IMPROVING AND BENCHMARKING OF ALGORITHMS FOR DECISION MAKING WITH LOWER PREVISIONS

NAWAPON NAKHARUTAI, MATTHIAS C. M. TROFFAES, AND CAMILA C. S. CAIADO

Abstract. Maximality, interval dominance, and E-admissibility are three well-known criteria for decision making under severe uncertainty using lower previsions. We present a new fast algorithm for finding maximal gambles. We compare its performance to existing algorithms, one proposed by Troffaes and Hable (2014), and one by Jansen, Augustin, and Schollmeyer (2017). To do so, we develop a new method for generating random decision problems with pre-specified ratios of maximal and interval dominant gambles.

Based on earlier work, we present efficient ways to find common feasible starting points in these algorithms. We then exploit these feasible starting points to develop early stopping criteria for the primal-dual interior point method, further improving efficiency. We find that the primal-dual interior point method works best.

We also investigate the use of interval dominance to eliminate non-maximal gambles. This can make the problem smaller, and we observe that this benefits Jansen et al.'s algorithm, but perhaps surprisingly, not the other two algorithms. We find that our algorithm, without using interval dominance, outperforms all other algorithms in all scenarios in our benchmarking.

1. Introduction

Consider a subject who needs to choose from a set of possible decisions. Each decision leads to an uncertain reward, depending on her decision and on the state of nature revealed after the decision. The reward could be anything, for example, money, food, or a lottery ticket. For simplicity, we will assume that rewards are expressed on a utility scale. In this way, we can view an uncertain reward, and thereby, each decision, as a bounded real-valued function on the set of states of nature. Such function will be called a *gamble*.

The subject wants to choose gambles that lead to the best possible reward. We assume that, for any set of gambles, the subject can identify a subset of gambles that she does not want to choose. We say that a gamble is *optimal* in a given set of gambles if the subject is not committed to eliminate it.

If we assume that, under the usual rationality assumptions, the subject can specify a complete probability measure on the state of nature, then she should simply choose a gamble that maximises her expected utility [Anscombe and Aumann, 1963]. However, when only little information is available, the subject may not be able to specify a complete probability measure. In that case, the subject might consider other ways to express her beliefs. In this study, we will assume that the subject is able to model her beliefs using a *lower prevision*, or equivalently, through

Key words and phrases. decision; maximality; primal-dual; algorithm; benchmarking; imprecise probability.

probability bounding [Williams, 1975, 2007]. *Maximality* [Walley, 1991, §3.9.1-3.9.3, pp. 160-162] and *interval dominance* [Zaffalon et al., 2003, §2.3.3, pp.68-69] are well-known decision criteria induced by strict partial orders associated with lower previsions [Troffaes, 2007].

Several authors proposed algorithms for finding maximal gambles, for example, Kikuti et al. [2011], Troffaes and Hable [2014, p. 336] and Jansen et al. [2017].

Troffaes and Hable [2014, Algorithm 16.4, p. 336] present an incremental algorithm where once some maximal gambles in the sets are known, we should compare the remaining gambles against those maximal gambles first. Additionally, Troffaes and Hable [2014, p. 336] suggest that sorting all gambles in advance, e.g. by expectation, could help the algorithm to perform better. In this paper, we propose a new algorithm that incorporates this suggestion, and we confirm that this leads to a considerable speed-up.

In the algorithms proposed in Kikuti et al. [2011] and Troffaes and Hable [2014, p. 336], to check maximality of each gamble, one has to evaluate the sign of the lower prevision of several gambles. This can be done by solving several linear programming problems [Troffaes and Hable, 2014, p. 331]. From earlier work, we know that the primal-dual interior point method is particularly suitable when working with lower previsions [Nakharutai et al., 2018]. We propose early stopping criteria to determine more quickly the sign of lower previsions, exploiting the fact that primal-dual methods solve both the primal and dual simultaneously. We also use results from Nakharutai et al. [2018] to quickly obtain feasible starting points, further improving the efficiency of these methods.

Jansen et al. [2017] proposed an algorithm that can verify whether a gamble is maximal by solving a single larger linear program. To improve this algorithm, we exploit the fact that if a gamble is not maximal in a given iteration, then it can be excluded from all future iterations. We verify that this improved version performs slightly faster.

As all maximal gambles are also interval dominant, Troffaes [2007] suggested to eliminate non-maximal gambles by applying interval dominance first. When most gambles are not interval dominant, this can eliminate many non-maximal gambles early on. This might be useful, because interval dominance is easier to check. In this paper, we will compare the above mentioned algorithms for finding maximal gambles, with and without applying interval dominance.

The contributions of the paper are as follows. We propose a new algorithm for finding maximal gambles and compare its performance to the two algorithms proposed by Troffaes and Hable [2014, p. 336] and Jansen et al. [2017]. For benchmarking, we propose an algorithm for generating sets of gambles which have a precisely given number of maximal and interval dominant gambles. For the algorithm in Troffaes and Hable [2014, p. 336], and also for our new algorithm, we solve a sequence of linear programs by the primal-dual method, because we can easily find a common feasible starting point. For further improvement, we also develop early stopping criteria, so the method can stop iterating before achieving an optimal solution.

The paper is organised as follows. In section 2, we present the basic concepts used in the remained of the paper. First, we give a brief outline of lower previsions and natural extension. Then, we present several decision criteria for lower previsions. In section 3, we discuss several algorithms from the literature, and propose a new algorithm for finding maximal gambles. To benchmark these algorithms, in section 4, we provide an algorithm for generating sets of gambles, and compare the performance of various decision algorithms on generated sets. Proofs of technical results from this section are provided in A. Section 5 concludes the paper.

2. Decision making with lower previsions

In this section, we first explain lower previsions and natural extension. Then, we present three decision criteria: maximality, E-admissibility, and interval dominance. For more about the relation between these criteria and their advantages and disadvantages, we refer to Huntley et al. [2014] and Troffaes [2007].

2.1. Lower previsions.

Let Ω denote the set of states of nature, and let $\mathcal{L}$ denote the set of all gambles (i.e. bounded real-valued functions) on Ω. A *lower prevision* $\underline{P}$ is a real-valued function defined on some subset of $\mathcal{L}$. We denote the domain of $\underline{P}$ by $\operatorname{dom}\underline{P}$. Given a gamble $f \in \operatorname{dom}\underline{P}$, we interpret $\underline{P}(f)$ as the subject's supremum buying price for f, i.e. for all $\alpha < \underline{P}(f)$, she is willing to accept the gamble $f-\alpha$. It has been argued that this is a good way for the subject to model her uncertainty about Ω, especially under severe uncertainty [Walley, 1991, Miranda, 2008, Miranda and de Cooman, 2014, Troffaes and de Cooman, 2014].

We say $\underline{P}$ *avoids sure loss* if for all $n \in \mathbb{N}$, all $\lambda_1, \dots, \lambda_n \geq 0$, and all $f_1, \dots, f_n \in \operatorname{dom}\underline{P}$, it holds that [Troffaes and de Cooman, 2014, p. 42]:

$$\max_{\omega\in\Omega}\left(\sum_{i=1}^{n}\lambda_i\left[f_i(\omega)-\underline{P}(f_i)\right]\right)\geq 0. \tag{1}$$

If $\underline{P}$ does not avoid sure loss, then there is a finite combination of gambles $f_1, \dots, f_n \in \operatorname{dom}\underline{P}$ such that for some $\lambda_1, \dots, \lambda_n \geq 0$:

$$\max_{\omega\in\Omega}\left(\sum_{i=1}^{n}\lambda_i f_i(\omega)\right)<\sum_{i=1}^{n}\lambda_i\underline{P}(f_i) \tag{2}$$

which means that the subject is willing to pay more than what she could ever gain, which does not make sense [Troffaes and de Cooman, 2014, p. 44]. Therefore, throughout this study, we assume that all lower previsions avoid sure loss.

The conjugate upper prevision $\overline{P}$ on $-\operatorname{dom}\underline{P} \coloneqq \{-f : f \in \operatorname{dom}\underline{P}\}$ is defined by $\overline{P}(f) \coloneqq -\underline{P}(-f)$. It represents the subject's infimum selling price for f [Troffaes and de Cooman, 2014, p. 41].

We can extend $\underline{P}$ to the set of all gambles $\mathcal{L}$ via its *natural extension* $\underline{E}$. For any gamble g, the natural extension $\underline{E}(g)$ corresponds to the supremum price a subject should be willing to pay for g, given the prices $\underline{P}(f)$ for all $f \in \operatorname{dom}\underline{P}$ [Troffaes and de Cooman, 2014, p. 47].

