On computing the maximum-entropy probability consistent with a capacity

Pietro Baroni
Dip. Elettronica per l'Automazione
University of Brescia
Via Branze 38, I-25123 Brescia, Italy
baroni@ing.unibs.it

Paolo Vicig
Dip. Matematica Applicata "B. de Finetti"
University of Trieste
Piazzale Europa 1, I-34127 Trieste, Italy
paolo.vicig@econ.units.it

Abstract

The problem of computing the maximum-entropy precise probability consistent with a 2-monotone capacity is solved by an algorithm devised by Jaffray. After reviewing its properties, with particular reference to its applicability scope, we introduce a modified version of the algorithm, showing that it works correctly on a strictly larger family of capacities. Finally, an empirical analysis about behavior and performance of the proposed algorithm is carried out.

Keywords: Capacities, Imprecise probabilities, Maximum entropy.

1 Introduction

The problem of computing the maximum-entropy probability consistent with a capacity attracts the interest of researchers from several perspectives. Among them, the maximum-entropy probability can be regarded as a "synthetic" representation of a more general uncertainty assignment (e.g. a belief function or an imprecise probability). Further, the entropy value of this (precise) probability was considered when measuring uncertainty of an imprecise probability [1, 6]. An algorithm to solve this problem was first devised in [8] for belief functions. The result was generalized by Jaffray who proposed in [5] an algorithm, referred to as *algorithm J* in this paper, to compute the maximum-entropy precise probability consistent with a 2-monotone capacity. It has to be remarked that, while the analysis carried out in [5] is restricted to 2-monotone capacities, the applicability scope of algorithm J is actually larger. This is probably the root of a recent claim that the algorithm "which was proven correct for belief functions theory, is applicable without any change when belief functions are replaced with arbitrary lower probability functions of any other kind" ([6], p. 327). Unfortunately this claim does not hold in general (cf. [2, 3] or the later Example 1). This paper aims at providing a twofold contribution. First of all, we analyze in section 2 the applicability scope of algorithm J, pointing out that it actually works correctly on a set of uncertainty assignments which strictly includes 2-monotone capacities but does not cover more general imprecise probabilities. Then we introduce in section 3 a generalized version of algorithm J, called *algorithm G*, and analyze its properties in section 4, showing that it works on a strictly larger family of capacities. This result is complemented in section 5 by an empirical analysis carried out over a large number of randomly generated imprecise probabilities and concerning the applicability scope and performances of the algorithms. Section 6 concludes the paper.

2 Jaffray's algorithm

Algorithm J receives in input an upper measure $\overline{P}(\cdot)$ defined on the powerset 2^{Ω} of a given universal set $\Omega = \{\omega_1, \ldots, \omega_N\}$ and produces a precise probability $P_J(\cdot)$ on 2^{Ω}. In [5], $\overline{P}$ is *submodular* (or *2-alternating*), namely such that $\overline{P}(S_1 \cup S_2) + \overline{P}(S_1 \cap S_2) \leq \overline{P}(S_1) + \overline{P}(S_2)$,

$\forall S_1, S_2 \in 2^\Omega$; this is equivalent to its conjugate $\underline{P}$ being *supermodular* (or *2-monotone*). Then P_J respects the consistency constraint:

$$\forall S \in 2^\Omega, P_J(S) \leq \overline{P}(S) \tag{1}$$

and is determined by a distribution p_J on the atoms of Ω, constructed as follows.

Definition of algorithm J

```
Step 1. put $A_0^J = B_0^J = \emptyset$; $k = 1$;
BEGIN MAIN LOOP
Step 2. Determine $A_k^J$ such that:
$\emptyset \neq A_k^J \subseteq (\Omega \setminus B_{k-1}^J)$ and
$\frac{\overline{P}(B_{k-1}^J \cup A_k^J) - \overline{P}(B_{k-1}^J)}{|A_k^J|} =$
$= \min_{\emptyset \neq E \subseteq (\Omega \setminus B_{k-1}^J)} \left( \frac{\overline{P}(B_{k-1}^J \cup E) - \overline{P}(B_{k-1}^J)}{|E|} \right)$;
Step 3. put $B_k^J = B_{k-1}^J \cup A_k^J$;
put $\alpha_k^J = \frac{\overline{P}(B_{k-1}^J \cup A_k^J) - \overline{P}(B_{k-1}^J)}{|A_k^J|}$;
Step 4. $\forall \omega \in A_k^J$ put $p_J(\omega) = \alpha_k^J$;
Step 5. if $B_k^J = \Omega$ then EXIT;
else put $k = k + 1$;
Step 6. goto Step 2;
END MAIN LOOP
```

In words, algorithm J identifies a sequence of nested sets $\emptyset = B_0^J \subsetneq B_1^J \subsetneq \ldots \subsetneq B_K^J = \Omega$, with $K \geq 1$, by suitably selecting at each main loop iteration a set $A_k^J \subseteq (\Omega \setminus B_{k-1}^J)$ to be added to B_{k-1}^J (therefore $B_k^J \setminus B_{k-1}^J = A_k^J$). The selection criterion for A_k^J in Step 2 ensures that the probability values p_J are assigned in a way as uniform as allowed by the consistency constraint (1). The choice of A_k^J is *nondeterministic* when the minimum at Step 2 is attained by more than one set in some iteration. Without any assumption on the nondeterministic choices, it is proved in [5], Proposition 2, that any probability P_J produced by algorithm J is an optimal solution of a convex programming problem related to entropy maximization. Since this problem admits a unique optimal solution, P_J is the same for any nondeterministic choice.

