Upper entropy for 2-monotone lower probabilities

Tuan-Anh Vu
UMR CNRS 7253 Heudiasyc
Université de Technologie de Compiègne
France
tuan-anh.vu@hds.utc.fr

Sébastien Destercke
UMR CNRS 7253 Heudiasyc
Université de Technologie de Compiègne
France
sebastien.destercke@hds.utc.fr

Frédéric Pichon
Laboratoire de Génie Informatique et d'Automatique de l'Artois (LGI2A)
Université d'Artois, UR 3926, France
frederic.pichon@univ-artois.fr

Abstract

Uncertainty quantification is a key aspect in many tasks such as model selection/regularization, or quantifying prediction uncertainties to perform active learning or OOD detection. Within credal approaches that consider modeling uncertainty as probability sets, upper entropy plays a central role as an uncertainty measure. This paper is devoted to the computational aspect of upper entropies, providing an exhaustive algorithmic and complexity analysis of the problem. In particular, we show that the problem has a strongly polynomial solution, and propose many significant improvements over past algorithms proposed for 2-monotone lower probabilities and their specific cases.

1 Introduction

Imprecise probabilistic or *credal* [Levi, 1980] approaches offer rich representations of uncertainty, usually coming in the form of convex sets of probabilities or of equivalent representations such as lower envelope of expectations [Troffaes and De Cooman, 2014], that we will call here *credal sets*. They have been used in many facets of AI, such as symbolic reasoning [Hansen et al., 2000], graphical models [Cozman, 2000] or machine learning [Caprio et al., 2024], with this latter trend gaining traction, as show many recent papers on the topic [Singh et al., 2024, Wang et al., 2024, Nguyen et al., 2025, Löhr et al., 2025]. Uncertainty quantification, by which we mean the fact of quantifying some aspect of an uncertainty representation by a value, is another important aspect of reasoning under uncertainty, and is a cornerstone of, e.g., Klir's theory [Klir, 2006]. Such uncertainty quantifications can serve, e.g., to select a peculiar model among many possible ones [Dubey et al., 2018], as a regularization tool when used as a penalty term [Grandvalet and Bengio, 2006], or as a way to quantify a predictive uncertainty.

Within credal approaches, upper entropy plays a particular role, as it can be shown to satisfy quite a number of axioms [Klir, 2006, Sec. 6.7.] when considered as measure of total uncertainty. It can also be associated to a robust, minimax version of using the logarithmic scoring rule: if your uncertainty is represented by a credal set $\mathcal{P}$, and if $\overline{\mathbb{E}}_{\mathcal{P}}[-\ln(Q)]$ is the maximum loss of choosing Q under knowledge $\mathcal{P}$ with a logarithmic scoring rule, then the distribution Q minimizing $\overline{\mathbb{E}}_{\mathcal{P}}[-\ln(Q)]$ is the one of maximal entropy [Walley, 1991, Sec. 5.12.]. It is also commonly used in various learning settings, such as the learning of decision trees [Abellán and Moral, 2005] or as a means to quantify total uncertainties of predictions [Hüllermeier and Waegeman, 2021, Javanmardi et al., 2024].

Given this importance of upper entropy, it is surprising that the computational aspects of calculating it for large families of credal sets has not been much explored. Abellán and Moral [2006] provide an algorithm for credal sets induced by 2-monotone capacities, yet only conclude that it is a "difficult problem". This algorithm is derived from the MRW algorithm [Meyerowitz et al., 1994, Harmanec et al., 1996] that is dedicated to belief functions and is claimed to be of exponential complexity in the size of the considered space by Huynh and Nakamori [2009]. Recent improvements of the MRW do not discuss complexity either [Abellán et al., 2023]. A more efficient algorithm, that is quadratic in the size of the space, exists for the specific case of probability intervals [Abellan and Moral, 2003].

The goal of this paper is to revisit the problem of computing upper entropies for credal sets $\mathcal{P}$ induced by 2-monotone (a.k.a. supermodular) capacities, that include many special cases such as belief functions, probability intervals, ϵ-contamination models or possibility distributions. Our main contributions to this problem are the following:

- We demonstrate that by using supermodular optimization, the exact algorithm proposed by Abellán and Moral [2006] can be made polynomial by making a number of calls to supermodular function maximization (SFM) that is quadratic in the space size. This insight also allows us to propose a new algorithm whose number of calls to SFM is linear in the space size. These results both show that the problem is not as hard as previously claimed, and improve significantly upon existing methods. This is elaborated in Section 3.

- We provide improved algorithms for the specific cases of belief functions, possibility distributions and probability intervals, that are commonly considered in applications (see, e.g., [Caprio et al., 2025, Chau et al., 2025]). These are detailed in Section 4.

- We propose in Section 5 an approximation algorithm using the Frank–Wolfe algorithm [Bach, 2013] that frames the true results between a lower and a upper bounds, thereby allowing us to control the approximation error.

Finally, we propose in Section 6 some first experiments that compare our algorithms but also prove that our improvements make the associated computations scalable. Before that, let us start with some necessary preliminaries.

2 Preliminaries

2.1 2-monotone lower probability

Throughout this paper, we consider a finite space $\Omega = \{1, 2, \ldots, n\}$. A set function $\nu : 2^\Omega \to \mathbb{R}$ with $\nu(\emptyset) = 0$ is called *submodular* if

$$\nu(A) + \nu(B) \geq \nu(A \cup B) + \nu(A \cap B) \,\forall A, B \subseteq \Omega. \tag{1}$$

Similarly, ν is called *supermodular* if $-\nu$ is submodular, i.e., the inequality in (1) is reversed. As listing 2^n values is intractable, we assume that ν is accessed through an oracle that returns $\nu(A)$ for any query $A \subseteq \Omega$ [Grötschel et al., 1993, Chapter 10] and let EO be the time required for one query. The *base polyhedron* $B(\nu) \subseteq \mathbb{R}^n$ of ν is defined as

$$B(\nu) = \{x : \sum_{i \in A} x_i \leq \nu(A) \,\forall A \subseteq \Omega, \sum_{i=1}^n x_i = \nu(\Omega)\}. \tag{2}$$

Submodular (supermodular) functions appear in many domains, including the theory of imprecise probabilities [Walley, 1991]. This uncertainty theory generalizes probability theory, that uses *additive measures*, notably by considering probability sets that can be induced by *lower probabilities*, that are *non-additive measures*. A set function $\mu : 2^\Omega \to [0, 1]$ is called a lower probability if $\mu(\Omega) = 1$, $\mu(\emptyset) = 0$ and $\mu(A) \leq \mu(B)$ if $A \subseteq B$. The term "lower probability" comes from the interpretation that the true probability on Ω is unknown and μ encodes the available (limited) information in the form of lower bounds on event probabilities. Hence, the so-called *credal set* consisting of probabilities on Ω that are compatible with μ is $\mathcal{P}(\mu) := \{p \in \Delta_n : p(A) \geq \mu(A) \,\forall A \subseteq \Omega\}$.

Many important lower probabilities are supermodular, and have a specific name in imprecise probabilities.

Definition 1. *A lower probability μ is called* 2-monotone *if it is supermodular.*

Remark 1. *For a lower probability μ, its dual, called an* upper probability *$\bar{\mu}$, is given by $\bar{\mu}(A) = 1 - \mu(\Omega \setminus A) \,\forall A$. It can be seen that $\mathcal{P}(\mu) = \{p : p(A) \leq \bar{\mu}(A) \,\forall A \subseteq \Omega\}$. Moreover, 2-monotonicity of μ implies that $\bar{\mu}$ is submodular, so $\mathcal{P}(\mu)$ is exactly the base polyhedron $B(\bar{\mu})$ of $\bar{\mu}$ (see (2)).*

Below, we highlight some notable special cases of 2-monotone lower probabilities.

