# Scaling multi-label conformal prediction with label interactions for a large number of labels

Ghassan Najjar, Céline Berthou, and Héléna Vorobieva

Safran Tech, Rue des Jeunes Bois, Châteaufort, 78114 Magny-Les-Hameaux, France
{ghassan.najjar, celine.berthou, helena.vorobieva}@safrangroup.com

**Abstract.** Multi-label classification is the task where a single instance may belong to multiple classes simultaneously. The Label Powerset approach (LP) allows to apply Inductive Conformal Prediction (ICP) on multi-label classification tasks, by considering each label set as a single class and by assigning a non-conformity score to each of them. The construction of the prediction set $\mathcal{C}$ requires selecting all the label sets –represented as binary vectors– that satisfy a given conformity criterion. Since the number of possible outputs is exponentially growing with the number of classes, constructing $\mathcal{C}$ by testing the conformity criterion on all cases is unaffordable. We propose an algorithm that efficiently computes $\mathcal{C}$, even in the difficult case where the non-conformity score involves label interactions. It is based on a customized partial order relation on the set of binary vectors coupled with a monotone lower bound of the non-conformity score. Our tests confirm the algorithm's efficiency, even with a high class count.

**Keywords:** Multi-label Classification · Inductive Conformal Prediction · Label Powerset · Computational Efficiency · Trees

## 1 Introduction

In the realm of machine learning, multi-label classification is a complex and challenging problem, where a single data instance can belong to multiple classes simultaneously. This scenario is ubiquitous in various real-world applications, such as text classification and medical diagnosis. For instance, in medical diagnosis, the patient may suffer from multiple diseases at the same time. In the case of multi-label classification with $c$ classes, given an input, the classifier predicts a set of classes, instead of predicting one single class.
In high-sensitivity tasks and applications requiring certification such as in the aeronautical or health industries, it is valuable to provide statistically valid uncertainty quantification. Conformal Prediction [9] tackles this challenge by extending traditional classification outputs, providing a set of multiple possible outputs with explicit guarantees under the assumption of data exchangeability. Within the family of conformal prediction methods, several techniques have been proposed for multi-label classification, such as Binary Relevance Conformal Prediction (BRCP) [2] and the Label-Powerset Conformal Prediction (LPCP) [5].

BRCP transforms the problem into a binary classification task for each class, but does not natively account for dependencies between different classes: for a given $\alpha > 0$, it applies Conformal Prediction (CP) independently with a significance level of $\frac{\alpha}{c}$ for each label, applying then the union bound gives a significance level of $\alpha$ for the whole prediction. This method is not adapted for large values of $c$ because of the $\frac{\alpha}{c}$ fraction. Another approach [8] employs a hierarchical tree and uses the technique of multiple hypothesis testing to address multi-label conformal prediction. However, this method requires to construct a tree with more than $2^c$ nodes, which is infeasible for very large $c$.
LPCP modifies the problem into a multi-class mono-label one by treating each label set as a unique class, and applies classical CP methods for mono-label problems. LPCP has the advantage of enabling the use of non-conformity scores that account for label dependencies. We therefore chose to follow the LPCP method and model each label set as a vector $y \in \{0,1\}^c$: a 1 coordinate at position $i$ corresponding to the presence of class $i$.
By calibration for a chosen threshold $\alpha > 0$, under exchangeability assumption of data, ICP [6] returns prediction regions with marginal coverage guarantee of at least $1 - \alpha$. This method needs a preliminary definition of a non-conformity score, which assigns a real number to each label set. The non-conformity score introduced in [5] accounts for interactions between labels by penalizing combinations that never simultaneously appeared in the data. This allows to significantly reduce the size of the conformal prediction sets, as shown in [5] and [4].

The main disadvantage of LPCP is its high computational complexity. The work [3] solves this problem in the case of a large number of labels (tested up to $c = 90$ labels), with an $L^p$ non-conformity score, which does not account for label interactions.
The present work aims to reduce the computational complexity in the case of non-conformity scores that account for label interactions, as in [5] and [4]. We aim to find an algorithm that constructs the prediction set efficiently in the challenging case of label-interacting non-conformity scores.
For the purpose of certification, it is essential to compute the prediction set exactly, ensuring that no elements are omitted. The following section recalls the ICP framework and states the problem. Section 3 gives a first algorithmic solution (ECP) of the problem of prediction set construction and an optimized variant (CAECP). In section 4, we present an even more optimized algorithm (PBECP) based on CAECP.

## 2 Problem Statement

### 2.1 Preliminaries

ICP assumes we have a calibration set denoted as $(X_i, Y_i)_{i \leq n}$, where $X_i$ and $Y_i$ are inputs and outputs. The $X_i$ and $Y_i$ are considered here as random variables in $\mathcal{X} \times \{0,1\}^c$ with $\mathcal{X}$ the input space, $c$, the number of classes and $n$ the number of data. Let $(X_{test}, Y_{test})$, denote a test point such that the dataset

$((X_i, Y_i)_{i\leq n}, (X_{test}, Y_{test}))$ is exchangeable.
Assume we have a pre-trained model $\hat{f}$ trained on the set $\mathcal{D}_{pre-trained}$ assumed independent of the calibration data and test point. The output of $\hat{f}$ can be seen as scores assigned to each class:

$$\hat{f} : \mathcal{X} \to [0,1]^c$$

Then ICP can use this underlying model through a *non-conformity* score, which is allowed to depend on $\mathcal{D}_{pre-trained}$:

$$s : [0,1]^c \times \{0,1\}^c \to \mathbb{R}$$

$s(\hat{f}(x), y)$ is a measure of dissimilarity between the model prediction $\hat{f}(x)$ and the candidate output $y$. For each input $X_i$, we denote by $\hat{Y}_i := \hat{f}(X_i) \in [0,1]^c$. Let $\alpha > 0$ and let $q$, be the empirical quantile of order $\frac{(n+1)(1-\alpha)}{n}$ of the calibration scores $s(\hat{Y}_i, Y_i)$. This quantile is used to form a prediction set for the new input $X_{test}$:

$$C(X_{test}) = \{y \in \{0,1\}^c | s(\hat{Y}_{test}, y) \leq q\} \tag{1}$$

Under exchangeability assumption of all the $n+1$ samples, the marginal coverage guarantee holds:

$$\mathbb{P}\left(Y_{test} \in C(X_{test})\right) \geq 1 - \alpha \tag{2}$$

Where the probability is taken over both the calibration data and the test point. During inference, for a given sample $x_{test}$, which is a realization of $X_{test}$, the prediction set $C(x_{test})$ is useful in practice only if $\text{Card}(C(x_{test}))$ is low. It is crucial to thoughtfully design the $s$ function to achieve this goal. Thus, we opted for a non-conformity score that considers label interactions. This approach allows us to incorporate practical prior knowledge, such as: "if two labels have never appeared simultaneously in the training data, then it is unlikely that they to co-occur in the future". The non-conformity score introduced in [5] addresses the two mentioned requirements:

$$\forall\ \hat{y} \in [0,1]^c, \forall\ y \in \{0,1\}^c, s_\mu(\hat{y}, y) = \sum_{i=1}^{c} |\hat{y}_i - y_i|^\phi + \lambda \sum_{1\leq i\leq j\leq c} \mu_{i,j} y_i y_j \tag{3}$$

Where $\lambda \geq 0$, $\phi > 0$, $\mu_{i,j} = 0$ if labels $i$ and $j$ have been simultaneously observed in the training set, and $\mu_{i,j} = 1$ otherwise. This non-conformity score allows the reduction of the cardinality of the prediction set $C(x_{n+1})$. However, constructing $C(x_{n+1})$ is computationally very demanding: a naive approach would require generating the $2^c$ vectors $y$ to test whether $s(\hat{y}_{n+1}, y) \leq q$.