Let us discuss the main properties of algorithm J, proved in [5], from the perspective of extension to more general capacities.

Properties of algorithm J

1. $\overline{P}(\cdot)$ may be assumed to be strictly positive on $2_*^\Omega \triangleq 2^\Omega \setminus \{\emptyset\}$;
2. $0 < \alpha_k^J \leq \alpha_{k+1}^J$ for $k = 1, \ldots, K-1$;
3. $\forall S \in 2^\Omega, P_J(S) \leq \overline{P}(S)$;
4. $\sum_{k=1}^K \alpha_k^J |A_k^J| = \sum_{\omega \in \Omega} p_J(\omega) = 1$;
5. $P_J(\cdot)$ is the maximum entropy probability consistent with $\overline{P}(S)$.

Property 1 is based on the fact that $\overline{P}$ is *zero-additive*, i.e. $\forall S_1, S_2 \in 2^\Omega, \overline{P}(S_1) = \overline{P}(S_2) = 0 \Rightarrow \overline{P}(S_1 \cup S_2) = 0$. This implies that there is a largest set $S \in 2^\Omega$ such that $\overline{P}(S) = 0$ (hence the restriction of $\overline{P}$ to $2^\Omega \setminus S$ is considered in algorithm J). As remarked in [5], submodular capacities are zero-additive. Such are also the measures in the (larger) family of coherent upper probabilities [9], because of their nonnegativity and sublinearity. Capacities outside this family may or may not be zero-additive. Property 2 depends only on Property 1 and the definition of algorithm J, whilst Property 3 relies on submodularity of $\overline{P}$ and algorithm J's definition, which entail that $\forall S \in 2^\Omega$, $P_J(S \cap A_k^J) \leq \overline{P}(S \cap B_k^J) - \overline{P}(S \cap B_{k-1}^J)$, for $1 \leq k \leq K$. Then $P_J(S) = \sum_{k=1}^K P_J(S \cap A_k^J) \leq \sum_{k=1}^K \left(\overline{P}(S \cap B_k^J) - \overline{P}(S \cap B_{k-1}^J)\right) = \overline{P}(S)$. Property 4 derives from the definition of algorithm J since $B_K^J = \Omega$. The proof of Property 5 in [5] relies on algorithm J definition and Properties 2, 3, and 4, thus exploiting submodularity of $\overline{P}$ only through Property 3. Turning now to more general uncertainty measures, note first that the problem we are discussing makes sense only when the set of consistent precise probabilities is non-empty, that is iff the capacity is an ASL (Avoiding Sure Loss) imprecise probability [9]. The ASL condition may be referred to either upper or lower capacities, because of conjugacy. To parallel Jaffray's procedure, we consider upper capacities. Hence the uncertainty measures we are concerned with in the sequel, denoted with $\overline{P}$, are ASL zero-additive upper capacities defined on 2^Ω (zero-additivity is required to ensure Property 1).

A crucial point is that Property 5 of algorithm J is preserved also if $\overline{P}$ is not submodular, provided that Property 3 still holds. We make two remarks on this. First, submodularity is a sufficient, but not necessary condition for Property 3. For instance, it is easy to see that when $\overline{P}$ dominates the uniform probability the procedure ends in one iteration, by selecting $A_1^J = B_1^J = \Omega$ independently of any other property of $\overline{P}$. On the other hand, examples are known where Jaffray's algorithm produces a precise probability which is not consistent with the given upper probability [2, 3]. Let us see this.

Example 1 Let $\Omega = \{a, b, c, d\}$, and $\overline{P}$ be the coherent upper probability on 2^Ω which is the upper envelope of the two precise probabilities P_1, P_2, determined by orderly assigning the following values on the atoms a, b, c, d: P_1 - values $[0.29, 0.34, 0.04, 0.33]$, P_2 - values $[0.76, 0.11, 0.08, 0.05]$. At the first main loop iteration $\frac{\overline{P}(S)}{|S|}$ is evaluated for all $S \in 2^\Omega_*$ and the set S giving rise to the minimum value, actually $\{c\}$, is selected as A_1^J. Then $\alpha_1^J = \frac{\overline{P}(A_1^J)}{|A_1^J|} = 0.08$; $p_J(c) = 0.08$; $B_1^J = A_1^J$. At the second iteration, $\frac{\overline{P}(S) - \overline{P}(B_1^J)}{|S \setminus B_1^J|}$ is evaluated only for the proper supersets S of B_1^J. The minimum value of such a ratio, actually 0.29, is obtained for $S = \{c, d\}$. Then $A_2^J = \{d\}$; $p_J(d) = \alpha_2^J = 0.29$. Then the supersets S of $B_2^J = \{c, d\}$ are considered: $\min(\frac{\overline{P}(S) - \overline{P}(B_2^J)}{|S \setminus B_2^J|})$ is achieved for $S = \Omega$. Then $A_3^J = \{a, b\}$; $p_J(a) = p_J(b) = \alpha_3^J = 0.315$, and the algorithm terminates since $B_3^J = \Omega$. However the probability P_J obtained by additivity from p_J is not consistent with $\overline{P}$ since $P_J(\{b, c\}) = 0.395 > \overline{P}(\{b, c\}) = 0.38$. □