Belief Function A mass function on Ω is a mapping $m : 2^\Omega \to [0, 1]$ satisfying $\sum_{A \subseteq \Omega} m(A) = 1$ and $m(\emptyset) = 0$ [Shafer, 1976]. A subset A is called a *focal set* of m if $m(A) > 0$. m induces a special lower probability called *belief function* defined as $\mathrm{Bel}(B) := \sum_{A \subseteq B} m(A)$. The dual of Bel, the plausibility function, is defined as $\mathrm{Pl}(B) := \sum_{A \cap B \neq \emptyset} m(A)$. In this case, EO is polynomial in $|\{A : m(A) > 0\}|$, so is tractable if we do consider a mass function positive on a limited amount of subsets: this is for instance the case with k-additive belief functions [Grabisch, 1997] or imprecise cumulative distributions [Montes and Destercke, 2017].

Possibility Distribution   A possibility distribution π on Ω is a vector in $[0, 1]^n$ where each component π_i represents the degree of possibility of element i. π induces a lower probability on Ω, called the necessity measure, defined as $\mathrm{N}(A) := 1 - \max_{i \in \Omega \setminus A} \pi_i \ \forall A$. The dual of N, the possibility measure, is $\Pi(A) := \max_{i \in A} \pi_i \ \forall A$. Again, this means EO is polynomial.

Probability Intervals   In this case, the credal set is given directly as $\mathcal{P} = \{p \in \Delta_n : l_i \leq p_i \leq u_i \ \forall i\}$, where each intervals $[l_i, u_i] \subseteq [0, 1] \ \forall i \in \{1, \ldots, n\}$. Moreover, we require that $\sum_{i=1}^{n} l_i \leq 1 \leq \sum_{i=1}^{n} u_i$ so that $\mathcal{P} \neq \emptyset$. De Campos et al. [1994] showed that the set function μ where $\mu(A) := \min_{p \in \mathcal{P}} p(A) \ \forall A$ is a 2-monotone lower probability. Note that De Campos et al. [1994] provide formulas such that EO is polynomial in $|\Omega|$.

2.2   Upper Entropy

The upper entropy $\mathrm{UE}(\mu)$ of a lower probability μ is defined as

$$\mathrm{UE}(\mu) := \max_{p \in \mathcal{P}(\mu)} -\sum_{i=1}^{n} p_i \log p_i. \tag{3}$$

Algorithm 1 presents the Abellán and Moral [2006] procedure for computing $\mathrm{UE}(\mu)$ when μ is 2-monotone. Setting $\mu = \mathrm{Bel}$ yields the MRW algorithm [Meyerowitz et al., 1994, Harmanec et al., 1996] for belief functions. However and as said in the introduction, none of these papers proceed through a detailed complexity analysis of these algorithms, merely concluding that the problem is "difficult", mostly due to Line 1 in Algorithm 1.

Algorithm 1: Abellán-Moral's algorithm

Input: 2-monotone μ on $\Omega = \{1, \ldots, n\}$.

1 Find A such that $\frac{\mu(A)}{|A|}$ is maximum. If A is not unique, the largest such set is selected

2 For each $i \in A$, $p_i = \frac{\mu(A)}{|A|}$

3 For each $B \subseteq \Omega \setminus A$, $\mu(B) = \mu(B \cup A) - \mu(A)$

4 $\Omega = \Omega \setminus A$

5 **if** $\Omega \neq \emptyset$ *and* $\mu(\Omega) > 0$ **then** go to 1

6 **if** $\mu(\Omega) = 0$ *and* $\Omega \neq \emptyset$ **then** $p_i = 0$ for all $i \in \Omega$

7 **return** $-\sum_{i=1}^{n} p_i \log p_i$

---

3   Strong Polynomiality of computing Upper Entropy

We call an algorithm whose input is a supermodular (submodular) function *strongly polynomial* if it performs a number of oracle queries and arithmetic operations [1] bounded by $\mathrm{poly}(n)$. A notable example is supermodular function maximization (SFM) (equivalently, submodular function minimization), which seeks a subset maximizing (resp. minimizing) a supermodular (resp. submodular) set function; several algorithms are known, e.g., Orlin's algorithm Orlin [2009] runs in time $O(n^5 \mathrm{EO} + n^6)$. Upon closer inspection, Algorithm 1 can be implemented so that it is strongly polynomial. The key observation is that although the optimization problem in Line 1 is not an SFM, it can be transformed into a series of SFM.

Lemma 1. *Finding $A \in \arg\max_{\emptyset \neq B \subseteq \Omega} \frac{\mu(B)}{|B|}$ and $|A|$ is maximum amounts to solving $O(n)$ SFM.*

The key of the proof, given in Appendix A, is that the function $\mu(A) - \lambda|A|$ is supermodular if μ is supermodular, and that finding the argmax within Lemma 1 is equivalent to successively finding the set A maximizing $\mu(A) - \lambda|A|$ (a SFM procedure, known to be polynomial), for $O(n)$ suitable values of λ.

Proposition 1. *Algorithm 1 boils down to solving $O(n^2)$ SFM, and thus is strongly polynomial.*

Proof. Line 1 of Algorithm 1 requires solving $O(n)$ SFM (Lemma 1). Moreover, the update in Lines 3–4 still results in a supermodular function on $\Omega \setminus A$, e.g., [Abellán and Moral, 2006, Property 1], whose evaluation oracle is given by Line 3. As Ω shrinks by at least one element per iteration (Line 4), Algorithm 1 terminates after at most n passes through the loop in Lines 1–5. Hence, it requires solving $O(n^2)$ SFM. □

---

[1]Arithmetic operations include addition, subtraction, multiplication, division, comparison operations, etc.

As it turns out, we can compute $\text{UE}(\mu)$ with improved complexity by using a decomposition algorithm [Nagano and Kawahara, 2013, Fujishige, 2005]. Let ν be a submodular function and b be a positive vector in $\mathbb{R}^n$, Nagano and Kawahara [2013, Sec. 5] shows that solving

$$\min_{x \in B(\nu)} \sum_{i=1}^n (x_i \log \frac{x_i}{b_i} + b_i - x_i) \tag{4}$$

reduces to finding a chain $\emptyset = S_0 \subset \cdots \subset S_l = \Omega$ and breakpoints $0 = \alpha_0 < \alpha_1 < \cdots < \alpha_l < \alpha_{l+1} = +\infty$ such that, for any $\alpha \in [\alpha_j, \alpha_{j+1})$, S_j is the unique minimizer of maximum cardinality of $\min_{A \subseteq \Omega} \nu(A) - \alpha \sum_{i \in A} b_i$. The minimizer of (4) is, for each $i \in S_j \setminus S_{j-1}$ $(j \in 1, \ldots, l)$

$$x_i = \frac{\nu(S_j) - \nu(S_{j-1})}{\sum_{i \in S_j \setminus S_{j-1}} b_i} b_i. \tag{5}$$

Algorithm 2 aims to compute the chain $\{S_j\}$. Given two sets $S_j \subset S_k$ in the chain (initially $S_j = \emptyset$, $S_k = \Omega$), it recursively checks if S_j and S_k are consecutive, and if not, it finds a new chain set strictly between them and repeats the process until the whole chain is revealed. In total, Nagano and Kawahara [2013] shows that Algorithm 2 requires solving $O(n)$ SFM (Line 3).