### 2.2 Main Contribution and General Setting of the problem

Our work proposes an efficient way to compute exactly the prediction set $C(x_{n+1})$ for the non-conformity score (3) and even for a more general one:

$$s_\mu(\hat{y}, y) = \sum_{i=1}^{c} \varphi(|\hat{y}_i - y_i|) + \sum_{1\leq i\leq j\leq c} \mu_{i,j} y_i y_j \tag{4}$$

Where the parameters $(\mu_{i,j})_{1\leq i\leq j\leq c}$ are arbitrary elements of $\mathbb{R}^+$ and $\varphi : \mathbb{R}^+ \to \mathbb{R}$ is an arbitrary increasing function. $(\mu_{i,j})_{1\leq i\leq j\leq c}$ can be chosen *a priori* or can be the result of a data-based optimization on $\mathcal{D}_{pre-trained}$.

In order to compute the prediction set, we make the following assumptions. Firstly, we assume that $q$ has been calibrated and that we are at inference stage. Given a new input $x$ -or more precisely the underlying classifier's raw predictions $\hat{y}$- we want to compute:

$$C_{\mu,q}(\hat{y}) = \{y \in \{0,1\}^c | s_\mu(\hat{y}, y) \leq q\} \tag{5}$$

Fixing $\hat{y}$ for the rest of the paper, we can simplify the notation and write:

$$s(y) := s_\mu(\hat{y}, y) \tag{6}$$

Secondly, we assume from now on that the coordinates of the vector $\hat{y} \in [0,1]^c$ are in **decreasing order**. To achieve that, one can apply a permutation $\sigma \in \mathfrak{S}_c$ such that the coordinates of $\hat{y}^\sigma := (\hat{y}_{\sigma(i)})_i$ are in decreasing order. If we consider the permuted vector $y^\sigma := (y_{\sigma(i)})_i$ and the new family $(\mu^\sigma_{i,j})_{i\leq j}$ defined by

$$\mu^\sigma_{i,j} = \begin{cases} \mu_{\sigma(i),\sigma(j)} & \text{if } \sigma(i) \leq \sigma(j) \\ \mu_{\sigma(j),\sigma(i)} & \text{else} \end{cases} \tag{7}$$

Then we have this permutation invariance property:

**Proposition 1.**

$$s_{\mu^\sigma}(\hat{y}^\sigma, y^\sigma) = s_\mu(\hat{y}, y) \tag{8}$$

See proof in appendix.[1]
Using $\sigma$ is not constraining: if the coordinates of $\hat{y}$ are not in a decreasing order, we solve first the problem with $\mu^\sigma$ and $\hat{y}^\sigma$. From the solution $C_{\mu^\sigma,q}(\hat{y}^\sigma)$ we then get the solution of the initial problem $C_{\mu,q}(\hat{y})$ applying the permutation $\sigma^{-1}$ to each vector of $C_{\mu^\sigma,q}(\hat{y}^\sigma)$.

## 3 (Children-Anticipating) Efficient Conformal Prediction: An efficient computation of the prediction set based on single-root candidate generation

### 3.1 Definition of an exploration tree of $\{0,1\}^c$

Our goal is to construct a tree $\mathcal{T}$ whose nodes are the elements of $\{0,1\}^c$. This tree explores these elements in an order which enables pruning and saves time.

[1] The key proofs of the paper are presented in the main document, while less critical proofs are included in the appendix: https://github.com/Ghassan01/appendix_icp/

**Definition 1.** *The max-1-position of a binary vector $y \in \{0,1\}^c$, denoted as $l(y)$ is defined the following way:*

$$l(y) = \begin{cases} max\{i \in [\![1;c]\!] | y_i = 1\} & \text{if } \{i \in [\![1;c]\!] | y_i = 1\} \neq \emptyset \\ 0 & \text{else} \end{cases} \tag{9}$$

It is the last index of 1-coordinate in $y$ if there exists at least one.

**Definition 2.** *We denote by $l_0$ the max-1-position of the vector $(1_{\hat{y}_i > 0.5})_i$.*

From the decreasing order assumption of $\hat{y}$, $l_0$ is also the number of coordinates of $\hat{y}$ that are above 0.5.

**Definition 3.** *We define the binary relation $\leq$ on $\{0,1\}^c$ the following way:*

$$\forall y, z \in \{0,1\}^c, \quad y \leq z \Leftrightarrow (y_i)_{i \leq l(y)} = (z_i)_{i \leq l(y)} \tag{10}$$

Meaning $y \leq z$ and $y \neq z$ if and only if $z$ is obtained by replacing at least one of the right zeros (the zeros at a position greater than $l(y)$) of $y$ by a 1. We clearly observe that:

**Proposition 2.** *The binary relation $\leq$ on $\{0,1\}^c$ is a partial order relation.*

We can now construct the tree $\mathcal{T}$ whose nodes are the elements of $\{0,1\}^c$, rooted at the null vector $0 \in \{0,1\}^c$, such that $y \in \{0,1\}^c$ is an ascendant of $z \in \{0,1\}^c$ if and only if $y \leq z$. Figure 1 shows $\mathcal{T}$ in the case $c = 4$. The partial order relation provides a natural and beneficial order of exploration of the elements of $\{0,1\}^c$ following this tree. It is clear that:

**Proposition 3.** *The depth $l$ of tree $\mathcal{T}$ is composed of binary vectors containing exactly a number $l$ of 1 coordinates*

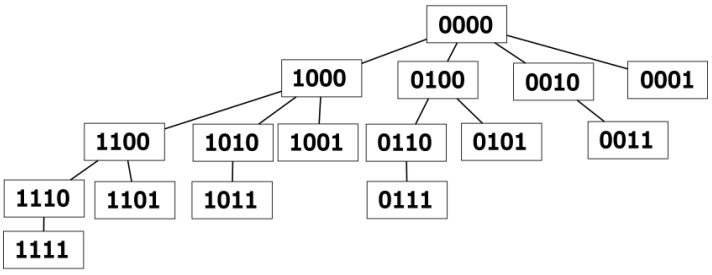


Fig. 1: $\mathcal{T}$ in the case $c = 4$

Next, we explore the nodes of the tree in a depth-first-search manner to construct $C_{\mu,q}(\hat{y})$ efficiently. At each node $y$, we choose whether to include it in $C_{\mu,q}(\hat{y})$, and in the next section, we establish a criterion to test whether we can prune all the descendants of $y$ to exclude them from the search process.

