Efficient Confidence Set Enumeration for Multi-label Conformal Classification

Arthur Ledaguenel[1,2,†] **Florent Capelli**[1] **Jean-Marie Lagniez**[1]

[1]Univ. Artois, CNRS, UMR 8188, Centre de Recherche en Informatique de Lens (CRIL), F-62300 Lens, France
[2]Laboratoire d'Informatique Signal et Image de la Côte d'Opale (LISIC), Université du Littoral Côte d'Opale (ULCO), Calais, France
[†]Corresponding author

Abstract

Conformal prediction is a distribution-free and model agnostic framework that can provide statistical guarantees to machine learning algorithms: a point-wise predictor is transformed into a conformal predictor that outputs sets of predictions which include the ground truth with a user defined confidence rate. In multi-label conformal classification however, the exponential growth of the output space makes confidence sets prohibitively complex to compute in the general case. In this paper, we analyze this challenge under the prism of enumeration complexity. We detail a general approach for confidence set enumeration based on the flashlight method, prove a sufficient condition for efficient enumeration and apply this method on three types of problems. First, we give an enumeration algorithm with linear delay for modular non-conformity losses. Then, we discuss informed conformal classification where a boolean circuit specifies a set of valid label combinations and prove a sufficient condition for efficient enumeration based on standard properties of the circuit. Moreover, we leverage our results on informed conformal classification to tackle label interactions in the loss. Finally, we illustrate the benefits of our approach with a few experiments.

1 INTRODUCTION

The last decades have seen a drastic improvement of machine learning, mainly driven by deep learning. However, despite their resounding successes in many domains, one important limitation of deep learning systems is their poor **calibration**: the indications they provide regarding the confidence of their predictions are untrustworthy [Guo et al., 2017].

Conformal prediction is a distribution-free and model agnostic framework that can provide statistical guarantees to machine learning algorithms: a point-wise predictor is transformed into a conformal predictor that outputs sets of predictions (called **confidence sets**) which include the ground truth with a confidence level $1 - \epsilon$, where ϵ is a user-defined **miscoverage rate** (also called **significance-level**). Computationally prohibitive in its original formulation, called transductive conformal prediction, it was later made more amenable to machine learning systems with inductive conformal prediction [Papadopoulos et al., 2002], while preserving the statistical guarantee.

Applying the paradigm of inductive conformal prediction to multi-label classification tasks, where each input may be classified into several labels, represents a computational challenge: the exponential growth of the output space makes confidence sets excessively expansive to compute in the general case.

In this paper, we frame the problem of computing confidence sets as an enumeration problem, which provides a formal characterization of its computational complexity [Johnson et al., 1988]. We describe a general approach for confidence set enumeration based on the *flashlight method* [Avis and Fukuda, 1996] and prove a sufficient condition to perform the enumeration with polynomial delay in the number of variables. Then, we apply this approach on three multi-label conformal classification cases and detail sufficient conditions for efficient enumeration for each of them. First, we provide an enumeration algorithm with linear delay for all modular non-conformity losses. Then, we introduce informed conformal classification where a propositional formula discriminates valid from invalid labelsets. We propose two methods to leverage prior knowledge in multi-label conformal classification and detail sufficient conditions for efficient enumeration based on standard properties of boolean circuits [Darwiche and Marquis, 2002]: smoothness, decomposability and determinism. Moreover, we show how this approach can be leveraged to provide efficient enumeration algorithms in other cases, such as non-conformity losses

Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence (UAI 2026), PMLR 337:3298–3322.

with label interactions. Finally, we provide a few experimental results that illustrate the benefits of our approach. All the code and data necessary to reproduce our experiments is openly available[1].

Outline The outline of the paper is the following. We first review related works in Section 2. We define preliminary notions on multi-label conformal classification, enumeration complexity and boolean circuits in Section 3. We describe the flashlight method and prove a sufficient condition for polynomial delay enumeration in Section 4. We then examine three different use-cases based on our approach: we analyze modular non-conformity functions in Section 5, we discuss informed conformal classification in Section 6 and tackle pairwise label interactions in Section 7. We present our experimental results in Section 8. Finally, we conclude our paper and discuss future research directions in Section 9.

2 RELATED WORKS

Several approaches have been developed to deal with multi-label conformal classification.

First, the methods developed to extend binary and multi-class classification frameworks to the multi-label setting were adapted to address multi-label conformal classification. Problem transformation approaches, like Label Powerset [Wang et al., 2015], Binary Relevance [Wang et al., 2015, Lambrou and Papadopoulos, 2016] or Random k-Labelsets (RAKEL) [Yang et al., 2017], decompose the task into several simpler binary or multi-class classification tasks in order to apply standard conformal classification algorithms. However, the decomposition of the task also requires to reduce the miscoverage rate for each sub-task, which often leads to an explosion of the size of confidence sets. Algorithm adaptation approaches [Papadopoulos, 2014] adapt standard classification algorithms to the multi-label task but do not solve the problem of confidence set enumeration.

Relaxation methods introduced in Lambrou and Papadopoulos [2016] adapts the statistical guarantee of conformal classification to the specific case of multi-label classification.

Several papers tackled the complexity of confidence set enumeration for specific non-conformity measures: a search space reduction method was developed for L^p norms in Maltoudoglou et al. [2022], while [Najjar et al., 2026] additionally takes into account label interactions. However, both their algorithms fail to guarantee a polynomial delay.

Finally, at the junction of conformal prediction and neurosymbolic artificial intelligence, Ramalingam et al. [2024] discuss how to compose conformal predictors based on symbolic knowledge.

[1] see the repository https://github.com/LedaguenelArthur/UAI26-Efficient-Multi-label-Confidence-Set-Enumeration

3 PRELIMINARIES

We use capital letters Y, Z and bold capital letters $\mathbf{Y}, \mathbf{Z}$ to denote boolean variables and sets of boolean variables respectively. Similarly, the instantiation of a boolean variable Y is noted with a lowercase letter $y \in \{0,1\}$, whereas the joint instantiation of a set of boolean variables $\mathbf{Y}$, also called an instance, is denoted using a bold lowercase letter $\mathbf{y}$. An instance $\mathbf{y}$ is understood equivalently as a subset of $\mathbf{Y}$ or as a function from $\mathbf{Y}$ to $\{0,1\}$. The set of all instances on $\mathbf{Y}$ is noted $\mathcal{Y} = \{0,1\}^{\mathbf{Y}}$. A partial instance on $\mathbf{Y}$ is an instance on any subset of $\mathbf{Y}$. Given two instances $\mathbf{y}$ and $\mathbf{z}$ on variables $\mathbf{Y}$ and $\mathbf{Z}$ such that $\mathbf{Z} \subset \mathbf{Y}$, we say that $\mathbf{y}$ is an extension of $\mathbf{z}$ if its restriction to $\mathbf{Z}$ is equal to $\mathbf{z}$. Given two instances $\mathbf{y}$ and $\mathbf{z}$ on two disjoint sets of variables $\mathbf{Y}$ and $\mathbf{Z}$, $\mathbf{yz}$ is the combined instance on $\mathbf{Y} \cup \mathbf{Z}$.

When $\mathbf{Y} = \{Y_1, ..., Y_n\}$, we also view $\mathbf{y}$ as a binary vector in $\{0,1\}^n$ and we note y_i the corresponding instantiation of Y_i. A k-prefix π is an instance on $\{Y_1, ..., Y_k\}$. By convention, we note $[\,]$ the only 0-prefix that does not fix the value of any variable. A n-prefix corresponds to an instance on $\mathbf{Y}$. For any k-prefix (with $0 \leqslant k \leqslant n-1$), we note π^{+0} (resp. π^{+1}) the direct extension of π such that $\pi^{+0}_{k+1} = 0$ (resp. $\pi^{+1}_{k+1} = 1$).

3.1 MULTI-LABEL CONFORMAL CLASSIFICATION

Let $\mathcal{X}$ be an input space and $\mathbf{Y} = \{Y_1, ..., Y_n\}$ be a set of labels. In multi-label classification, an element of the output space $\mathbf{y} \in \mathcal{Y}$, called a **labelset**, is an instance on $\mathbf{Y}$. We assume a trained model $m : \mathcal{X} \to [0,1]^n$ which maps each input sample $x \in \mathcal{X}$ to a *likelihood* score for each class $Y_i \in \mathbf{Y}$. This model can be combined with a loss function $\ell : [0,1]^n \times \{0,1\}^n \to \mathbb{R}^+$ to build a non-conformity measure μ on the space $\mathcal{X} \times \mathcal{Y}$ such that:

$$\mu : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+, (x, \mathbf{y}) \mapsto \ell(m(x), \mathbf{y})$$

Remark 1. *Typically, m is a neural network trained on a collection of labeled samples $\mathcal{D}_{train}$ and ℓ is the loss function used during training.*

In inductive conformal prediction, in addition to the training set, we are given another collection of labeled samples $\mathcal{D}_{cal} = (x^i, \mathbf{y}^i)_{1 \leqslant i \leqslant d}$, called a **calibration** set. We assume that $\mathcal{D}_{cal}$ was i.i.d. sampled according to a distribution $\mathcal{P}_{\mathcal{X} \times \mathcal{Y}}$, to which we do not have access.

Consider a new input x^{d+1}: we would like to make a confident guess about what could be the corresponding output $\mathbf{y}^{d+1}$, knowing that $(x^{d+1}, \mathbf{y}^{d+1})$ was sampled according to $\mathcal{P}_{\mathcal{X} \times \mathcal{Y}}$. To do so, we compare the non-conformity of every

possible pair $(x^{d+1}, \mathbf{y}^{d+1})$ to that of known ground truths in $\mathcal{D}_{cal}$.

Remark 2. *The i.i.d. assumption between $\mathcal{D}_{cal}$ and $(x^{d+1}, \mathbf{y}^{d+1})$ ensures the statistical guarantees of conformal classification. No assumption is required on $\mathcal{D}_{train}$ for the conformal guarantee. However, its distribution should be close so that learned parameters on $\mathcal{D}_{train}$ transfer well on $\mathcal{D}_{cal}$ and $(x^{d+1}, \mathbf{y}^{d+1})$.*

Non-conformity scores are defined for each calibration sample:

$$\alpha_i = \ell(m(x^i), \mathbf{y}^i)$$

We assume increasing calibration non-conformity scores *w.l.o.g.* then compute their $(1-\epsilon)^{th}$-quantile $\tau_\epsilon = \alpha_{q_\epsilon}$ where:

$$q_\epsilon = \lfloor \frac{(d+1)(1-\epsilon)}{d} \rfloor$$

In other terms, the threshold τ_ϵ is such that the proportion of calibration samples $(x^i, \mathbf{y}^i)_{1 \leqslant i \leqslant d}$ such that $\ell(m(x^i), \mathbf{y}^i) \leqslant \tau_\epsilon$ is $1-\epsilon$.

Then, the confidence set is directly defined based on the threshold and the non-conformity measure of each output:

$$C_\epsilon(x) := \{\mathbf{y} \in \mathcal{Y} | \ell(m(x), \mathbf{y}) \leqslant \tau_\epsilon\} \tag{1}$$

Fortunately, inductive conformal classification preserves the statistical guarantee of transductive conformal classification [Papadopoulos et al., 2002]:

$$\mathcal{P}(\mathbf{y}^t \in C_\epsilon(x)) \geqslant 1 - \epsilon \tag{2}$$

Remark 3. *Equation 2 is a slight abuse of notation since the statistical guarantee also takes into account the sampling of $\mathcal{D}_{cal}$.*

3.2 ENUMERATION COMPLEXITY

The main problem we wish to solve is the following.

Definition 1 (Confidence set enumeration)**.** *Given a loss function $\ell : [0,1]^n \times \{0,1\}^n \to \mathbb{R}^+$, the enumeration problem* $\texttt{Enum}(\ell)$ *takes as input a vector of scores $\mathbf{a} \in [0,1]^n$ produced by a trained neural network and a non-conformity threshold τ_ϵ and outputs the confidence set:*

$$C_\epsilon(\mathbf{a}) = \{\mathbf{y} \in \mathcal{Y} | \ell(\mathbf{a}, \mathbf{y}) \leqslant \tau_\epsilon\} \tag{3}$$

The exponential size of the output domain $\mathcal{Y}$ poses two challenges to the prediction phase that are unique to multi-label conformal classification:

1. Confidence sets can themselves be exponential in size.
2. Computing the confidence set by *brute force* requires an exponential amount of computations (*i.e.*, browsing the entire output domain $\mathcal{Y}$ to select the labelsets $\mathbf{y}$ such that $\ell(\mathbf{a}, \mathbf{y}) \leqslant \tau_\epsilon$).