It is useful for the sequel to state a condition which characterizes Property 3 referring to algorithm J. Such a condition can be identified by considering that, at the k-th iteration of the algorithm, P_J is determined for all sets $S : S \subseteq B_k^J, S \not\subset B_{k-1}^J$ and that for any such S it holds $P_J(S) = P_J(S \cap B_{k-1}^J) + \alpha_k^J |S \setminus B_{k-1}^J|$. Now, requiring consistency of P_J is clearly equivalent to imposing, for each iteration k, that $\overline{P}(S) \geq P_J(S)$ for all sets S whose P_J is determined at iteration k. Therefore we have:

Proposition 1 *Let P_J be a precise probability produced applying algorithm J to $\overline{P}(\cdot)$. Then, $P_J(S) \leq \overline{P}(S), \forall S \in 2^\Omega$ if and only if $\forall k$, $1 \leq k \leq K$, $\forall S \subseteq B_k^J$, $S \not\subset B_{k-1}^J$:*

$$\frac{\overline{P}(S) - P_J(S \cap B_{k-1}^J)}{|S \setminus B_{k-1}^J|} \geq \alpha_k^J \tag{2}$$

where $\alpha_k^J = \frac{\overline{P}(B_k^J) - \overline{P}(B_{k-1}^J)}{|B_k^J \setminus B_{k-1}^J|}$.

As we will now see, condition (2) is the starting point for generalizing algorithm J.

3 Generalizing Jaffray's algorithm

Condition (2) provides a constraint on the sets $S \not\subset B_{k-1}^J$, $S \subseteq B_k^J$ for each iteration k. However Step 2 of algorithm J enforces (2) only for the sets S such that $B_{k-1}^J \subsetneq S \subseteq B_k^J$. Then, the produced result is correct if $\overline{P}$ is such that (2) holds also for any set S such that $B_{k-1}^J \not\subseteq S \subseteq B_k^J$. In Example 1, this is not the case for the set $\{b, c, e\}$.

On the basis of this observation it is possible to devise a modified version of Jaffray's algorithm, which enforces the constraint on all the sets. This generalized algorithm, called *Algorithm G*, constructs a distribution $p_G(\cdot)$ and the relevant additive measure $P_G(\cdot)$ as follows.

Definition of algorithm G

```
Step 1. put $A_0^G = B_0^G = \emptyset$;
put $k = 1$; $P_G(\emptyset) = 0$;

BEGIN MAIN LOOP

Step 2. Determine $A_k^G \in 2^\Omega$ such that:
$A_k^G \not\subseteq B_{k-1}^G$ and
$\frac{\overline{P}(A_k^G) - P_G(A_k^G \cap B_{k-1}^G)}{|A_k^G \setminus B_{k-1}^G|} =$
$= \min_{E \in 2^\Omega, E \not\subseteq B_{k-1}^G} \left( \frac{\overline{P}(E) - P_G(E \cap B_{k-1}^G)}{|E \setminus B_{k-1}^G|} \right)$;

Step 3. put $B_k^G = B_{k-1}^G \cup A_k^G$;
put $\alpha_k^G = \frac{\overline{P}(A_k^G) - P_G(A_k^G \cap B_{k-1}^G)}{|A_k^G \setminus B_{k-1}^G|}$;

Step 4. $\forall \omega \in (A_k^G \setminus B_{k-1}^G)$ put $p_G(\omega) = \alpha_k^G$;

Step 5. if $B_k^G = \Omega$ then EXIT;
else put $k = k + 1$;

Step 6. goto Step 2;
END MAIN LOOP
```

Also algorithm G operates on a sequence of nested sets $\emptyset = B_0^G \subsetneq B_1^G \subsetneq \ldots \subsetneq B_K^G = \Omega$, with $K \geq 1$, by properly selecting, at each main loop iteration, a set A_k^G to be added to B_{k-1}^G. The core difference with respect to algorithm J is that A_k^G is selected among all the sets $S \not\subseteq B_{k-1}^G$, while $A_k^J \subseteq (\Omega \setminus B_{k-1}^J)$ in algorithm J. In other words, it may be $A_k^G \cap B_{k-1}^G \neq \emptyset$, while $A_k^J \cap B_{k-1}^J = \emptyset$. Therefore a larger set of candidates for A_k^G is considered and, as a side effect, not necessarily $B_k^G \setminus B_{k-1}^G = A_k^G$, while $B_k^J \setminus B_{k-1}^J = A_k^J$ in algorithm J, by construction. Consequently the difference $\overline{P}(A_k^G) - P_G(A_k^G \cap B_{k-1}^G)$ replaces $\overline{P}(B_{k-1}^J \cup A_k^J) - \overline{P}(B_{k-1}^J)$ and the denominator is $|A_k^G \setminus B_{k-1}^G|$ instead of $|A_k^J|$.

Note that Step 4 implies also that:

$$P_G(S) = \alpha_k^G |S|, \forall S \subseteq (A_k^G \setminus B_{k-1}^G). \qquad (3)$$

Further, using (3) with $S = A_k^G \setminus B_{k-1}^G$ in the definition of α_k^G (Step 3), we obtain $P_G(A_k^G \setminus B_{k-1}^G) = \overline{P}(A_k^G) - P_G(A_k^G \cap B_{k-1}^G)$, and hence

$$\overline{P}(A_k^G) = P_G(A_k^G), \forall k \qquad (4)$$

To exemplify the difference between algorithms G and J, consider again Example 1. Clearly, the first main loop iteration is always the same for the two algorithms. Therefore $A_1^G = \{c\}$. Then $\alpha_1^G = 0.08$; $p_G(c) = 0.08$; $B_1^G = A_1^G$. At the second iteration, $\forall S \not\subseteq B_1^G$, $\frac{\overline{P}(S) - P_G(S \cap B_1^G)}{|S \setminus B_1^G|}$ is evaluated. Note that $P_G(S \cap B_{k-1}^G)$ is already available at iteration k, since p_G has necessarily been assigned in previous iteration(s) to all atoms $\omega \in B_{k-1}^G$ (one atom in the first iteration of our example). It turns out that (again, as in algorithm J) $A_2^G = \{c, d\}$, hence $p_G(d) = \alpha_2^G = 0.29$; $B_2^G = \{c, d\}$.