### 3.2 Lower bound of the non-conformity score

**Definition 4.** *We define the function $m$ defined on $\{0,1\}^c$, the following way:*

$$m(y) = \sum_{i=1}^{l(y)} \varphi(|\hat{y}_i - y_i|) + \sum_{i=l(y)+1}^{c} \varphi(|\hat{y}_i - 1_{\hat{y}_i>0.5}|) + \sum_{1\leq i\leq j\leq c} \mu_{i,j} y_i y_j \qquad (11)$$

**Proposition 4.** *$m$ is a lower bound of $s$.*

*Proof.* $s(y) - m(y) = \sum_{i=l(y)+1}^{c} \varphi(|\hat{y}_i - y_i|) - \varphi(|\hat{y}_i - 1_{\hat{y}_i>0.5}|)$.
Since the $y_i$ are in $\{0,1\}$, it is clear that $\forall i \in [\![1;c]\!], \quad |\hat{y}_i - 1_{\hat{y}_i>0.5}| \leq |\hat{y}_i - y_i|$.
We conclude using the increasing property of $\varphi$. □

*Remark 1.* Given that the higher $l(y)$ is, the closer $m(y)$ is to $s(y)$, we remark that $m$ is a relatively close lower bound of $s$. If $l_0$ is the max-1-position of the vector $(1_{\hat{y}_i>0.5})_{1\leq i\leq c}$, the proposition 5 gives a bound of the minoration gap.

**Proposition 5.**

$$\forall y \in \{0,1\}^c, \quad 0 \leq s(y) - m(y) \leq (l_0 - l(y))^+ ||\varphi||_\infty$$

*With the convention $0 \times \infty = 0$ if $(l_0 - l(y))^+ = 0$ and $||\varphi||_\infty = \infty$.*

*Remark 2.* $(.)^+$ denotes the positive part function. In particular, if $y$ contains a 1 coordinate at a position higher than $l_0$, then we have the strong equality $s(y) = m(y)$. As $l_0$ is often expected to be low in practice, $m$ is a high-quality lower bound.

The introduced partial order on $\{0,1\}^c$ and the lower bound $m$ are particularly significant as $m$ is monotonic:

**Proposition 6.** *The lower bound $m$ is increasing for $"\leq"$.*

*Proof.* Let $y, z \in \{0,1\}^c$ such that $y \leq z$. Hence, $l(y) \leq l(z)$ and we obtain

$$\begin{aligned} m(z) - m(y) = &\sum_{i=l(y)+1}^{l(z)} \varphi(|\hat{y}_i - z_i|) - \varphi(|\hat{y}_i - 1_{\hat{y}_i>0.5}|) \\ &+ \sum_{1\leq i\leq j\leq c} \mu_{i,j}((z_i - y_i)z_j + (z_j - y_j)y_i) \end{aligned}$$

The terms $\varphi(|\hat{y}_i - z_i|) - \varphi(|\hat{y}_i - 1_{\hat{y}_i>0.5}|)$ are non-negative since $\varphi$ is increasing. The terms $\mu_{i,j}((z_i - y_i)z_j + (z_j - y_j)y_i)$ are also non-negative because $l(y) \leq l(z)$ implies that $\forall i \in [\![1;c]\!], \quad y_i \leq z_i$. □

This monotonicity implies in $\mathcal{T}$ higher values of $m$ for the descendants than for the father: then, excluding a node $y$ following the criterion $m(y) > q$ allows to exclude the whole descendance of $y$.

### 3.3 A first candidate generation algorithm

In this section, we present Algorithm 1, Efficient Conformal Prediction (ECP), which generates a reasonable amount of candidates for the prediction set through a clever node exploration of the tree $\mathcal{T}$. The cornerstone results from:

**Proposition 7.** *If $y \in \{0,1\}^c$ and $m(y) > q$, then*

$$\forall z \in \{0,1\}^c, \quad z \geq y \implies z \notin C_{\mu,q}(\hat{y})$$

*Proof.* This follows directly from: 1) $m$ is a lower bound of $s$, 2) the elements of $C_{\mu,q}(\hat{y})$ satisfy $s(y) \leq q$, 3) $m$ is an increasing function. □

**Definition 5.** *We denote by $M(r)$, the descendance of $r$ in the tree, ie.*

$$M(r) = \{y \in \{0,1\}^c | r \leq y\} \tag{12}$$

*Remark 3.* If $m(y) > q$, then neither $y$ nor elements of $M(y)$ are in $C_{\mu,q}(\hat{y})$.

ECP performs a tree traversal, pruning great parts as it progresses. Importantly, constructing the entire tree is unnecessary (generating all the $2^c$ nodes is impractical for large $c$ values). Instead, only the direct children of the current node are generated, when beneficial, and explored recursively. Using Proposition 7, ECP efficiently avoids generating many unnecessary nodes.

**Algorithm 1** Efficient Conformal Prediction (ECP)

**Input:** $r$, $\hat{y}$, $\mu$, $q$
**Output:** $C_{\mu,q}(\hat{y}) \cap M(r)$

```
1: C ← ∅
2: E ← {r}                                   ▷ Nodes to explore
3: while E ≠ ∅ do
4:     Let p ∈ E
5:     E ← E − {p}
6:     if m(p) ≤ q then
7:         Generate F, set of children of p
8:         E ← E ∪ F
9:         if s_μ(ŷ, p) ≤ q then C ← C ∪ {p}
10: return C
```

*ECP algorithm iteratively performs pruning at line 6 and generates children at line 7*

Algorithm 1 returns $C_{\mu,q}(\hat{y}) \cap M(r)$, given $r$ as input, that is to mean all the elements of $C_{\mu,q}(\hat{y})$ in the descendance of a given node $r$.

*Remark 4.* $M((0)_{1 \leq i \leq c}) = \{0,1\}^c$, which implies the Proposition 8.

**Proposition 8.** ***Algorithm 1, applied with $r = 0$ solves the problem of constructing $C_{\mu,q}(\hat{y})$.***

ECP hence provides a first solution to our problem.

### 3.4 CAECP: an optimized ECP based on children anticipation

In Algorithm 1, a node is generated if its parent $p$ has been generated and satisfies $m(p) \leq q$. Here we propose an optimization of ECP, which consists in adding a supplementary condition on $p$ to generate its children. In this section, we introduce Algorithm 2, Children-Anticipating Efficient Conformal Prediction (CAECP), which may stop exploring a branch one step earlier than ECP.