Importantly, the two problems are nested, but of different nature. The first problem depends on the ability of the underlying model m to identify ground truth labelsets: the better the model, the lower the non-conformity threshold q_ϵ and the smaller the confidence sets. Obviously, when the size of the confidence set blows up exponentially, there is no hope of computing it in a reasonable time. However, even when the model is accurate and confidence sets are small, computing them using a naive algorithm that iterates through each possible labelset still requires an exponential amount of computations.

Therefore, assuming that an accurate model yields small confidence sets, it is worth studying under what conditions these can be computed in a reasonable amount of time.

A first step in that direction was made in Maltoudoglou et al. [2022]: based on the hamming distance with the most conform labelset, a significant part of $\mathcal{Y}$ is eliminated without exploration because it cannot contain labelsets below the conformity threshold defined by the quantile q_ϵ. However, this method fails to deliver guarantees regarding enumeration complexity.

Framing the calculation of confidence sets as an enumeration problem provides an interesting formal characterization of its computational complexity. Indeed, instead of looking at the complexity as a function of the input size, enumeration complexity classes take into account the size of the output as well. This allows to discriminate between enumeration problems that are hard because the size of the output is prohibitive and enumeration problems that are hard because finding every solution requires a lot of computations even when there are few of them.

Standard enumeration complexity classes are total output polynomial `TotalP`, which means that solutions can be enumerated in time polynomial in the combined size of the input and the output, and delay polynomial `DelayP`, which means that solutions can be enumerated with a delay between two solutions that is polynomial in the size of the input [Johnson et al., 1988]. It comes directly that problems in `DelayP` are also in `TotalP`.

Although `TotalP` would be enough in the context of confidence set enumeration for multi-label conformal classification, `DelayP` provides the possibility to interrupt the enumeration process whenever a specific condition is met (*e.g.* the ground truth label has been enumerated in test mode, the number of enumerated labelsets or their cumulated probabilities has exceeded a certain threshold, etc.) which can prove useful in practice.

3.3 BOOLEAN CIRCUITS

A literal on $\mathbf{Y}$ is Y or $\neg Y$ with $Y \in \mathbf{Y}$. The set of literals on $\mathbf{Y}$ is noted $\mathrm{lits}(\mathbf{Y})$.

A boolean circuit $\kappa(\mathbf{Y})$ in Negation Normal Form (NNF) [Darwiche and Marquis, 2002] is a labeled directed acyclic graph where each leaf node is labeled with $\top, \bot$ or a literal in $\mathrm{lits}(\mathbf{Y})$ and internal nodes are labeled with logical connectives $\wedge$ or $\vee$. The size of a boolean circuit κ, noted $|\kappa|$, is its number of edges. An internal node $u \in \kappa$ is often conflated with the sub-circuit of κ rooted in u. The set of direct children of u in κ is noted $\mathrm{ch}(u)$ and the set of variables featured in the sub-circuit is noted $\mathrm{var}(u)$.

To know if an instance $\mathbf{y}$ on $\mathbf{Y}$ satisfies a circuit $\kappa(\mathbf{Y})$ in NNF, we evaluate the circuit bottom up, mapping each node u to 0 or 1. Leaf nodes labeled $\top$ (resp. $\bot$) are mapped to 1 (resp. 0). Leaf nodes labeled Y (resp $\neg Y$) are mapped 1 iff Y is positive (resp. negative) in $\mathbf{y}$. Internal nodes u labeled $\wedge$ (resp. $\vee$) are mapped to 1 iff all its children (resp. at least one child) are mapped to 1. The instance $\mathbf{y}$ satisfies κ, noted $\mathbf{y} \models \kappa$, if the root is valued at 1. Such an instance is called a **model** of κ (not to be confused with the machine learning model mentioned previously). A circuit is **consistent** (or satisfiable) if it has at least one model, otherwise it is inconsistent. Two circuits are **equivalent** if they are satisfied by the same instances. We sometimes say that two circuits are inconsistent if their conjunction is inconsistent.

A disjunction (resp. conjunction) of literals is called a clause (resp. term). A boolean circuit is in conjunctive normal form (CNF) iff it is a conjunction of clauses. A $\wedge$-node is **decomposable** if its children do not share variables. A NNF circuit is in Decomposable Negation Normal Form (DNNF) iff all of its $\wedge$-nodes are decomposable. A $\vee$-node is **smooth** if all its children have identical set of variables. A DNNF circuit is in sDNNF if all of its $\vee$-nodes are smooth. A $\vee$-node u is **deterministic** if its children are inconsistent. A sDNNF circuit is in smooth deterministic Decomposable Negation Normal Form (sd-DNNF) iff all of its $\vee$-nodes are deterministic.

4 FLASHLIGHT ENUMERATION

In this section, we introduce a general method for confidence set enumeration based on *backtrack search* or the *flashlight method* [Avis and Fukuda, 1996] and give a simple sufficient condition for this algorithm to be in DelayP. The algorithm shown on Algorithm 1 can be seen as the depth first exploration of a binary tree where each node is a prefix: the root is the 0-prefix [ ], each internal node π has two children π^{+0} and π^{+1} and leaf nodes correspond to labelsets. To avoid exploring branches *useless* branches, each branch is first tested to know if it leads to at least one

Algorithm 1 Enumerate the confidence set

Require: Variables $\mathbf{Y} = \{Y_1, ..., Y_n\}$, a vector of scores $\mathbf{a} \in \mathbb{R}^n$ (produced by the trained neural network), a loss function $\ell : [0,1]^n \times \{0,1\}^n \to \mathbb{R}^+$ and a non-conformity threshold τ_ϵ

Ensure: $\mathcal{C}$ contains the confidence set $\mathcal{C}_\epsilon(\mathbf{a})$

```
1:  function ENUMERATE(a, ℓ, τ_ε)
2:      INITIALIZE an empty confidence set C
3:      INITIALIZE an empty Last-In First-Out queue Q
4:      m ← MIN(ℓ, a, [ ])
5:      if m ⩽ τ_ε then
6:          PUSH(Q, [ ])
7:      else
8:          return ∅
9:      while Q is not empty do
10:         π ← POP(Q)
11:         if π is a labelset then
12:             APPEND π to C
13:         else
14:             for b ∈ {0, 1} do
15:                 m^{+b} ← MIN(ℓ, a, π^{+b})
16:                 if m^{+b} ⩽ τ_ε then
17:                     PUSH(Q, π^{+b})
18:     return C
```

labelset that belongs to the output confidence set (*i.e.*, its non-conformity score is below τ_ϵ). To do so, we assume that we dispose of a subroutine MIN which computes, for a given prefix π, a set of scores $\mathbf{a}$ and a loss function ℓ, the minimum non-conformity score that can be reached by an extension of π. See the supplementary materials for a visual explanation of the algorithm.

We assume every instruction INITIALIZE, APPEND, PUSH and POP can be done in constant time. Therefore, the complexity of ENUMERATE essentially depends on the complexity of the MIN sub-routine and the number of iterations in the **while** loop.

Definition 2 (Loss minimization)**.** *Given a loss function $\ell : [0,1]^n \times \{0,1\}^n \to \mathbb{R}^+$, the optimization problem* $\mathtt{Min}(\ell)$ *takes as input a vector of scores $\mathbf{a} \in [0,1]^n$ and a partial instance π on $\mathbf{Y}$ and outputs the minimum non-conformity score that can be reached by any extension of π in $\mathcal{Y}$:*

$$\min_{\substack{\mathbf{y} \in \mathcal{Y}, \\ \mathbf{y} \models \pi}} \ell(\mathbf{a}, \mathbf{y}) \tag{4}$$

Proposition 1. *Given a loss function $\ell : [0,1]^n \times \{0,1\}^n \to \mathbb{R}^+$, if* $\mathtt{Min}(\ell)$ *can be solved in $\mathcal{O}(n^p)$ with $p \in \mathbb{N}$, then* $\mathtt{Enum}(\ell)$ *can be computed with a delay in $\mathcal{O}(n^{p+1})$.*

Proof. Algorithm 1 provides a practical proof.

Let's analyze an iteration of the **while** loop where the prefix π that is popped from the queue is not a labelset. If π was placed in the queue in the first place, it can be extended into a labelset in $\mathcal{C}_\epsilon(x)$, and therefore one of its direct extensions π^{+0} and π^{+1} will be placed at the top of the queue. Thus, between two labelsets, the size of the prefix π at the top of the queue grows strictly at each iteration. Hence, there can be at most n iterations of the loop between two labelsets. Each iteration of the loop calls MIN two times.

Therefore, if MIN can be computed in $\mathcal{O}(n^p)$, the delay between two labelsets in ENUMERATE is at most in $\mathcal{O}(n^{p+1})$. □

Remark 4. ENUMERATE *can be slightly tweaked into* RANKEDENUMERATE *to output elements of the confidence set in increasing order of non-conformity with an increased delay of* $\mathcal{O}(n^{p+1} \times \log(n \cdot k))$ *where* k *is the number of labelsets enumerated so far (see Algorithm 3 in the Appendix). If the maximum non-conformity score that can be reached by an extension of* π *can also be computed efficiently,* ENUMERATE *can be easily turned into a counting algorithm that computes the size of the confidence set more efficiently (see Algorithm 4 in the Appendix).*

5 MODULAR LOSSES

The conditions for efficient minimization of discrete functions are widely studied and notoriously include modular (resp. sub-modular) functions, which can be minimized in linear (resp. polynomial) time. Combined with Proposition 1, using these losses provides an efficient algorithm for confidence set enumeration. In this section, we dive deeper in the specific case of modular multi-label loss functions, which are frequently used in the literature. Table 1 gives a few examples of common modular losses.

Definition 3. *We say that a loss function* $\ell : [0,1]^n \times \{0,1\}^n \mapsto \mathbb{R}^+$ *is* ***modular*** *when there are functions* $f_i : [0,1] \times \{0,1\} \mapsto \mathbb{R}^+$ *for* $1 \leqslant i \leqslant n$ *such that:*

$$\ell(\mathbf{a}, \mathbf{y}) = \sum_{1 \leqslant i \leqslant n} f_i(a_i, y_i)$$

Remark 5. *If not mentioned otherwise, we assume that all functions* $(f_i)_{1 \leqslant i \leqslant n}$ *are equal to the same function* f*, which does not impact any of the results proven in the paper.*

Linear minimization of modular losses comes easily: for each variable Y_i not instantiated by the input partial instance, pick the value y_i that minimizes $f_i(a_i, y_i)$. Naively combining the linear minimization of modular losses with Proposition 1 provides an enumeration algorithm with quadratic delay (in the number of variables). We show below that a tighter integration with Algorithm 1 yields a linear delay.

Proposition 2. *If* $\ell : [0,1]^n \times \{0,1\}^n \to \mathbb{R}^+$ *is a modular loss function, then* $\texttt{Enum}(\ell)$ *can be computed with a linear delay.*

Proof. Instead of running the MIN sub-routine from scratch in each iteration of the **while** loop, notice that the minimum non-conformity score $\text{MIN}(\ell, \mathbf{a}, \pi)$ reached from an inner node π can be updated in constant time to yield $\text{MIN}(\ell, \mathbf{a}, \pi^{+b})$ of a direct extensions π^{+b} by adding $f_i(a_i, b) - \min_{b \in \{0,1\}} f_i(a_i, b)$ for $b \in \{0,1\}$. Therefore, if $\text{MIN}(\ell, \mathbf{a}, \pi)$ is cached alongside π in the queue Q to be retrieved when useful, each call to the MIN sub-routine inside the **while** loop can be replaced by a constant time operation. The delay is therefore only determined by the number of iterations of the loop between two labelsets, which we have proven to be linear in Proposition 1. □

A concrete implementation of the algorithm underlying Proposition 2 can be found in Appendix A.4.

Remark 6. *In all generality, Definition 3 and Proposition 2 can be extended to pseudo-modular loss functions, i.e., the composition* $\ell = g \circ h$ *of two* ***known*** *functions where* h *is a modular loss function and* g *is a monotone function. One standard example of a pseudo-modular loss function is the* L^p *norm loss (defined in Table 1 without its monotone wrapping function).*

6 INFORMED CONFORMAL CLASSIFICATION

We say that a task of supervised multi-label classification is **informed** when it comes attached with prior knowledge that specifies which labelsets in the output space are **semantically valid**. Such prior knowledge can be expressed with a boolean circuit $\kappa(\mathbf{Y})$ whose models correspond to valid labelsets. Successfully integrating this knowledge in multi-label conformal classification may reduce the size of output confidence sets without breaking the statistical guarantee. We introduce to this end two informed conformal classification methods that integrate prior knowledge and detail their computational complexity.