Definition 1. *[Troffaes and de Cooman, 2014, p. 47] Let $\underline{P}$ be a lower prevision. The natural extension of $\underline{P}$ is defined on all $f \in \mathcal{L}$ by:*

$$\underline{E}(f)\coloneqq\sup\left\{\alpha\in\mathbb{R}\colon f-\alpha\geq\sum_{i=1}^{n}\lambda_i(f_i-\underline{P}(f_i)), n\in\mathbb{N},\, f_i\in\operatorname{dom}\underline{P},\, \lambda_i\geq 0\right\}. \tag{3}$$

Note that $\underline{E}$ is finite, and therefore, is a lower prevision, if an only if $\underline{P}$ avoids sure loss [Troffaes and de Cooman, 2014, p. 68]. In the case that both Ω and $\operatorname{dom}\underline{P}$

are finite, for any gamble f, $\underline{E}(f)$ can be calculated by solving a linear program [Troffaes and Hable, 2014, p. 331].

We denote the conjugate of $\underline{E}$ by $\overline{E}$ which is given by

$$\overline{E}(f) \coloneqq -\underline{E}(-f) \tag{4}$$

$$= \inf\left\{\beta \in \mathbb{R} \colon \beta - f \geq \sum_{i=1}^{n} \lambda_i (f_i - \underline{P}(f_i)), n \in \mathbb{N}, f_i \in \operatorname{dom}\underline{P}, \lambda_i \geq 0\right\}. \tag{5}$$

2.2. **Decision criteria.** We first define two strict partial orderings on $\mathcal{L}$, and then define optimality through maximality with respect to either of these two strict partial orderings.

Definition 2. *For any two gambles f and g, we say that $f \succ g$ whenever*

$$\underline{E}(f-g) > 0. \tag{6}$$

Note that Walley [1991, §3.8.1] uses a stronger ordering, which also includes pointwise dominance. Here, we follow Troffaes [2007], Troffaes and Hable [2014, §16.3.2] and Jansen et al. [2017] and simply omit pointwise dominance from our definition.

Definition 3. *[Huntley et al., 2014, p. 194] For any two gambles f and g, we say that $f \sqsupset g$ whenever*

$$\underline{E}(f) > \overline{E}(g). \tag{7}$$

Given any strict partial order on $\mathcal{L}$, we can define a notion of optimality through maximality with respect to that order:

Definition 4. *Let $>$ be a strict partial order on $\mathcal{L}$, and let $\mathcal{K}$ be a finite subset of $\mathcal{L}$. The* set of maximal gambles in $\mathcal{K}$ with respect to $>$ *is then defined by:*

$$\operatorname{opt}_{>}(\mathcal{K}) \coloneqq \{f \in \mathcal{K} \colon (\forall g \in \mathcal{K})(g \not> f)\}. \tag{8}$$

We call $\operatorname{opt}_{\succ}(\mathcal{K})$ the set of *maximal* gambles in $\mathcal{K}$ and $\operatorname{opt}_{\sqsupset}(\mathcal{K})$ the set of *interval dominant* gambles in $\mathcal{K}$.

Finally, we also need to define E-admissibility, which is yet another decision criterion. First, we need some notation. The unit simplex is the set of all probability mass functions:

$$\Delta \coloneqq \left\{p \in \mathbb{R}^{\Omega} \colon p \geq 0 \text{ and } \sum_{\omega \in \Omega} p(\omega) = 1\right\}. \tag{9}$$

The *credal set* of a lower prevision $\underline{P}$ is defined by [Miranda and de Cooman, 2014, p.37]:

$$\mathcal{M} \coloneqq \{p \in \Delta \colon \forall f \in \operatorname{dom}\underline{P},\ E_p(f) \geq \underline{P}(f)\}. \tag{10}$$

Definition 5 (E-admissibility)**.** *[Troffaes and Hable, 2014, p. 336] A gamble f is* E-admissible *in $\mathcal{K}$ if there is $p \in \mathcal{M}$ such that*

$$\forall g \in \mathcal{K} \colon E_p(f) \geq E_p(g). \tag{11}$$

The set of all E-admissible gambles in $\mathcal{K}$ is denoted by $\operatorname{opt}_{\mathcal{M}}(\mathcal{K})$.

Note that [Troffaes, 2007]:

$$\operatorname{opt}_{\mathcal{M}}(\mathcal{K}) \subseteq \operatorname{opt}_{\succ}(\mathcal{K}) \subseteq \operatorname{opt}_{\sqsupset}(\mathcal{K}). \tag{12}$$

If a gamble is not interval dominant, then it is not maximal. Consequently, if there are many gambles in the set, one may want to eliminate non-maximal gambles in $\mathcal{K}$ by applying interval dominance first [Troffaes, 2007].

Similarly, if we find an E-admissible gamble f in $\mathcal{K}$, then f is immediately maximal [Walley, 1991, §3.9.4]. In section 3, we will show how we can quickly find an E-admissible gamble in $\mathcal{K}$ to speed up algorithms for finding $\mathrm{opt}_{\succ}(\mathcal{K})$.

3. Improving algorithms for finding maximal gambles

3.1. **Algorithms for finding maximal gambles.** In this section, we will discuss algorithms for finding $\mathrm{opt}_{\succ}(\mathcal{K})$. We study two algorithms from the literature, and propose a new algorithm based on a suggestion from Troffaes and Hable [2014, p. 336].

One can see that a gamble f is maximal in $\mathcal{K}$ only if

$$\forall g \in \mathcal{K} : \overline{E}(f-g) \geq 0. \tag{13}$$

Suppose that there are m possible outcomes in Ω, k gambles in $\mathcal{K}$ and n gambles in $\operatorname{dom}\underline{P}$ where $\underline{P}$ avoids sure loss. To check whether f is maximal in $\mathcal{K}$, we have to calculate $\overline{E}(f-g)$ for all $g \in \mathcal{K} \setminus \{f\}$. Let $h := g - f$, we can calculate $\underline{E}(h)$ through either (P1) or (D1):

$$\text{(P1)} \quad \min \quad \sum_{\omega\in\Omega} h(\omega)p(\omega) \tag{P1a}$$

$$\text{subject to} \quad \forall g_i \in \operatorname{dom}\underline{P}\colon \sum_{\omega\in\Omega} (g_i(\omega) - \underline{P}(g_i))p(\omega) \geq 0 \tag{P1b}$$

$$\sum_{\omega\in\Omega} p(\omega) = 1 \tag{P1c}$$

$$\text{where} \quad \forall \omega\colon p(\omega) \geq 0, \tag{P1d}$$

$$\text{(D1)} \quad \max \quad \alpha \tag{D1a}$$

$$\text{subject to} \quad \forall \omega \in \Omega\colon \sum_{i=1}^{n} (g_i(\omega) - \underline{P}(g_i))\lambda_i + \alpha \leq h(\omega) \tag{D1b}$$

$$\text{where} \quad \forall i\colon \lambda_i \geq 0 \quad (\alpha \text{ free}). \tag{D1c}$$

$\underline{E}(h)$ is precisely the optimal value of (P1) (or (D1)). The problem (D1) is an unconditional case of the linear program in [Augustin et al., 2014, p. 331]. Note that (P1) has $n+1$ constraints and m variables. So, to determine all maximal gambles, we must solve (at most) $k(k-1)$ of these linear programs.

Troffaes and Hable [2014, algorithm 16.4, p. 336] proposed the following strategy for finding maximal gambles: once a non-maximal gamble is detected, it is no longer compared with other gambles. Indeed, if f is non-maximal, then there will be some gamble g that dominates f. However, if g dominates f, and f dominates h, then g will also dominates h as well. Consequently, every non-maximal gamble is dominated by at least one maximal gamble in $\mathcal{K}$. Therefore, the algorithm no longer needs to consider non-maximal gambles as soon as they are deemed non-maximal (see algorithm 1).