**Definition 6.** *Assume that $l(p) \neq c$, meaning $p$ has a non empty set of children, then we define $Q(p)$ as:*

$$Q(p) = \left(\sum_{i=1}^{l(p)} \varphi(|\hat{y}_i - p_i|)\right) + \varphi(|\hat{y}_{l(p)+1} - 1|) + \sum_{i=l(p)+2}^{c} \varphi(|\hat{y}_i - 1_{\hat{y}_i > 0.5}|) + \sum_{1 \leq i \leq j \leq c} \mu_{i,j} p_i p_j$$

**Proposition 9.** *If $y$ is a direct child of $p$, then $m(y) \geq Q(p)$.*

**Corollary 1.** $\forall\ p \in \{0,1\}^c : l(p) \neq c, \quad Q(p) > q \implies (\forall\ y > p, \quad y \notin C_{\mu,q}(\hat{y}))$

*Proof.* This follows from: if $y > p$, then there exists a direct child $y'$ of $p$ such that $y \geq y'$. □

The correctness of CAECP is a direct consequence of corollary 1. Next proposition shows that the condition $Q(p) \leq q$ is stronger than $m(p) \leq q$, which proves that CAECP is an optimization of ECP, generating less useless vectors.

**Algorithm 2** Children-Anticipating Efficient Conformal Prediction (CAECP)

**Input:** root $r$, $\hat{y}$, $\mu$, $q$
**Output:** $C_{\mu,q}(\hat{y}) \cap M(r)$

```
1: C ← ∅
2: E ← {r}                                          ▷ Nodes to explore
3: while E ≠ ∅ do
4:     Let p ∈ E
5:     E ← E − {p}
6:     if s_μ(ŷ, p) ≤ q then C ← C ∪ {p}
7:     if l(p) ≠ c then
8:         if Q(p) ≤ q then                         ▷ Stronger condition than m(p) ≤ q
9:             Generate F, set of children of p
10:            E ← E ∪ F
11: return C
```

*CAECP behaves as ECP, with a stronger condition for children generation (line 8)*

**Proposition 10.** *The condition $Q(p) \leq q$ is stronger than $m(p) \leq q$ since:*

$$\forall\ p \in \{0,1\}^c \text{ such that } l(p) \neq c, \quad m(p) \leq Q(p)$$

*Remark 5.* The proof of this result (cf. appendix) shows that the introduction of $Q(p)$ is useful only for those $p$ that satisfy $l(p) \geq l_0$. Otherwise, $Q(p) = m(p)$.

ECP is conceptually simpler and is sufficient in most cases in practice. Since CAECP is an optimization of ECP, and takes the same input and returns the same output, we only mention CAECP until the end of the paper.

## 4 Prefix-Based ECP (PBECP): An optimized algorithm based on multi-root candidate generation

Given any root vector $r$, CAECP returns $C_{\mu,q}(\hat{y}) \cap M(r)$ . To construct $C_{\mu,q}(\hat{y})$ one just needs to apply CAECP with $r = 0$. Here we expose a new algorithm which is useful when the parameters $(\mu_{i,j})_{1\leq i\leq j\leq c}$ have many zero values.
**Preliminary example**. Let $c = 100$ and $l_0 = 15$ ( ie. $\forall i \in [\![1; 15]\!], \quad \hat{y}_i > 0.5$ since the $(\hat{y}_i)_i$ are in decreasing order). In that case, the binary vectors composing $C_{\mu,q}(\hat{y})$ are likely to contain many 1 coordinates (approximately 15, for small $q$). According to Proposition 3, those vectors are at depth around 15 in $\mathcal{T}$. Let $y \in C_{\mu,q}(\hat{y})$, if we apply directly CAECP with root $r$=(0), all vectors on the path between (0) and $y$ in $\mathcal{T}$ are generated.
We understand on this example that we could have an optimized version if instead of executing CAECP with root $r = (0)$, we execute CAECP with more 1-containing roots such as $(1, 1, 1, 1, 1, 0, ..., 0)$. The difficulty is to ensure that we do not forget any of the smaller binary vectors. In this section, to select roots without exploring all of them, we use a differently built tree with another goal.

### 4.1 Definitions

**Definition 7.** *Assuming that* $\{k \in [\![1; l_0]\!] | \sum_{1\leq i\leq j\leq k} \mu_{i,j} = 0\} \neq \emptyset$*, we define*

$$l_1 := max\{k \in [\![1; l_0]\!] | \sum_{1\leq i\leq j\leq k} \mu_{i,j} = 0\} \tag{13}$$

This is the maximum integer $l_1$ such that $y := (1_{i\leq l_1})_{i\leq c}$ has no penalization $\sum_{1\leq i\leq j\leq k} \mu_{i,j} y_i y_j$. We suppose from now on that $l_1 \geq 2$.

**Definition 8.** *The set of* $l_1$*-prefixes is defined as*

$$Pref_{l_1} := \{y \in \{0, 1\}^c | (y_i)_{i>l_1} = (0)\} \subset \{0, 1\}^c \tag{14}$$

**Definition 9.** *We define the* $l_1$*-completion of a prefix* $p \in Pref_{l_1}$ *as:*

$$Comp_{l_1}(p) := \{z \in \{0, 1\}^c | (z_i)_{i\leq l_1} = (p_i)_{i\leq l_1}\} \subset \{0, 1\}^c \tag{15}$$

*Remark 6.* For a prefix $p \in Pref_{l_1}$, $Comp_{l_1}(p) \subset M(p)$ but we do not have mutual inclusion, because we do not necessarily have $l_1 = l(p)$.

*Remark 7.* $\bigcup_{p\in Pref_{l_1}} Comp_{l_1}(p) = \{0, 1\}^c$ and the union is disjoint.

### 4.2 Construction of a tree on $Pref_{l_1}$

We start by defining a tree on $Pref_{l_1}$, to establish a prefix exploration order.

**Definition 10.** *We define the tree $\mathcal{T}_{pref}$ rooted on $r = (1_{i\leq l_1})_{i\leq c}$: the only child of $r$ is $(1_{i\leq l_1-1})_{i\leq c}$ and given a node $p \neq r$ of $\mathcal{T}_{pref}$, let's denote by $i_{first0}$ the smallest index of a zero coordinate of $(p_i)_{i\leq l_1}$.*

1. *If $i_{first0} > 1$ then its left child $p^{(left)}$ is obtained from $p$ by exchanging $p_{i_{first0}}$ and $p_{i_{first0}-1}$ and the right child $p^{(right)}$, by imposing $p^{(right)}_{i_{first0}-1} = 0$.*
2. *If $i_{first0} = 1$ (that is to mean $p$ starts by a 0), then $p$ has no child.*

Figure 2 shows $\mathcal{T}_{pref}$ in the case $c = 6$ and $l_1 = 4$.