Algorithm 2: Decomposition algorithm

Input: submodular ν on $\Omega = \{1, \ldots, n\}$; $b \in \mathbb{R}^n$.

1 **Function** DA(S, S'):

2     $\alpha = \dfrac{\nu(S') - \nu(S)}{\sum_{i \in S' \setminus S} b_i}$

3   Take $S'' \in \arg\min_{A \subseteq \Omega} \nu(A) - \alpha \sum_{i \in A} b_i$ s.t $|S''|$ is largest

4   **if** $S'' = S'$ **then return** $\{S, S'\}$

5   **else return** DA$(S, S'') \cup$ DA(S'', S')

6 **return** $\mathcal{S}^* = $ DA$(\emptyset, \Omega)$

Proposition 2. *Computing* $\text{UE}(\mu)$ *amounts to running Algorithm 2 with* $\bar{\mu}$ *(the dual of* μ*) and* $b = \mathbf{1}$*, which requires* $O(n)$ *SFM.*

Proof. With the provided input, Problem (4) becomes $\min_{x \in B(\nu)} \left(\sum_{i=1}^n x_i \log x_i \right) + n - 1$. It has the same optimal solution as (3) because of Remark 1. Algorithm 2 outputs the chain $\{S_j\}$ and the optimal p of (3) is obtained via (5). $\qquad \square$

Remark 2. *There is a tight connection between Algorithm 1 and Algorithm 2: the sets* $A_1, A_2, \ldots$ *successively computed in Line 1 of the former are exactly the sets* $S_l \setminus S_{l-1}, S_{l-1} \setminus S_{l-2}, \ldots$ *from the chain* $\emptyset = S_0 \subset \cdots \subset S_l = \Omega$ *returned by the latter with the input* $\bar{\mu}$ *and* $\mathbf{1}$*.*

Example 1. *Consider a 2-monotone* μ *and its dual* $\bar{\mu}$ *on* $\Omega = \{1, 2, 3\}$*. Their values are given below (excluding the obvious values of* $\emptyset$ *and* Ω*). Algorithm 1, with input* μ*, outputs the sets* $\{2, 3\}$ *and* $\{1\}$ *in this order at Line 1. Hence, the optimal probability is* $p = (0.2, 0.4, 0.4)$*.*

A	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$
$\mu(A)$	0.1	0.4	0.4	0.5	0.5	0.8
$\bar{\mu}(A)$	0.2	0.5	0.5	0.6	0.6	0.9

Let us illustrate Algorithm 2 with $\bar{\mu}$ *and* $b = \mathbf{1}$*. First, we run* DA$(\emptyset, \Omega)$*, for which the associated* $\alpha = \frac{\bar{\mu}(\Omega) - \bar{\mu}(\emptyset)}{|\Omega|} = \frac{1}{3}$*. Second, we solve* $\min_{A \subseteq \Omega} \bar{\mu}(A) - \frac{1}{3}|A|$ *and obtain* $A = \{1\}$ *the maximal minimizer. Third, we would proceed recursively by executing* DA$(\emptyset, \{1\})$ *and* DA$(\{1\}, \Omega)$*; but since there is no set strictly between* $\emptyset$ *and* $\{1\}$*, we only need to run* DA$(\{1\}, \Omega)$*, for which* $\alpha = \frac{\bar{\mu}(\Omega) - \bar{\mu}(\{1\})}{|\Omega \setminus \{1\}|} = 0.4$*. Finally, solving* $\min_{A \subseteq \Omega} \bar{\mu}(A) - 0.4|A|$*, we find that* Ω *is the maximal minimizer, implying there is no set in the chain strictly between* $\{1\}$ *and* Ω*, so we stop. Hence, Algorithm 2 outputs the chain* $\emptyset \subset \{1\} \subset \Omega$*. Applying (5), we also get* $p = (0.2, 0.4, 0.4)$*.*

4 Computing upper entropy in special cases

4.1 Belief function

Let m be a mass function on Ω with Pl its plausibility function and $\mathcal{F}$ its set of focal sets. We consider a directed bipartite graph with partitions L and R in which each *left vertex* in L (resp. *right vertex* in R) corresponds to an element of Ω (resp. focal set of m), and an edge from j to F_i iff $j \in F_i$. From this graph, we construct a flow network G by adding a source s and a sink t, together with edges (s, j) for all elements $j \in \Omega$ and (F_i, t) for all focal sets F_i of m. Define a capacity function c on edges as $c(s, j) = \alpha$, $c(j, F_i) = +\infty$, and $c(F_i, t) = m(F_i)$. A cut (S, T) of G is simply a partition of its vertices into a *source set* S (where $s \in S$) and a *sink set* $T = V \setminus S$ (where $t \in T$) [Cormen et al., 2009, Chapter 26]. The capacity of (S, T) is the sum of capacities of edges leaving S, i.e., $c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v)$. As the cut (S, T) where $T = \{t\}$ has capacity 1, the minimum cut capacity is at most 1.

In this case, Line 3 of Algorithm 2 boils down to minimum-cut/maximum-flow computations and is thus much faster than general SFM.

Proposition 3. *Solving $\min_{A \subseteq \Omega} \mathrm{Pl}(A) - \alpha|A|$ with $\alpha > 0$ is equivalent to finding a minimum cut (S, T) in G.*

Proof. Consider a minimum cut $X = (S, T)$ where $S = s \cup L' \cup R'$, with L' a set of left vertices (elements of Ω) and R' a set of right vertices (focal sets of m). If an element $i \in L'$, then R' contain all focal sets F such that $i \in F$, otherwise there is an edge with capacity $+\infty$ leaving S. Furthermore, if there is a focal set $F \in R'$, then $L' \cap F \neq \emptyset$, otherwise the cut formed by removing F from R' has capacity of $c(X) - m(F) < c(X)$. Therefore, X is of the form with $S = \{s\} \cup A \cup \{F \in \mathcal{F} : F \cap A \neq \emptyset\}$, for a subset $A \subseteq \Omega$. By the construction of G,

$$c(X) = \alpha|\Omega \setminus A| + \sum_{F \cap A \neq \emptyset} m(F)$$

$$= \sum_{F \cap A \neq \emptyset} m(F) - \alpha|A| + n\alpha = \mathrm{Pl}(A) - \alpha|A| + n\alpha,$$

so $\min_{A \subseteq \Omega} \mathrm{Pl}(A) - \alpha|A| = c^* - n\alpha$ with c^* is the minimum cut capacity. The proposition holds as $n\alpha$ is constant. $\square$

Example 2. *Consider a mass m on $\Omega := \{1, 2, 3, 4\}$ whose focal sets are $F_1 = \{1, 2\}$, $F_2 = \{2, 3\}$ and $F_3 = \{3, 4\}$. The flow network from Proposition 3 is shown in Figure 1, which we use to illustrate its proof. Namely, the cut with $S = \{s\} \cup \{1\} \cup \{F_1\}$ is not minimum because edge $(1, F_2)$ has capacity $+\infty$. The cut with $S = \{s\} \cup \{1, 2\} \cup \{F_1, F_2, F_3\}$ is not minimum because removing F_3 from S results in a cut with strictly smaller capacity.*

We still need to select $A^* \in \arg\min_{A \subseteq \Omega} \mathrm{Pl}(A) - \alpha|A|$ with maximum cardinality. It is well-known that for a minimum cut (S^*, T^*) such that $|S^*|$ is maximum, S^* is the union of all minimum cut source sets [Picard and Queyranne, 1980]. It follows that A^* consists of exactly the left vertices in S^*, found as follows. By the max-flow min-cut theorem [Cormen et al., 2009, Theorem 26.6], we first compute a maximum flow of G and construct the associated residual graph. Then A^* is the set of left vertices that cannot reach t in this residual graph [Picard and Queyranne, 1980]. Hence, we obtain:

Proposition 4. *For belief functions, computing upper entropy via Algorithm 2 requires $O(n)$ maximum-flow computations and $O(n)$ depth-first searches.*

As the number of edges of G and its residual graphs are bounded by $O(n|\mathcal{F}|)$, the running time of Algorithm 2 is $\mathrm{poly}(n, |\mathcal{F}|)$ with the exact complexity depending on the chosen maximum-flow algorithm.
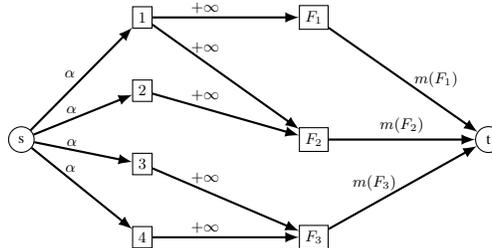

Figure 1: Flow network in Example 2.