For algorithm 1, if the first considered gamble happens to be the only maximal gamble in $\mathcal{K}$, then the algorithm only needs to solve $2(k-1)$ linear programs

Algorithm 1 Find the set of maximal gambles in $\mathcal{K}$

```
Input: a set of $k$ gambles $\mathcal{K} = \{f_1, \dots, f_k\}$
Output: an index set of $\operatorname{opt}_{\succ}(\mathcal{K})$
 1: $I \leftarrow \emptyset$                                  ▷ an index set of $\operatorname{opt}_{\succ}(\mathcal{K})$
 2: for $i = 1 : k$ do
 3:     if IsNotDominated1$(I, i)$ then
 4:         $I \leftarrow I \cup \{i\}$                       ▷ $f_i$ is maximal
 5:     end if
 6: end for
 7: return $I$
 8: where IsNotDominated1$(I, i)$
 9:     for $j \in I \cup \{i+1, \dots, k\}$
10:         if $\underline{E}(f_j - f_i) > 0$ then return False    ▷ $f_i$ is dominated by $f_j$
11:     return True
```

[Troffaes and Hable, 2014, p. 336]. Specifically, to verify that none of gambles in the set dominate the first gamble, the algorithm first needs to solve $k-1$ linear programs. Next, for each of the remaining gambles, the algorithm compares it with the only existing maximal gamble, so the algorithm additionally solves $k-1$ linear programs. If all gambles in $\mathcal{K}$ are maximal, then the method needs to solve $k(k-1)$ linear programs, because for each gamble, the algorithm has to compare it with the other $k-1$ gambles [Troffaes and Hable, 2014, p. 335].

This shows that we could speed up algorithm 1 by early identifying some maximal gambles in $\mathcal{K}$, for example, via E-admissibility. Specifically, if a gamble f is E-admissible in $\mathcal{K}$, then f is also maximal in $\mathcal{K}$ [Huntley et al., 2014, p. 196]. We can simply find one of E-admissible gambles as follows. We first sort all gambles in $\mathcal{K}$ as $f_1, \dots, f_k$ such that for some $p \in \mathcal{M}$, for all $j > i$:

$$E_p(f_j) - E_p(f_i) \geq 0. \tag{14}$$

Then f_k has the highest expectation, and therefore f_k is E-admissible in $\mathcal{K}$. We can then improve algorithm 1 by initially setting $\operatorname{opt}_{\succ}(\mathcal{K}) = \{f_k\}$. To obtain such $p \in \mathcal{M}$, we can solve (P1) with $h = 0$. In principle, one could identify further maximal gambles by finding all E-admissible gambles, for example by using one of the algorithms proposed by Kikuti et al. [2005] or Utkin and Augustin [2005]. However, these algorithms require solving k linear programs where $k = |\mathcal{K}|$, and they are more complex than (P1) with $h = 0$. Therefore, we do not apply those algorithms here.

In addition to identifying one E-admissible gamble in $\mathcal{K}$, sorting gambles with respect to the expectation also saves a lot of comparison steps in algorithm 1 for finding $\operatorname{opt}_{\succ}(\mathcal{K})$. Specifically, to determine whether gamble f_i is maximal in $\mathcal{K}$, we need to evaluate only $\underline{E}(f_j - f_i)$ when $j > i$, because we immediately know that

$$\forall i \leq j \colon \underline{E}(f_i - f_j) \leq E_p(f_i - f_j) \leq 0. \tag{15}$$

An algorithm for finding maximal gambles that exploits sorting gambles is presented in algorithm 2.

Even though we have to do extra work to sort gambles at the beginning, we do not have to make as many comparisons in algorithm 2 as in algorithm 1. In particular, in the case that the set $\mathcal{K}$ has one maximal gamble, algorithm 2 only

Algorithm 2 Find the set of maximal gambles in $\mathcal{K}$

```
Input: a set of k gambles $\mathcal{K} = \{f_1, \dots, f_k\}$ such that for some $p \in \mathcal{M}$, we have
    that $E_p(f_1) \leq E_p(f_2) \leq \cdots \leq E_p(f_k)$.
Output: an index set of $\mathrm{opt}_{\succ}(\mathcal{K})$
 1: $I \leftarrow \{k\}$                                  ▷ an index set of $\mathrm{opt}_{\succ}(\mathcal{K})$
 2: for $i = 1 : k-1$ do
 3:     if IsNotDominated2($i$) then
 4:         $I \leftarrow I \cup \{i\}$                   ▷ $f_i$ is maximal
 5:     end if
 6: end for
 7: return $I$
 8: where IsNotDominated2($i$)
 9:     for $j \in \{k, k-1, \dots, i+1\}$
10:         if $\underline{E}(f_j - f_i) > 0$ then return False   ▷ $f_i$ is dominated by $f_j$
11:     return True
```

needs to solve $k-1$ linear programs. On the other hand, if all gambles are maximal, algorithm 2 needs to evaluate $\frac{k(k-1)}{2}$ linear programs. In both cases, this is only half of the number of comparisons of algorithm 1.

Instead of solving multiple linear programs, Jansen et al. [2017] suggest to solve just a single linear program per gamble in $\mathcal{K}$:

$$\text{(P0a)} \quad \text{(P0)} \quad \max \quad \sum_{j=1}^{k} \sum_{\omega \in \Omega} p_j(\omega)$$

$$\text{(P0b)} \quad \text{subject to} \quad \forall j = 1, \dots, k \colon \sum_{\omega \in \Omega} p_j(\omega) \leq 1$$

$$\text{(P0c)} \quad \forall j = 1, \dots, k, \ \forall g_i \in \operatorname{dom} \underline{P} \colon \sum_{\omega \in \Omega} (g_i(\omega) - \underline{P}(g_i)) p_j(\omega) \geq 0$$

$$\text{(P0d)} \quad \forall j = 1, \dots, k \colon \sum_{\omega \in \Omega} (f(\omega) - f_j(\omega))\, p_j(\omega) \geq 0$$

$$\text{(P0e)} \quad \text{where} \quad \forall j = 1, \dots, k, \ \forall \omega \colon p_j(\omega) \geq 0.$$

If the optimal value of (P0) is equal to k, then f is a maximal gamble in $\mathcal{K}$ [Jansen et al., 2017]. Therefore, to determine those k gambles, we solve only k linear programs (see algorithm 3). However, the size of linear program is much bigger as it has $k(3+n)$ constraints and mk variables.

Note that if we modify the constraint eq. (P0b) to the following equality:

$$\text{(P0b')} \quad \forall j = 1, \dots, k \colon \sum_{\omega \in \Omega} p_j(\omega) = 1,$$

then every feasible solution of (P0'):

$$\text{(16)} \quad \text{(P0')} \quad \max 0 \text{ subject to eqs. (P0b'), (P0c), (P0d) and (P0e),}$$

is also an optimal solution of (P0). Therefore, if (P0') has a feasible solution, then f is a maximal gamble in $\mathcal{K}$. In our simulation study, we solve (P0') because it is in a more suitable format for the primal-dual method, as it needs fewer artificial variables.

Algorithm 3 Find the set of maximal gambles in $\mathcal{K}$

```
Input: a set of $k$ gambles $\mathcal{K} = \{f_1, \dots, f_k\}$
Output: an index set of $\mathrm{opt}_{\succ}(\mathcal{K})$
1: $I \leftarrow \emptyset$                                   ▷ an index set of $\mathrm{opt}_{\succ}(\mathcal{K})$
2: for $i = 1\colon k$ do
3:     if (P0') with respect to $\mathcal{K}$ and $f_i$ has a feasible solution then
4:         $I \leftarrow I \cup \{i\}$                          ▷ $f_i$ is maximal
5:     end if
6: end for
7: return $I$
```

Remember that if a gamble is not maximal in a given iteration, then we can exclude it from all further iterations. We use this idea to improve algorithm 3 and present it in algorithm 4.

Algorithm 4 Find the set of maximal gambles in $\mathcal{K}$

```
Input: a set of $k$ gambles $\mathcal{K} = \{f_1, \dots, f_k\}$
Output: an index set of $\mathrm{opt}_{\succ}(\mathcal{K})$
1: $I \leftarrow \emptyset$                                   ▷ an index set of $\mathrm{opt}_{\succ}(\mathcal{K})$
2: for $i = 1\colon k$ do
3:     if IsNotDominated4$(I, i)$ then
4:             $I \leftarrow I \cup \{i\}$                      ▷ $f_i$ is maximal
5:     end if
6: end for
7: return $I$
8: where IsNotDominated4$(I, i)$
9:     $\mathcal{G} = \{f_j \in \mathcal{K}\colon j \in I \cup \{i+1, \dots, k\}\}$
10:    if (P0') with respect to $\mathcal{G}$ and $f_i$ has a feasible solution then
11:        return True
```

Algorithms 1, 2, 3 and 4 will be benchmarked later in section 4.

