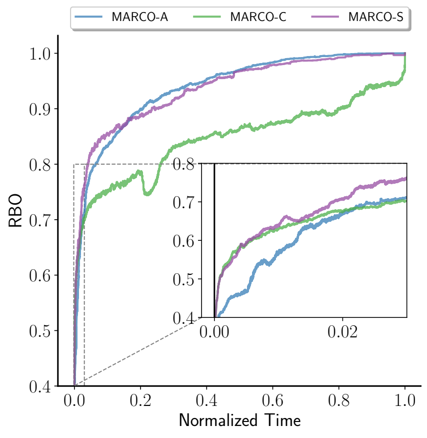


(a) Mean

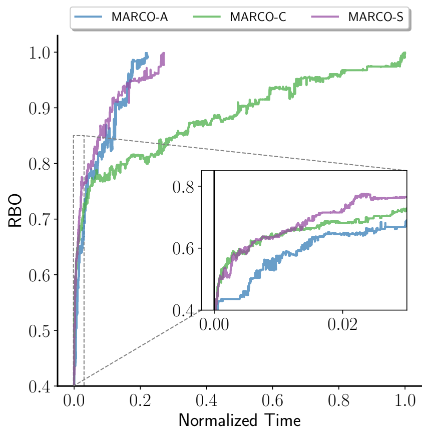


(b) Median

Figure 4 Kendall's Tau over time.



(a) Mean

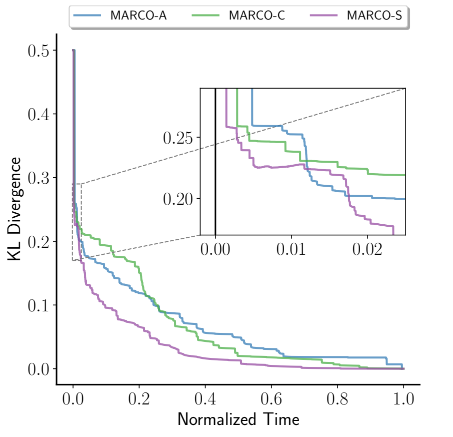


(b) Median

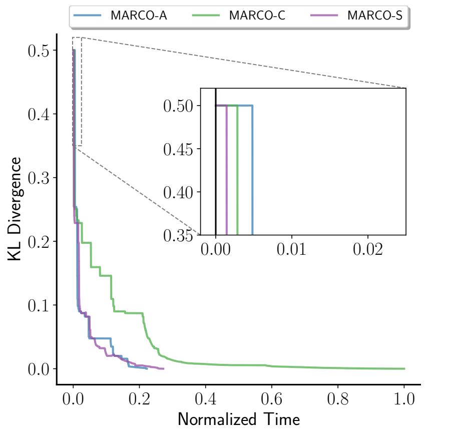
Figure 5 RBO over time.

by KL divergence.[3] Kendall's Tau and RBO produce values within the range of $[-1, 1]$ and $[0, 1]$, respectively, where higher values in both metrics indicate stronger agreement or similarity between the approximate FFA and the exact FFA. KL-divergence ranges from 0 to ∞, with the value approaching 0 reflecting better alignment between approximate FFA distribution and the exact FFA distribution. Note that if a feature in the exact FFA distribution holds a non-zero probability but is assigned a zero probability in the approximate one, the KL value becomes ∞. Finally, we also compare the efficiency of generating AXp's in the aforementioned approaches.

[3] Kendall's Tau is a correlation coefficient metric that evaluates the ordinal association between two ranked lists, providing a similarity measure for the order of values, while RBO quantifies similarity between two ranked lists, considering both the order and depth of the overlap. KL-divergence measures the statistical distance between two probability distributions.