4.2 Possibility distribution

Klir [2006, Algorithm 6.2] has particularized Algorithm 1 for the case of a possibility distribution π under the standard convention that $1 = \pi_1 \geq \pi_2 \geq \ldots \geq \pi_n > 0$. Unsurprisingly, this particularization inherits the quadratic complexity (see Proposition 1), yielding an $O(n^2)$ running time. We will, however, design an algorithm achieving $O(n)$ complexity.

Recall in Section 3 that our goal is to compute the chain $\emptyset = S_0 \subset \cdots \subset S_l = \Omega$ where S_j is the unique maximal minimizer of $\min_{A \subseteq \Omega} \Pi(A) - \alpha|A|$ $\forall \alpha \in [\alpha_j, \alpha_{j+1})$. To this end, we employ a procedure different from Algorithm 2 that specifically exploits the structure of π. Due to the ordering convention, $\Pi(A) = \pi_j$ where j is the minimum element of A, and thus $\{j, \ldots, n\}$ is the largest set having the same possibility as A. Therefore, $\forall \alpha \geq 0$,

$$\min_{A \subseteq \Omega} \Pi(A) - \alpha|A| = \min_{j \in \{1, \ldots, n, n+1\}} \pi_j - \alpha(n - j + 1), \tag{6}$$

with $\pi_{n+1} := 0$ to handle the case when $\emptyset$ minimizes the left-hand side at $\alpha = 0$. Let $g_j(\alpha) := \pi_j - \alpha(n - j + 1)$ and $g := \min_{j \in \{1, \ldots, n+1\}} g_j$. To find the desired chain, we compute all breakpoints in $[0, +\infty)$ of g and select the smallest j such that $g_j = g$ at each breakpoint. This ensures that the associated minimizer of the left-hand side of (6) has maximum size, which is $\{j, \ldots, n\}$.

Our task can be done via a classic problem in computational geometry. More precisely, in this field, g is called the *lower envelope* of lines g_j. Under the duality that maps a line $\ell : y = m\alpha + b$ to its dual point $\ell^* := (m, -b)$, g corresponds one-to-one with the *upper convex hull* [2] of the set of dual points $\{g_j^* : 1 \leq j \leq n + 1\}$ where $g_j^* = (j - n - 1, -\pi_j)$ [De Berg et al., 2008, Chapter 11.4].

We employ this correspondence to our setting to solve $\min_{A \subseteq \Omega} \Pi(A) - \alpha|A|$ $\forall \alpha$, described in Algorithm 3. Because of this correspondence, we simply traverse the upper hull's vertices from right to left, recording the breakpoints as the slopes of the visited edges (Line 5). Furthermore, the smallest minimizing indices associated with the breakpoints are computed in Line 6, from which the maximal minimizers are obtained immediately (Line 8). Finally, the complexity is dominated by the upper convex hull computation (Line 2), which is $O(n \log n)$ in general but is only $O(n)$ here since the dual points are already sorted by x-coordinate (the slopes of g_j are strictly monotone)[De Berg et al., 2008, Theorem 1.1]. Hence, we obtain the following.

Proposition 5. *Algorithm 3 runs in $O(n)$ time.*

Algorithm 3: Solve $\min_{A \subseteq \Omega} \Pi(A) - \alpha|A|$ $\forall \alpha$

Input: possibility distribution π s.t $1 = \pi_1 \geq \pi_2 \geq \ldots \geq \pi_n > \pi_{n+1} = 0$.

1 Form points $p^j = (j - n - 1, -\pi_j)$, each has attribute $p^j.\mathrm{idx} = j$.

2 Compute upper hull vertices (from right to left) of the set $\{p^j : 1 \leq j \leq n + 1\}$ and store in array V

3 N= length(V); B = [ ]; I = [ ]

4 for $i = 1$ **to** $N - 1$ **do**

5   Add slope of edge joining $V[i]$ and $V[i + 1]$ to B

6   Add $V[i + 1].\mathrm{idx}$ to I

7 for $j \in$ I **do**

8 $S_j = \{j, \ldots, n\}$

9 return B *and* $\{S_j\}$

Example 3. *Consider the distribution $\pi = (1, 0.6, 0.3)$. We have $g_1(\alpha) = 1 - 3\alpha$, $g_2(\alpha) = 0.6 - 2\alpha$, $g_3(\alpha) = 0.3 - \alpha$, and $g_4(\alpha) = 0$. The graph of $g = \min_i g_i$ is illustrated in Figure 2a in which the intersections of active lines are marked in black. At the first breakpoint $\alpha = 0.3$, the smallest j satisfying $g_j = g$ is $j = 2$. At the second (and last) breakpoint $\alpha = 0.4$, the smallest j satisfying $g_j = g$ is $j = 1$. From these breakpoints, we read the chain $\{S_j\}$ associated to $\min_{A \subseteq \Omega} \Pi(A) - \alpha|A|$ as: $\forall \alpha \in [0.3, 0.4)$, the maximal minimizer is $\{2, 3\}$; $\forall \alpha \in [0.4, +\infty)$, the maximal minimizer is $\{1, 2, 3\}$. Of course, $\forall \alpha \in [0, 0.3)$, the maximal minimizer is $\emptyset$.*

Let g_i^* be the dual points of lines g_i. Figure 2b illustrates the convex hull of these points in which the upper convex hull is marked in red. As g_3^* is not a vertex, the edges in this upper hull from right to left are (g_4^*, g_2^*) and (g_2^*, g_1^*), which correspond one to one with the intersections of active lines of g (see Figure 2a). We also see that the slopes of edges (g_4^*, g_2^*) and (g_2^*, g_1^*) are 0.3 and 0.4 matching exactly the breakpoints of g calculated before.

---

[2] The upper convex hull is the chain of edges on the convex hull's upper boundary.