6.1 SEMANTIC FILTERING

A first approach works as a simple extension of inductive conformal classification by simply removing labelsets that do not satisfy κ from the output confidence set.

Definition 4 (Filtered loss function)**.** *Given a loss function* $\ell : [0,1]^n \times \{0,1\}^n \to \mathbb{R}^+$ *and a boolean circuit* κ *on variables* $\{Y_1, ..., Y_n\}$*, we note* ℓ^κ *the filtered loss function such that for any vector of scores* $\mathbf{a} \in [0,1]^n$ *and instance* $\mathbf{y} \in \mathcal{Y}$*:*

$$\ell^\kappa(\mathbf{a}, \mathbf{y}) = \begin{cases} \ell(\mathbf{a}, \mathbf{y}) & \text{if } \mathbf{y} \models \kappa \\ +\infty & \text{otherwise} \end{cases} \quad (5)$$

Loss	Expression
Binary Cross-Entropy	$\sum_{1 \leqslant i \leqslant n} -\log(a_i) \cdot y_i - \log(1-a_i) \cdot (1-y_i)$
Basic Back-Propagation	$\sum_{1 \leqslant i \leqslant n} (a_i - y_i^{\pm})^2$, where $y_i^{\pm} = 1$ if $y_i = 1$ and -1 otherwise
L^p norms	$\sum_{1 \leqslant i \leqslant n} \lvert a_i - y_i \rvert^p$, for $p \in \mathbf{N}^*$

Table 1: Examples of modular losses.

The enumeration problem $\texttt{Enum}(\ell^\kappa)$ *takes as input a vector of scores* $\mathbf{a} \in [0,1]^n$ *produced by a trained neural network and a non-conformity threshold* τ_ϵ *and outputs the confidence set:*

$$C_\epsilon^\kappa(\mathbf{a}) = \{\mathbf{y} \in \mathcal{Y} | \ell(\mathbf{a}, \mathbf{y}) \leqslant \tau_\epsilon, \mathbf{y} \models \kappa\} \tag{6}$$

The optimization problem $\texttt{Min}(\ell^\kappa)$ *takes as input a vector of scores* $\mathbf{a} \in [0,1]^n$ *produced by a trained neural network and a partial instance* π *and outputs:*

$$\min_{\substack{\mathbf{y} \in \mathcal{Y}, \\ \mathbf{y} \models \kappa \wedge \pi}} \ell(\mathbf{a}, \mathbf{y}) \tag{7}$$

Interestingly, as long as the consistency hypothesis about the test set (*i.e.*, all labels in the test set satisfy κ) is maintained, filtering can be applied after training and calibration without affecting the statistical guarantee that comes with conformal prediction (see Equation 2). Hence, filtered confidence sets are smaller for a given miscoverage rate, or alternatively the miscoverage rate can be decreased while maintaining manageable confidence sets. Obviously, this size reduction is of little value if filtered confidence sets can not be enumerated efficiently.

Since testing if a labelset satisfies a boolean circuit κ can be done in $\mathcal{O}(|\kappa|)$, filtering labelsets along the way represents a minor overhead to Algorithm 1 when κ is small enough. However, as $C_\epsilon^\kappa(\mathbf{a})$ may be drastically smaller than $C_\epsilon(\mathbf{a})$, many labelsets can be discarded during the enumeration and this naive approach does not guarantee a polynomial delay for $\texttt{Enum}(\ell^\kappa)$, even for modular loss functions.

Proposition 3. *If* κ *is in* sDNNF *and* ℓ *is modular, then* $\texttt{Min}(\ell^\kappa)$ *can be solved in* $\mathcal{O}(|\kappa|)$.

Proof. The optimization problem $\texttt{Min}(\ell^\kappa)$ for a modular loss ℓ can be reduced to a linear optimization problem on the models of κ, which is known to be solvable in $\mathcal{O}(|\kappa|)$ for sDNNF circuits [Darwiche and Marquis, 2004, Audemard et al., 2020]. See Algorithm 7 in Appendix A.5 for a practical implementation. □

Corollary 4. *If* κ *is in* sDNNF *and* ℓ *is modular, then* $\texttt{Enum}(\ell^\kappa)$ *can be solved with a delay in* $\mathcal{O}(n \cdot |\kappa|)$.

Proof. The proof is similar to that of Proposition 1 with the size $|\kappa|$ taken as a fixed parameter. □

6.2 SEMANTIC CONDITIONING

As mentioned previously, semantic filtering does not alter the loss function on datasets that respect the consistency hypothesis. In particular, substituting ℓ for ℓ^κ during training has no impact if the training set also respects the consistency hypothesis. However, prior knowledge can be integrated in the loss function in a way that also impacts training, for example using the informed loss of semantic conditioning [Ahmed et al., 2022, Ledaguenel et al., 2024] or the regularized semantic loss [Xu et al., 2018]. To avoid a fatal misalignment between the non-conformity measure and the training loss, the same informed loss can be used during calibration and prediction, which can drastically impact the complexity of confidence set enumeration.

In this section, we analyze the complexity of confidence set enumeration for the loss of semantic conditioning defined as:

$$\ell_{sc}^\kappa(\mathbf{a}, \mathbf{y}) = \begin{cases} -\log(\frac{p(\mathbf{y}|\mathbf{a})}{p(\kappa|\mathbf{a})}) & \text{if } \mathbf{y} \models \kappa \\ +\infty & \text{otherwise} \end{cases} \tag{8}$$

where $p(\mathbf{y}|\mathbf{a}) = \prod_{1 \leqslant i \leqslant n} a_i \cdot y_i + (1-a_i) \cdot (1-y_i)$ and $p(\kappa|\mathbf{a}) = \sum_{\mathbf{y} \models \kappa} p(\mathbf{y}|\mathbf{a})$.

In semantic conditioning, contrary to semantic filtering on modular losses, computing the non-conformity measure for even a single labelset $\mathbf{y}$ is no longer tractable in general as it may require to solve a complex weighted model counting problem: if $\mathbf{y} \models \kappa$ than the value of $\ell_{sc}^\kappa(\mathbf{a}, \mathbf{y})$ can directly lead to the value of $p(\kappa|\mathbf{a})$, which is known to be #P-hard for arbitrary boolean circuits.

Proposition 5. *If* κ *is in* sd-DNNF *and* ℓ *is modular, then* $C_\epsilon^\kappa(\mathbf{a}) = \{\mathbf{y} \in \mathcal{Y} | \ell_{sc}^\kappa(\mathbf{a}, \mathbf{y}) \leqslant \tau_\epsilon\}$ *can be enumerated with a delay in* $\mathcal{O}(n \cdot |\kappa|)$.

Proof. Notice that:

$$\begin{aligned} C_\epsilon^\kappa(\mathbf{a}) &= \{\mathbf{y} \in \mathcal{Y} | \ell_{sc}^\kappa(\mathbf{a}, \mathbf{y}) \leqslant \tau_\epsilon\} \\ &= \{\mathbf{y} \in \mathcal{Y} | \ell_{BCE}^\kappa(\mathbf{a}, \mathbf{y}) \leqslant \tau_\epsilon - \log(p(\kappa|\mathbf{a}))\} \end{aligned} \tag{9}$$

where $\ell_{BCE}^\kappa(\mathbf{a}, \mathbf{y})$ corresponds to the binary cross-entropy loss displayed in Table 1, filtered by the circuit κ.

Since κ is in sd-DNNF, $p(\kappa|\mathbf{a})$ can be computed once in time $\mathcal{O}(|\kappa|)$ [Kimmig et al., 2017]. Then, since ℓ_{BCE} is modular, labelsets in $C_\epsilon^\kappa(\mathbf{a})$ can be enumerated with a delay in $\mathcal{O}(n \cdot |\kappa|)$ according to Proposition 4. □

7 WEIGHTED FUNCTIONAL ENCODINGS

Even when no prior knowledge is available on a multi-label conformal classification task, Proposition 3 provides a powerful tool to solve confidence set enumeration of non-modular loss functions.

Definition 5. *Given a set of boolean variables $\mathbf{Y}$ and a function $\ell : \mathcal{X} \times \mathbb{R}^d \to \mathbb{R}^+$, a **weighted functional encoding** of ℓ is a tuple $(\kappa(\mathbf{Y}, \mathbf{Z}), f, \nu)$ such that:*

- $\mathbf{Z}$ *is a set of p auxiliary variables*
- κ *is a boolean circuit such that for all $\mathbf{y} \in \mathcal{Y}$ there exists a unique $\mathbf{z} \in \mathcal{Z}$ such that $\mathbf{yz} \models \kappa$ (we note $\mathbf{z} = \kappa(\mathbf{y})$)*
- $f : [0,1] \times \{0,1\} \to \mathbb{R}^+$ *and* $\nu : \text{lits}(\mathbf{Z}) \to \mathbb{R}$ *are such that:*

$$\forall \mathbf{y} \in \mathcal{Y}, \ell(\mathbf{y}, \mathbf{a}) = \sum_{1 \leqslant i \leqslant n} f(a_i, y_i) + \sum_{\substack{l \in \text{lits}(\mathbf{Z}), \\ \kappa(\mathbf{y}) \models l}} \nu(l)$$

Proposition 6. *Given a loss function ℓ with a weighted functional encoding $(\kappa(\mathbf{Y}, \mathbf{Z}), f, \nu)$ where κ is in sDNNF, then $\texttt{Min}(\ell)$ can be computed in $\mathcal{O}(|\kappa|)$.*

Proof. Let us note, for any $\mathbf{yz} \in \mathcal{Y} \times \mathcal{Z}$:

$$\hat{\ell}(\mathbf{yz}, \mathbf{a}) = \sum_{1 \leqslant i \leqslant n} f(a_i, y_i) + \sum_{\substack{l \in \text{lits}(\mathbf{Z}), \\ \mathbf{z} \models l}} \nu(l)$$

Notice that $\hat{\ell}$ can be viewed as a modular loss function on $\mathbf{Y} \cup \mathbf{Z}$.

Then, for any $\mathbf{yz} \in \mathcal{Y} \times \mathcal{Z}$ such that $\mathbf{yz} \models \kappa$, we have $\ell(\mathbf{y}, \mathbf{a}) = \hat{\ell}(\mathbf{yz}, \mathbf{a})$.

Therefore:

$$\min_{\substack{\mathbf{y} \in \mathcal{Y} \\ \mathbf{y} \models \pi}} \ell(\mathbf{y}, \mathbf{a}) = \min_{\substack{\mathbf{yz} \in \mathcal{Y} \times \mathcal{Z} \\ \mathbf{yz} \models \kappa \wedge \pi}} \hat{\ell}(\mathbf{yz}, \mathbf{a}) \tag{10}$$

Finally, by the same reduction to the linear optimization on sDNNF boolean circuits used in Proposition 3, $\texttt{Min}(\ell)$ can be computed in $\mathcal{O}(|\kappa|)$. □

Corollary 7. *Given a loss function ℓ with a weighted functional encoding $(\kappa(\mathbf{Y}, \mathbf{Z}), f, \nu)$ where κ is in sDNNF, then $\texttt{Enum}(\ell)$ can be solved with a delay $\mathcal{O}(n \cdot |\kappa|)$.*

We illustrate how weighted functional encodings can be used in practice with the following use case. In order to penalize uncommon label combinations, it is possible to add a pairwise interaction term to a modular loss:

$$\ell_\mu(\mathbf{a}, \mathbf{y}) = \sum_{1 \leqslant i \leqslant n} f(a_i, y_i) + \sum_{1 \leqslant i < j \leqslant n} \mu_{ij} y_i y_j$$

where $\mu_{ij} = 1$ if labels i and j have never been observed jointly in the training set, and $\mu_{ij} = 0$ otherwise.

Unfortunately, this pairwise term breaks the modularity of the loss and therefore Proposition 2 cannot be leveraged. A practical algorithm is introduced in Najjar et al. [2026] but offers no guarantee of polynomial delay. In the rest of section, we introduce two weighted functional encodings for ℓ_μ (depending on the density of μ).