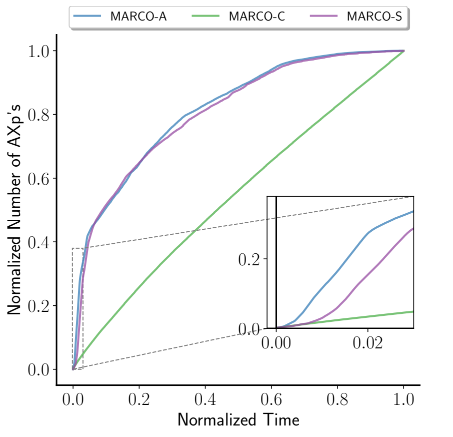


(a) Mean

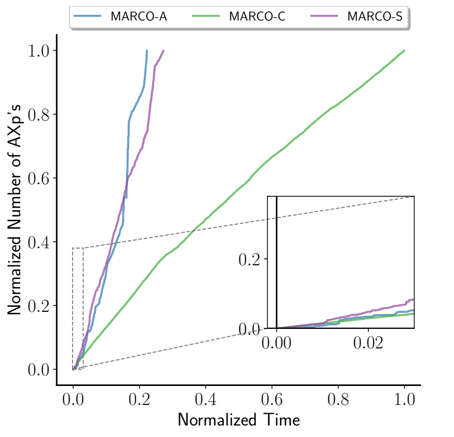


(b) Median

Figure 6 KL divergence over time.



(a) Mean



(b) Median

Figure 7 Number of AXp's over time.

Dataset	Min	Mean	Max
MNIST1vs3	2916	15780.87	46576
MNIST1vs7	461	4028.27	10790
PneumoniaMNIST	21	8802.87	30996
Sarcasm	1056	12542.13	20024
Disaster	128	22853.20	35804

(a) Numbers of MUSes/AXp's.

Dataset	Min	Mean	Max
MNIST1vs3	992	17158.07	55108
MNIST1vs7	394	3558.80	9228
PneumoniaMNIST	30	6148.67	42308
Sarcasm	73	487.73	641
Disaster	88	4792.20	7028

(b) Numbers of MCSes/CXp's.

Table 1 The absolute numbers of MUSes (AXp's) and MCSes (CXp's) for different datasets.

4.2 Overview of Experimental Results

This section compares the proposed approach against the original MARCO algorithms for both AXp enumeration and CXp enumeration within the examined datasets. Figures 3 to 7 illustrate the results of approximate FFA in terms of the five aforementioned metrics, namely, errors, Kendall's Tau, RBO, KL divergence, and the number of AXp's. These results are obtained by averaging values across all instances. Note that since KL-divergence is ∞ when there exists a feature in the exact FFA distribution that holds a non-zero probability but is assigned a zero probability in the approximate one, to address this issue we assign 0.5 as the KL-divergence value instead of ∞ in this case.[4] The average runtime to extract exact FFA is 3255.30s (from 2.15s to 29191.42s), 19311.87s (from 9.39s to 55951.57s), and 3509.50s (from 9.26s to 30881.55s) for MARCO-A, MARCO-C, and MARCO-S, respectively. Switching from CXp to AXp enumeration in MARCO-S occurs on average at 106.77s. Note that since MARCO-A and MARCO-S tend to finish the enumeration process much earlier than MARCO-C, we also plot the median information because it better reflects the typical performance of these approaches in practice (which may be hard to see on the average data). Since the runtime required to get exact FFA varies, we normalized the runtime in each instance into [0, 1], where the longest time across three compared approaches in each instance is normalized to 1. Furthermore, we normalized the number of AXp's in each instance into the interval of [0, 1], as Table 1a shows that the numbers of AXp's (MUSes) vary across different instances and datasets. (Similarly, Table 1b indicates that the numbers of CXp's (MCSes) also differ across instances and datasets.) FFA approximation errors are also normalized into [0, 1] for each instance. Finally, switching from CXp to AXp enumeration in MARCO-S occurs at the time point of 0.0055 on the normalized scale (recall that it equals $\approx$106.77s).