(a) Lower envelope of lines g_j

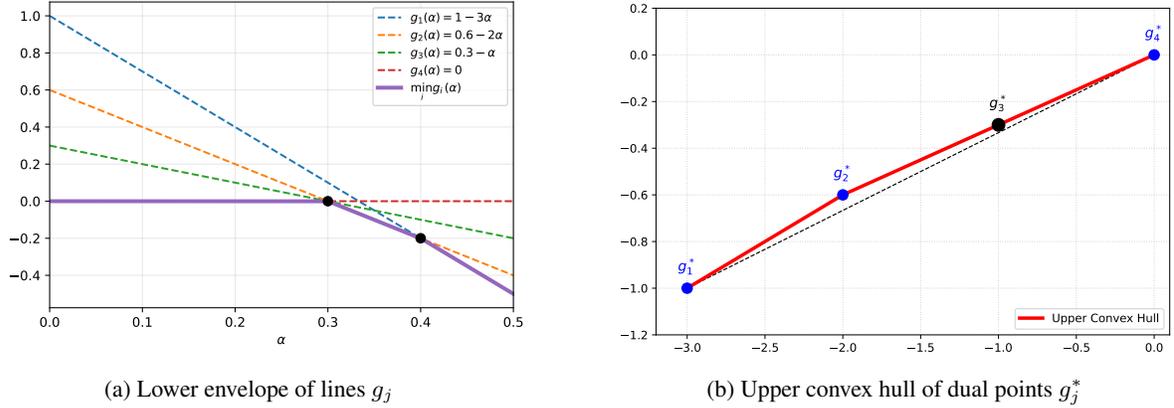(b) Upper convex hull of dual points g_j^*

Figure 2: Illustration of Example 3

4.3 Probability intervals

In this case, computing the upper entropy becomes solving

$$\max_{p \in \mathcal{P}} - \sum_{i=1}^{n} p_i \log p_i, \tag{7}$$

where $\mathcal{P} = \{p \in \Delta_n : l_i \leq p_i \leq u_i \ \forall i\}$.

Abellan and Moral [2003] designed an algorithm for solving (7) with $O(n^2)$ complexity. We now present an algorithm with better complexity. By using the KKT conditions [Boyd and Vandenberghe, 2004], the optimal solution is characterized as follows.

Proposition 6. *p is optimal of Problem 7 if and only if there exists x such that $\sum_{i=1}^{n} \min\{\max\{x, l_i\}, u_i\} = 1$ and $p_i = \min\{\max\{x, l_i\}, u_i\} \ \forall i$.*

Let $f_i(x) := \min\{\max\{x, l_i\}, u_i\}$ or explicitly

$$f_i(x) = \begin{cases} l_i, & x \leq l_i, \\ x, & l_i < x < u_i \\ u_i, & x \geq u_i. \end{cases}$$

From Proposition 6, whose technical proof is in the Appendix B, it suffices to solve $f(x) = 1$ where $f(x) := \sum_{i=1}^{n} f_i(x)$. Because each f_i is non-decreasing and piecewise affine, the same holds for f. Let $a := \min_i l_i$ and $b := \max_i u_i$, then $f(a) = \sum_{i=1}^{n} l_i$ and $f(b) = \sum_{i=1}^{n} u_i$. As f is continuous and $\sum_{i=1}^{n} l_i \leq 1 \leq \sum_{i=1}^{n} u_i$, from the intermediate value theorem there is a solution to $f(x) = 1$ in the interval $[a, b]$. Figure 3 illustrates a plot of f.
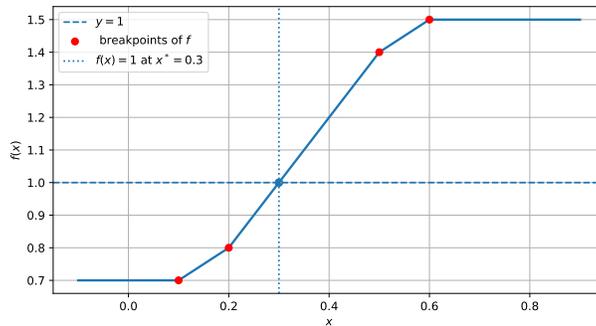

Figure 3: Plot of $f(x)$ with $[l_1, u_1] = [0.1, 0.4]$, $[l_2, u_2] = [0.4, 0.5]$ and $[l_3, u_3] = [0.2, 0.6]$.

Because f is non-decreasing, we can solve $f(x) = 1$ by binary search: at each iteration, set $x = \frac{a+b}{2}$; if $f(x) < 1$, continue on $[x, b]$, otherwise continue on $[a, x]$, and repeat. This procedure is already quite efficient. In fact, suppose

---

Algorithm 4: Hybrid Newton–Binary Search

Input: n-intervals $[l_i, u_i] \subseteq [0,1]$; tolerance ϵ

1   $a = \min_i l_i,\ b = \max_i u_i,\ x = \frac{a+b}{2}$
2   **while** $|f(x) - 1| > \epsilon$ **do**
3   **if** $f(x) - 1 < 0$ **then**
4         $a = x$ and $d = |\{i : l_i \leq x < u_i\}|$
5   **else**
6         $b = x$ and $d = |\{i : l_i < x \leq u_i\}|$
7   **if** $d = 0$ **then**
8         $x = \frac{a+b}{2}$
9   **else**
10        $x' = x - \frac{f(x)-1}{d}$
11   **if** $x' \in (a,b)$ **then** $x = x'$ **else** $x = \frac{a+b}{2}$
12   **return** $p_i = \min\{\max\{x, l_i\}, u_i\}\ \forall i$

---

we aim to solve $f(x) = 1$ with accuracy ϵ, i.e., $|f(x) - f(x^*)| \leq \epsilon$ where x^* is a solution. Because $|f(x) - f(x^*)| \leq n|x - x^*|$, we stop the binary search once the interval width is less than $\frac{\epsilon}{n}$, which requires at most $\lceil \log_2 \frac{n}{\epsilon} \rceil$ iterations (as $[a, b] \subseteq [0,1]$). For example, if $n = 10^7$ and $\epsilon = 10^{-10}$, this amounts to about 56 iterations, each needing only an evaluation of f.

We can speed up the computation even further by combining binary search with the celebrated Newton method. Start with an initial guess x_0, until a solution is found, Newton's method iteratively updates x_k for $k = 1, 2, \ldots$ as

$$x_k = x_{k-1} - \frac{f(x_{k-1}) - 1}{f'(x_{k-1})}. \tag{8}$$

Note that f is not differentiable at its breakpoints. However, this is not problematic since at a given x, if $f(x) < 1$ (resp. $f(x) > 1$), the solution lies to the right (resp. the left) of x, and we replace the derivative in (8) with the right derivative $f'_+(x)$ (resp. left derivative $f'_-(x)$). It is easy to see that $\forall i$, $(f'_i)_+(x) = 1$ if $l_i \leq x < u_i$ and $(f'_i)_+(x) = 0$ otherwise. Summing these right derivatives, we obtain $f'_+(x) = |\{i : l_i \leq x < u_i\}|$. Similarly, $f'_-(x) = |\{i : l_i < x \leq u_i\}|$. Finally, the Newton step in (8) may step out of the current interval which contains the solution or be undefined due to a zero derivative. In these cases, we fall back to a binary-search step [Press et al., 1992, Chapter 9]. Wrapping up, the approach is described in Algorithm 4. At worst, it may perform binary search only, and thus we obtain:

Proposition 7. *Algorithm 4 runs in $O(n \log \frac{n}{\epsilon})$ time.*

We note that in practice, Algorithm 4 is often considerably faster due to the Newton steps.

5 Computing upper entropy via convex optimization

The bottleneck of Algorithm 2 lies in solving $O(n)$ SFM since algorithms for solving a general SFM are slow due to their high-order polynomial complexity. Hence, for general 2-monotonicity, it is worthwhile to explore efficient approximate ways to compute upper entropy.