7.1 SPARSE INTERACTIONS

We first design a weighted functional encoding that works best when μ is sparse. Given a binary matrix $\mu \in \{0,1\}^{n \times n}$, let Z_{ij} be an auxiliary variable for each pair of labels (Y_i, Y_j) such that $i < j$ and $\mu_{ij} = 1$. We note $F^\mu(\mathbf{Y}, \mathbf{Z})$ a circuit defined as:

$$F^\mu \equiv \bigwedge_{\substack{1 \leqslant i < j \leqslant n, \\ \mu_{ij} = 1}} Z_{ij} \iff (Y_i \wedge Y_j) \tag{11}$$

Finally, let $\nu : \text{lits}(\mathbf{Z}) \to \mathbb{R}$ be the score function such that, for $Z_{ij} \in \mathbf{Z}$:

$$\nu(Z_{ij}) = 1 \qquad \nu(\neg Z_{ij}) = 0 \tag{12}$$

Proposition 8. *$(F^\mu(\mathbf{Y}, \mathbf{Z}), f, \nu)$ defines a weighted functional encoding for ℓ_μ.*

Proof. A detailed proof can be found in Appendix B.1.1. □

7.2 DENSE INTERACTIONS

In case the interaction matrix is dense and therefore F^μ is too large to be compiled to a small sDNNF, we can take the dual approach by using $F^{1-\mu}$ for looking at the gaps in μ. Notice that the interaction term can be decomposed as:

$$\sum_{1 \leqslant i < j \leqslant n} \mu_{ij} y_i y_j = \sum_{1 \leqslant i < j \leqslant n} y_i y_j - \sum_{1 \leqslant i < j \leqslant n} (1 - \mu_{ij}) y_i y_j \tag{13}$$

such that:

1. the term $\sum_{1 \leqslant i < j \leqslant n} y_i y_j$ only depends on the number of variables Y_i set to 1 in $\mathbf{y}$
2. the term $\sum_{1 \leqslant i < j \leqslant n} (1 - \mu_{ij}) y_i y_j$ is similar to the term $\sum_{1 \leqslant i < j \leqslant n} \mu_{ij} y_i y_j$ with $1 - \mu$ in place of μ

Let us assume circuits $F_0^{1-\mu}(\mathbf{Y}), ..., F_n^{1-\mu}(\mathbf{Y})$ such that:

$$\forall 0 \leqslant k \leqslant n, \forall \mathbf{y} \in \mathcal{Y}, \forall \mathbf{z} \in \mathcal{Z},$$
$$\mathbf{yz} \models F_k^{1-\mu} \iff \mathbf{yz} \models F^{1-\mu} \text{ and } \sum_{1 \leqslant i \leqslant n} y_i = k$$

We introduce new variables $\mathbf{S} = \{S_0, ..., S_n\}$ and recursively define, for $0 \leqslant k \leqslant n-1$:

$$F^{1-\mu}_{\geqslant k} = (S_k \wedge F^{1-\mu}_k \wedge_{k+1 \leqslant m \leqslant n} \neg S_m) \vee (\neg S_k \wedge F^{1-\mu}_{\geqslant k+1})$$
$$F^{1-\mu}_{\geqslant n} = S_n \wedge F^{1-\mu}_n \tag{14}$$

Finally, let us define a new score function $\nu : \text{lits}(\mathbf{Z} \cup \mathbf{S}) \to \mathbb{R}$ such that, for $Z_{i,j} \in \mathbf{Z}$ and $S_k \in \mathbf{S}$:

$$\begin{array}{ll} \nu(Z_{ij}) = -1 & \nu(\neg Z_{ij}) = 0 \\ \nu(S_k) = \binom{k}{2} & \nu(\neg S_k) = 0 \end{array} \tag{15}$$

Proposition 9. *$(F^{1-\mu}_{\geqslant 0}, f, \nu)$ is a weighted functional encoding for ℓ_μ.*

Proposition 10. *If $F^{1-\mu}$ is in sDNNF, then $F^{1-\mu}_{\geqslant 0}$ can be compiled in a sDNNF of size $\mathcal{O}(n^2 \cdot |F^{1-\mu}|)$.*

Proof. Detailed proofs for Propositions 9 and 10 can be found in Appendix B.1.2. □

8 EXPERIMENTS

In this section, we quickly put to the test the enumeration algorithms presented in this paper: we compare modular and filtered enumeration algorithms presented in Sections 5 and 6 respectively against the search space reduction approach introduced in Maltoudoglou et al. [2022]. To do so we use the Warcraft Shortest Path dataset Pogančić et al. [2019], for which simple path constraints can be efficiently compiled in a sDNNF (see Appendix C.1). We train a simple multi-label neural network on a train split using a Binary Cross-Entropy loss (see Table 1). We compute loss thresholds for several user-defined miscoverage rates using a calibration split. Finally, for each miscoverage rate, we compare the different methods based on the size of their search spaces and their timeout rates for various timewalls.

8.1 SEARCH SPACE SIZE

The search space is defined relative to each method, in order to measure the portion of the output space that is explored during enumeration. For modular and filtered enumeration, the size search space can be effectively measured by the size of the final confidence set, as only those labelsets will be *visited* during the enumeration. For Maltoudoglou et al. [2022], the search space is defined based on the maximal hamming distance from the *best* (*i.e.*, the least non-conform) labelset reached by an element of the output confidence set. The size of the search space is an interesting proxy of enumeration efficiency for two reasons: first, all three methods have an enumeration time proportional to the size of their search space, second, it is faster to compute than it is to enumerate

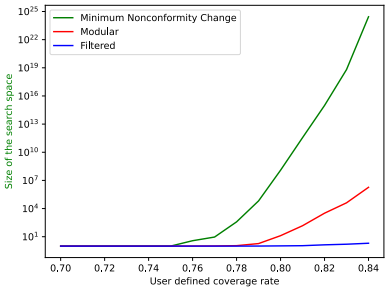


Figure 1: Size of the search space depending on the user-defined coverage rate for each enumeration method

the confidence sets. By definition, search spaces of reduced, modular and filtered enumeration are of decreasing sizes (the reduced search space of Maltoudoglou et al. [2022] contains the final confidence set, which itself contains the filtered confidence set). The exponential separation in the sizes of search spaces illustrated on Figure 1 clearly shows the magnitude of this gap in practice. Finally, as smaller confidence sets provide more information to the user, the size difference between unfiltered and filtered confidence sets (*i.e.*, modular and filtered search spaces) showed on Figure 1 has a great significance beyond enumeration efficiency.

8.2 TIMEOUTS

Although search space sizes can be considered a good proxy of a method's scalability, it does not take into account the practical compilation costs that can be hidden in the constant factor between the size of search space and the runtime of enumeration. In practice, this constant could easily be high enough to completely overweigh the scalability of filtered enumeration. Our results show that modular enumeration is systematically faster than reduced search space. Regarding filtered enumeration, the answer depends on the timewall chosen to stop enumeration before completion. For a time-

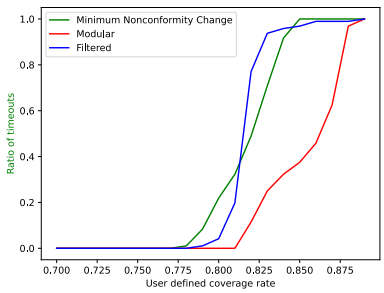


Figure 2: Proportion of timeouts after 1 second of enumeration time depending on the user-defined coverage rate for each enumeration method

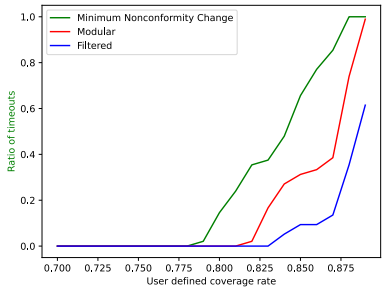


Figure 3: Proportion of timeouts after 5 seconds of enumeration time depending on the user-defined coverage rate for each enumeration method

wall of 1s, Figure 2 shows that filtered enumeration only scales better for large coverage rates (*i.e.*, when the confidence sets are larger). For a timewall of 5s however, Figure 3 shows that filtered enumeration is competitive for small confidence rates and maintains scalability for much longer than reduced search space and modular enumeration. This tells us that, in practice, the choice of the most efficient method between filtered and unfiltered modular enumeration may depend on the user defined coverage rate and the expected size of the corresponding confidence sets.

8.3 WEIGHTED FUNCTIONAL ENCODINGS

Finally, we evaluate in this section the applicability of weighted functional encodings to linearize pairwise interactions in the non-conformity measure. We applied the encoding for dense interaction matrices described in Section 7.2 to the `arxiv_categories` dataset [Schopf et al., 2024] used in Najjar et al. [2026]. The interaction matrix is shown on Figure 7b in Appendix C.2. Given the density of μ, we decided to test the weighted functional encoding based on $F^{1-\mu}$: there are 196 gaps in the upper right half of the matrix μ, resulting in as many auxiliary variables in $F^{1-\mu}$. We compiled $F^{1-\mu}$ in a `sDNNF` using the `d4` compiler introduced in Lagniez and Marquis [2017], which yielded in less than 5ms a circuit with a few thousands edges, much smaller than the exponential size 2^{130} of the output space. Hence, by exploiting jointly Corollary 7 with Propositions 9 and 10, we can provide an enumeration algorithm with a delay in $\mathcal{O}(n^3 \cdot |F^{1-\mu}|)$ that works for any vector of scores $\mathbf{a} \in [0,1]^n$ produced by the neural network.

9 CONCLUSION

This paper analyzes the complexity of confidence set enumeration in multi-label conformal classification. We first detail a general method for confidence set enumeration based on the flashlight method and prove a sufficient condition for efficient enumeration. Then, we applied this method on three specific types of problems: modular losses, informed conformal classification and weighted functional encodings for complex loss functions. We designed either polynomial delay or fixed-parameter polynomial delay enumeration algorithms for each of these cases. Finally, we illustrated the efficiency of our algorithms with a few experiments.

Future research directions include more experimental evaluations to validate the practical benefits of our algorithms. In particular, we would like to analyze to which extent informed conformal classification allows to increase the coverage rates while maintaining manageable confidence sets. Eventually, in case where confidence sets become too large, boolean circuits can also be leveraged to build a compressed representation of confidence sets which can then be queried.

Acknowledgements

This work was supported by State funding managed by the French National Research Agency (ANR) under France 2030 and by the European Union through NextGenerationEU, reference 22-EXES-0009.

References

Kareem Ahmed, Stefano Teso, Kai-Wei Chang, Guy den Broeck, and Antonio Vergari. Semantic Probabilistic Layers for Neuro-Symbolic Learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 29944–29959. Curran Associates, Inc., 2022.

Gilles Audemard, Frédéric Koriche, and Pierre Marquis. On Tractable XAI Queries based on Compiled Representations. *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 17(1):838–849, July 2020. ISSN 2334-1033. doi: 10.24963/kr.2020/86. URL `https://proceedings.kr.org/2020/86/`. Conference Name: Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning.

David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1):21–46, 1996. ISSN 0166-218X. doi: https://doi.org/10.1016/0166-218X(95)00026-N. URL `https://www.sciencedirect.com/science/article/pii/0166218X9500026N`.

Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17(1):229–264, September 2002. ISSN 1076-9757.

Adnan Darwiche and Pierre Marquis. Compiling propositional weighted bases. *Artificial Intelligence*, 157(1-2):81–113, August 2004. ISSN 00043702. doi: 10.1016/j.artint.2004.04.005.

URL https://linkinghub.elsevier.com/retrieve/pii/S000437020400058X.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks, August 2017. URL http://arxiv.org/abs/1706.04599. Issue: arXiv:1706.04599 arXiv: 1706.04599 [cs].

David S. Johnson, Mihalis Yannakakis, and Christos H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988. ISSN 0020-0190. doi: https://doi.org/10.1016/0020-0190(88)90065-8. URL https://www.sciencedirect.com/science/article/pii/0020019088900658.

Angelika Kimmig, Guy Van den Broeck, and Luc De Raedt. Algebraic model counting. 22:46–62, 2017. ISSN 1570-8683. doi: 10.1016/j.jal.2016.11.031. URL https://doi.org/10.1016/j.jal.2016.11.031.

Jean-Marie Lagniez and Pierre Marquis. An Improved Decision-DNNF Compiler. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 667–673, 2017. doi: 10.24963/ijcai.2017/93. URL https://doi.org/10.24963/ijcai.2017/93.

Antonis Lambrou and Harris Papadopoulos. Binary relevance multi-label conformal predictor. In Alexander Gammerman, Zhiyuan Luo, Jesús Vega, and Vladimir Vovk, editors, *Conformal and Probabilistic Prediction with Applications*, pages 90–104. Springer International Publishing, 2016. ISBN 978-3-319-33395-3. doi: 10.1007/978-3-319-33395-3_7.