Approximation Errors. Figure 3 displays the average and median errors of approximate FFA across all instances over time. Observe that in the early period, MARCO-C obtains more accurate approximate FFA regarding errors compared with MARCO-A, while beyond the 0.02 time fraction, the latter surpasses the former and eventually achieves 0 error faster, which also indicates that MARCO-A requires less time to acquire the exact FFA. Motivated by the above observation, the proposed approach aims at replicating the "best of two worlds" during the FFA approximation process. Observe that MARCO-S commences with CXp enumeration and so replicates the superior behavior of MARCO-C during the initial stage. Over time, MARCO-S triggers a switch criterion and transitions to targeting AXp's, thus emulating the behavior of the better competitor beyond the early stage, i.e. MARCO-A. Finally, MARCO-S acquires FFA with 0 error (i.e. exact FFA) as efficiently as MARCO-A.

Feature Ranking. The results of Kendall's Tau and RBO are depicted in Figures 4 and 5. Initially, MARCO-C outperforms MARCO-A in terms of both feature ranking metrics. As time progresses, MARCO-A starts to surpass MARCO-C since 0.01 time fraction until the point of acquiring the exact FFA. Figures 4 and 5 demonstrate that initially MARCO-S manages to keep close to the better performing MARCO-C. When MARCO-A starts dominating, MARCO-S switches the target phase from CXp's to AXp's, replicating the superior performance displayed by MARCO-A.

Distribution. Figure 6 depicts the average and median results of KL divergence over time. Similar to feature ranking, MARCO-C is initially capable of computing an FFA distribution closer to the exact FFA distribution. Beyond the initial stage, MARCO-A exhibits the ability to generate closer FFA distribution. Once again, MARCO-S replicates the superior behavior between MARCO-A and MARCO-C most of the time. During the initial stage, MARCO-S

[4] According the experimental results we obtained, the maximum of *non-infinity* KL-divergence values is not greater than 0.5.

Approach	MNIST-1vs3	MNIST-1vs7	Pneumoniamnist	Sarcasm	Disaster
MARCO-A	9350.79	2844.15	1972.41	669.91	1439.24
MARCO-C	14787.22	7412.40	8343.55	33391.29	32624.89
MARCO-S	9970.55	2959.15	2016.49	975.31	1626.01

Table 2 Average runtime(s) in each dataset.

reproduces the behavior of MARCO-C, and switch to target AXp's directly when the switch criterion is met. Surprisingly, MARCO-S *outperforms both competitors* throughout (almost) the entire time interval.

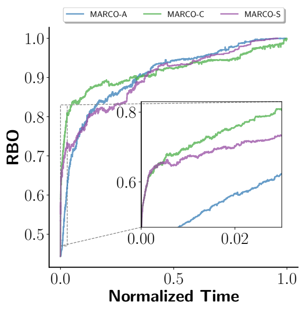
Number of AXp's. The average and median results of the normalized number of AXp's are illustrated in Figure 7. MARCO-A generates AXp's faster and finishes earlier than MARCO-C. Observe that the proposed approach MARCO-S is able to avoid the inferior performance between MARCO-A and MARCO-C throughout the process. Initially, MARCO-S replicates the behavior of MARCO-C and then switches to target AXp's to replicate the performance of MARCO-A.

Summary. MARCO-S can replicate the behavior of the superior competitor for most of the computation duration, leading to efficient and good approximation of FFA. As illustrated by Figures 3–6 in terms of FFA errors, Kendall's Tau, RBO, and KL divergence, starting from CXp enumeration and switching to AXp enumeration based on the criteria (6)–(7) successfully replicates the behavior of the winning configuration of MARCO, thus getting close to the *virtual best solver*. Although in terms of the number of AXp's shown in Figure 7 MARCO-A consistently outperforms MARCO-C, those AXp's are not diverse enough to allow MARCO-A to beat MARCO-C in other relevant metrics. This is alleviated by MARCO-S, which manages to get enough diverse AXp's initially and then switches to target AXp's to catch up with the performance of MARCO-A.