3.2. **Algorithm for finding interval dominant gambles.** As we mentioned before, every maximal gamble in $\mathcal{K}$ is also interval dominant. Therefore, before running each algorithm, we can eliminate some non-maximal gambles in $\mathcal{K}$ by finding $\mathrm{opt}_{\sqsupset}(\mathcal{K})$. To check whether a gamble f is interval dominant in $\mathcal{K}$, we first calculate $\max_{g\in\mathcal{K}} \underline{E}(g)$. Then f is interval dominant if

$$\overline{E}(f) \geq \max_{g\in\mathcal{K}} \underline{E}(g). \tag{17}$$

Overall, to handle k gambles, we have to solve $2k-1$ linear programs [Troffaes and Hable, 2014, p. 337]. This algorithm for finding interval dominant gambles in $\mathcal{K}$ is summarized in algorithm 5. So, in section 4, in addition to benchmarking those three algorithms for finding $\mathrm{opt}_{\succ}(\mathcal{K})$, we will also run algorithm 5 to helping those three algorithms to identify maximal gambles. Specifically, we will run algorithm 5 at the beginning to eliminate non-maximal gambles in $\mathcal{K}$, and then run those three algorithms on $\mathrm{opt}_{\sqsupset}(\mathcal{K})$.

Algorithm 5 Find the set of interval dominant gambles in $\mathcal{K}$

```
Input: a set of k gambles $\mathcal{K} = \{f_1, \dots, f_k\}$
Output: an index set of $\mathrm{opt}_{\sqsupset}(\mathcal{K})$
 1: for $j \in \{1, 2, \dots, k\}$ do
 2:     $e_j \leftarrow \underline{E}(f_j)$
 3: end for
 4: $\ell \leftarrow \arg\max_{j=1}^{k} e_j$
 5: $I \leftarrow \{\ell\}$                          ▷ an index set of $\mathrm{opt}_{\sqsupset}(\mathcal{K})$
 6: for $i \in \{1, 2, \dots, k\} \setminus \{\ell\}$ do
 7:     if $\overline{E}(f_i) \geq e_\ell$ then
 8:         $I \leftarrow I \cup \{i\}$               ▷ $f_i$ is interval dominant
 9:     end if
10: end for
11: return $I$
```

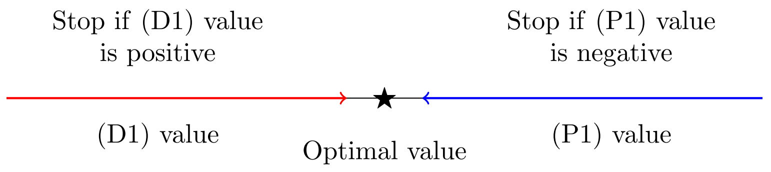

Stop if (D1) value is positive
Stop if (P1) value is negative
(D1) value
Optimal value
(P1) value

FIGURE 1. Early stopping criterion

3.3. **Fast evaluation of natural extensions inside algorithms.** We can also speed up the process of evaluating the natural extension through (P1) or (D1). To do so, we exploit the fact that we only need to find the sign of $\underline{E}(g - f)$, and not its exact value, to verify whether f is dominated by g or not.

As we minimize the objective function in (P1), the optimal value of (P1) is less or equal to other feasible objective values. So, we can stop as soon as we find a feasible solution that achieves a negative objective value, because then we know that the optimal value of (P1) is negative. In this case, f is not maximal in $\mathcal{K}$.

Similarly, as we maximize the objective function in (D1), the optimal value of (D1) is larger or equal to other feasible objective values. Consequently, we can stop as soon as we find a feasible solution that achieves a positive objective value. In this case, f is not dominated by g, so we must continue to compare f to other gambles in $\mathcal{K}$.

These extra stopping criteria are illustrated in fig. 1.

We can solve (P1) and (D1) by many linear programming methods, for example, the simplex, the affine scaling or the primal-dual interior point methods. However, we only focus on the primal-dual method as this method solves both primal and dual problems simultaneously, and can therefore exploit both stopping criteria simultaneously. We also know from earlier work that the primal-dual interior point method is particularly suitable for working with lower previsions [Nakharutai et al., 2018]. Finally, the primal-dual interior point method is widely regarded as one of the best general purpose linear programming methods [Griva et al., 2009, §10.2].

Note that, in practice, the primal-dual method can start with an arbitrary point and then generates a sequence of (not necessarily feasible) points that converges to an optimal feasible solution [Fang and Puthenpura, 1993, §7.3]. On the other hand, given initial feasible points, the method will generate a sequence of feasible points converging to an optimal solution [Fang and Puthenpura, 1993, §7.3]. Therefore, to apply the extra stopping criterion, we first have to find initial feasible points for (P1) and (D1). Fortunately, there is an efficient way to obtain initial feasible points for the linear programs (P1) and (D1).

For (P1), we can apply the first phase of the two-phase method to obtain an initial feasible probability mass function $p(\omega)$ [Nakharutai et al., 2018, §4.2]. This technique is usually used for obtaining interior feasible points for the affine scaling method [Fang and Puthenpura, 1993, §7.1.2]. Since the constraints of (P1) do not change, once we find a feasible starting point, we can reuse it for other problems (P1) with different objective functions. So we only need to do this once for any given lower prevision, and it is independent of the decision problem.

For (D1), we can very quickly calculate a feasible starting point without solving a linear program, using a result from Nakharutai et al. [2018, Theorem 7].

Unfortunately, there is no direct way to obtain feasible starting points for the linear programming problem (P0') (otherwise we would have immediately solved the problem).

4. Benchmarking

4.1. **Generating sets of gambles for benchmarking.**

As we mentioned before, we would like to generate a set of gambles $\mathcal{K}$ for benchmarking algorithms 1, 2, 3 and 4 for finding $\operatorname{opt}_{\succ}(\mathcal{K})$ and algorithm 5 for finding $\operatorname{opt}_{\sqsupset}(\mathcal{K})$. Can we generate a set $\mathcal{K}$ such that $|\mathcal{K}| = k$, $|\operatorname{opt}_{\succ}(\mathcal{K})| = m$ and $|\operatorname{opt}_{\sqsupset}(\mathcal{K})| = n$ where $m \leq n \leq k$?

A naive idea is to first generate $\mathcal{K} = \{g\}$, so obviously, $\operatorname{opt}_{\succ}(\mathcal{K}) = \operatorname{opt}_{\sqsupset}(\mathcal{K}) = \{g\}$. Next, we generate a gamble h such that

$$\operatorname{opt}_{\succ}(\mathcal{K} \cup \{h\}) = \operatorname{opt}_{\succ}(\mathcal{K}) \cup \{h\} \quad \& \quad \operatorname{opt}_{\sqsupset}(\mathcal{K} \cup \{h\}) = \operatorname{opt}_{\sqsupset}(\mathcal{K}) \cup \{h\}, \tag{18}$$

and we add h to $\mathcal{K}$. We repeat this process until we have $|\mathcal{K}| = |\operatorname{opt}_{\succ}(\mathcal{K})| = |\operatorname{opt}_{\sqsupset}(\mathcal{K})| = m$. After that, we generate a gamble h that satisfies

$$\operatorname{opt}_{\succ}(\mathcal{K} \cup \{h\}) = \operatorname{opt}_{\succ}(\mathcal{K}) \quad \& \quad \operatorname{opt}_{\sqsupset}(\mathcal{K} \cup \{h\}) = \operatorname{opt}_{\sqsupset}(\mathcal{K}) \cup \{h\}. \tag{19}$$

Again, we add h to $\mathcal{K}$ and repeat this process until we have $|\mathcal{K}| = |\operatorname{opt}_{\sqsupset}(\mathcal{K})| = n$. However, $|\operatorname{opt}_{\succ}(\mathcal{K})| = m$. Finally, we generate a gamble h such that

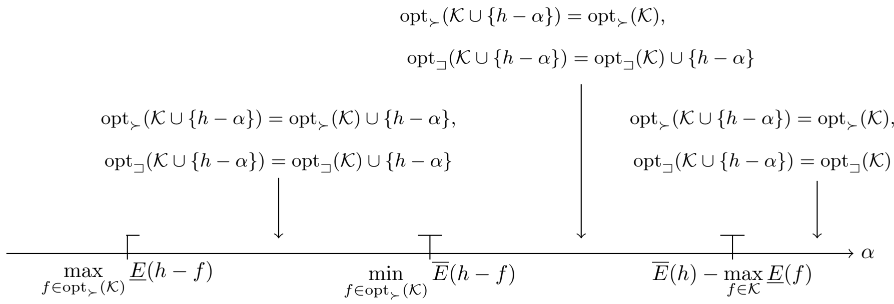
$$\operatorname{opt}_{\succ}(\mathcal{K} \cup \{h\}) = \operatorname{opt}_{\succ}(\mathcal{K}) \quad \& \quad \operatorname{opt}_{\sqsupset}(\mathcal{K} \cup \{h\}) = \operatorname{opt}_{\sqsupset}(\mathcal{K}), \tag{20}$$

we add h to $\mathcal{K}$, and repeat this process until we have $|\mathcal{K}| = k$, $|\operatorname{opt}_{\succ}(\mathcal{K})| = m$ and $|\operatorname{opt}_{\sqsupset}(\mathcal{K})| = n$ as we want.