Arthur Ledaguenel, Céline Hudelot, and Mostepha Khouadjia. Improving neural-based classification with logical background knowledge, 2024. URL https://arxiv.org/abs/2402.13019.

Arthur Ledaguenel, Céline Hudelot, and Mostepha Khouadjia. A Complexity Map of Probabilistic Reasoning for Neurosymbolic Classification Techniques. *Mathematics in Computer Science*, 19(1):8, August 2025. ISSN 1661-8289. doi: 10.1007/s11786-025-00603-7. URL https://doi.org/10.1007/s11786-025-00603-7.

Lysimachos Maltoudoglou, Andreas Paisios, Ladislav Lenc, Jiří Martínek, Pavel Král, and Harris Papadopoulos. Well-calibrated confidence measures for multi-label text classification with a large number of labels. *Pattern Recognition*, 122:108271, February 2022. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2021.108271. URL https://www.sciencedirect.com/science/article/pii/S0031320321004519.

Ghassan Najjar, Céline Berthou, and Héléna Vorobieva. Scaling Multi-label Conformal Prediction with Label Interactions for a Large Number of Labels. In Rita P. Ribeiro, Bernhard Pfahringer, Nathalie Japkowicz, Pedro Larrañaga, Alípio M. Jorge, Carlos Soares, Pedro H. Abreu, and João Gama, editors, *Machine Learning and Knowledge Discovery in Databases. Research Track*, pages 111–128, Cham, 2026. Springer Nature Switzerland. ISBN 978-3-032-06109-6. doi: 10.1007/978-3-032-06109-6_7.

Mathias Niepert, Pasquale Minervini, and Luca Franceschi. Implicit MLE: Backpropagating through discrete exponential family distributions. In *Advances in Neural Information Processing Systems*, volume 34, pages 14567–14579. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/7a430339c10c642c4b2251756fd1b484-Abstract.html.

Harris Papadopoulos. A cross-conformal predictor for multi-label classification. In Lazaros Iliadis, Ilias Maglogiannis, Harris Papadopoulos, Spyros Sioutas, and Christos Makris, editors, *Artificial Intelligence Applications and Innovations*, pages 241–250. Springer, 2014. ISBN 978-3-662-44722-2. doi: 10.1007/978-3-662-44722-2_26.

Harris Papadopoulos, Vladimir Vovk, and Alex Gammerman. Qualified prediction for large data sets in the case of pattern recognition. In M. Arif Wani, Hamid R. Arabnia, Krzysztof J. Cios, Khalid Hafeez, and Graham Kendall, editors, *Proceedings of the 2002 International Conference on Machine Learning and Applications - ICMLA 2002, June 24-27, 2002, Las Vegas, Nevada, USA*, pages 159–163. CSREA Press, 2002. URL https://dblp.org/rec/conf/icmla/PapadopoulosVG02.bib.

Marin Vlastelica Pogančić, Anselm Paulus, Vit Musil, Georg Martius, and Michal Rolinek. Differentiation of Blackbox Combinatorial Solvers. September 2019. URL https://openreview.net/forum?id=BkevoJSYPB.

Ramya Ramalingam, Sangdon Park, and Osbert Bastani. Uncertainty quantification for neurosymbolic programs via compositional conformal prediction. abs/2405.15912, 2024. doi: 10.48550/ARXIV.2405.15912. URL https://doi.org/10.48550/arXiv.2405.15912.

Tim Schopf, Alexander Blatzheim, Nektarios Machner, and Florian Matthes. Efficient few-shot learning for multi-label classification of scientific documents with many classes. In Mourad Abbas and Abed Alhakim Freihat, editors, *Proceedings of the 7th International Conference on Natural Language and Speech Processing (ICNLSP 2024)*, pages 186–198, Trento, October 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.icnlsp-1.21/.

Huazhen Wang, Xin Liu, Ilia Nouretdinov, and Zhiyuan Luo. A Comparison of Three Implementations of Multi-Label Conformal Prediction. In Alexander Gammerman, Vladimir Vovk, and Harris Papadopoulos, editors, *Statistical Learning and Data Sciences*, pages 241–250, Cham, 2015. Springer International Publishing. ISBN 978-3-319-17091-6. doi: 10.1007/978-3-319-17091-6_19.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van Den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *35th International Conference on Machine Learning, ICML 2018*, volume 12, pages 8752–8760. International Machine Learning Society (IMLS), 2018. ISBN 9781510867963.

Fan Yang, Xiaolu Gan, Huazhen Wang, Lei Feng, and Yongxuan Lai. CP-RA\k{EL: Improving Random \k\-labelsets with Conformal Prediction for Multi-label Classification. In *Proceedings of the Sixth Workshop on Conformal and Probabilistic Prediction and Applications*, pages 266–279. PMLR, May 2017. URL `https://proceedings.mlr.press/v60/yang17a.html`.

Zhun Yang, Adam Ishay, and Joohyung Lee. NeurASP: Embracing neural networks into answer set programming. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 1755–1762. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-6-5. doi: 10.24963/ijcai.2020/243.

Efficient Confidence Set Enumeration for Multi-label Conformal Classification (Supplementary Material)

Arthur Ledaguenel[1,2,†] **Florent Capelli**[1] **Jean-Marie Lagniez**[1]

[1]Univ. Artois, CNRS, UMR 8188, Centre de Recherche en Informatique de Lens (CRIL), F-62300 Lens, France
[2]Laboratoire d'Informatique Signal et Image de la Côte d'Opale (LISIC), Université du Littoral Côte d'Opale (ULCO), Calais, France
[†]Corresponding author

A ALGORITHMS

A.1 FLASHLIGHT ENUMERATION

In this section, we graphically detail the behavior of Algorithm 1 for confidence set enumeration based on the flashlight method, which we recall below.

Algorithm 2 Enumerate the confidence set

Require: Variables $\mathbf{Y} = \{Y_1, ..., Y_n\}$, a vector of scores $\mathbf{a} \in \mathbb{R}^n$ (produced by the trained neural network), a loss function $\ell : [0,1]^n \times \{0,1\}^n \to \mathbb{R}^+$ and a non-conformity threshold τ_ϵ
Ensure: $\mathcal{C}$ contains the confidence set $\mathcal{C}_\epsilon(\mathbf{a})$

function ENUMERATE($\mathbf{a}, \ell, \tau_\epsilon$)
  INITIALIZE an empty confidence set $\mathcal{C}$
  INITIALIZE an empty First-In First-Out queue Q
  $m \leftarrow$ MIN($\ell, \mathbf{a}, [\,]$)
  if $m \leqslant \tau_\epsilon$ **then**
    PUSH($Q, [\,]$)
  else
    return $\varnothing$
  while Q is not empty **do**
    $\pi \leftarrow$ POP(Q)
    if π is a labelset **then**
      APPEND π to $\mathcal{C}$
    else
      for $b \in \{0,1\}$ **do**
        $m^{+b} \leftarrow$ MIN($\ell, \mathbf{a}, \pi^{+b}$)
        if $m^{+b} \leqslant \tau_\epsilon$ **then**
          PUSH(Q, π^{+b})
  return $\mathcal{C}$

First, let $\mathcal{T}$ be the binary tree such that:

- the root corresponds to the 0-prefix $[\,]$
- each internal vertice v corresponds to a k-**prefix** $\pi_v \in \{0,1\}^k$
- the left (resp. right) children of v in $\mathcal{T}$ corresponds to π_v^{+0} (resp. π_v^{+1})
- leaves correspond to labelsets

Now let us consider the subtree of $\mathcal{T}$ induced by leaf nodes which correspond to labelsets in $\mathcal{C}_\epsilon(x)$ and all the nodes that lead to them, which we note $\mathcal{T}_\epsilon(x)$. Algorithm 1 consists in a depth first exploration of $\mathcal{T}_\epsilon(x)$. To do this without traversing the complete tree $\mathcal{T}$, which is exponential in the number n of variables, we check before exploring any branch that it leads to at least one labelset that belongs to the confidence set. Doing so, the number of visited vertices is bounded by $n \times k$ with k the size of the confidence set.

Example 1. *An example of $\mathcal{T}$ for three variables is shown on Figure 4. The left and right children of $[\,]$ are $[0]$ and $[1]$ respectively. The subtree induced by the confidence set $\mathcal{C}_\epsilon(x) = \{[0,0,1],[1,0,0],[1,0,1]\}$ is shown on Figure 5.*

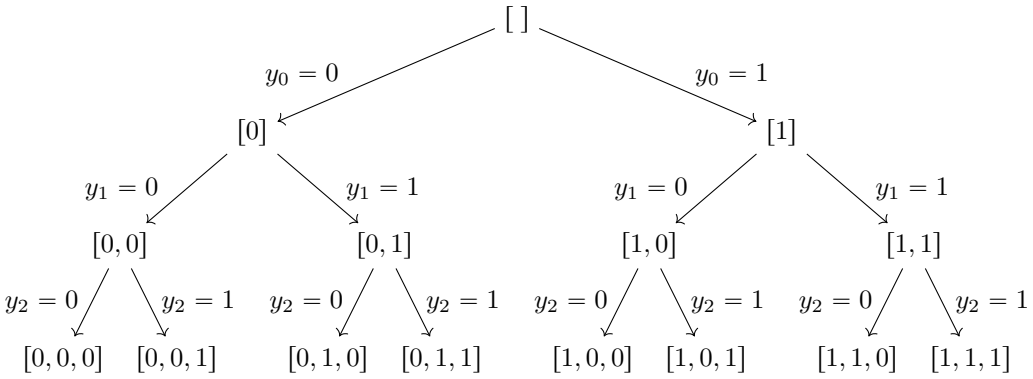


Figure 4: Illustration of the complete binary tree for 3 variables

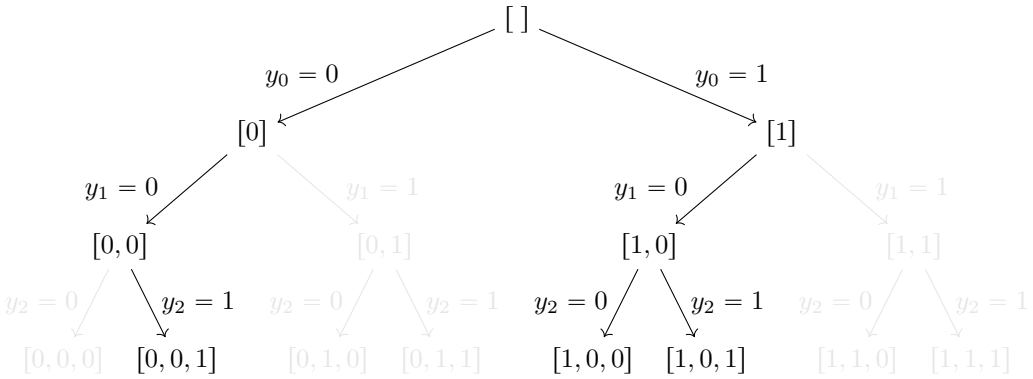


Figure 5: Illustration of the subtree $\mathcal{T}_\epsilon(x)$ induced by the confidence set $\mathcal{C}_\epsilon(x) = \{[0,0,1],[1,0,0],[1,0,1]\}$

A.2 RANKED ENUMERATION

Algorithm 3 presents a modified version of Algorithm 1 which outputs elements of the confidence set in increasing order of non-conformity with an increased delay of $\mathcal{O}(n^{p+1} \times \log(n \cdot k))$ where k is the number of labelsets enumerated so far. The only difference with Algorithm 1 is that the first-in first-out queue is replaced by a priority queue where elements (m, π) of the queue are sorted in increasing order of m (ties are broken by recency of insertion). Changing the type of the queue does not alter the delay between two labelsets: when a couple (m, π) is popped from the queue, either π is a labelset or the minimal non-conformity extension of π will end-up on top of the queue. Therefore, the size of the prefix still strictly grows at each step, preventing more than n steps between two output labelsets. However, operations PUSH and POP can no longer be performed in constant time and require instead a logarithmic cost in the size of the queue. Observe that at each step, at most two elements are pushed into the queue and at least one is popped from the queue. Therefore, the number of elements in the queue is bounded by the number of steps so far, which is itself bounded by $n \times k$ where n is the number of variables and k is the number of labelsets enumerated so far.