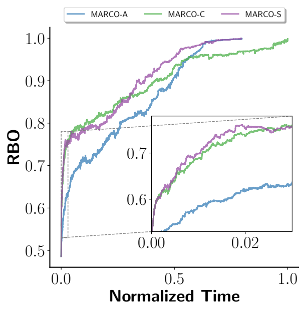
4.3 Detailed Experimental Results

This section compares the proposed approach (MARCO-S against the original MARCO algorithms for targeting AXp's (MARCO-A) and CXp's (MARCO-C) in each considered dataset. Figures 8 to 10 depict the average results of the comparison between the approximate FFA and the exact FFA using 3 metrics, namely, RBO, KL divergence, and the number of AXp's. The results show the mean values across 15 selected instances in a dataset. The average runtime of the three methods to acquire the exact FFA in each datadset is summarized in Table 2.

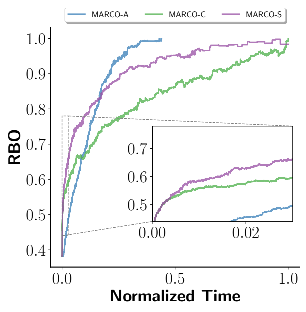
Feature Ranking. Figure 8 illustrates the results of RBO in each dataset. Observe that in all datasets but *Sarcasm*, MARCO-C performs better initially than MARCO-A, except in the *Sarcasm* dataset. Over time, MARCO-A gradually overtakes MARCO-C until reaching the point of obtaining the exact FFA. This figure demonstrates that MARCO-S maintains close to the superior performance exhibited by MARCO-C initially and then switches to targeting AXp's, replicating the superior performance demonstrated by MARCO-A. Nevertheless, in the *Sarcasm* dataset, MARCO-A consistently displays the superior performance. In the *Sarcasm* dataset, switching from CXp to AXp enumeration beyond the initial stage avoids reproducing the inferior performance between MARCO-A and MARCO-C in most of time.



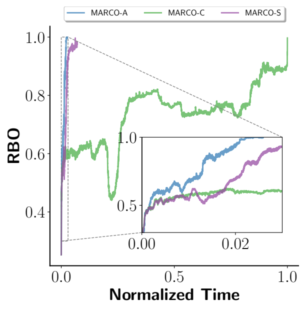
(a) MNIST-1vs3



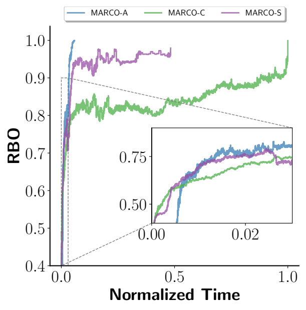
(b) MNIST-1vs7



(c) PneumoniaMNIST

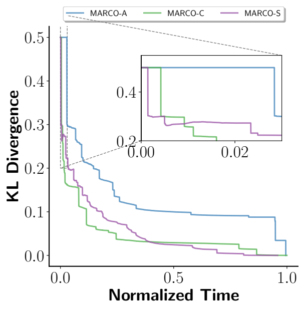


(d) Sarcasm

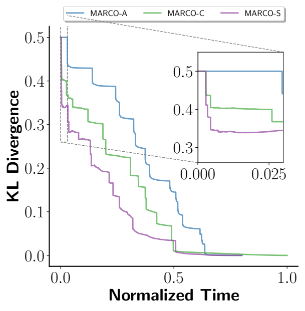


(e) Disaster

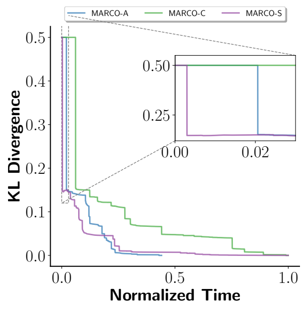
Figure 8 Mean RBO over time in each dataset.