We now adopt the well-known Frank–Wolfe (FW) algorithm [Bach, 2013] to solve Problem (3). Denote the entropy function by $H(p) := -\sum_{i=1}^n p_i \log p_i$. In this method, we start with a $p^0 \in \mathcal{P}(\mu)$. At each iteration $k = 0, 1, 2, \ldots$, we solve the so-called linear oracle

$$\max_{p \in \mathcal{P}(\mu)} \nabla H(p^k)^T (p - p^k), \tag{9}$$

for which the maximum is attained at v^k. Since $\mathcal{P}(\mu)$ is convex, the segment between p^k and v^k lies in $\mathcal{P}(\mu)$, and the next iterate is taken on this segment as the point with the best objective, i.e., $p^{k+1} = (1 - \gamma_k)p^k + \gamma_k v^k$ where

$$\gamma_k \in \arg\max_{\gamma \in [0,1]} H((1 - \gamma)p^k + \gamma v^k). \tag{10}$$

If p^* is the optimal probability then the concavity of H ensures that $H(p^k) \leq H(p^*) \leq H(p^k) + \nabla H(p^k)^T(v^k - p^k)$. Hence, the maximum value of (9) (also called the *duality gap*) serves as a stopping certificate: we terminate if it is

smaller than a threshold. As H is not differentiable at p when some $p_i = 0$, we apply the algorithm by starting with an initial point $p^0 > 0$ and restricting the line search (10) to $\gamma \in [0, 1)$. Thus, the gradient is always defined.

Remark 3. *A necessary and sufficient condition for the existence of such $p^0 > 0$ is $\bar{\mu}(\{i\}) > 0 \; \forall i$. Since $\bar{\mu}(\{i\}) = 0$ means i is impossible (as $\bar{\mu}$ provides upper bounds for event probabilities), this condition makes all outcomes i possible. Moreover, p^0 can be found in $O(n^2 EO)$ (see Appendix C).*

The key step in the algorithm is solving (9). Fortunately, $\mathcal{P}(\mu)$ coincides with the base polyhedron of $\bar{\mu}$ (see Remark 1) and Problem (9) is linear in p. Hence, it is efficiently solvable due to the following result.

Proposition 8 ([Fujishige, 2005]). *Let $w \in \mathbb{R}^n$ and ϕ be a permutation such that $w_{\phi(1)} \geq \ldots \geq w_{\phi(n)}$. Define $S_i := \{\phi(1), \ldots, \phi(i)\} \; \forall i \in \{1, \ldots, n\}$ and $S_0 = \emptyset$. If ν is a submodular function on Ω, then a maximizer of $\max_{x \in B(\nu)} w^T x$ is given as $x_i = \nu(S_i) - \nu(S_{i-1}) \; \forall i$.*

6 Experiments

Our experiments are conducted on a laptop with 16GB RAM and Intel Core i9 where we code in Python with NumPy package for efficient array operations [Harris et al., 2020]. Our code is available in the supplementary materials.

6.1 2-monotonicity

To construct a 2-monotone μ, we do as follows [Bednarski, 1981]: let $f : [0, 1] \to [0, 1]$ be any convex function where $f(0) = 0$ and p^* be any probability on Ω; define $\mu(\Omega) = 1$, $\mu(A) = f(p^*(A)) \; \forall A \neq \Omega$. We have chosen p^* as a vector sampled from a Dirichlet distribution where all concentration parameters are 1. Our choices of f and sizes of Ω are listed in Table 1.

We implement the FW algorithm for this type of 2-monotonicity. We first generate the initial point $p^0 > 0$ in the credal set which takes $O(n^2 EO)$ (see Remark 3). The resolution is then carried out starting from p^0. At any iteration, we obtain an approximate entropy H_a and a gap for which the true entropy $H^* \in [H_a, H_a + \text{gap}]$. Hence, the relative error $\frac{H^* - H_a}{H^*}$ is bounded by $\frac{\text{gap}}{H_a + \text{gap}}$ and we stop once this bound is less than 0.1%. Table 1 reports the average running times (in seconds) for initial-point generation and the subsequent resolution, obtained by repeating each $(f, |\Omega|)$ experiment with five random seeds used to construct μ. From Table 1, we observe that the approach is quite efficient, yet most of the computation time is spent generating the initial point $p^0 > 0$ rather than the resolution. This is because the former has quadratic complexity, making it the bottleneck as $|\Omega|$ grows.

Therefore, when the space is large, we may adopt the workaround of optimizing the smooth surrogate $\tilde{H}(p) := -\sum_{i=1}^{n} p_i \log(p_i + \epsilon)$ for a fixed, sufficiently small $\epsilon > 0$. The gap between the optimal values of H and $\tilde{H}$ is bounded by $n\epsilon$ [Hazan et al., 2019, Lemma A.1]. With this surrogate, we can start from any feasible point in the credal set, which is generated in $O(n)$ time using a single permutation on Ω (see Appendix C). The results in Table 2 show that this approach is suitable for large spaces, e.g., $|\Omega| = 5 \times 10^5$.

In general, this approximation approach is likely to outperform in efficiency the exact Algorithm 2, mostly because solving the SFM problems will require polynomial methods whose exponent will be too high. We would therefore recommend it for very high values of $|\Omega|$, for which Algorithm 2 will struggle.

Table 1: Initial point construction and resolution times (s).

f	$|\Omega|$	Create p^0	Resolution
	5000	0.284 ± 0.006	0.013 ± 0.001
x^2	10000	1.035 ± 0.018	0.024 ± 0.001
	20000	3.961 ± 0.072	0.206 ± 0.027
	5000	0.307 ± 0.003	0.009 ± 0.001
$1 - \sqrt{1 - x}$	10000	1.149 ± 0.016	0.017 ± 0.001
	20000	4.333 ± 0.014	0.147 ± 0.014
	5000	0.397 ± 0.003	0.008 ± 0.001
$\frac{e^{2x} - 1}{e^2 - 1}$	10000	1.481 ± 0.010	0.015 ± 0.001
	20000	5.713 ± 0.019	0.139 ± 0.010

Table 2: Initial point construction and resolution times (s) for $|\Omega| = 5 \times 10^5$ with the surrogate.

f	Create p^0	Resolution
x^2	0.010 ± 0.001	6.327 ± 0.291
$1 - \sqrt{1-x}$	0.012 ± 0.001	4.326 ± 0.088
$\frac{e^{2x}-1}{e^2-1}$	0.019 ± 0.001	3.617 ± 0.074

6.2 Probability intervals

To ensure a nonempty credal set, we construct n random intervals $[l_i, u_i]$ as follows. First, we draw all u_i independently and uniformly from $(0,1]$ and compute $\sum_{i=1}^n u_i$. If this sum is less than 1, we discard the draw and resample. Second, each l_i is drawn uniformly from $[0, u_i)$. Finally, we rescale l_i so that $\sum_{i=1}^n l_i = 1 - \text{margin}$ for a fixed $0 < \text{margin} < 1$. We compare Algorithm 4 against the one in [Abellan and Moral, 2003] and report in Table 3 the average running times (in seconds) over five random seeds for each (margin, $|\Omega|$) pair.

As mentioned earlier, Algorithm 4 has better complexity than Abellán & Moral's algorithm ($O(n \log \frac{n}{\epsilon})$ vs. $O(n^2)$). From Table 3, we also observe a significant improvement in the experiment. Moreover, the results suggest that their algorithm is influenced not only by $|\Omega|$ but also by the size of the feasible set: for a fixed $|\Omega|$, increasing margin enlarges the credal set and their algorithm struggles more. Our algorithm, in contrast, appears to depend only on $|\Omega|$.