In the next iteration, the difference between algorithms becomes effective: the minimum value for $\frac{\overline{P}(S) - P_G(S \cap B_2^G)}{|S \setminus B_2^G|}$, actually 0.3, is obtained for $S = \{b, c\}$, therefore $A_3^G = \{b, c\}$; $A_3^G \setminus B_2^G = \{b\}$; $p_J(b) = \alpha_3^G = 0.3$; $B_3^G = \{b, c, d\}$. Since only one atom remains to be added, the fourth iteration is necessarily the last one. For all sets $S \supseteq \{a\}$, $\frac{\overline{P}(S) - P_G(S \cap B_3^G)}{1}$ is evaluated: its minimum value 0.33 is attained for $S = \Omega = \{a, b, c, d\}$, completing the p_J assignment with $p_J(a) = 0.33$.

Unlike algorithm J, the resulting probability measure is consistent with $\overline{P}$.

4 Properties of algorithm G

Algorithm G returns the same result as algorithm J, for any $\overline{P}$ satisfying condition (2). To prove this, we first show in Proposition 2 that any solution satisfying (2) produced by algorithm J may also be obtained by algorithm G, possibly through some nondeterministic choice. Then, we will show in Proposition 4 that algorithm G produces the same solution for any nondeterministic choice.

Proposition 2 *Let $\mathcal{S}_J(\overline{P})$ and $\mathcal{S}_G(\overline{P})$ be the sets of all $\{A_k^J\}$, $\{A_k^G\}$ sequences that can be produced by applying, respectively, algorithms J and G to $\overline{P}(\cdot)$. If condition (2) holds for all $\{A_k^J\} \in \mathcal{S}_J(\overline{P})$, then $\mathcal{S}_J(\overline{P}) \subseteq \mathcal{S}_G(\overline{P})$.*

Proof. For a given $\overline{P}$, let $A_0^J, \ldots, A_K^J$ be a sequence in $\mathcal{S}_J(\overline{P})$: let us show that it also belongs to $\mathcal{S}_G(\overline{P})$. First note that, by definition, for any $\overline{P}$ it holds that $A_0^J = A_0^G = \emptyset = B_0^J = B_0^G$, and therefore also $P_J(S \cap B_0^J) = P_G(S \cap B_0^G)$, for any $S \in 2^\Omega$. Moreover $\overline{P}(B_0^G) = P_G(B_0^G) = 0$.

Then, let inductively be, for $k \geq 1$, $A_{k-1}^G = A_{k-1}^J$; $B_{k-1}^G = B_{k-1}^J$; $P_G(B_{k-1}^G) = \overline{P}(B_{k-1}^G)$; $P_G(S \cap B_{k-1}^G) = P_J(S \cap B_{k-1}^J)$, $\forall S \in 2^\Omega$.

We show that the inductive hypothesis and condition (2) imply $A_k^G = A_k^J$ (and hence $B_k^G = B_k^J$; $\alpha_k^G = \alpha_k^J$; $P_G(B_k^G) = \overline{P}(B_k^G)$; $P_G(S \cap B_k^G) = P_J(S \cap B_k^J)$, $\forall S \in 2^\Omega$).

In fact, substituting $P_J(S \cap B_{k-1}^J) = P_G(S \cap B_{k-1}^G)$ and $B_{k-1}^J = B_{k-1}^G$ in (2) and in the ratio defining α_k^J we obtain:
$r_k^G(S) \triangleq \frac{\overline{P}(S) - P_G(S \cap B_{k-1}^G)}{|S \setminus B_{k-1}^G|} \geq \alpha_k^J = \frac{\overline{P}(B_k^J) - \overline{P}(B_{k-1}^G)}{|B_k^J \setminus B_{k-1}^G|}, \forall S \subseteq B_k^J, S \not\subseteq B_{k-1}^J$.

Recalling $\overline{P}(B_{k-1}^G) = P_G(B_{k-1}^G)$ and noting that $B_{k-1}^G = B_{k-1}^J = (B_k^J \cap B_{k-1}^J)$, this entails: $\alpha_k^J = \frac{\overline{P}(B_k^J) - \overline{P}(B_k^J \cap B_{k-1}^G)}{|B_k^J \setminus B_{k-1}^G|} = r_k^G(B_k^J) \leq r_k^G(S), \forall S \subseteq B_k^J, S \not\subseteq B_{k-1}^J$.

Since $\alpha_k^G = \min_{S \not\subseteq B_{k-1}^G} r_k^G(S)$, the above condition guarantees that, at iteration k, the set B_k^J is at least as preferred as any set $S \subseteq B_k^J, S \not\subseteq B_{k-1}^J$ in the selection by algorithm G. Let us now show that B_k^J is also preferred to any set $T \not\subseteq B_k^J$: this ensures that B_k^J can be the actual (possibly nondeterministic) choice of algorithm G at iteration k and completes the proof.