Algorithm 3 Enumerate the confidence set in increasing order of non-conformity

Require: Variables $\mathbf{Y} = \{Y_1, ..., Y_n\}$, a vector of scores $\mathbf{a} \in \mathbb{R}^n$ (produced by the trained neural network), a loss function $\ell : [0, 1]^n \times \{0, 1\}^n \to \mathbb{R}^+$ and a non-conformity threshold τ_ϵ
Ensure: $\mathcal{C}$ contains the confidence set $\mathcal{C}_\epsilon(\mathbf{a})$

function RANKEDENUMERATE($\mathbf{a}, \ell, \tau_\epsilon$)
    INITIALIZE an empty confidence set $\mathcal{C}$
    INITIALIZE an empty min priority queue Q
    $m \leftarrow$ MIN($\ell, \mathbf{a}, [\,]$)
    if $m \leqslant \tau_\epsilon$ **then**
        PUSH($Q, (m, [\,])$)
    else
        return $\varnothing$
    while Q is not empty **do**
        $(m, \pi) \leftarrow$ POP(Q)
        if π is a labelset **then**
            APPEND π to $\mathcal{C}$
        else
            for $b \in \{0, 1\}$ **do**
                $m^{+b} \leftarrow$ MIN($\ell, \mathbf{a}, \pi^{+b}$)
                if $m^{+b} \leqslant \tau_\epsilon$ **then**
                    PUSH($Q, (m^{+b}, \pi^{+b})$)
    return $\mathcal{C}$

A.3 COUNTING

Algorithm 4 presents a modified version of Algorithm 1 which counts the number of labelsets in the confidence set. Instead of simply checking the minimum of the branch as in Algorithm 1, we also check the maximum of the branch to know if all extensions of the current prefix are contained in the confidence set, which avoids useless and expansive exploration time.

Algorithm 4 Count the number of labelsets in the confidence set

Require: Variables $\mathbf{Y} = \{Y_1, ..., Y_n\}$, a vector of scores $\mathbf{a} \in \mathbb{R}^n$ (produced by the trained neural network), a loss function $\ell : [0,1]^n \times \{0,1\}^n \to \mathbb{R}^+$ and a non-conformity threshold τ_ϵ
Ensure: c is equal to the size of the confidence set $|\mathcal{C}_\epsilon(\mathbf{a})|$

function COUNT($\mathbf{a}, \ell, \tau_\epsilon$)
    INITIALIZE an empty confidence set $\mathcal{C}$
    INITIALIZE an empty First-In First-Out queue Q
    $ma \leftarrow \text{MAX}(\ell, \mathbf{a}, [\,])$
    if $ma \leqslant \tau_\epsilon$ **then**
        return 2^n
    $mi \leftarrow \text{MIN}(\ell, \mathbf{a}, [\,])$
    if $m \leqslant \tau_\epsilon$ **then**
        PUSH($Q, (0, [\,])$)
    else
        return $\varnothing$
    while Q is not empty **do**
        $l, \pi \leftarrow \text{POP}(Q)$
        if $l = n$ **then**
            $c \leftarrow c + 1$
        else
            for $b \in \{0, 1\}$ **do**
                $ma^{+b} \leftarrow \text{MAX}(\ell, \mathbf{a}, \pi^{+b})$
                if $ma^{+b} \leqslant \tau_\epsilon$ **then**
                    $c \leftarrow c + 2^{n-l-1}$
                $mi^{+b} \leftarrow \text{MIN}(\ell, \mathbf{a}, \pi^{+b})$
                if $mi^{+b} \leqslant \tau_\epsilon$ **then**
                    PUSH($Q, (l + 1, \pi^{+b})$)
    return c

A.4 MODULAR FUNCTIONS

Algorithm 5 is a detailed version of the optimization algorithm that solves $\texttt{Min}(\ell)$ in $\mathcal{O}(n)$ for modular loss functions $\ell(\mathbf{a}, \mathbf{y}) = \sum_{1 \leqslant i \leqslant n} f(a_i, y_i)$ as sketched in the proof of Proposition 2. Besides, as mentioned in Section 5, this algorithm coupled with the flashlight method presented in Algorithm 1 provides an enumeration algorithm for modular function with a quadratic delay.

Algorithm 5 Computes the minimum non-conformity extension of a partial instantiation for a modular loss function

Require: Variables $\mathbf{Y} = \{Y_1, ..., Y_n\}$, a vector of scores $\mathbf{a} \in \mathbb{R}^n$ (produced by the trained neural network), a modular loss $\ell(\mathbf{a}, \mathbf{y}) = \sum_{1 \leqslant i \leqslant n} f(a_i, y_i)$ and a partial instantiation π
Ensure: m is equal to $\min_{\substack{\mathbf{y} \in \mathcal{Y}, \\ \mathbf{y} \models \pi}} \ell(\mathbf{a}, \mathbf{y})$

function DECMIN($f, \mathbf{a}, \pi$)
    $I \leftarrow \varnothing$
    $m \leftarrow 0$
    for $l \in \pi$ **do**
        $i \leftarrow \texttt{var}(l)$
        $m \leftarrow m + f(a_i, l)$
        APPEND i to I
    for $j \in \{1, ..., n\} \backslash I$ **do**
        $m \leftarrow m + \min(f(a_j, 0), f(a_j, 1))$
    return m

In fact, we prove in Proposition 2 that a tighter integration of Algorithms 1 and 5 enables to enumerate confidence sets of modular functions with a delay in $\mathcal{O}(n)$. We provide below the corresponding Algorithm 6.

Algorithm 6 Enumerate the confidence set for modular loss functions

Require: Variables $\mathbf{Y} = \{Y_1, ..., Y_n\}$, a vector of scores $\mathbf{a} \in \mathbb{R}^n$ (produced by the trained neural network), a modular loss $\ell(\mathbf{a}, \mathbf{y}) = \sum_{1 \leqslant i \leqslant n} f(a_i, y_i)$ and a non-conformity threshold τ_ϵ
Ensure: $\mathcal{C}$ contains the confidence set $\mathcal{C}_\epsilon(\mathbf{a})$

function DECENUMERATE($\mathbf{a}, \ell, \tau_\epsilon$)
    INITIALIZE an empty confidence set $\mathcal{C}$
    INITIALIZE an empty last-in first-out queue Q
    $mins \leftarrow \varnothing$
    $indx \leftarrow \varnothing$
    for $i \in \{0, ..., n\}$ **do**
        APPEND $\min_{b \in \{0,1\}} f(a_i, b)$ to $mins$
        APPEND $\operatorname{argmin}_{b \in \{0,1\}} f(a_i, b)$ to $indx$
    $m \leftarrow \sum_{1 \leqslant i \leqslant n} mins[i]$
    if $m \leqslant \tau_\epsilon$ **then**
        PUSH$(Q, (m, [\,]))$
    else
        return $\varnothing$
    while Q is not empty **do**
        $(m, \pi) \leftarrow$ POP(Q)
        $l \leftarrow$ LEN(π)
        if $l = n$ **then**
            APPEND π to $\mathcal{C}$
        else
            $b \leftarrow indx[l + 1]$
            PUSH$(Q, (m, \pi^{+b}))$
            $m \leftarrow m + f(a_i, 1 - b) - f(a_i, b)$
            if $m \leqslant \tau_\epsilon$ **then**
                $b \leftarrow 1 - b$
                PUSH$(Q, (m, \pi^{+b}))$
    return $\mathcal{C}$

A.5 LINEAR OPTIMIZATION OF BOOLEAN CIRCUITS

To support our demonstration of Proposition 3 we provide below an algorithm for linear optimization of boolean circuits in sDNNF (see Algorithm 7).

Algorithm 7 Computes the minimum of a modular loss reached by any extension of a partial instantiation that satisfies a circuit in sDNNF

Require: Variables $\mathbf{Y} = \{Y_1, ..., Y_n\}$, a vector of scores $\mathbf{a} \in \mathbb{R}^n$ (produced by the trained neural network), a modular loss $\ell(\mathbf{a}, \mathbf{y}) = \sum_{1 \leqslant i \leqslant n} f(a_i, y_i)$, a circuit κ in sDNNF and a partial instantiation π

Ensure: m is equal to $\min_{\substack{\mathbf{y} \in \mathcal{Y}, \\ \mathbf{y} \models \kappa \wedge \pi}} \ell(\mathbf{a}, \mathbf{y})$

function sDNNFMin($f, \kappa, \mathbf{a}, \pi$)
  $r \leftarrow \texttt{root}(\kappa)$
  if r is Y_i **then**
    if $\pi \models \neg Y_i$ **then**
      return $+\infty$
    else
      return $f(a_i, 1)$
  else if r is $\neg Y_i$ **then**
    if $\pi \models Y_i$ **then**
      return $+\infty$
    else
      return $f(a_i, 0)$
  else if r is $\wedge$ **then**
    return $\sum_{c \in \texttt{ch}(r)}$ sDNNFMin$(f, c, \mathbf{a}, \pi)$
  else if r is $\vee$ **then**
    return $\min_{c \in \texttt{ch}(r)}$ sDNNFMin$(f, c, \mathbf{a}, \pi)$

B PROOFS

In this section we provide detailed proofs for Propositions 8, 9 and 10.

B.1 WEIGHTED FUNCTIONAL ENCODINGS

We assume a modular loss function $\ell(\mathbf{a}, \mathbf{y}) = \sum_{1 \leqslant i \leqslant n} f(a_i, y_i)$, a matrix of pairwise label interactions $\mu \in \{0,1\}^{n \times n}$ where $\mu_{ij} = 1$ if labels i and j have never been observed jointly in the training set, and $\mu_{ij} = 0$ otherwise.

We define the loss function $\ell_\mu : [0,1]^n \times \{0,1\}^n \to \mathbb{R}^+$ which includes pairwise label interactions as:

$$\ell_\mu(\mathbf{a}, \mathbf{y}) = \sum_{1 \leqslant i \leqslant n} f(a_i, y_i) + \sum_{1 \leqslant i < j \leqslant n} \mu_{ij} y_i y_j$$

We remind below the two encodings designed in Sections 7.1 and 7.2 then prove that they are well defined weighted functional encodings.

B.1.1 Sparse interactions

Let Z_{ij} be an auxiliary variable for each pair of labels (Y_i, Y_j) such that $i < j$ and $\mu_{ij} = 1$. We note $F^\mu(\mathbf{Y}, \mathbf{Z})$ the circuit in `CNF` defined as:

$$\begin{aligned} F^\mu &= \bigwedge_{\substack{1 \leqslant i < j \leqslant n, \\ \mu_{ij} = 1}} (Y_i \vee \neg Z_{ij}) \wedge (Y_j \vee \neg Z_{ij}) \wedge (Z_{ij} \vee \neg Y_i \vee \neg Y_j) \\ &\equiv \bigwedge_{\substack{1 \leqslant i < j \leqslant n, \\ \mu_{ij} = 1}} Z_{ij} \iff (Y_i \wedge Y_j) \end{aligned} \tag{16}$$

Finally, let $\nu : \mathrm{lits}(\mathbf{Z}) \to \mathbb{R}$ be the score function such that, for $Z_{ij} \in \mathbf{Z}$:

$$\nu(Z_{ij}) = 1 \qquad \nu(\neg Z_{ij}) = 0 \tag{17}$$

Proposition (Reminder of Proposition 8)**.** *$(F^\mu(\mathbf{Y}, \mathbf{Z}), f, \nu)$ defines a weighted functional encoding for ℓ_μ.*

Proof. First we show that F^μ enforces a functional dependency from $\mathbf{Y}$ to $\mathbf{Z}$, *i.e.*, for all $\mathbf{y} \in \mathcal{Y}$ there exists a unique $\mathbf{z} \in \mathcal{Z}$ such that $\mathbf{yz} \models F^\mu$. Suppose $\mathbf{y} \in \mathcal{Y}$, we note $\mathbf{z} \in \mathcal{Z}$ such that:

$$\forall 1 \leqslant i < j \leqslant n, \mu_{ij} = 1, z_{ij} = y_i \cdot y_j = \begin{cases} 1 & \text{if } y_i = y_j = 1 \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

One can easily notice that the $\mathbf{Y} \cup \mathbf{Z}$-instantiation $\mathbf{yz}$ satisfies all clauses of F^μ, therefore $\mathbf{yz} \models F^\mu$.

Suppose now another $\tilde{\mathbf{z}} \in \mathcal{Z}$ such that $\mathbf{y}\tilde{\mathbf{z}} \models F^\mu$. In particular, for any $1 \leqslant i < j \leqslant n$ such that $\mu_{ij} \neq 0$, we know that $\mathbf{y}\tilde{\mathbf{z}} \models Z_{ij} \iff (Y_i \wedge Y_j)$, which implies that $\tilde{z}_{ij} = y_i \cdot y_j = z_{ij}$. Therefore, F^μ enforces a functional dependency from $\mathbf{Y}$ to $\mathbf{Z}$.