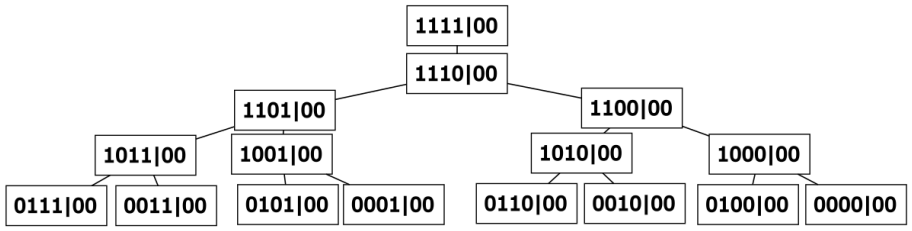


Fig. 2: $\mathcal{T}_{pref}$ for c=6 and $l_1$=4. The bar | delimits pre-$l_1$ from post-$l_1$ coordinates

**Proposition 11.** *The set of nodes of $\mathcal{T}_{pref}$ is the whole set $Pref_{l_1}$.*

Hence we do not forget any element of $Pref_{l_1}$ by exploring $\mathcal{T}_{pref}$.

**Proposition 12.** *$\mathcal{T}_{pref}$ naturally defines a partial order relation $\leq_{pre}$ on its nodes $Pref_{l_1}$ ($p_1 \leq_{pre} p_2 \iff p_1$ is an ascendant of $p_2$), and the quantity:*

$$g(p) = \sum_{i=1}^{l_1} \varphi(|\hat{y}_i - p_i|) + \sum_{i=l_1+1}^{c} \varphi(|\hat{y}_i - 1_{\hat{y}_i>0.5}|) \tag{16}$$

*is an increasing function of $p$ for $\leq_{pre}$ and is a lower bound of $s$. In addition, for any node $p$ having two children, $g(p^{(left)}) \leq g(p^{(right)})$.*

*Proof.* The lower bound property and $\leq_{pre}$ defining a partial order are clear. The monotonicity of $g$ for $\leq_{pre}$, and the comparison between both children follow from $l_1 \leq l_0$ and the decreasing order assumption of $\hat{y}_i$. □

**Definition 11.** *We define the set of selected prefixes $SelPref_{l_1}$ as:*

$$SelPref_{l_1} := \{p \in Pref_{l_1} | g(p) \leq q\} \tag{17}$$

*Remark 8.* The non-conformity score $s$ also increases on this tree, but to uphold Proposition 13, $s$ cannot replace $g$ to construct $SelPref_{l_1}$, unless $l_1$ is maximal (cf. Proposition 15). $SelPref_{l_1}$ is smaller than $Pref_{l_1}$: this filtering narrows down the subparts of $\mathcal{T}$ on which to apply CAECP. Proposition 13 guarantees that this approach suffices to fully construct $C_{\mu,q}(\hat{y})$.

We use Proposition 12 to design Algorithm 3: "Efficient Prefix Selection" (EPS). EPS returns $SelPref_{l_1}$ efficiently, following the same pruning strategy than the ECP algorithm, replacing the function $m$ by the function $g$. Furthermore, we use the supplementary optimization of generating $p^{(right)}$ only if $g(p^{(left)}) \leq q$.

**Algorithm 3** Efficient Prefix Selection (EPS)

**Input:** $\hat{y}$, $\mu$, $q$
**Output:** $SelPref_{l_1}$

Compute $l_1$ (function of $\hat{y}$ and $\mu$)
$r = (1_{i \leq l_1})_{i \leq c}$
$P \leftarrow \emptyset$
$E \leftarrow \emptyset$ ▷ Nodes that are already in $P$ and whose children are to be explored
**if** $g(r) \leq q$ **then** $E \leftarrow \{r\}$, $P \leftarrow \{r\}$
**while** $E \neq \emptyset$ **do**
    Let $p \in E$
    $E \leftarrow E - \{p\}$
    **if** $p$ is the root node and is not a leaf **then**
        Generate $p'$, the only child of $p$ in $\mathcal{T}_{pref}$
        **if** $g(p') \leq q$ **then** $E \leftarrow E \cup \{p'\}$, $P \leftarrow P \cup \{p'\}$
    **if** $p$ is not the root node and is not a leaf **then**
        Generate $p^{(left)}$
        **if** $g(p^{(left)}) \leq q$ **then**
            $E \leftarrow E \cup \{p^{(left)}\}$, $P \leftarrow P \cup \{p^{(left)}\}$
            Generate $p^{(right)}$
            **if** $g(p^{(right)}) \leq q$ **then** $E \leftarrow E \cup \{p^{(right)}\}$, $P \leftarrow P \cup \{p^{(right)}\}$
**return** $P$

*EPS pre-selects relevant prefixes for CAECP algorithm. Like ECP, EPS applies a pruning framework but limited to prefixes, with another increasing lower bound $g$ and on another tree $\mathcal{T}_{pref}$. In addition, we first compute $g$ on the left child before computing $g$ on the right one.*

### 4.3 PBECP algorithm

In this section, we present the Prefix-Based Efficient Conformal Prediction algorithm (PBECP), using EPS as an upstream component.

**Definition 12.** *We define the first-stage-$l_1$-completion of a prefix $p \in Pref_{l_1}$:*

$$FSC_{l_1}(p) := \{z \in \{0,1\}^c | (z_i)_{i \leq l_1} = (p_i)_{i \leq l_1} \text{ and } \exists! j > l_1, \quad z_j = 1\} \tag{18}$$

We use this concept in Proposition 13 which states that the computation of $C_{\mu,q}(\hat{y})$ reduces to multiple calls of CAECP.

**Proposition 13.** *The set $C_{\mu,q}(\hat{y})$ is decomposable into a **disjoint union** of simpler sets:*

$$C_{\mu,q}(\hat{y}) = \bigcup_{p \in SelPref_{l_1}} \left( (C_{\mu,q}(\hat{y}) \cap \{p\}) \cup \bigcup_{p' \in FSC_{l_1}(p)} C_{\mu,q}(\hat{y}) \cap M(p') \right)$$

The terms $C_{\mu,q}(\hat{y}) \cap M(p')$ of Proposition 13 can be computed by calling CAECP with $r = p'$, and since the unions are all disjoint, PBECP does not explore the same parts of $\mathcal{T}$ twice. We always have $SelPref_{l_1} \subset Pref_{l_1}$, and our goal is for $SelPref_{l_1}$ to be a very smaller subset. This would ensure that PBECP explores only a fraction of the tree $\mathcal{T}$ compared to CAECP, which is the cornerstone of its efficiency.