For any set $T \not\subseteq B_k^J$ there is an index m such that $T \subseteq B_m^J, T \not\subseteq B_{m-1}^J$, with $m > k$. Since condition (2) holds, P_J is consistent with $\overline{P}$ and in particular $\overline{P}(T) \geq P_J(T)$. Then $\overline{P}(T) - P_J(T \cap B_{k-1}^J) \geq P_J(T) - P_J(T \cap B_{k-1}^J) = \sum_{i=k}^m P_J(T \cap A_i^J) = \sum_{i=k}^m \alpha_i^J |T \cap A_i^J|$.

Now, since $\alpha_i^J \leq \alpha_{i+1}^J$ for any i (Property 2) and $\sum_{i=k}^m |T \cap A_i^J| = |T \setminus B_{k-1}^J|$ it holds that $\sum_{i=k}^m \alpha_i^J |T \cap A_i^J| \geq \alpha_k^J |T \setminus B_{k-1}^J|$.
Hence, $\overline{P}(T) - P_J(T \cap B_{k-1}^J) \geq \alpha_k^J |T \setminus B_{k-1}^J|$; recalling $B_{k-1}^J = B_{k-1}^G$ and $P_J(T \cap B_{k-1}^J) = P_G(T \cap B_{k-1}^G)$, we obtain: $\forall T \not\subseteq B_k^J$,
$r_k^G(B_k^J) = \alpha_k^J \leq \frac{\overline{P}(T) - P_G(T \cap B_{k-1}^G)}{|T \setminus B_{k-1}^G|} = r_k^G(T)$. □

Let us now investigate the properties of algorithm G. We first verify that it satifies a property exactly corresponding to Property 2.

Proposition 3 *Let $\alpha_1^G, \ldots, \alpha_K^G$ be a sequence of values produced by algorithm G on $\overline{P}(\cdot)$. Then $0 < \alpha_k^G \leq \alpha_{k+1}^G$, for $k = 1, \ldots, K-1$.*

Proof. $0 < \alpha_1^G$ derives from zero-additivity. For $k \geq 1$, consider now $\alpha_k^G =$
$\min_{E \in 2^\Omega, E \not\subseteq B_{k-1}^G} \frac{\overline{P}(E) - P_G(E \cap B_{k-1}^G)}{|E \setminus B_{k-1}^G|}$.

We show by contradiction that $\alpha_{k+1}^G \geq \alpha_k^G$. By (4), $\overline{P}(A_{k+1}^G) = P_G(A_{k+1}^G) = P_G(A_{k+1}^G \cap B_k^G) + P_G(A_{k+1}^G \setminus B_k^G)$.
Since (recalling $B_{k-1}^G \subsetneq B_k^G$) $A_{k+1}^G \cap B_k^G = (A_{k+1}^G \cap B_{k-1}^G) \cup (A_{k+1}^G \cap (B_k^G \setminus B_{k-1}^G))$, and any $\omega \in (A_{k+1}^G \cap (B_k^G \setminus B_{k-1}^G))$ is given the value $p_G(\omega) = \alpha_k^G$, it is $P_G(A_{k+1}^G \cap B_k^G) = P_G(A_{k+1}^G \cap B_{k-1}^G) + \alpha_k^G |A_{k+1}^G \cap (B_k^G \setminus B_{k-1}^G)|$. Moreover, $P_G(A_{k+1}^G \setminus B_k^G) = \alpha_{k+1}^G |A_{k+1}^G \setminus B_k^G|$, by (3).

Assuming now $\alpha_{k+1}^G < \alpha_k^G$, we have $\overline{P}(A_{k+1}^G) = P_G(A_{k+1}^G \cap B_{k-1}^G) + \alpha_k^G |A_{k+1}^G \cap (B_k^G \setminus B_{k-1}^G)| + \alpha_{k+1}^G |A_{k+1}^G \setminus B_k^G| < P_G(A_{k+1}^G \cap B_{k-1}^G) + \alpha_k^G |A_{k+1}^G \setminus B_{k-1}^G|$. This implies $\frac{\overline{P}(A_{k+1}^G) - P_G(A_{k+1}^G \cap B_{k-1}^G)}{|A_{k+1}^G \setminus B_{k-1}^G|} < \alpha_k^G =$
$= \min_{E \in 2^\Omega, E \not\subseteq B_{k-1}^G} (\frac{\overline{P}(E) - P_G(E \cap B_{k-1}^G)}{|E \setminus B_{k-1}^G|})$.
Contradiction. □

Using Proposition 3, it is possible to show that Algorithm G produces the same P_G independently of the possibly nondeterministic selections of A_k^G at Step 2, since the atoms of any set A' eligible at iteration k are in any case assigned the same value α_k^G as if A' were the actual selection at iteration k.

Proposition 4 *Let A_k^G be the set selected by Algorithm G at iteration k. For any $A' \neq A_k^G, A' \not\subseteq B_{k-1}^G$ such that $\frac{\overline{P}(A') - P_G(A' \cap B_{k-1}^G)}{|A' \setminus B_{k-1}^G|} = \alpha_k^G$, it is $p_G(\omega) = \alpha_k^G$, $\forall \omega \in (A' \setminus B_{k-1}^G)$.*

Proof. Define $D' = A' \setminus B_{k-1}^G$ and $D = A_k^G \setminus B_{k-1}^G$. By Step 4 of the algorithm, $\forall \omega \in D$, $p_G(\omega) = \alpha_k^G$. Therefore the conclusion is immediate if $D' \subseteq D$. If not, $A' \not\subseteq B_k^G$, which means that A' is considered as a candidate A_{k+1}^G in the $(k+1)$-th main loop iteration. In particular, using $|A' \setminus B_k^G| = |D'| - |D \cap A'|$, (3), and the hypothesis it is: $\frac{\overline{P}(A') - P_G(A' \cap B_k^G)}{|A' \setminus B_k^G|} =$
$= \frac{\overline{P}(A') - P_G(A' \cap B_{k-1}^G) - \alpha_k^G |D \cap A'|}{|A' \setminus B_k^G|} =$
$= \frac{\alpha_k^G |D'| - \alpha_k^G |D \cap A'|}{|A' \setminus B_k^G|} = \alpha_k^G$.