Then, let us show that:

$$\forall \mathbf{y} \in \mathcal{Y}, \ell_\mu(\mathbf{y}, \mathbf{a}) = \sum_{1 \leqslant i \leqslant n} f(a_i, y_i) + \sum_{\substack{l \in \mathrm{lits}(\mathbf{Z}), \\ F^\mu(\mathbf{y}) \models l}} \nu(l)$$

We have from Equation 18:

$$\forall \mathbf{y} \in \mathcal{Y}, \sum_{1 \leqslant i < j \leqslant n} \mu_{ij} y_i y_j = \sum_{\substack{1 \leqslant i < j \leqslant n \\ \mu_{ij} = 1}} y_i y_j = \sum_{\substack{1 \leqslant i < j \leqslant n \\ \mu_{ij} = 1}} F^\mu(\mathbf{y})_{ij}$$

Moreover, notice that for any literal $l \in \text{lits}(\mathbf{Z})$ in $F^\mu(\mathbf{y})$, either $l = Z_{ij}$ and $\nu(l) = 1 = F^\mu(\mathbf{y})_{ij}$, either $l = \neg Z_{ij}$ and $\nu(l) = 0 = F^\mu(\mathbf{y})_{ij}$. Hence:

$$\forall \mathbf{y} \in \mathcal{Y}, \sum_{1 \leqslant i < j \leqslant n} \mu_{ij} y_i y_j = \sum_{\substack{l \in \text{lits}(\mathbf{Z}), \\ F^\mu(\mathbf{y}) \models l}} \nu(l)$$

Finally we have:

$$\forall \mathbf{y} \in \mathcal{Y}, \ell_\mu(\mathbf{y}, \mathbf{a}) = \sum_{1 \leqslant i \leqslant n} f(a_i, y_i) + \sum_{1 \leqslant i < j \leqslant n} \mu_{ij} y_i y_j = \sum_{1 \leqslant i \leqslant n} f(a_i, y_i) + \sum_{\substack{l \in \text{lits}(\mathbf{Z}), \\ F^\mu(\mathbf{y}) \models l}} \nu(l)$$

Therefore, $(F^\mu(\mathbf{Y}, \mathbf{Z}), f, \nu)$ defines a weighted functional encoding for ℓ_μ. □

B.1.2 Dense interactions

We assume $F^{1-\mu}$ defined as in Equation 16 for $1 - \mu$ instead of μ.

Let us also assume circuits $F_0^{1-\mu}(\mathbf{Y}), ..., F_n^{1-\mu}(\mathbf{Y})$ such that:

$$\begin{gathered} \forall 0 \leqslant k \leqslant n, \forall \mathbf{y} \in \mathcal{Y}, \forall \mathbf{z} \in \mathcal{Z}, \\ \mathbf{yz} \models F_k^{1-\mu} \iff \mathbf{yz} \models F^{1-\mu} \text{ and } \sum_{1 \leqslant i \leqslant n} y_i = k \end{gathered} \tag{19}$$

We introduce new variables $\mathbf{S} = \{S_0, ..., S_n\}$ and recursively define, for $0 \leqslant k \leqslant n-1$:

$$\begin{aligned} F_{\geqslant k}^{1-\mu} &= (S_k \wedge F_k^{1-\mu} \wedge_{k+1 \leqslant m \leqslant n} \neg S_m) \vee (\neg S_k \wedge F_{\geqslant k+1}^{1-\mu}) \\ F_{\geqslant n}^{1-\mu} &= S_n \wedge F_n^{1-\mu} \end{aligned} \tag{20}$$

Finally, we define a new score function $\nu : \text{lits}(\mathbf{Z} \cup \mathbf{S}) \to \mathbb{R}$ such that, for any $Z_{i,j} \in \mathbf{Z}$ and $S_k \in \mathbf{S}$:

$$\begin{aligned} \nu(Z_{ij}) &= -1 & \nu(\neg Z_{ij}) &= 0 \\ \nu(S_k) &= \tbinom{k}{2} & \nu(\neg S_k) &= 0 \end{aligned} \tag{21}$$

Proposition (Reminder of Proposition 9)**.** *$(F_{\geqslant 0}^{1-\mu}, f, \nu)$ is a weighted functional encoding for ℓ_μ.*

Proof. Suppose $\mathbf{y} \in \mathcal{Y}$ and note $r = \sum_{1 \leqslant i \leqslant n} y_i$, $\mathbf{z}$ such that $z_{ij} = y_i \cdot y_j$ for all $1 \leqslant i < j \leqslant n$ with $\mu_{ij} = 0$ and $\mathbf{s} \in \mathcal{S}$ such that $s_r = 1$ and $s_k = 0$ for $k \neq r$.

First we show that $F_{\geqslant 0}^{1-\mu}$ enforces a functional dependency from $\mathbf{Y}$ to $\mathbf{Z} \cup \mathbf{S}$, *i.e.*, $\mathbf{z}$ and $\mathbf{s}$ are the only instantiations such that $\mathbf{yzs} \models F_{\geqslant 0}^{1-\mu}$.

Notice that by definition of $F_r^{1-\mu}$ in Equation 19 $\mathbf{yz} \models F_r^{1-\mu}$. Besides, by definition $\mathbf{s} \models S_r$ and $\mathbf{s} \models \neg S_k$ for $k \neq r$. Therefore, recursively developing Equation 20, we have $\mathbf{yzs} \models S_r \wedge F_r^{1-\mu} \wedge_{r+1 \leqslant m \leqslant n} \neg S_m \models F_{\geqslant r}^{1-\mu}$ and for $k = r-1, ..., 0$, $\mathbf{yzs} \models \neg S_k \wedge F_{\geqslant k+1}^{1-\mu} \models F_{\geqslant k}^{1-\mu}$. Hence, $\mathbf{yzs} \models F_{\geqslant 0}^{1-\mu}$.

Besides, one can see from Equations 19 and 20 that $F_{\geqslant 0}^{1-\mu} \models \bigvee_{0 \leqslant k \leqslant n} F_n^{1-\mu} \models F^{1-\mu}$, which proves the unicity of $\mathbf{z}$ based on the proof of Proposition 8 given in Appendix B.1.1. Finally, suppose $\tilde{\mathbf{s}} \in \mathcal{S}$ such that $\mathbf{yz}\tilde{\mathbf{s}} \models F_{\geqslant 0}^{1-\mu}$. Then by developing Equation 20, we have:

$$\tilde{\mathbf{s}} \models \bigvee_{0 \leqslant k \leqslant n} (S_k \bigwedge_{\substack{0 \leqslant m \leqslant n \\ m \neq k}} \neg S_m)$$

In particular, suppose that $\tilde{\mathbf{s}} \models S_m$ with $m \neq r$, then necessarily $\mathbf{yz}\tilde{\mathbf{s}} \models F_m^{1-\mu}$ which is impossible since $\sum_{1 \leqslant i \leqslant n} y_i = r \neq m$. Therefore, $\mathbf{s}$ is unique as well.

Then, let us show that:

$$\ell_\mu(\mathbf{y},\mathbf{a}) = \sum_{1\leqslant i\leqslant n} f(a_i,y_i) + \sum_{\substack{l\in\mathrm{lits}(\mathbf{Z}\cup\mathbf{S}),\\ \mathbf{zs}\models l}} \nu(l)$$

We have:

$$\sum_{1\leqslant i<j\leqslant n} \mu_{ij}y_iy_j = \sum_{1\leqslant i<j\leqslant n} y_iy_j - \sum_{1\leqslant i<j\leqslant n}(1-\mu_{ij})y_iy_j = \sum_{\substack{1\leqslant i<j\leqslant n,\\ y_i=y_j=1}} 1 - \sum_{\substack{1\leqslant i<j\leqslant n,\\ \mu_{ij}=0}} y_iy_j = \binom{r}{2} - \sum_{\substack{1\leqslant i<j\leqslant n,\\ \mu_{ij}=0}} z_{ij} \quad (22)$$

Moreover, notice that for any literal $l\in\mathrm{lits}(\mathbf{Z})$ such that $\mathbf{z}\models l$, either $l=Z_{ij}$ and $\nu(l)=-1=-z_{ij}$, either $l=\neg Z_{ij}$ and $\nu(l)=0=-z_{ij}$. Hence

$$-\sum_{\substack{1\leqslant i<j\leqslant n,\\ \mu_{ij}=0}} z_{ij} = \sum_{\substack{l\in\mathrm{lits}(\mathbf{Z}),\\ \mathbf{z}\models l}} \nu(l) \quad (23)$$

Likewise for any literal $l\in\mathrm{lits}(\mathbf{S})$ such that $\mathbf{s}\models l$, either $l=S_r$ and $\nu(l)=\binom{r}{2}$ or $l=\neg S_k$ with $k\neq r$ and $\nu(l)=0$. Hence:

$$\binom{r}{2} = \sum_{\substack{l\in\mathrm{lits}(\mathbf{S}),\\ \mathbf{s}\models l}} \nu(l) \quad (24)$$

Combining Equations 22, 23 and 24 we have:

$$\ell_\mu(\mathbf{y},\mathbf{a}) = \sum_{1\leqslant i\leqslant n} f(a_i,y_i) + \binom{r}{2} - \sum_{\substack{1\leqslant i<j\leqslant n,\\ \mu_{ij}=0}} z_{ij} = \sum_{1\leqslant i\leqslant n} f(a_i,y_i) + \sum_{\substack{l\in\mathrm{lits}(\mathbf{Z}\cup\mathbf{S}),\\ \mathbf{zs}\models l}} \nu(l)$$

Therefore $(F^{1-\mu}_{\geqslant 0}, f, \nu)$ is a weighted functional encoding for ℓ_μ. □

Proposition (Reminder of Proposition 10)**.** *If $F^{1-\mu}$ is in sDNNF, then $F^{1-\mu}_{\geqslant 0}$ can be compiled in a sDNNF of size $\mathcal{O}(n^2\cdot|F^{1-\mu}|)$.*

If first start to prove the following Lemma:

Lemma 11. *Given a circuit κ in sDNNF on variables $\{Y_1,...,Y_n\}$ and $p\leqslant n$, we can build circuits $\kappa_0,...,\kappa_p$ in sDNNF of combined sizes $\mathcal{O}(p^2\cdot|\kappa|)$ such that:*

$$\forall 0\leqslant k\leqslant p, \mathbf{y}\models\kappa_k \iff \mathbf{y}\models\kappa \text{ and } \sum_{1\leqslant i\leqslant n} y_i = k$$

Remark 7. *By combined size we do not mean the sum of their sizes taken individually but the number of edges needed if they were to all appear in the same circuit. This reduces the size up to a factor $\mathcal{O}(p)$ since these circuits share a lot of sub-circuits.*

Proof. We develop all $\wedge$-nodes with more than two children until we only have binary $\wedge$-nodes left. This can be done without more than doubling the size of the circuit.

For each node u in κ, we inductively build nodes $u_0,...,u_p$ in the following way:

- if u is either $\top$ or $\bot$, then $u_0=u$ and $u_k=\bot$ for $0<k\leqslant p$
- if u is Y_i, then $u_0=\bot$, $u_1=Y_i$ and $u_k=\bot$ for $1<k\leqslant p$
- if u is $\neg Y_i$, then $u_0=\neg Y_i$ and $u_k=\bot$ for $0<k\leqslant p$
- if u is $\bigvee_{c\in\mathtt{ch}(u)} c$, then:

$$\forall 0\leqslant k\leqslant p, u_k = \bigvee_{c\in\mathtt{ch}(u)} c_k$$

- if u is $\bigwedge_{c \in \mathtt{ch}(u)} c$, with children c^1, c^2, then:

$$\forall 0 \leqslant k \leqslant p, u_k = \bigvee_{0 \leqslant m \leqslant k} c^1_m \wedge c^2_{k-m}$$

Notice that new $\vee$-nodes are still smooth (included those added in $\bigvee_{0 \leqslant m \leqslant k} c^1_m \wedge c^2_{k-m}$) and new $\wedge$-nodes are still decomposable.