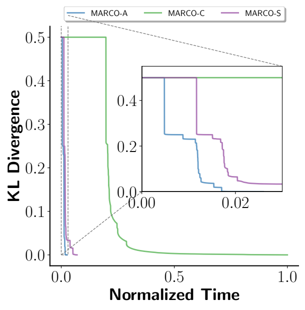
(a) MNIST-1vs3



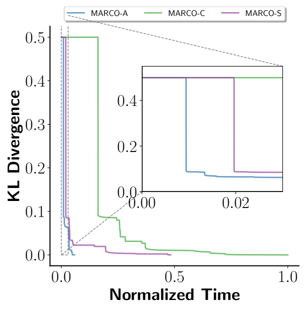
(b) MNIST-1vs7



(c) PneumoniaMNIST



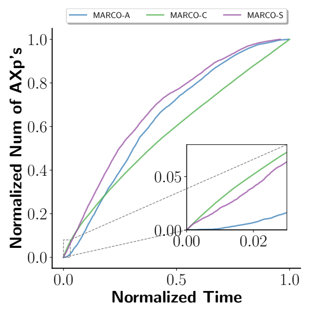
(d) Sarcasm



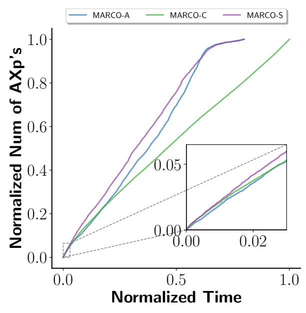
(e) Disaster

Figure 9 Mean KL-divergence over time in each dataset.

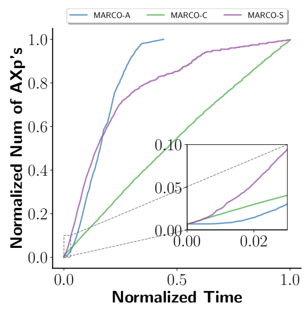
565 **Distribution.** The average results of KL divergence over time are depicted in Figure 9.
566 MARCO-C is initially capable of generating an FFA distribution more similar to the exact



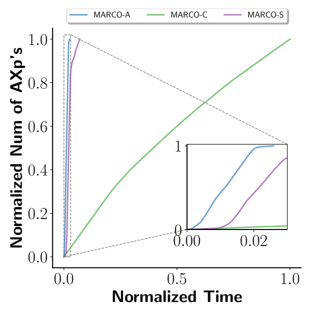
(a) MNIST-1vs3



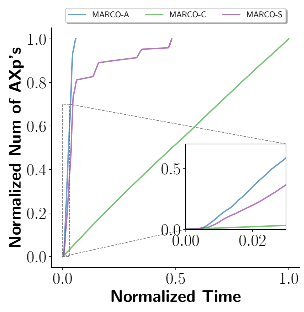
(b) MNIST-1vs7



(c) PneumoniaMNIST

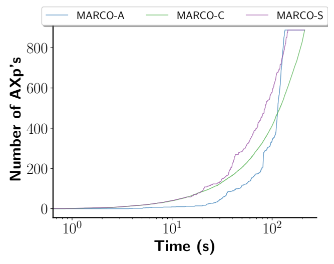


(d) Sarcasm

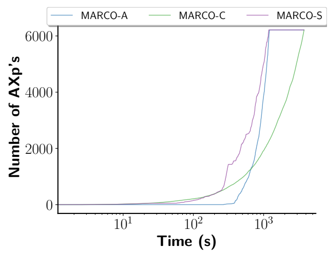


(e) Disaster

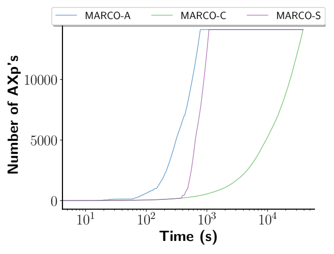
Figure 10 Mean number of AXp's over time in each dataset.



(a) MNIST-1vs7 instance



(b) PneumoniaMNIST instance



(c) Sarcasm instance

Figure 11 Number of AXp's over time in example instances.