Since $\alpha_{k+1}^G \geq \alpha_k^G$ (Proposition 3) and A' is a candidate for A_{k+1}^G, it is then necessarily $\alpha_{k+1}^G = \alpha_k^G$. Now, if A' is actually selected as A_{k+1}^G or turns out to be included in B_{k+1}^G, the conclusion follows. Otherwise we can iterate the same argument in the subsequent iterations. In summary, we obtain a sequence $1, \ldots, p$, $p \geq 1$, such that $A' \not\subseteq B_{k+i}^G$, $i < p$, $A' \subseteq B_{k+p}^G$, $\alpha_k^G = \alpha_{k+1}^G = \ldots = \alpha_{k+p}^G$. Hence $\forall \omega \in (D' \setminus D)$, $p_G(\omega) = \alpha_k^G$. □

The following proposition ensures the satisfaction of the consistency requirement.

Proposition 5 *Let P_G be the assignment on 2^Ω produced by applying algorithm G to $\overline{P}(\cdot)$. Then $\forall S \in 2^\Omega, P_G(S) \leq \overline{P}(S)$.*

Proof. There is an index k such that $S \not\subseteq B^G_{k-1}, S \subseteq B^G_k$. Recalling that $\alpha^G_k \leq \frac{\overline{P}(S)-P_G(S\cap B^G_{k-1})}{|S\setminus B^G_{k-1}|}$, we obtain (use (3) for the first equality) $\overline{P}(S) \geq \alpha^G_k |S\setminus B^G_{k-1}| + P_G(S\cap B^G_{k-1}) = P_G(S\setminus B^G_{k-1}) + P_G(S\cap B^G_{k-1}) = P_G(S)$. □

Proposition 5 shows that, unlike algorithm J, algorithm G always produces a consistent assignment on 2^Ω. There is however also a downside: the result of algorithm J is always a precise probability, while it may occur using algorithm G that $\sum_{\omega\in\Omega} p_G(\omega) < 1$.

Example 2 Let $\Omega = \{a,b,c,d,e,f\}$, and $\overline{P}$ be the coherent upper probability on 2^Ω which is the upper envelope of the two precise probabilities P_1, P_2, determined by orderly assigning the following values on the atoms a,b,c,d,e,f: P_1 - values $[0.59, 0.12, 0.01, 0.26, 0.01, 0.01]$, P_2 - values $[0.44, 0.21, 0.18, 0.02, 0.03, 0.12]$. At the first main loop iteration the set $\{e\}$ is selected as A^G_1. Then $\alpha^G_1 = \frac{\overline{P}(A^G_1)}{|A^G_1|} = p_G(e) = 0.03$ and $B^G_1 = A^G_1$. At the second iteration, the set $\{c,d,f\}$ is selected as A^G_2. Then (since $A^G_2 \cap B^G_1 = \emptyset$) $\alpha^G_2 = \frac{\overline{P}(\{c,d,f\})}{|\{c,d,f\}|} = 0.10\overline{6}$, and, therefore, $p_G(c) = p_G(d) = p_G(f) = \alpha^G_2$; $B^G_2 = \{c,d,e,f\}$. Third iteration: $A^G_3 = \{b,d,e,f\}$, $\alpha^G_3 = \frac{\overline{P}(\{b,d,e,f\})-P_G(\{d,e,f\})}{|\{b\}|} = 0.15\overline{6}$; $p_G(b) = \alpha^G_3$; $B^G_3 = \{b,c,d,e,f\}$. Fourth iteration: $A^G_4 = \{a,e,f\}$; $\alpha^G_4 = \frac{\overline{P}(\{a,e,f\})-P_G(\{e,f\})}{|\{a\}|} = 0.47\overline{3}$ and $p_G(a) = \alpha^G_3$. Since $B^G_4 = \Omega$, the algorithm terminates. Unfortunately, $\sum_{\omega\in\Omega} p_G(\omega) = 0.98$. □

We show now that algorithm G produces a precise probability if and only if Ω is selected (or is among the possible non deterministic choices) at the algorithm last iteration.

Proposition 6 *Let $A^G_1,\ldots,A^G_K$ be a sequence of sets obtained by applying algorithm G on $\overline{P}(\cdot)$, and $p_G(\cdot)$ the relevant distribution on Ω. Then $\sum_{\omega\in\Omega} p_G(\omega) = 1$ iff*

$$\alpha^G_K = \frac{\overline{P}(\Omega) - P_G(\Omega\cap B^G_{K-1})}{|\Omega\setminus B^G_{K-1}|}. \tag{5}$$

Proof. Of course $\sum_{\omega\in\Omega} p_G(\omega) = 1 \Leftrightarrow P_G(\Omega) = 1$. Let us first note that if (5) holds, we can assume $A^G_K = \Omega$ by Proposition 4. Then $P_G(\Omega) = \overline{P}(\Omega) = 1$, by (4). Conversely, assume $P_G(\Omega) = \overline{P}(\Omega) = 1$. Then (5) follows replacing $P_G(\Omega)$ with $\overline{P}(\Omega)$ into $P_G(\Omega) = \alpha^G_K|\Omega\setminus B^G_{K-1}| + P_G(\Omega\cap B^G_{K-1})$. □

In the frame of the empirical analysis described in Section 5, we verified on randomly generated imprecise probabilities that condition (5) is not particularly restrictive. The percentage of cases where it holds is quite high and slightly decreases with the cardinality of Ω: for instance, it is about 99.7 for $|\Omega| = 6$, 99.1 for $|\Omega| = 8$, and 98.4 for $|\Omega| = 10$.