In practice, a randomly generated gamble h may not easily satisfy eq. (18), eq. (19) or eq. (20). We may need to sample many gambles until we satisfy the desired conditions, and therefore it may take a while to obtain the set $\mathcal{K}$ that we want.

Surprisingly, for any generated gamble h, we can modify h by shifting it by α, for some $\alpha \in \mathbb{R}$, so that a new gamble $h - \alpha$ meets any of the above requirements. Next, we explain for what range of α, the gamble $h - \alpha$ satisfies either eq. (18), eq. (19) or eq. (20). Specifically, we identify for which values of α we have one of the following:

(i) $\operatorname{opt}_{\succ}(\mathcal{K} \cup \{h - \alpha\}) = \operatorname{opt}_{\succ}(\mathcal{K}) \cup \{h - \alpha\}$,
(ii) $\operatorname{opt}_{\succ}(\mathcal{K} \cup \{h-\alpha\}) = \operatorname{opt}_{\succ}(\mathcal{K})$ and $\operatorname{opt}_{\sqsupset}(\mathcal{K} \cup \{h-\alpha\}) = \operatorname{opt}_{\sqsupset}(\mathcal{K}) \cup \{h-\alpha\}$,
(iii) $\operatorname{opt}_{\sqsupset}(\mathcal{K} \cup \{h - \alpha\}) = \operatorname{opt}_{\sqsupset}(\mathcal{K})$



$\mathrm{opt}_{\succ}(\mathcal{K} \cup \{h-\alpha\}) = \mathrm{opt}_{\succ}(\mathcal{K})$,
$\mathrm{opt}_{\sqsupset}(\mathcal{K} \cup \{h-\alpha\}) = \mathrm{opt}_{\sqsupset}(\mathcal{K}) \cup \{h-\alpha\}$
$\mathrm{opt}_{\succ}(\mathcal{K} \cup \{h-\alpha\}) = \mathrm{opt}_{\succ}(\mathcal{K}) \cup \{h-\alpha\}$,
$\mathrm{opt}_{\sqsupset}(\mathcal{K} \cup \{h-\alpha\}) = \mathrm{opt}_{\sqsupset}(\mathcal{K}) \cup \{h-\alpha\}$
$\mathrm{opt}_{\succ}(\mathcal{K} \cup \{h-\alpha\}) = \mathrm{opt}_{\succ}(\mathcal{K})$,
$\mathrm{opt}_{\sqsupset}(\mathcal{K} \cup \{h-\alpha\}) = \mathrm{opt}_{\sqsupset}(\mathcal{K})$
α
$\max_{f \in \mathrm{opt}_{\succ}(\mathcal{K})} \underline{E}(h-f)$
$\min_{f \in \mathrm{opt}_{\succ}(\mathcal{K})} \overline{E}(h-f)$
$\overline{E}(h) - \max_{f \in \mathcal{K}} \underline{E}(f)$

FIGURE 2. Ranges for α such that $h-\alpha$ satisfies either of the situations described in theorem 1.

Let $\mathcal{K}$ be a set of gambles. Given any gamble h, lemma 1 shows for which α, $h-\alpha$ is a maximal gamble in $\mathcal{K} \cup \{h-\alpha\}$.

Lemma 1. *Let $\mathcal{K}$ be a set of gambles and let h be another gamble and $\alpha \in \mathbb{R}$. Then $h-\alpha$ is maximal in $\mathcal{K} \cup \{h-\alpha\}$ if and only if*

$$\min_{f \in \mathrm{opt}_{\succ}(\mathcal{K})} \overline{E}(h-f) \geq \alpha. \tag{21}$$

Lemma 1 provides an upper bound on α for which $h-\alpha$ is maximal in $\mathcal{K} \cup \{h-\alpha\}$. However, if we set α too low, then $h-\alpha$ may dominate other maximal gambles in $\mathrm{opt}_{\succ}(\mathcal{K})$, that is, we risk having gambles f for which $f \in \mathrm{opt}_{\succ}(\mathcal{K})$ but $f \notin \mathrm{opt}_{\succ}(\mathcal{K} \cup \{h-\alpha\})$. The following lemma tells us how to prevent this situation.

Lemma 2. *Let $\mathcal{K}$ be a set of gambles and let h be another gamble and $\alpha \in \mathbb{R}$. Then all maximal gambles in $\mathcal{K}$ are still maximal in $\mathcal{K} \cup \{h-\alpha\}$ if and only if*

$$\max_{f \in \mathrm{opt}_{\succ}(\mathcal{K})} \underline{E}(h-f) \leq \alpha. \tag{22}$$

Lemma 2 provides a lower bound on α such that $h-\alpha$ does not dominate any other maximal gambles in $\mathcal{K} \cup \{h-\alpha\}$.

Finally, by eq. (17), we know that $h-\alpha$ is interval dominant in $\mathcal{K} \cup \{h-\alpha\}$ if and only if

$$\overline{E}(h-\alpha) \geq \max_{f \in \mathcal{K}} \underline{E}(f). \tag{23}$$

This is equivalent to

$$\alpha \leq \overline{E}(h) - \max_{f \in \mathcal{K}} \underline{E}(f). \tag{24}$$

The next lemma ensures that these bounds on α are always ordered in the same way:

Lemma 3. *Let $\mathcal{K}$ be a set of gambles and let h be another gamble. Then, the following holds:*

$$\max_{f \in \mathrm{opt}_{\succ}(\mathcal{K})} \underline{E}(h-f) \leq \min_{f \in \mathrm{opt}_{\succ}(\mathcal{K})} \overline{E}(h-f) \leq \overline{E}(h) - \max_{f \in \mathcal{K}} \underline{E}(f). \tag{25}$$

Theorem 1 brings everything together, and summarises for which ranges of α we have that $h-\alpha$ satisfies either eq. (18), eq. (19) or eq. (20).

Theorem 1. *Let $\mathcal{K}$ be a set of gambles and let h be another gamble and $\alpha \in \mathbb{R}$.*

(1) *If we choose*

$$\max_{f\in\operatorname{opt}_{\succ}(\mathcal{K})} \underline{E}(h-f) \le \alpha \le \min_{f\in\operatorname{opt}_{\succ}(\mathcal{K})} \overline{E}(h-f) \tag{26}$$

then

$$\operatorname{opt}_{\succ}(\mathcal{K}\cup\{h-\alpha\}) = \operatorname{opt}_{\succ}(\mathcal{K})\cup\{h-\alpha\} \tag{27}$$

$$\operatorname{opt}_{\sqsupset}(\mathcal{K}\cup\{h-\alpha\}) = \operatorname{opt}_{\sqsupset}(\mathcal{K})\cup\{h-\alpha\}. \tag{28}$$

(2) *If we choose*

$$\min_{f\in\operatorname{opt}_{\succ}(\mathcal{K})} \overline{E}(h-f) < \alpha \le \overline{E}(h) - \max_{f\in\mathcal{K}} \underline{E}(f), \tag{29}$$

then

$$\operatorname{opt}_{\succ}(\mathcal{K}\cup\{h-\alpha\}) = \operatorname{opt}_{\succ}(\mathcal{K}) \tag{30}$$

$$\operatorname{opt}_{\sqsupset}(\mathcal{K}\cup\{h-\alpha\}) = \operatorname{opt}_{\sqsupset}(\mathcal{K})\cup\{h-\alpha\}. \tag{31}$$

(3) *If we choose*

$$\alpha > \overline{E}(h) - \max_{f\in\mathcal{K}} \underline{E}(f), \tag{32}$$

then

$$\operatorname{opt}_{\succ}(\mathcal{K}\cup\{h-\alpha\}) = \operatorname{opt}_{\succ}(\mathcal{K}) \tag{33}$$

$$\operatorname{opt}_{\sqsupset}(\mathcal{K}\cup\{h-\alpha\}) = \operatorname{opt}_{\sqsupset}(\mathcal{K}). \tag{34}$$

Figure 2 illustrates theorem 1.