**The Prefix-Based Efficient Conformal Prediction algorithm (PBECP)** Using Proposition 13, we propose Algorithm 4 (PBECP), which first performs prefix selection and then calls CAECP on judiciously chosen roots.

**Algorithm 4** Prefix-Based Efficient Conformal Prediction (PBECP)

**Input:** $\hat{y}$, $\mu$, $q$
**Output:** $C_{\mu,q}(\hat{y})$

1: Compute $l_1$ (function of $\hat{y}$ and $\mu$)
2: Execute EPS: $SelPref_{l_1} = EPS(l_1, \hat{y}, \mu, q)$
3: $SP \leftarrow SelPref_{l_1}$
4: $C \leftarrow \emptyset$
5: **while** $SP \neq \emptyset$ **do**
6: Let $p \in SP$
7: $SP \leftarrow SP - \{p\}$
8: **if** $s_\mu(\hat{y}, p) \leq q$ **then** $C \leftarrow C \cup \{p\}$
9: **for** $p' \in FSC_{l_1}(p)$ **do**
10: $C \leftarrow C \cup CAECP(r = p')$ ▷ Calls of CAECP with different roots
11: **return** $C$

*PBECP is an alternative to directly calling CAECP(r=0), it rather calls CAECP(r) for multiple judiciously chosen $r$. At line 10, we call CAECP(r) for $r$ varying in the first-stage completions of the elements of SP.*

**Commentary on the optimization** For computational efficiency, we must ensure that the multiple calls of CAECP($r = p'$) in Algorithm 4 do not lead to redundant computations.

**Proposition 14.** *In PBECP, we do not explore the same node of $\mathcal{T}$ twice. More precisely, for two different $p'_1$ and $p'_2$ appearing at line 10 of Algorithm 4 - corresponding to different stages of the execution- the respective calls CAECP($r$=$p'_1$) and CAECP($r$=$p'_2$) explore disjoint sets of vectors.*

In the frequent case where $l_1 = l_0$, the following proposition explains the optimality of lower bound functions used in PBECP in this case.

**Proposition 15.** *If $l_1$ is maximal (ie. $l_1 = l_0$), then:*

1. *During the execution of EPS at line 2 of Algorithm 4, the lower bound $g$ of the non-conformity score $s$ is always equal to $s$.*
2. *During the execution of CAECP($r = p'$) at line 10 of Algorithm 4, the lower bound $m$ of the non-conformity score $s$ is always equal to $s$.*

**Optimal case.** If $l_1 = l_0$, then PBECP operates as if traversing a virtual tree $\mathcal{T}_{virtual}$ defined over all binary vectors. The non-conformity score is an increasing function on that tree and is directly used for the pruning of the descendants. $\mathcal{T}_{virtual}$ is obtained from $\mathcal{T}_{pref}$ with the following steps:

1. Start with $\mathcal{T}_{pref}$.
2. For each node $p$ of $\mathcal{T}_{pref}$, let $F$ be its set of children in $\mathcal{T}_{pref}$, then $F \leftarrow F \cup FSC_{l_1}(p)$. We obtain the tree $\mathcal{T}'$.
3. For each leaf $y$ of $\mathcal{T}'$ element of a certain $FSC_{l_1}(p)$, we expand it into the subtree of $\mathcal{T}$ rooted at this $y$.
4. **We obtain a tree on the whole set $\{0,1\}^c$ on which the non-conformity score is increasing**.

This shows the optimality of PBECP in practical cases where $l_1$ is maximal, which appears when $\mu$ is defined as in [5] and where the labels predicted by $(1_{\hat{y}_i>0.5})_i$ have been seen simultaneously pairwise in the training dataset (which is very common).

For computational complexity computation, we introduce the following:

**Definition 13.** *For a given algorithm $A$, we denote by $\mathcal{G}(A)$ the following set:*

$$\mathcal{G}(A) = \{y \in \{0,1\}^c |\ y \text{ is generated during execution of } A\}$$

*Remark 9.* The correctness of algorithms 2, 3, and 4 implies that:

$$C_{\mu,q}(\hat{y}) \subset \mathcal{G}(CAECP(r=0)),\ SelPref_{l_1} \subset \mathcal{G}(EPS),\ C_{\mu,q}(\hat{y}) \subset \mathcal{G}(PBECP)$$

*Remark 10.* $\text{Card}\,(\mathcal{G}(A))$ is a measure of the complexity of algorithm $A$, and we aim to minimize this value. It is worth noting that, in any case, the naive method yields a complexity of $\text{Card}\,(\mathcal{G}(naive)) = 2^c$.
During execution, many binary vectors are generated, but they are not all kept in the result $C_{\mu,q}(\hat{y})$. Note that since $A$ is an algorithm which computes $C_{\mu,q}(\hat{y})$, we at least generate $\text{Card}(C_{\mu,q}(\hat{y}))$ vectors: hence $\text{Card}(C_{\mu,q}(\hat{y})) \leq \text{Card}\,(\mathcal{G}(A))$.

So Card $(\mathcal{G}(A))$ strongly depends on $\text{Card}(C_{\mu,q}(\hat{y}))$. To better assess algorithmic efficiency, it is natural to consider not just the total number of generated vectors, but the proportion of them that are actually useful. This motivates definition 14, introducing an inverse measure of complexity, called *usefulness ratio*, which we aim to maximize.

**Definition 14.** *The usefulness ratio for algorithm $A$ executed on $(\hat{y}, q)$ inputs, is defined as:*

$$R(A, \hat{y}, q) = \frac{\text{Card}(C_{\mu,q}(\hat{y}))}{\text{Card}(\mathcal{G}(A(\hat{y}, q)))}$$

*Remark 11.* A usefulness ratio of 0.1 means that 10% of the generated vectors are useful, and such a value would typically indicate a high level of efficiency, especially when compared to the naive algorithm. Indeed, the usefulness ratio of the naive algorithm which generates all the $2^c$ vectors is $\frac{\text{Card}(C_{\mu,q}(\hat{y}))}{2^c} \sim \frac{1}{2^c}$ when $\text{Card}(C_{\mu,q}(\hat{y}))$ is low. For instance, for $c = 100$, and $\text{Card}(C_{\mu,q}(\hat{y})) = 1$ we have $R(\text{naive}, \hat{y}, q) \sim 10^{-30}$, and only a proportion of about $10^{-30}$ of the generated vectors is useful.

The following proposition states that in the optimal case where $l_1 = l_0$, the computational complexity of PBECP grows only linearly with $c$, in contrast to the exponential complexity of the naive approach.