As for the optimality of P_G, checking it can be reduced to checking the compatibility of a standard linear programming problem. Actually, P_G is the maximum entropy probability consistent with $\overline{P}$ iff $p_G(\omega_1),\ldots,p_G(\omega_N)$ solves the following constrained minimization problem for the opposite of the entropy function $H(P_G) = -\sum_{i=1}^N p_G(\omega_i) log_2 p_G(\omega_i)$:

$$\min \sum_{i=1}^N p_G(\omega_i) log_2 p_G(\omega_i)$$

$$\sum_{\omega_i\in S} p_G(\omega_i) \leq \overline{P}(S), \forall S \in 2^\Omega_*;$$

$$-p_G(\omega_i) \leq 0, \forall\omega_i\in\Omega; \sum_{i=1}^N p_G(\omega_i) = 1.$$

As also discussed in [5], thanks to the properties of the objective function, Karush-Kuhn-Tucker necessary and sufficient conditions (see [4] Chap. 4) ensure that $\hat{p_G}$ is an optimal solution iff there exist real values ρ_{ω_i} $(\omega_i\in\Omega)$, λ_S $(S\in 2^\Omega_*)$, and μ such that:

$$\rho_{\omega_i} \geq 0, \forall\omega_i\in\Omega; \lambda_S \geq 0, \forall S\in 2^\Omega_*; \tag{6}$$

$$-K(\omega_i) + \sum_{S:\omega_i\in S} \lambda_S + \mu - \rho_{\omega_i} = 0, \forall\omega_i\in\Omega; \tag{7}$$

$$\lambda_S(\sum_{S:\omega_i\in S} \hat{p_G}(\omega_i) - \overline{P}(S)) = 0, \forall S\in 2^\Omega_*; \tag{8}$$

$$\rho_{\omega_i}\hat{p_G}(\omega_i) = 0, \forall\omega_i\in\Omega; \tag{9}$$

where $K(\omega_i) = -1 - log_2\hat{p_G}(\omega_i)$.

By Proposition 3, $\hat{p_G}(\omega_i) > 0$, therefore (9) holds iff $\rho_{\omega_i} = 0, \forall i$. From (8), $\lambda_S = 0$, $\forall S : \hat{P_G}(S) < \overline{P}(S)$, i.e. λ_S may be non zero only for those S such that $\hat{P_G}(S) = \overline{P}(S)$. This equality holds at least for $S \in \mathcal{A} = \{A_1^G, \ldots, A_K^G\}$, by (4). Putting $\mathcal{A}^*(\supseteq \mathcal{A}) = \{S : \hat{P_G}(S) = \overline{P}(S)\}$, and $\lambda_S = 0$, $\forall S \notin \mathcal{A}^*$, conditions (6)-(9) simplify as follows:

$$\lambda_S \geq 0, \forall S \in \mathcal{A}^* \quad (10)$$

$$\sum_{S \in \mathcal{A}^* : \omega_i \in S} \lambda_S + \mu = K(\omega_i), \forall \omega_i \in \Omega \quad (11)$$

Checking the optimality of $\hat{p_G}(\cdot)$ amounts therefore to verifying whether the linear system (10)-(11) is compatible, a task for which well known procedures are available.

Finally, we carried out a complexity comparison between algorithms J and G. Due to space limitations, we only state without proof that the worst case orders of magnitude of the total number of operations are, respectively, $2 \cdot 2^{|\Omega|}$ (algorithm J) and $(|\Omega| - 1) \cdot 2^{|\Omega|}$ (algorithm G).

5 An empirical analysis

We have shown that algorithm J works correctly on a family of upper capacities $\mathcal{J}$, characterized by condition (2), which is a strict superset of the set $\mathcal{S}$ of submodular upper capacities and that algorithm G works on a still larger family $\mathcal{G} \supsetneq \mathcal{J}$, indirectly characterized by condition (5) and the linear system (10)-(11). One may now wonder whether this achievement may have some practical relevance. To answer this, two kinds of tests can be carried out: on one hand, it has to be checked whether the improvement with respect to algorithm J is significant, on the other hand, since maximum entropy can also be computed by resorting to nonlinear programming, the use of algorithm G has to be justified with respect to this alternative. Tests were focused on coherent upper probabilities [9], a very general family of uncertainty measures whose practical relevance is widely acknowledged. We wrote a set of Java classes whose features include the random generation of coherent upper probabilities, the implementation of algorithms J and G, and the capability to use the Java API of the Lindo [7] package for linear and nonlinear programming tasks.

To compare algorithms J and G, we wrote a program whose main loop randomly generates a coherent upper probability $\overline{P}$ and verifies whether $\overline{P}$ belongs to each of the sets $\mathcal{S}$,$\mathcal{J}$,$\mathcal{G}$.