For any node u, we show below that nodes $u_0, ..., u_p$ are such that:

$$\mathbf{y} \models u_k \iff \mathbf{y} \models u \text{ and } \sum_{\substack{1 \leqslant i \leqslant n, \\ Y_i \in \mathtt{vars}(u)}} y_i = k$$

The property is trivial to see for leaf nodes. For a node $u = \bigvee_{c \in \mathtt{ch}(u)} c$, smoothness guarantees that for each child $c \in \mathtt{ch}(u)$ we have $\mathtt{vars}(u) = \mathtt{vars}(c)$, which implies that for any $0 \leqslant k \leqslant |\mathtt{vars}(u)|$:

$$\begin{aligned} u_k = \bigvee_{c \in \mathtt{ch}(u)} c_k &\equiv \bigvee_{c \in \mathtt{ch}(u)} c \wedge \Big(\sum_{\substack{1 \leqslant i \leqslant n, \\ Y_i \in \mathtt{vars}(c)}} y_i = k \Big) \\ &\equiv \bigvee_{c \in \mathtt{ch}(u)} c \bigwedge \Big(\sum_{\substack{1 \leqslant i \leqslant n, \\ Y_i \in \mathtt{vars}(u)}} y_i = k \Big) \\ &\equiv u \bigwedge \Big(\sum_{\substack{1 \leqslant i \leqslant n, \\ Y_i \in \mathtt{vars}(u)}} y_i = k \Big) \end{aligned}$$

Finally for nodes $u = \bigwedge_{c \in \mathtt{ch}(u)}$, decomposability ensures disjoint variable sets between its children, therefore:

$$\begin{aligned} u_k = \bigvee_{\substack{m_1, ..., m_q \\ m_1 + ... + m_q = k}} \bigwedge_{1 \leqslant i \leqslant q} c^i_{m_i} &\equiv \bigvee_{\substack{m_1, ..., m_q \\ m_1 + ... + m_q = k}} \bigwedge_{1 \leqslant i \leqslant q} c^i \wedge \Big(\sum_{\substack{1 \leqslant i \leqslant n, \\ Y_j \in \mathtt{vars}(c^j)}} y_j = m_i \Big) \\ &\equiv \bigvee_{\substack{m_1, ..., m_q \\ m_1 + ... + m_q = k}} u \wedge \bigwedge_{1 \leqslant i \leqslant q} \Big(\sum_{\substack{1 \leqslant i \leqslant n, \\ Y_j \in \mathtt{vars}(c^j)}} y_j = m_i \Big) \\ &\equiv u \wedge \bigvee_{\substack{m_1, ..., m_q \\ m_1 + ... + m_q = k}} \bigwedge_{1 \leqslant i \leqslant q} \Big(\sum_{\substack{1 \leqslant i \leqslant n, \\ Y_j \in \mathtt{vars}(c^j)}} y_j = m_i \Big) \\ &\equiv u \wedge \Big(\sum_{\substack{1 \leqslant i \leqslant n, \\ Y_i \in \mathtt{vars}(u)}} y_i = k \Big) \end{aligned}$$

The circuits $\kappa_0, ..., \kappa_p$ simply correspond to $r_0, ..., r_p$ where r is the root of κ.

Besides, notice that new leaf nodes do not add new edges. New $\vee$-nodes add at most a factor p to their number of edges: original edges (u, c) are turned into edges (u_k, c_k) for each $0 \leqslant k \leqslant p$. Finally new $\wedge$-nodes add at most a factor $3p^2$ to their number of edges: original edges (u, c) are turned into edges $(u_k, \vee_m)$, $(\vee_m, c^1_m)$ and $(\vee_m, c^2_{k-m})$ for each $0 \leqslant k \leqslant p$ and $0 \leqslant m \leqslant k$.

Therefore, the worst combined size of the circuits is in $\mathcal{O}(p^2 \cdot |\kappa|)$. □

Now we can prove Proposition 10.

Proof. We first build circuits $F^{1-\mu}_0, ..., F^{1-\mu}_n$ in sDNNF based on Lemma 11 then we build $F^{1-\mu}_{\geqslant 0}$ following Equation 20.

We can see by recurrence on k that for all $0 \leqslant k \leqslant n$:

$$\mathtt{var}(F^{1-\mu}_k) = \mathbf{Y} \cup \mathbf{Z} \quad \text{and} \quad \mathtt{var}(F^{1-\mu}_{\geqslant k}) = \mathbf{Y} \cup \mathbf{Z} \cup \{S_k, ..., S_n\}$$

This directly implies that for all $0 \leqslant k \leqslant n$, $\wedge$-nodes in $S_k \wedge F_k^{1-\mu} \wedge_{k+1 \leqslant m \leqslant n} \neg S_m$ and $\neg S_k \wedge F_{\geqslant k+1}^{1-\mu}$ are decomposable.

Besides, for all $0 \leqslant k \leqslant n$:

$$\texttt{var}(S_k \wedge F_k^{1-\mu} \wedge_{k+1 \leqslant m \leqslant n} \neg S_m) = \texttt{var}(\neg S_k \wedge F_{\geqslant k+1}^{1-\mu}) = \mathbf{Y} \cup \mathbf{Z} \cup \{S_k, ..., S_n\}$$

This directly implies that for all $0 \leqslant k \leqslant n$, $\vee$-nodes in $F_{\geqslant k}^{1-\mu}$ are smooth.

Therefore in particular, $F_{\geqslant 0}^{1-\mu}$ is in sDNNF.

Finally, writing $F_{\geqslant 0}^{1-\mu}$ only requires at most $\mathcal{O}(n^2)$ additional edges on top of $F_0^{1-\mu}, ..., F_n^{1-\mu}$ which already has a combined size in $\mathcal{O}(n^2 \cdot |F^{1-\mu}|)$.

Therefore, the size of $F_{\geqslant 0}^{1-\mu}$ is also in $\mathcal{O}(n^2 \cdot |F^{1-\mu}|)$. □

C DATASETS

In this section we provide more details about the datasets used in our experiments.

C.1 WARCRAFT SHORTEST PATH DATASET

The Warcraft Shortest Path task [Pogančić et al., 2019] is a staple of neurosymbolic AI [Yang et al., Niepert et al., Ahmed et al., 2022]. Inputs consist of randomly generated images of terrain maps from the Warcraft II tileset, and the objective is to predict the shortest path in the map from the image alone. Maps are built on a 12×12 directed grid (each vertex is connected to all its *neighbors*), and to each vertex of the grid corresponds a tile of the tileset. Each tile is a RGB image that depicts a specific terrain, with a fixed traveling cost. For each map, a labelset encodes its shortest s-t path, *i.e.*, a path from the upper-left to the lower-right corners. The weight of the path is the sum of the traveling costs of all terrains (*i.e.*, grid vertices) on the path. The terrain costs are used to produce the dataset but are not provided during training or inference. In the original dataset [Pogančić et al., 2019], output variables correspond to vertices in the grid, and a labelset satisfies the simple path constraint if the vertices set to 1 can be connected into a simple s-t path. A sample from the dataset is shown in Figure 6.

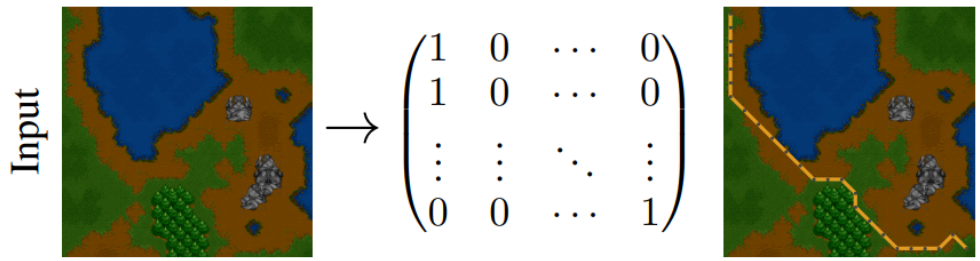


Figure 6: A sample from the Warcraft Shortest Path dataset, taken from [Ahmed et al., 2022]. The input sample is the map shown on the left picture. The labelset shown on the central array encodes a simple path displayed on the right picture.

This representation comes with several issues. First, as noted in [Ahmed et al., 2022], the set of vertices ambiguously encodes more than one path (because of cycles in the grid, there are several possible simple paths that go through the same vertices). Besides, weighted optimization and counting problems on simple path constraints for general directed graphs are respectively **NP**-hard and #**P**-hard [Ledaguenel et al., 2025]. To deal with these issues, [Ahmed et al., 2022] transforms the output space in the following way: edges of the grid are chosen as output variables instead of vertices, and only simple paths with a maximal length of 29 (the maximal length found in the training set) are kept. This implies that the test set might not be consistent with the constraints since it might contain a path longer than 29 edges. Besides, such a method would not scale to larger grids since the length of paths would grow exponentially.

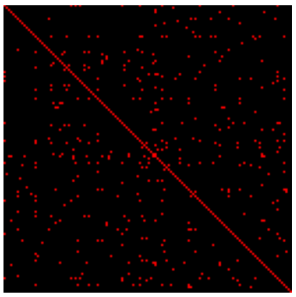
In our experiments, we adopt a different approach. We keep the set of edges as our output variables and additionally turn the grid into an acyclic graph by only connecting vertices to their right and lower neighbors. It is shown in Ledaguenel et al. [2025] that acyclicity is a sufficient condition to compile simple path constraints in sd-DNNF circuits (even OBDD in fact). As we changed the set of edges, we recomputed labelsets for the new output space using the terrain costs.

C.2 ARXIV_CATEGORIES

The `arxiv_categories` dataset [Schopf et al., 2024] is a text multi-label classification dataset that labels titles and abstracts of arxiv articles with 130 categories taken from the arXiv category taxonomy[1]. Statistics about the dataset provided in [Schopf et al., 2024] and Najjar et al. [2026] are reproduced in Table 7a. The interaction matrix is shown on Figure 7b.

#Classes	Split sizes	Interaction density
130	163k / 20k / 20k	96%

(a) Statistics about the `arxiv_categories` dataset: number of classes, split sizes for train, validation and test sets, and density of the interaction matrix μ

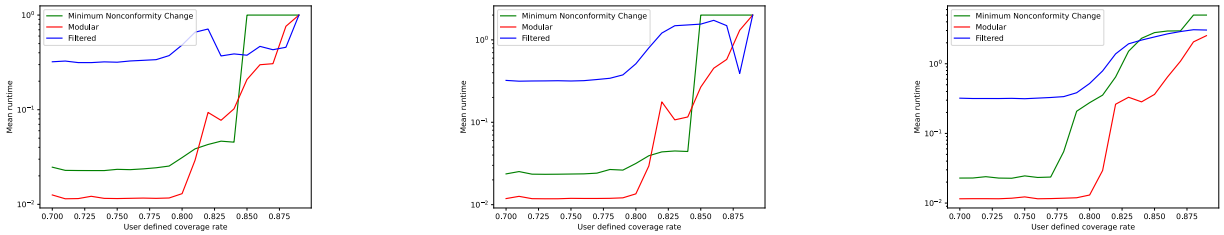


(b) Interaction matrix for `arxiv_categories`: red dots (i, j) correspond to pairs of labels (Y_i, Y_j) which have been observed jointly in the data (*i.e.*, $\mu_{ij} = 0$)).

Figure 7: Information about the `arxiv_categories` dataset

D RUNTIMES

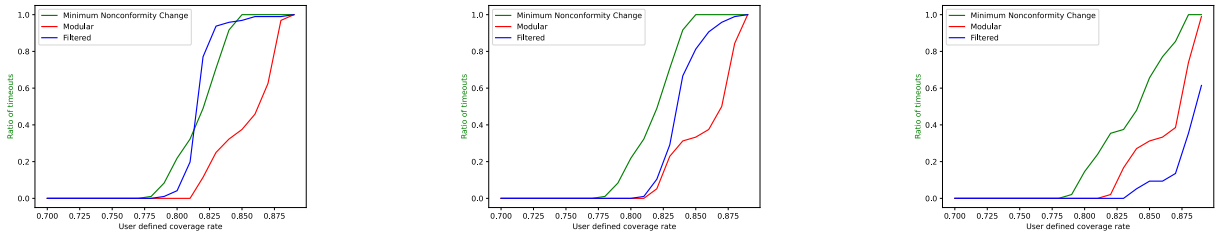
Figure 8 shows the average runtimes of finished runs of the different methods on the Warcraft Shortest Path dataset. This shows that on instances that are computable below the timewall, filtered enumeration is the slowest method of the three. Combined with the rates of timeouts shown on Figure 9, this strengthens our intuition that filtered enumeration has an edge in scenarios where the search space of other methods explodes exponentially.



(a) 1 second (b) 2 seconds (c) 5 seconds

Figure 8: Runtimes of finished runs for various timewalls depending on the user-defined coverage rate for each enumeration method

[1]The taxonomy can be found at: https://arxiv.org/category_taxonomy



(a) 1 second (b) 2 seconds (c) 5 seconds

Figure 9: Rates of timeouts for various timewalls depending on the user-defined coverage rate for each enumeration method