**Proposition 16.** *If $\mu$ and $\hat{y}$ are such that $l_1$ is maximal (ie. $l_1 = l_0$), then we can ensure that we do not generate too many vectors:*

$$\text{Card}(\mathcal{G}(PBECP)) \leq (c - l_0 + 3) \times \text{Card}(C_{\mu,q}(\hat{y})) + 1$$

*Stated differently, if $C_{\mu,q}(\hat{y}) \neq \emptyset$, we give a lower bound of the usefulness ratio:*

$$\forall\ q > 0, \quad \frac{1}{c - l_0 + 3 + \frac{1}{\text{Card}(C_{\mu,q}(\hat{y}))}} \leq R(PBECP, \hat{y}, q)$$

*Remark 12.* Hence, since $\text{Card}(C_{\mu,q}(\hat{y})) \geq 1$, we always have

$$R(naive, \hat{y}, q) \sim \frac{1}{2^c} << \frac{1}{(c - l_0 + 4)} \leq R(PBECP, \hat{y}, q) \tag{19}$$

## 5 Experiments

The primary goal of this article is to develop an algorithm capable of efficiently computing the set $C_{\mu,q}(\hat{y})$, given a specific quantile $q$ and an operational vector $\hat{y}$. In this section, we evaluate whether the proposed algorithms can compute prediction sets within reasonable time frames, even as the number of labels increases. Additionally, we compare the computational complexities of these algorithms as a supplementary analysis.

To assess performance, we use the usefulness ratio as the main evaluation metric across various quantile values ($q \in [q_{min}, q_{max}]$), noting that a higher usefulness ratio indicates greater algorithmic efficiency. To conduct these experiments, we try to compute the set $C_{\mu,q}(\hat{y})$ for different values of $(\mu_{i,j})_{1\leq i\leq j\leq c}$ (one for each dataset), for $q$ varying in a range $[q_{min}, q_{max}]$ and for a fixed challenging vector $\hat{y}$ (one per dataset). To illustrate our algorithms in practical contexts, we select two real datasets: **arxiv_category** [7], **ASRS** [1], and one **dummy** dataset.

**arxiv_category** is composed of 203,961 titles and abstracts categorized into 130 different classes. We used 163,168 samples to construct $\mu$.
**ASRS** (Aviation Safety Reporting System) database stands out as the most renowned incident reporting dataset. We used 96,986 incident reports of US flights from 2000 to 2022 categorized into 63 different classes to construct $\mu$.
We construct a **dummy dataset** with a large number of classes to explore a more complicated case: $c = 150$. It contains $N = 65$ random training samples (elements of $\{0,1\}^c$) with a maximum of 28 simultaneously observed labels.

For all the experiments, we choose $q_{min}$ so that $\text{Card}(C_{\mu,q_{min}}(\hat{y})) = 1$ and $q_{max}$ so that $\text{Card}(C_{\mu,q_{max}}(\hat{y}))$ is very high and beyond any reasonably exploitable prediction set's cardinality. Being able to construct very big and concretely unexploitable prediction sets (corresponding to high $q$) indicates we are able to construct any exploitable prediction set (for lower $q$). We used $\varphi = (.)^2$ and constructed $(\mu_{i,j})_{i,j}$ following the methodology described in Preliminaries (see Appendix for visualization). We denote by $density(\mu)$ the proportion of ones in $(\mu_{i,j})_{1\leq i\leq j\leq c}$ and gives an indication of $\text{Card}(C_{\mu,q}(\hat{y}))$.

Table 1 lists important properties of each experiment. One $\hat{y}$ is chosen so that its $l_0$ is maximal while its $L^2$-closest binary vector $(1_{\hat{y}_i>0.5})_i$ is an element of dataset (to ensure that $l_1 = l_0$). This approach ensures that we focus on constructing the prediction set for a particularly challenging vector, one with numerous 1 entries (relatively deep in the tree $\mathcal{T}$). By doing so, we can restrict our experiments to a single $\hat{y}$ per dataset. Successfully handling such a difficult case gives us confidence that the algorithm will also be effective for other, less complex vectors. Note that having $\mu$ and $\hat{y}$ such that $l_1 = l_0$ is very frequent since in most cases we want to predict labels that already appeared pairwise simultaneously.

| | $\#classes\ c$ | $l_0 = \|\|(1_{\hat{y}_i>0.5})_i\|\|_1$ | $density(\mu)$ | $[q_{min}, q_{max}]$ | $\#C_{\mu,q_{max}}(\hat{y})$ |
|---|---|---|---|---|---|
| **dummy** | 150 | 28 | 51% | $[0.4, 3.6]$ | 4 320 |
| **arxiv_category** | 130 | 4 | 96% | $[0.4, 7]$ | 16 712 |
| **ASRS** | 63 | 13 | 37% | $[0.4, 4]$ | 39 254 |

Table 1: Properties of Datasets and Experiments

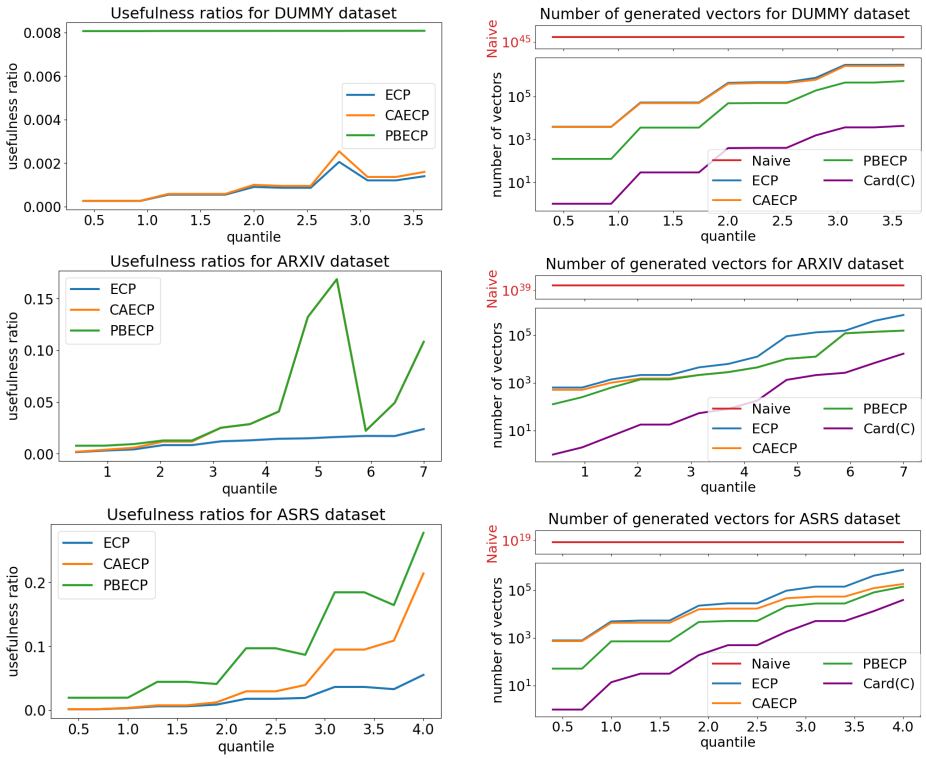


Fig. 3: Evolution of $R(A, \hat{y}, q)$ (left) and evolution of $\text{Card}(\mathcal{G}(A(\hat{y}, q)))$ and $\text{Card}\left(C_{\mu,q}(\hat{y})\right)$ (right) as functions of $q$.