Table 3: Runtimes (s) of Algorithm 4 vs. Abellán & Moral

margin	$|\Omega|$	Algorithm 4	Abellán & Moral
0.1	10^7	0.305 ± 0.002	12.906 ± 0.036
	10^8	3.133 ± 0.042	15.699 ± 0.144
0.2	10^7	0.265 ± 0.003	19.375 ± 0.047
	10^8	2.729 ± 0.037	19.589 ± 0.245
0.3	10^7	0.281 ± 0.015	25.054 ± 0.060
	10^8	2.843 ± 0.147	27.069 ± 0.179

7 Conclusion

In this paper, we revisited the problem of computing upper entropy for 2-monotone lower probabilities, from theoretical and practical viewpoints. Upper entropy indeed plays an important role when it comes to credal uncertainty quantification, hence the interest to fully explore its computational aspects.

We analyzed Abellán & Moral's algorithm, well-known in imprecise probability but whose exact complexity was previously unknown. In contrast with previous claims saying that it was difficult, we showed that it was strongly polynomial, and proposed an improved algorithm based on submodular optimization. For important special cases, namely belief functions, probability intervals and possibility distributions, we also developed algorithms with notable improvements over existing approaches in the literature. Along with those improved methods, our results give a rather comprehensive picture of how to compute upper entropies for practical credal sets routinely used in applications.

Also, as the algorithm for computing upper entropy in general cases has high-degree polynomial complexity, we proposed an alternative approach using the classic Frank-Wolfe method. Initial experiments indicates that this alternative is practical and achieves good precision quickly, thus offering a highly efficient alternative to exact methods. In the future, we aim to perform much more comprehensive experiments. In particular, for belief functions, it is interesting to compare the exact resolution approach using maximum-flow algorithms with the approximate one via Frank-Wolfe.

A more general question to which this paper contributes is to know to which extent classical results issued from submodular optimization can help to solve commonly encountered computational issues for credal sets, and more specifically those induced by 2-monotone lower probabilities? One could think of other robust versions of classical uncertainty quantification tools, such as KL divergences, Wasserstein metrics. Another interesting topic to explore under this algorithmic and computational aspect is optimal transport [Lorenzini et al., 2025, Caprio, 2025].

References

Joaquin Abellan and Serafin Moral. Maximum of entropy for credal sets. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 11(05):587–597, 2003.

Joaquín Abellán and Serafín Moral. Upper entropy of credal sets. applications to credal classification. *International Journal of Approximate Reasoning*, 39(2-3):235–255, 2005.

Joaquín Abellán and Serafín Moral. An algorithm to compute the upper entropy for order-2 capacities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 14(02):141–154, 2006.

Joaquín Abellán, Alejandro Pérez-Lara, and Serafín Moral-García. A variation of the algorithm to achieve the maximum entropy for belief functions. *Entropy*, 25(6), 2023.

Francis R. Bach. Learning with submodular functions: A convex optimization perspective. *Found. Trends Mach. Learn.*, 6(2-3):145–373, 2013.

Tadeusz Bednarski. On solutions of minimax test problems for special capacities. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 58(3):397–405, 1981.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Michele Caprio. Optimal transport for ε-contaminated credal sets. In *International Symposium on Imprecise Probabilities: Theories and Applications*, pages 33–46. PMLR, 2025.

Michele Caprio, Maryam Sultana, Eleni G Elia, and Fabio Cuzzolin. Credal learning theory. *Advances in Neural Information Processing Systems*, 37:38665–38694, 2024.

Michele Caprio, Shireen K Manchingal, and Fabio Cuzzolin. Credal and interval deep evidential classifications. *arXiv preprint arXiv:2512.05526*, 2025.

Siu Lun Chau, Michele Caprio, and Krikamol Muandet. Integral imprecise probability metrics. *arXiv preprint arXiv:2505.16156*, 2025.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 3rd edition, 2009.

Fabio G Cozman. Credal networks. *Artificial Intelligence*, 120(2):199–233, 2000.

Mark De Berg, Otfried Cheong, Marc Van Kreveld, and Mark Overmars. *Computational geometry: algorithms and applications*. Springer, 2008.

Luis M De Campos, Juan F Huete, and Serafin Moral. Probability intervals: a tool for uncertain reasoning. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2(02):167–196, 1994.

Werner Dinkelbach. On nonlinear fractional programming. *Management Science*, 13(7):492–498, 1967.

Abhimanyu Dubey, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Maximum-entropy fine grained classification. *Advances in neural information processing systems*, 31, 2018.

Lisa Fleischer and Satoru Iwata. A push-relabel framework for submodular function minimization and applications to parametric optimization. *Discrete Applied Mathematics*, 131(2):311–322, 2003.

Satoru Fujishige. *Submodular functions and optimization*, volume 58. Elsevier, 2005.

Michel Grabisch. K-order additive discrete fuzzy measures and their representation. *Fuzzy Sets and Systems*, 92(2): 167–189, 1997.

Yves Grandvalet and Yoshua Bengio. Entropy regularization. *Semi-Supervised Learning*, pages 151–168, 2006.

M. Grötschel, L. Lovász, and A. Schrijver. *Geometric algorithms and combinatorial optimization*. Springer, 1993.

Pierre Hansen, Brigitte Jaumard, Marcus Poggi De Aragao, Fabien Chauny, and Sylvain Perron. Probabilistic satisfiability with imprecise probabilities. *International Journal of Approximate Reasoning*, 24(2-3):171–189, 2000.

D. Harmanec, G. Resconi, G. J. Klir, and Y. Pan. On the computation of uncertainty measure in Dempster–Shafer theory. *International Journal of General Systems*, 25(2):153–163, 1996.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020.

Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pages 2681–2691. PMLR, 2019.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

Van-Nam Huynh and Yoshiteru Nakamori. Notes on "reducing algorithm complexity for computing an aggregate uncertainty measure". *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 40(1): 205–209, 2009.

Alireza Javanmardi, David Stutz, and Eyke Hüllermeier. Conformalized credal set predictors. *Advances in Neural Information Processing Systems*, 37:116987–117014, 2024.

George J Klir. Uncertainty and information. *Foundations of Generalized Information Theory*, 2006.

Isaac Levi. *The enterprise of knowledge: An essay on knowledge, credal probability, and chance*. MIT press, 1980.

Timo Löhr, Paul Hofman, Felix Mohr, and Eyke Hüllermeier. Credal prediction based on relative likelihood. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=rKM3oqruN3`.

Silvia Lorenzini, Davide Petturiti, and Barbara Vantaggi. Choquet-wasserstein pseudo-distances via optimal transport under partially specified marginal probabilities. *Fuzzy Sets and Systems*, 515:109429, 2025.

Aaron Meyerowitz, Fred Richman, and Elbert Walker. Calculating maximum-entropy probability densities for belief functions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2(04):377–389, 1994.

Ignacio Montes and Sebastien Destercke. On extreme points of p-boxes and belief functions. *Annals of Mathematics and Artificial Intelligence*, 81(3):405–428, 2017.

Kiyohito Nagano and Yoshinobu Kawahara. Structured convex optimization under submodular constraints. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, UAI*. AUAI Press, 2013.

Vu-Linh Nguyen, Haifei Zhang, and Sébastien Destercke. Credal ensembling in multi-class classification. *Machine Learning*, 114(1):19, 2025.

James B Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251, 2009.

J. C. Picard and M. Queyranne. On the structure of all minimum cuts in a network and applications. *Mathematical Programming Study*, 13:8–16, 1980.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 1992.

Glenn Shafer. *A mathematical theory of evidence*. Princeton university press, 1976.

Anurag Singh, Siu Lun Chau, Shahine Bouabid, and Krikamol Muandet. Domain generalisation via imprecise learning. In *International Conference on Machine Learning*, pages 45544–45570. PMLR, 2024.

Matthias CM Troffaes and Gert De Cooman. *Lower previsions*. John Wiley & Sons, 2014.