FFA distribution in *MNIST-1vs3* and *MNIST-1vs7* datasets. Afterwards, MARCO-A exhibits the ability to compute FFA distribution more similar to the exact FFA attribution. However, MARCO-A consistently generate a closer FFA distribution than MARCO-C in the other datasets. Once again, MARCO-S emulates the superior behavior between MARCO-A and MARCO-C in most of time or avoids replicating the inferior performance for a long time due to the switch mechanism. MARCO-S initially reproduces the behavior of MARCO-C, and switches to target AXp's when meeting the switch criterion. Surprisingly, MARCO-S exhibits the best performance among the competitors in most of the entire time interval in *MNIST-1vs3* and *MNIST-1vs7*.

Number of AXp's. Figure 10 shows the average results of the normalized number of AXp's in each dataset. Observe that compared with MARCO-A, MARCO-C is capable of generating AXp's more efficiently during the early stage in *MNIST-1vs3* and *MNIST-1vs7* datasets, but MARCO-A starts to outperform MARCO-C as time progresses. In the other three datasets, MARCO-A achieves similar or better performance in the entire process. As demonstrated by Figure 10, the proposed approach MARCO-S is able to avoid the inferior performance between MARCO-A and MARCO-C for most of the duration. Initially, MARCO-S emulates the behavior of MARCO-C, and transitions to target AXp's to replicate the performance of MARCO-A afterwards, preventing the reproduction of inferior performance. Remarkably, in the *MNIST-1vs7* dataset, MARCO-S emerges as the best-performing approach for most of time. Figure 11 presents numbers of AXp's over time in three example instances, demonstrating that MARCO-S can avoid the inferior performance between MARCO-A and MARCO-C for most of time in these three instances.

Summary. In alignment with the results presented in Section 4.2, MARCO-S is able to replicate the behavior of the superior competitor between MARCO-A and MARCO-C throughout most of the computation period, resulting in fast and good approximation of FFA. Figures 8 and 9 display that switching from CXp to AXp enumeration based on criteria 6–7 can reproduce the performance of the top MARCO configuration, closely approaching their virtual best solver. While MARCO-A consistently exhibits better than MARCO-C in some datasets in terms of the number of AXp's depicted in Figure 10, the lack of diversity among these AXp's prevents MARCO-A from outperforming MARCO-C in other relevant metrics. MARCO-S addresses this diversity issue by initially obtaining a diverse set of AXp's and then transitioning to targeting them, thereby matching the performance of MARCO-A.

5 Conclusions

Formal feature attribution (FFA) defines a crisp and easily understood notion of feature importance to a decision. It builds on the concepts of formal abductive and contrastive explanations [36], which can be related to the concepts of minimal unsatisfiable subsets (MUSes) and minimal correction subsets (MCSes) in the context of SAT solving. Unfortunately, for many classifiers and datasets FFA is challenging to compute exactly. As our paper demonstrates, it remains hard even if the set of CXp's is provided. Hence, there is a need for *anytime* approaches to compute FFA. One approach to compute and approximate FFA values is by exploiting the duality between AXp's and CXp's and applying the MARCO-style algorithms [45, 27, 29] of exhaustive AXp (resp., MUS) and CXp (resp., MCS) enumeration. As exhaustive explanation enumeration can be done by targeting either AXp's or CXp's, it is not always clear which approach is more efficient in practice from the perspective of the raw number of explanations but also from the view of the quality of FFA value approximations. Surprisingly, using CXp enumeration to generate AXp's leads to fast good approximations of FFA, but in the longer term it is worse than simply enumerating AXp's. This paper shows how to combine the approaches by diligently switching the phase of enumeration, without losing information computed in the underlying MARCO enumeration algorithm. This gives a highly practical approach to computing FFA.