From theorem 1, we obtain an algorithm for generating a set $\mathcal{K}$ of k gambles such that $|\operatorname{opt}_{\succ}(\mathcal{K})| = m$ and $|\operatorname{opt}_{\sqsupset}(\mathcal{K})| = n$ where $m \le n \le k$. First, we generate a set $\mathcal{K}$ consisting of m gambles that are all maximal, so $|\operatorname{opt}_{\succ}(\mathcal{K})| = |\operatorname{opt}_{\sqsupset}(\mathcal{K})| = |\mathcal{K}| = m$. Next, we add $n-m$ further gambles to $\mathcal{K}$, where these gambles are interval dominant but not maximal, so $|\operatorname{opt}_{\sqsupset}(\mathcal{K})| = |\mathcal{K}| = n$ but it remains that $|\operatorname{opt}_{\succ}(\mathcal{K})| = m$. Finally, we add $k-n$ further gambles to $\mathcal{K}$, where these gambles are not interval dominant. We then end up with a set of gambles which has a precisely given number of maximal and interval dominant gambles, as required. For the full algorithm, see algorithm 6.

Note that if α in the third loop is much larger than $\overline{E}(h) - \max_{f\in\mathcal{K}} \underline{E}(f)$, then $h-\alpha$ can be more easily dominated. Therefore, if we want these non-maximal gambles to be difficult to detect, then we should set α to be only slightly larger than $\overline{E}(h) - \max_{f\in\mathcal{K}} \underline{E}(f)$. In our simulation study, we will choose α in the third loop to be $\overline{E}(h) - \max_{f\in\mathcal{K}} \underline{E}(f) + \epsilon$ where ϵ is uniformly sampled from $(0,1)$.

Also note that we require the following condition for all i and j:

$$\overline{E}(h_i - h_j) < \overline{E}(h_i) - \underline{E}(h_j). \tag{35}$$

This ensures that there exists an α satisfying the strict inequality in eq. (29), because in that case, the left hand side of eq. (29) will be strictly less than the right hand side of eq. (29) (see eq. (60) in the appendix; the inequality there will be a strict inequality under the assumed condition). In this way, there is always an α for which $h-\alpha$ is not maximal but still interval dominant. Equation (35) requires that $\underline{E}$ is non-linear (i.e. genuinely imprecise), and that the gambles h_i are non-constant and linearly independent. For example, if for each ω, we sample $h_i(\omega)$ uniformly from $[0,1]$, then this requirement is practically always satisfied, and if not, we can simply resample h_i until it is.

Algorithm 6 Generate a set of k gambles $\mathcal{K}$ such that $|\operatorname{opt}_{\succ}(\mathcal{K})| = m$ and $|\operatorname{opt}_{\sqsupset}(\mathcal{K})| = n$ where $m \leq n \leq k$

```
Input: (a) Numbers $m$, $n$, $k$ where $m \leq n \leq k$, and (b) a sequence of $k$ gambles
    $h_1, \dots, h_k$ such that $\overline{E}(h_i - h_j) < \overline{E}(h_i) - \underline{E}(h_j)$ for all $i, j \in \{1, \dots, k\}$
Output: a set of $k$ gambles $\mathcal{K}$ such that such that $|\operatorname{opt}_{\succ}(\mathcal{K})| = m$ and $|\operatorname{opt}_{\sqsupset}(\mathcal{K})| =$
    $n$ where $m \leq n \leq k$
1: $\mathcal{K} \leftarrow \{h_1\}$
2: for $i = 2 : m$ do                                      ▷ $m$ maximal and interval dominant
3:     Choose $\alpha$ such that $\max_{f \in \operatorname{opt}_{\succ}(\mathcal{K})} \underline{E}(h_i - f) \leq \alpha \leq \min_{f \in \operatorname{opt}_{\succ}(\mathcal{K})} \overline{E}(h_i - f)$
4:     $\mathcal{K} \leftarrow \mathcal{K} \cup \{h_i - \alpha\}$
5: end for
6: for $i = m + 1 : n$ do                              ▷ $n - m$ interval dominant but not maximal
7:     Choose $\alpha$ such that $\min_{f \in \operatorname{opt}_{\succ}(\mathcal{K})} \overline{E}(h_i - f) < \alpha \leq \overline{E}(h_i) - \max_{f \in \mathcal{K}} \underline{E}(f)$
8:     $\mathcal{K} \leftarrow \mathcal{K} \cup \{h_i - \alpha\}$
9: end for
10: for $i = n + 1 : k$ do                                          ▷ $k - n$ not interval dominant
11:     Choose $\alpha$ such that $\alpha > \overline{E}(h_i) - \max_{f \in \mathcal{K}} \underline{E}(f)$
12:     $\mathcal{K} \leftarrow \mathcal{K} \cup \{h_i - \alpha\}$
13: end for
14: return $\mathcal{K}$
```

Also note that algorithm 6 has to evaluate many natural extensions for each new generated gamble. This is required to ensure that the numbers of maximal gambles and interval dominant gambles in the sets are exactly as specified. Although this gives us very precise control over the range of decision problems that are generated, an obvious downside is that evaluating all these natural extensions requires a huge computational effort. For this reason, we had to limit ourselves to $|\Omega| \leq 2^6$ and $|\mathcal{K}| \leq 2^8$.

4.2. **Benchmarking results.** To benchmark those algorithms 1, 2, 3, 4 and 5 from section 3, in this section, we generate random sets of gambles. We consider the case that $|\Omega| = 2^2$ and $|\Omega| = 2^6$ and the number of gambles in $\mathcal{K}$ where $k = 2^j$ for $j \in \{4, 6, 8\}$. For each case, random sets of gambles $\mathcal{K}$ are generated as follows.

(1) We first generate a lower prevision $\underline{P}$ on a finite domain, that avoids sure loss. To do so, we use [Nakharutai et al., 2018, algorithm 2] with 2^4 coherent previsions to generate a lower prevision $\underline{E}$ on the set of all gambles, that avoids sure loss. Next, we use [Nakharutai et al., 2018, stages 1 and 2 in algorithm 4] to restrict $\underline{E}$ to a lower prevision $\underline{P}$ that avoids sure loss, with a given finite size of domain. In this simulation we consider $|\operatorname{dom} \underline{P}| = 2^i$ for $i \in \{2, 4, 6\}$. This ensures that the generated lower prevision has no specific structural properties (such as 2-monotonicity [Troffaes and de Cooman, 2014, Chapter 6]).
(2) We generate k gambles $h_1, \dots, h_k$ as follows. For each ω and i, we sample $h_i(\omega)$ uniformly from $[0, 1]$ and check whether they satisfy eq. (35).

(3) We use algorithm 6 to generate random sets $\mathcal{K}$ such that $|\mathcal{K}| = k$, $|\operatorname{opt}_{\succ}(\mathcal{K})| = m$ and $|\operatorname{opt}_{\sqsupset}(\mathcal{K})| = n$ where $m \leq n \leq k$, where we use the previously generated $\underline{P}$ to evaluate $\underline{E}$ and $\overline{E}$. Note that in algorithm 6, we choose α in the first loop as follows: we sample δ uniformly from $(0,1)$, and set

$$\alpha \coloneqq \delta \max_{f \in \operatorname{opt}_{\succ}(\mathcal{K})} \underline{E}(h_i - f) + (1-\delta) \min_{f \in \operatorname{opt}_{\succ}(\mathcal{K})} \overline{E}(h_i - f). \tag{36}$$

For α in the second loop, we choose it as follow: sample δ uniformly from $(0,1)$, and set

$$\alpha \coloneqq \delta \min_{f \in \operatorname{opt}_{\succ}(\mathcal{K})} \overline{E}(h_i - f) + (1-\delta) \left(\overline{E}(h_i) - \max_{f \in \mathcal{K}} \underline{E}(f) \right), \tag{37}$$

and in the last loop, we set $\alpha \coloneqq \overline{E}(h_i) - \max_{f \in \mathcal{K}} \underline{E}(f) + \epsilon$, where we sample ϵ uniformly from $(0,1)$.

In the simulation, we would like to cover a range of possible options of m, n, and k that satisfy $m \leq n \leq k$. For each different size of $\mathcal{K}$, we consider 10 options that vary the number of maximal gambles m and the number of interval dominant gambles n in $\mathcal{K}$ which are illustrated in fig. 3 and table 1.

These 10 options can be grouped as follows. Options a to d represent the cases where $m = 1$ while we increase n from 1 to k. Options d, g, i, and j represent the cases where $n = k$ while we increase m from 1 to k. Options a, e, h, and j represent the cases where $m = n$ while we increase them jointly from 1 to k. Option f represents a case where $m < n < k$.