Peter Walley. *Statistical reasoning with imprecise probabilities*, volume 42. Springer, 1991.

Kaizheng Wang, Fabio Cuzzolin, Keivan Shariatmadar, David Moens, Hans Hallez, et al. Credal deep ensembles for uncertainty quantification. *Advances in Neural Information Processing Systems*, 37:79540–79572, 2024.

A Proof of Lemma 1

Before proving this, we remark that if μ is supermodular, the set function $\mu_\lambda(A) := \mu(A) - \lambda|A|$ is also supermodular as $|A| + |B| = |A \cap B| + |A \cup B| \; \forall A, B$.

Part 1: Finding $A \in \arg\max_{\emptyset \neq B \subseteq \Omega} \frac{\mu(B)}{|B|}$ by solving $O(n)$ SFM.

For this part, we adapt the idea described in [Fleischer and Iwata, 2003, Sec. 4.1].

Proof. Define a function $g(\lambda) := \max_{\emptyset \neq B \subseteq \Omega} \mu(B) - \lambda|B|$. We claim that

$$\lambda^* := \max_{\emptyset \neq B \subseteq \Omega} \frac{\mu(B)}{|B|} \Leftrightarrow g(\lambda^*) = 0. \tag{11}$$

Indeed, if $\max_{B \neq \emptyset} \frac{\mu(B)}{|B|} > \lambda$ then $\max_{B \neq \emptyset} \frac{\mu(B) - \lambda|B|}{|B|} > 0$, and thus it follows that $g(\lambda) > 0$ because $|B| > 0$. Hence, we need to solve $g(\lambda) = 0$, for which the Dinkelbach's method Dinkelbach [1967] will be employed. Note that $\lambda^* \in (0, 1]$. Dinkelbach's method then constructs the sequence $\{\lambda_t\}$ starting from $\lambda_0 = 0$, evaluates $g(\lambda_t)$, and updates λ_t until it hits a solution of g as

1. Find $B_t \in \arg\max_{B \neq \emptyset} \mu(B) - \lambda_t|B|$. Evaluate $g(\lambda_t) = \mu(B_t) - \lambda_t|B_t|$.

2. If $g(\lambda_t) = 0$, then λ_t is a solution and B_t is optimal.

3. Otherwise, update $\lambda_{t+1} := \lambda_t + \frac{g(\lambda_t)}{|B_t|} = \frac{\mu(B_t)}{|B_t|}$.

Define $M_k := \max_{|B|=k} \mu(B)$ for $k = 1, \ldots, n$. At each iteration, because $|B_t|$ is a maximizer, if $|B_t| = k$, it follows that $\mu(B_t) = M_k$, and thus $\lambda_{t+1} = \frac{M_k}{k}$. Consequentiality, the sequence $\{\lambda_t\}$ can only take values in the finite set $\{\frac{M_1}{1}, \ldots, \frac{M_n}{n}\}$. Since $\{\lambda_t\}$ strictly increases until optimality with $g(\lambda_t) = 0$, the loop terminates in at most n iterations. Finally, at each iteration we solve $\max_B \mu(B) - \lambda_t|B|$ which is an SFM as μ is supermodular. In total, we need to solve $O(n)$ SFM. $\qquad\square$

Part 2: Finding the maximizer A of maximum size by solving $O(n)$ SFM.

Proof. Because the set of maximizers of a supermodular function is closed under union, after finding λ^* (see (11)) via Dinkelbach's method, we simply take the union of all maximizers of μ_{λ^*} where $\mu_{\lambda^*}(B) := \mu(B) - \lambda^*|B| \; \forall B$. Observe that $i \in \Omega$ belongs to a maximizer of μ_{λ^*} if and only if

$$\max_{B \subseteq \Omega \setminus \{i\}} \mu(B \cup \{i\}) - \lambda^*|B \cup \{i\}| = 0. \tag{12}$$

After solving (12) n times for each $i \in \Omega$, the desired set A consists of all i such that (12) holds. $\qquad\square$

B Proof of Proposition 6

WLOG, assume that $u_i > 0 \; \forall i$ (otherwise $p_i = 0$ and we discard p_i), $\sum_{i=1}^n l_i < 1$ (otherwise $p_i = l_i \; \forall i$), and $\sum_{i=1}^n u_i > 1$ (otherwise $p_i = u_i \; \forall i$). Under these assumptions, at the optimal p, we have $p_i > 0 \; \forall i$.

Proof. By the KKT conditions, p is optimal of Problem (7) if and only if the following system (with variables p_i, α_i, β_i and λ) is feasible:

$$l_i \leq p_i \leq u_i \text{ and } \sum_{i=1}^n p_i = 1 \tag{13}$$

$$\alpha_i, \beta_i \geq 0 \; \forall i \tag{14}$$

$$\alpha_i(-p_i + l_i) = 0 \text{ and } \beta_i(p_i - u_i) = 0 \; \forall i \tag{15}$$

$$\log p_i + 1 - \alpha_i + \beta_i + \lambda = 0 \; \forall i \tag{16}$$

Let $f(x) := \sum_{i=1}^n \min\{\max\{x, l_i\}, u_i\}$ and $b = \max_i u_i$. Hence, $f(0) = \sum_{i=1}^n l_i < 1$ and $f(b) = \sum_{i=1}^n u_i > 1$ (by our assumptions). As f is continuous, by the intermediate value theorem, there exists an $x \in (0, b)$ such that

$f(x) = 1$. We define $p_i = \min\{\max\{x, l_i\}, u_i\}$ $\forall i$. We claim that such p_i is feasible to (13-16) and thus is optimal. By the definition of p_i, (13) automatically holds. We set $\lambda = -1 - \log x$. For each i, if $l_i < u_i$, we consider three cases:

1. If $l_i < x < u_i$ then $p_i = x$, and we set $\alpha_i = \beta_i = 0$.

2. If $x \leq l_i$ then $p_i = l_i$, and we set $\alpha_i = \log \frac{l_i}{x}$ and $\beta_i = 0$.

3. If $x \geq u_i$ then $p_i = u_i$, and we set $\alpha_i = 0$ and $\beta_i = \log \frac{x}{u_i}$.

Finally, if $l_i = u_i$, then $p_i = l_i$, and we can set any α_i, β_i such that $-\alpha_i + \beta_i = \log \frac{x}{l_i}$. $\qquad\square$

C   Proof of Remark 3

We make use of the following well-known result about the structure of vertices of $\mathcal{P}(\mu)$ [Fujishige, 2005]. Let ϕ be a permutation on Ω, the probability p where $p_{\phi(i)} := \bar{\mu}(S_i^\phi) - \bar{\mu}(S_{i-1}^\phi)$ with $S_0^\phi := \emptyset$, $S_i^\phi := \{\phi(1), \ldots, \phi(i)\}$ $\forall i$, is a vertex of $\mathcal{P}(\mu)$.

Proof. If $\exists p \in \mathcal{P}(\mu)$ such that $p_i > 0$ $\forall i$ then $\bar{\mu}(\{i\}) > p_i > 0$. Conversely, assume that $\bar{\mu}(\{i\}) > 0$ $\forall i$. Consider n permutations $\phi^1, \ldots, \phi^n$ on Ω where $\phi^i(1) = i$ $\forall i$. Because of the above-mentioned result, each ϕ^i induces a vertex of $\mathcal{P}(\mu)$ for which its ith component is positive. Taking the average of these n vertices, we obtain a probability $p \in \mathcal{P}(\mu)$ such that $p_i > 0$ $\forall i$. Moreover, this p is found in $O(n^2\text{EO})$. $\qquad\square$