The proposed mechanism can be readily adapted to a multitude of other problems, e.g. in the domains of over-constrained systems or model-based diagnosis (MBD), where one wants to collect a *diverse* and representative set of MUSes or explanations as the same minimal hitting set duality exists in unsatisfiability and MBD between the concepts of MUSes and MCSes, and explanations and diagnoses, respectively [6, 48].

References

1 Fahiem Bacchus and George Katsirelos. Using minimal correction sets to more efficiently compute minimal unsatisfiable sets. In *CAV*, pages 70–86, 2015.

2 James Bailey and Peter J. Stuckey. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization. In *PADL*, pages 174–186, 2005.

3 Anton Belov, Ines Lynce, and Joao Marques-Silva. Towards efficient MUS extraction. *AI Commun.*, 25(2):97–116, 2012.

4 Jaroslav Bendík, Ivana Cerná, and Nikola Benes. Recursive online enumeration of all minimal unsatisfiable subsets. In *ATVA*, pages 143–159, 2018.

5 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.

6 Elazar Birnbaum and Eliezer L. Lozinskii. Consistent subsets of inconsistent systems: structure and behaviour. *J. Exp. Theor. Artif. Intell.*, 15(1):25–46, 2003.

7 Karthekeyan Chandrasekaran, Richard M. Karp, Erick Moreno-Centeno, and Santosh S. Vempala. Algorithms for implicit hitting set problems. In *SODA*, pages 614–629, 2011.

8 Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *KDD*, pages 785–794, 2016.

9 Peter Clark and Robin Boswell. Rule induction with CN2: some recent improvements. In *EWSL*, pages 151–163, 1991.

10 Li Deng. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

11 Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

12 Enrico Giunchiglia and Marco Maratea. Solving optimization problems with DLL. In *ECAI*, pages 377–381, 2006.

13 Éric Grégoire, Yacine Izza, and Jean-Marie Lagniez. Boosting MCSes enumeration. In *IJCAI*, pages 1309–1315, 2018.

14 Addison Howard, Devrishi, Phil Culliton, and Yufeng Guo. Natural language processing with disaster tweets, 2019. URL: `https://kaggle.com/competitions/nlp-getting-started`.

15 Xuanxiang Huang, Martin C. Cooper, António Morgado, Jordi Planes, and João Marques-Silva. Feature necessity & relevancy in ML classifier explanations. In *TACAS (1)*, pages 167–186, 2023.

16 Xuanxiang Huang and João Marques-Silva. The inadequacy of Shapley values for explainability. *CoRR*, abs/2302.08160, 2023.

17 Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *NIPS*, pages 4107–4115, 2016.

18 Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is NP-complete. *Inf. Process. Lett.*, 5(1):15–17, 1976.

19 Alexey Ignatiev, Mikolas Janota, and Joao Marques-Silva. Quantified maximum satisfiability. *Constraints An Int. J.*, 21(2):277–302, 2016.

20 Alexey Ignatiev, Nina Narodytska, Nicholas Asher, and Joao Marques-Silva. From contrastive to abductive explanations and back again. In *AI*IA*, pages 335–355, 2020.

21 Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva. Abduction-based explanations for machine learning models. In *AAAI*, pages 1511–1519, 2019.

22 Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

23 Ron Kohavi. Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid. In *KDD*, pages 202–207, 1996.

24 Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

25 Himabindu Lakkaraju, Stephen H. Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *KDD*, pages 1675–1684. ACM, 2016.

26 Mark Liffiton and Ammar Malik. Enumerating infeasibility: Finding multiple MUSes quickly. In *CPAIOR*, pages 160–175, 2013.

27 Mark Liffiton and Ammar Malik. Enumerating infeasibility: Finding multiple muses quickly. In *CPAIOR*, pages 160–175, 2013.

28 Mark H. Liffiton, Maher N. Mneimneh, Ines Lynce, Zaher S. Andraus, Joao Marques-Silva, and Karem A. Sakallah. A branch and bound algorithm for extracting smallest minimal unsatisfiable subformulas. *Constraints An Int. J.*, 14(4):415–442, 2009.