We then apply algorithms 1, 2, 3, 4 and 5 on each generated set of gambles $\mathcal{K}$. For algorithms 1, 2 and 5, we solve linear programs for evaluating upper and lower natural extensions by the improved primal-dual interior point method, including all of the improvements discussed in section 3.3, i.e. feasible starting points and early stopping criteria. As the MATLAB implementation of the primal-dual method cannot be easily modified, we had to write our own implementation of the improved primal-dual interior point method in MATLAB (R2018a) [MATLAB, 2018], based on the implementation used in Nakharutai et al. [2018]. For algorithms 3 and 4, we simply solve linear programs by the standard primal-dual method as the improvements cannot be applied. For fair comparison, we also used our own implementation of the standard primal-dual method here, rather than using the one that is available in MATLAB (R2018a).

To investigate whether interval dominance is helpful for finding maximal gambles, we also run each algorithm with and without algorithm 5. Specifically, we run algorithms 1, 2, 3 and 4 as such, but additionally we also run algorithm 5 to obtain a set of interval dominant gambles, and then again run each of algorithms 1, 2, 3 and 4 on just the resulting set of interval dominant gambles. In all cases, we measure the total computational time taken, i.e. including the time taken on algorithm 5 when applicable. Note that computational time to run algorithm 2 includes the time for sorting gambles from the lowest to the highest expectation as in eq. (14). To do so, we used *quicksort* which is available in MATLAB (R2018a) [MATLAB, 2018]. We repeat this process 100 times. Figure 4 summarizes the results.

Figure 4 shows the average computational time taken during each algorithm with and without algorithm 5. The average computational time taken for only

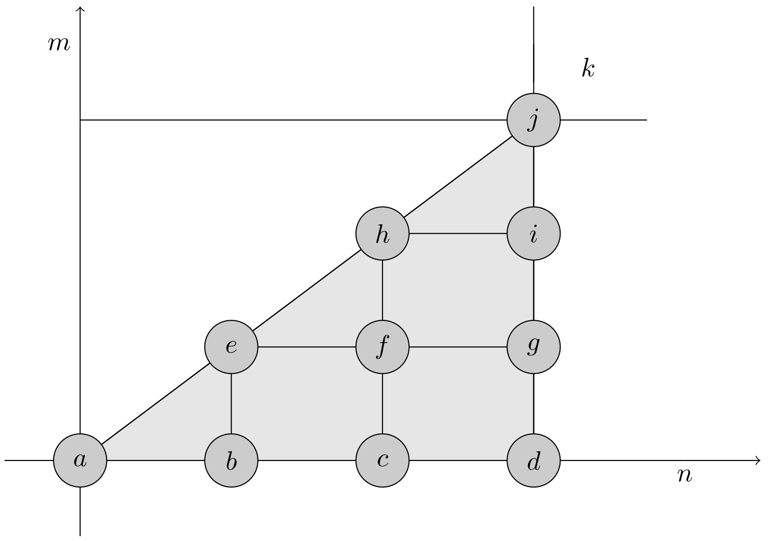


FIGURE 3. The area of $m \leq n \leq k$ and 10 options label the different m and n that we consider in the simulation (see table 1)

Options	$\lvert\mathcal{K}\rvert = 2^4$		$\lvert\mathcal{K}\rvert = 2^6$		$\lvert\mathcal{K}\rvert = 2^8$	
	m	n	m	n	m	n
a	1	1	1	1	1	1
b	1	5	1	21	1	85
c	1	11	1	42	1	170
d	1	16	1	64	1	256
e	5	5	21	21	85	85
f	5	11	21	42	85	170
g	5	16	21	64	85	256
h	11	11	42	42	170	170
i	11	16	42	64	170	256
j	16	16	64	64	256	256

TABLE 1. Table of points that indicate different sizes of set $\mathcal{K}$ with vary the number of maximal gambles m and the number of interval dominant gambles n in $\mathcal{K}$

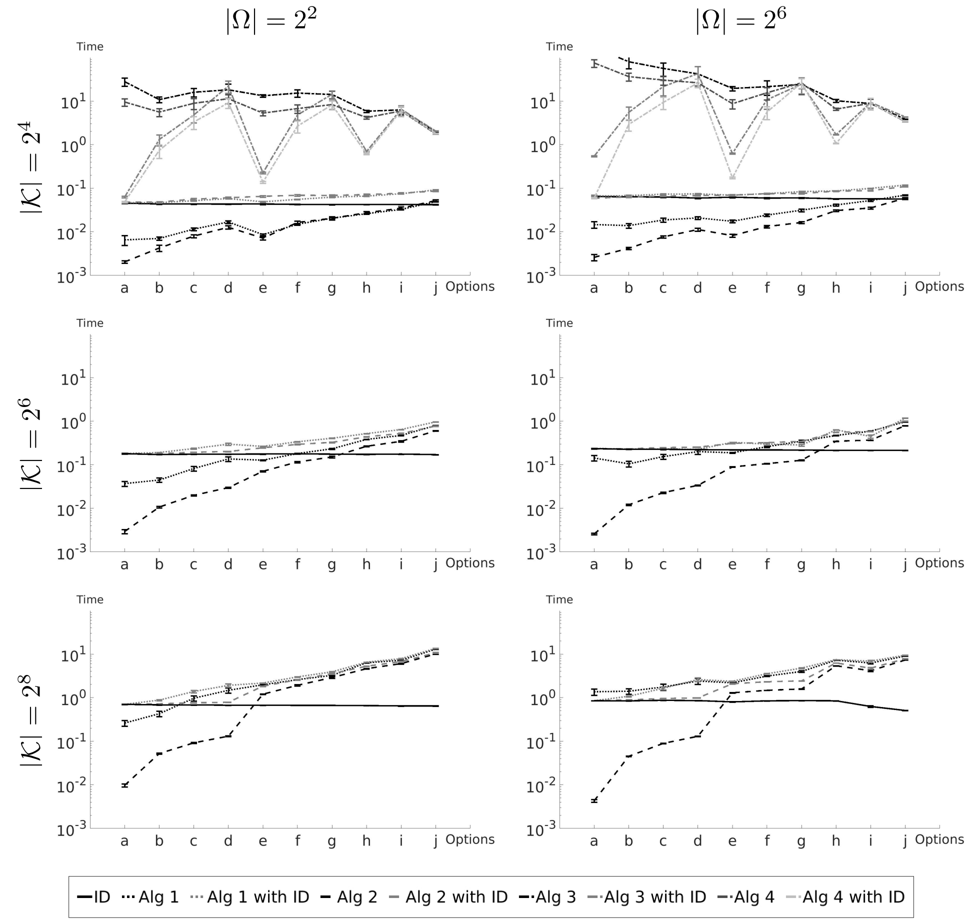


Figure 4. Comparison plots of the average computational time for algorithms 1, 2, 3 and 4 for finding maximal gambles and for algorithm 5 for finding interval dominant gambles. The number of outcomes in left column is 2^2 and 2^6 in the right column. Each row represents a different number of gambles with vary options of the numbers of maximal gambles and interval dominant gambles in the set (see table 1 for each option). The labels indicate algorithms with and without algorithm 5. We fix $|\operatorname{dom} \underline{P}| = 2^4$.

algorithm 5 is also presented there. In the top plots, we show the average computational time for algorithms 1, 2, 3 and 4. In the remaining plots, the average computational time for algorithms 3 and 4 are so high that their performance are completely dominated by the performance of the other algorithms, and is therefore not presented in these plots. In the left column, the number of outcomes is 2^2 and in the right column, it is 2^6. Each row represents a different size of $\mathcal{K}$.

We also consider an impact of the size of $\operatorname{dom} \underline{P}$. Figure 5 shows the average computational time taken for algorithms 1 and 2 with and without algorithm 5. In the left column, the number of gamble in the domain of $\underline{P}$ is 2^2 and in the right column, it is 2^6. Each row represents different numbers of outcomes and gambles.