The program was run for $|\Omega| = 4, \ldots, 10$ and generated 100000 imprecise probabilities for each cardinality[1]. Two main observations emerge from the results shown in Table 1.

Table 1: Percentage of coherent upper probabilities belonging to the families: $\mathcal{S}, \mathcal{J}, \mathcal{G}$

$\|\Omega\|$	$\mathcal{S}$	$\mathcal{J}$	$\mathcal{G}$
4	13.1	96.7	97.7
5	2.87	84.1	87.0
6	1.79	61.8	68.2
7	1.35	39.3	48.1
8	1.11	24.1	32.6
9	0.97	15.1	21.8
10	0.86	10.7	15.4

On one hand, while, as pointed out in [2], family $\mathcal{S}$ becomes very small for $|\Omega| \geq 5$, family $\mathcal{J}$ is definitely larger, though significantly decreasing with increasing cardinality. On the other hand, algorithm G provides the correct solution in a moderately but not negligibly larger number of cases than algorithm J.

Turning to the comparison with a state-of-the-art general nonlinear programming solver, algorithm G was first compared with Lindo NLP module in terms of computation times. Results of this comparison have to be taken very cautiously for two main reasons. First, due to the limitations of Lindo trial version, they could be carried out only when $|\Omega| = 4, \ldots, 7$. Second, the performances of the experimental Java implementation of algorithm G we used are strongly affected by the overhead of Java Virtual Machine, especially with increasing cardinality. Table 2 shows, for each cardinality, the average ratio AR between the computation times of Lindo and of algorithm G (including check of optimality of the computed solution). In every case algorithm G is several times faster than Lindo. Finally we evaluated the difference between the solutions produced by algorithm G, when they are non-optimal pre-

[1]Any coherent upper probability defined on 2^Ω with $|\Omega| \leq 3$ is submodular.

Table 2: Ratio between average computation time of Lindo and of algorithm G

$\|\Omega\|$	4	5	6	7
AR	12.8	8.89	6.46	4.58

cise probabilities, and Lindo's optimal ones, in order to verify whether algorithm G can be considered a suitable approximate algorithm when it does not produce the optimal solution. For $|\Omega| = 4, \ldots, 7$ we randomly generated 100000 cases where P_G is different from the optimal solution P^*. For each case i, we computed the difference between entropy values $DE_i = H(P_i^*) - H(P_{G_i})$ and the percent error $DE\%_i = (DE_i/H(P_i^*)) \cdot 100$. Table 3 displays the values $MDE = \max_i(DE_i)$, $ADE = \frac{\sum_i DE_i}{100000}$, $MDE\% = \max_i(DE\%_i)$, and $ADE\% = \frac{\sum_i DE\%_i}{100000}$. While, on the average, the approximation is quite good, the worst case error is relatively high.

Table 3: Evaluation of the non-optimal solutions produced by algorithm G

$\|\Omega\|$	MDE	ADE	MDE%	ADE%
4	0.082	0.010	10.2	0.69
5	0.185	0.0094	19.3	0.52
6	0.247	0.0093	14.0	0.45
7	0.261	0.0094	17.6	0.43

6 Conclusions

We provided a generalization of Jaffray's algorithm able to compute the maximum entropy probability on a family of capacities larger than the one where the original algorithm works correctly. We proved that whenever the solution P_J of algorithm J is correct, algorithm G returns the same solution $P_G = P_J$. In general, P_J is not consistent, but if it is so, it is also optimal. Algorithm G guarantees consistency, but P_G is not always a probability, in certain situations which we operationally characterized and which empirical evidence shows to be rather infrequent. Outside such situations, optimality of P_G can be checked via linear programming, and an empirical analysis shows that algorithm G is quicker than a standard non linear programming solver and often at least near-optimal. Among future research directions we mention devising algorithms for partially specified imprecise probabilities. A step in this direction is the algorithm proposed in [1] for *probability intervals*, i.e. imprecise probabilities assessed on elementary events only.

Acknowledgements

We thank the referees for their helpful comments.

References

[1] J. Abellan, S. Moral, Maximum of entropy for credal sets, *Int. J. of Uncertainty, Fuzziness and Knowledge-Based Systems*, **11** (2003) 587–597.

[2] P. Baroni and P. Vicig, Transformations from imprecise to precise probabilities, *Proc. of ECSQARU 2003,* (Aalborg, DK, 2003) 37–49.

[3] P. Baroni and P. Vicig, An uncertainty interchange format with imprecise probabilities, *Int. J. of Approximate Reasoning*, **40** (2005) 147–180.

[4] M. S. Bazaraa, H. D. Sherali, C. M. Shetty, *Nonlinear Programming*, 2nd ed., Wiley (New York, 1993).

[5] J.-Y. Jaffray, On the maximum-entropy probability which is consistent with a convex capacity, *Int. J. of Uncertainty, Fuzziness and Knowledge-Based Systems*, **3** (1995) 27–33.

[6] G. J. Klir, An update on generalized information theory, *Proc. of ISIPTA 03*, (Lugano, CH, 2003) 321–334.

[7] Lindo API 4.0 optimization engine, available at: http://www.lindo.com

[8] A. Meyerowitz, F. Richman, E. Walker, Calculating maximum-entropy probability densities for belief functions, *Int. J. of Uncertainty, Fuzziness and Knowledge-Based Systems*, **2** (1994) 377–389.

[9] P. Walley, *Statistical Reasoning with Imprecise Probabilities,* Chapman and Hall (London, 1991).