Figure 3 shows our ability to construct the prediction sets across all experiments. The usefulness ratios indicate that PBECP outperforms CAECP in this context, and that CAECP remains superior to ECP as theoretically proven. Higher usefulness ratios for PBECP means that it generates less superfluous vectors, highlighting its efficiency, particularly when $l_1$ is maximal, as described in Proposition 15. The right-hand side of Figure 3 presents the number of vectors generated by each algorithm (on a logarithmic scale), alongside the cardinal of the conformal prediction set. We observe that the algorithms do not generate too much superfluous vectors since their values of $\text{Card}(\mathcal{G}(A(\hat{y}, q)))$ are not too far from $\text{Card}\left(C_{\mu,q}(\hat{y})\right)$ (especially when compared with the naive algorithm).

Table 2 provides computation times and the associated cardinalities of the prediction sets for different values of $q$. While the specific $q$ values are not listed –since the main interest lies in the actual cardinalities $\text{Card}\left(C_{\mu,q}(\hat{y})\right)$– the results offer practical insight into the scaling behavior. All measurements were obtained using a single core of an AMD EPYC 9534 with 16 GB of RAM. Reported computation times are averaged over 10 runs to account for potential variability. The higher cardinalities already represent quite large prediction sets in practice, sug-

gesting that in typical operational contexts, the computation time would likely be shorter. Further evaluation in concrete applications would be necessary to confirm this hypothesis.

In another experiment not related here, we created a new dummy dataset with $c = 1000$. It is remarkable that the method remains tractable, producing sets of a few dozen elements within a few seconds. Nevertheless, practical limitations begin to emerge for higher dimensions: memory usage and computational demands for generating high-dimensional vectors become significant, rendering the approach infeasible for even larger problems. In operational set-

| Card $(C_{\mu,q}(\hat{y}))$ | $t$(s) |
|---|---|
| 1 | 0.008 |
| 5 | 0.020 |
| 29 | 0.093 |
| 407 | 0.807 |
| 1562 | 3.44 |
| 3683 | 7.45 |

**Dummy**

| Card $(C_{\mu,q}(\hat{y}))$ | $t$(s) |
|---|---|
| 1 | 0.013 |
| 70 | 0.049 |
| 388 | 0.099 |
| 2280 | 0.482 |
| 7358 | 2.21 |
| 16748 | 2.55 |

**Arxiv**

| Card $(C_{\mu,q}(\hat{y}))$ | $t$(s) |
|---|---|
| 1 | 0.019 |
| 315 | 0.089 |
| 501 | 0.096 |
| 851 | 0.256 |
| 4903 | 0.519 |
| 21254 | 1.945 |

**ASRS**

Table 2: Computation times (in seconds) for constructing Card $(C_{\mu,q}(\hat{y}))$ with PBECP across datasets (left to right) and $q$ values (top to bottom).

tings, interpreting the prediction set can be challenging, as it consists of multiple combinations of labels rather than a single outcome. Nevertheless, it offers valuable insight by revealing the full range of plausible scenarios. Unlike a simple list of possible labels, the prediction set captures not only which labels may be present, but also how they can realistically co-occur, providing a more nuanced understanding of potential outcomes.

## 6 Conclusion

In this study, we propose an efficient computation of prediction sets for inductive conformal prediction on multi-label classification using the Label-Powerset approach with a generic non-conformity score accounting for label interactions. To address the computational bottlenecks typically associated with the label-powerset strategy for prediction set generation, we developed ECP: a pruning algorithm based on a tailored exploration tree combined with a lower bound function of the non-conformity score.

We further introduced CAECP, an optimization of ECP that imposes an additional condition for child generation during tree exploration. Building on this, we presented PBECP, an even more efficient algorithm, which initiates with a careful selection of roots for CAECP and combines multiple but lighter

calls to CAECP. The experiments demonstrated that we are able to compute Card $(C_{\mu,q}(\hat{y}))$ for problems involving a large number of labels.

However, our work is limited to the scope of inductive conformal prediction and thus inherits its assumptions and constraints. Additionally, while the label-powerset approach enables flexible modeling of label interactions, the sets of label vectors it produces may lack interpretability, which highlights the potential benefit of employing complementary methods to analyze or better understand them.

**Acknowledgments.** Thanks to Dong Quan Vu[2] for providing the preprocessed ASRS data.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References

1. Billings, C.E., Reynard, W.D.: Human factors in aircraft incidents: results of a 7-year study. Aviation, space, and environmental medicine **55**(10), 960–965 (1984)
2. Lambrou, A., Papadopoulos, H.: Binary relevance multi-label conformal predictor. In: Conformal and Probabilistic Prediction with Applications: 5th International Symposium (COPA), Madrid, Spain, April 20-22. pp. 90–104. Springer (2016)
3. Maltoudoglou, L., Paisios, A., Lenc, L., Martínek, J., Král, P., Papadopoulos, H.: Well-calibrated confidence measures for multi-label text classification with a large number of labels. Pattern Recognition **122**, 108271 (2022)
4. Paisios, A., Lenc, L., Martínek, J., Král, P., Papadopoulos, H.: A deep neural network conformal predictor for multi-label text classification. In: Proceedings of the 8th Symposium on Conformal and Probabilistic Prediction and Applications. pp. 228–245. PMLR (2019)
5. Papadopoulos, H.: A cross-conformal predictor for multi-label classification. In: 10th IFIP International Conference on Artificial Intelligence Applications and Innovations (AIAI), Rhodes, Greece. pp. 241–250 (2014)
6. Papadopoulos, H., Proedrou, K., Vovk, V., Gammerman, A.: Inductive confidence machines for regression. In: 13th European conference on machine learning (ECML), Helsinki, Finland. pp. 345–356. Springer (2002)
7. Schopf, T., Blatzheim, A., Machner, N., Matthes, F.: Efficient few-shot learning for multi-label classification of scientific documents with many classes. In: Proceedings of the 7th International Conference on Natural Language and Speech Processing (ICNLSP). pp. 186–198. Association for Computational Linguistics, Trento (2024)
8. Tyagi, C., Guo, W.: Multi-label classification under uncertainty: a tree-based conformal prediction approach. In: Conformal and Probabilistic Prediction with Applications. pp. 488–512. PMLR (2023)
9. Vovk, V., Gammerman, A., Shafer, G.: Algorithmic Learning in a Random World. Springer-Verlag, Berlin, Heidelberg (2005)

[2] Safran Tech, Rue des Jeunes Bois, Châteaufort, 78114 Magny-Les-Hameaux