29 Mark H. Liffiton, Alessandro Previti, Ammar Malik, and Joao Marques-Silva. Fast, flexible MUS enumeration. *Constraints An Int. J.*, 21(2):223–250, 2016.

30 Mark H. Liffiton and Karem A. Sakallah. On finding all minimally unsatisfiable subformulas. In *SAT*, pages 173–186, 2005.

31 Mark H. Liffiton and Karem A. Sakallah. Algorithms for computing minimal unsatisfiable subsets of constraints. *J. Autom. Reasoning*, 40(1):1–33, 2008.

32 Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NeurIPS*, pages 4765–4774, 2017.

33 Joao Marques-Silva, Thomas Gerspacher, Martin C. Cooper, Alexey Ignatiev, and Nina Narodytska. Explanations for monotonic classifiers. In *ICML*, pages 7469–7479, 2021.

34 Joao Marques-Silva, Federico Heras, Mikolás Janota, Alessandro Previti, and Anton Belov. On computing minimal correction subsets. In *IJCAI*, pages 615–622, 2013.

35 João Marques-Silva and Xuanxiang Huang. Explainability is NOT a game. *CoRR*, abs/2307.07514, 2023.

36 João Marques-Silva and Alexey Ignatiev. Delivering trustworthy AI through formal XAI. In *AAAI*, pages 12342–12350, 2022.

37 Carlos Mencia, Alexey Ignatiev, Alessandro Previti, and Joao Marques-Silva. MCS extraction with sublinear oracle queries. In *SAT*, pages 342–360, 2016.

38 Carlos Mencia, Alessandro Previti, and Joao Marques-Silva. Literal-based MCS extraction. In *IJCAI*, pages 1973–1979, 2015.

39 Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.*, 267:1–38, 2019.

40 Rishabh Misra and Prahal Arora. Sarcasm detection using news headlines dataset. *AI Open*, 4:13–18, 2023.

41 Rishabh Misra and Jigyasa Grover. *Sculpting Data for ML: The first act of Machine Learning.* 01 2021.

42 Vinod Nair and Geoffrey Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

43 Nina Narodytska, Nikolaj Bjørner, Maria-Cristina V. Marinescu, and Mooly Sagiv. Core-guided minimal correction set and core enumeration. In *IJCAI*, pages 1353–1361, 2018.

44 Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, pages 8024–8035, 2019.

45 Alessandro Previti and João Marques-Silva. Partial MUS enumeration. In *AAAI*. AAAI Press, 2013.

46 Alessandro Previti, Carlos Mencía, Matti Järvisalo, and João Marques-Silva. Premise set caching for enumerating minimal correction subsets. In *AAAI*, pages 6633–6640, 2018.

47 J. Scott Provan and Michael O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM J. Comput.*, 12(4):777–788, 1983.

48 Raymond Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

49 Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?": Explaining the predictions of any classifier. In *KDD*, pages 1135–1144, 2016.

50 Ronald L. Rivest. Learning decision lists. *Mach. Learn.*, 2(3):229–246, 1987.

51 Andy Shih, Arthur Choi, and Adnan Darwiche. A symbolic approach to explaining Bayesian network classifiers. In *IJCAI*, pages 5103–5111, 2018.

52 Salil Vadhan. The complexity of counting in sparse, regular, and planar graphs. *SIAM J. Comput.*, 31(2):398–427, 2001.

53 L. G. Valiant. The complexity of computing the permanent. *Theoret. Comput. Sci.*, 8(2):189–201, 1979.

54 William Webber, Alistair Moffat, and Justin Zobel. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS)*, 28(4):1–38, 2010.

55 Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. MedMNIST v2-a large-scale lightweight benchmark for 2D and 3D biomedical image classification. *Scientific Data*, 10(1):41, 2023.

56 Jinqiang Yu, Alexey Ignatiev, and Peter J. Stuckey. On formal feature attribution and its approximation. *CoRR*, abs/2307.03380, 2023. `arXiv:2307.03380`.