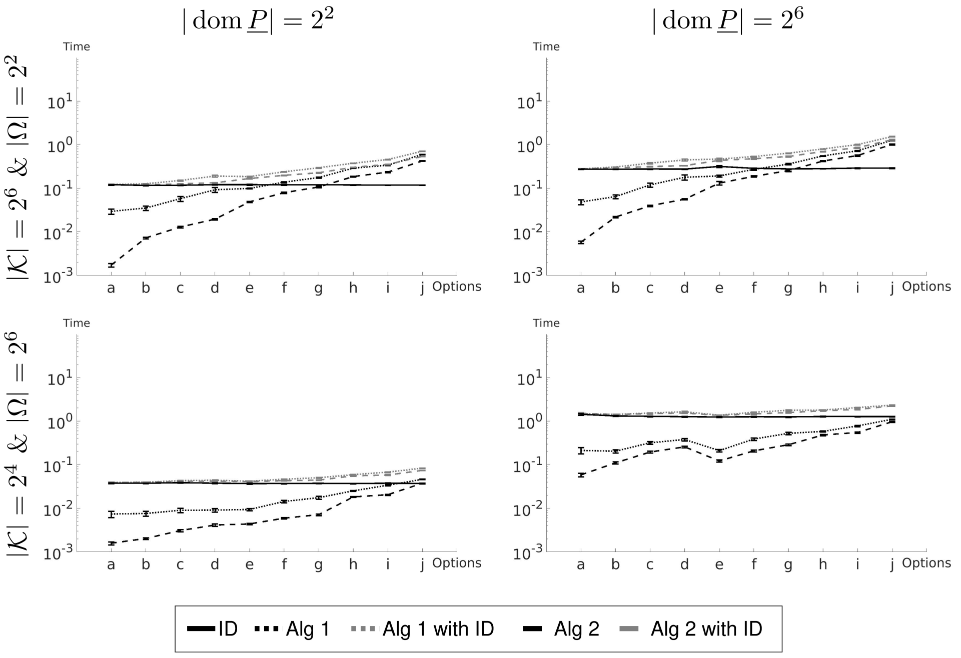


FIGURE 5. Comparison plots of the average computational time for algorithms 1 and 2 for finding maximal gambles and algorithm 5 for finding interval dominant gambles. In the left column, $|\operatorname{dom} \underline{P}| = 2^2$ and in the right column, $|\operatorname{dom} \underline{P}| = 2^6$. Each row represents different numbers of gambles and outcomes with vary options of the numbers of maximal gambles and interval dominant gambles in the set (see table 1 for each option). The labels indicate algorithms with and without algorithm 5.

In both figs. 4 and 5, the horizontal axis indicates different options of m, n, and k that we consider. The vertical axis presents the computational time which is averaged over 100 random generated sets of gambles. The error bars on the figure represent approximate 95% confidence intervals on the mean computational time.

We also solved (P0) by the simplex method in algorithm 3, using the default simplex method available in MATLAB (R2018a) [MATLAB, 2018]. However, this was still slower than algorithms 1 and 2. As there is no change in general conclusion, we do not show those plots here.

5. Discussion and conclusion

In this work, we proposed a new algorithm (algorithm 2) for finding maximal gambles and compared its performance with Troffaes and Hable [2014, p. 336]'s algorithms (algorithm 1) and Jansen et al. [2017] (algorithm 3). We further improved algorithm 3 by applying the fact that if a gamble is not maximal in one iteration, then it can be excluded from all subsequent iterations (algorithm 4). We also studied the impact of using interval dominance in algorithm 5 to eliminate non-maximal gambles as this can reduce the size of the problem.

To find the set of all maximal gambles, Jansen et al. [2017]'s algorithm solves a single large linear program for each gamble, while Troffaes and Hable [2014, p. 336]'s algorithm and our new algorithm solves a larger sequence of smaller linear programs. For the second case, we proposed early stopping criteria. We also applied common feasible starting points for the entire sequence of problems, based on earlier work [Nakharutai et al., 2018]. We found that the primal-dual method can exploit these improvements most effectively, and performs best overall.

To benchmark these algorithms, we presented a new algorithm for generating random sets of gambles with a pre-determined proportion of maximal and interval dominant gambles. This algorithm will be useful for others who want to test their algorithms. Whilst our benchmarking approach allows careful control over the properties of the generated decision problems (in particular, the fraction of optimal gambles according to different decision criteria), it does have severe computational limitations, due to the need to evaluate large numbers of natural extensions. Nevertheless, we hope this work provides a good starting point, and we hope that it inspires the development of further benchmarking frameworks for testing algorithms for decision making.

We compared computational performance of algorithms 1, 2, 3 and 4 with and without algorithm 5 on these generated sets. According to our numerical results, the relative performance of algorithms 1, 2, 3 and 4 depends on (i) the numbers of outcomes and gambles in the sets, (ii) the ratios of maximal and interval dominant gambles, and (iii) the number of gambles in the domain of lower previsions. If one of these numbers is increasing, then, generally, the average computational time taken on the algorithm is longer. In contrast, the average computational time taken on algorithm 5 does not depend on the ratios of maximal and interval dominant gambles, but it depends on the numbers of outcomes and gambles in the set and the number of gambles in the domain of lower previsions. This is because algorithm 5 has to evaluate the same number of natural extensions regardless of the structure of the problem.

We observed that applying interval dominance (algorithm 5) at the beginning benefits algorithms 3 and 4 as it makes the linear program smaller, especially if there are many non-interval dominant gambles. Therefore, when using algorithms 3 and 4, we would strongly suggest to run algorithm 5 first. We found that algorithm 4 slightly outperforms algorithm 3.

In contrast, perhaps surprisingly, interval dominance (algorithm 5), at least with our implementation of it, does not help algorithms 1 and 2. Even though using algorithm 5 can eliminate some non-maximal gambles, applying algorithm 5 first and then performing algorithm 1 or algorithm 2 is still slower than performing only algorithm 1 or algorithm 2. Therefore, we do not recommend applying algorithm 5 before algorithm 1 or algorithm 2. This said, there may still be ways to speed up the algorithm for interval dominance, for example, by adding more stopping criteria.

Overall, both algorithms 1 and 2 outperform algorithms 3 and 4 by an order of magnitude. Algorithm 2 also outperforms algorithm 1 in all cases in the experiment, especially when there is only one maximal gamble in the set. In the case that the number of maximal gambles in the set are increasing, there is no big difference in the time taken on algorithms 1 and 2, but algorithm 2 still slightly outperforms algorithm 1. When we vary the number of gambles in the domain of lower previsions, the conclusion does not change, i.e., algorithm 2 still outperforms algorithm 1.

Based on theoretical considerations, our newly proposed algorithm, algorithm 2, is a good choice for implementations as it reduces the number of linear programs, as well as the number of iterations. Our benchmarking study quantified these improvements, and we found that it outperformed all other algorithms tested over all scenarios considered.

Acknowledgements

We would like to acknowledge support for this project from Development and Promotion of Science and Technology Talents Project (Royal Government of Thailand scholarship).

References

F. J. Anscombe and R. J. Aumann. A definition of subjective probability. *Annals of Mathematical Statistics*, 34(1):199–205, March 1963.

Thomas Augustin, Frank P. A. Coolen, Gert De Cooman, and Matthias C. M. Troffaes, editors. *Introduction to Imprecise Probabilities.* Wiley Series in Probability and Statistics. Wiley, 2014. ISBN 978-0-470-97381-3. URL `http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470973811.html`.

Shu-Cherng Fang and Sarat Puthenpura. *Linear Optimization and Extensions: Theory and Algorithms.* Springer Science+Business Media New York, 1993.

Igor Griva, Stephen G. Nash, and Ariela Sofer. *Linear and Nonlinear Optimization Second edition.* SIAM, Philadelphia, 2009.

Nathan Huntley, Robert Hable, and Matthias C. M. Troffaes. *Introduction to Imprecise Probabilities*, chapter Decision making, pages 190–206. Wiley, 2014. doi:10.1002/9781118763117.ch8.

Christoph Jansen, Thomas Augustin, and Georg Schollmeyer. Decision theory meets linear optimization beyond computation. In Alessandro Antonucci, Laurence Cholvy, and Odile Papini, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 329–339, Cham, 2017. Springer International Publishing. ISBN 978-3-319-61581-3.

Daniel Kikuti, Fabio G. Cozman, and Cassio P. De Campos. Partially ordered preferences in decision trees: computing strategies with imprecision in probabilities. In *In IJCAI Workshop on Advances in Preference Handling*, pages 118–123, 2005.

Daniel Kikuti, Fabio Gagliardi Cozman, and Ricardo Shirota Filho. Sequential decision making with partially ordered preferences. *Artificial Intelligence*, 175(7):1346 – 1365, 2011. ISSN 0004-3702. doi:https://doi.org/10.1016/j.artint.2010.11.017. URL `http://www.sciencedirect.com/science/article/pii/S0004370210002067`. Representing, Processing, and Learning Preferences: Theoretical and Practical Challenges.

MATLAB. *version 9.4.0.813654 (R2018a).* The MathWorks Inc., Natick, Massachusetts, 2018.

Enrique Miranda. A survey of the theory of coherent lower previsions. *International Journal of Approximate Reasoning*, 48(2):628–658, 2008. doi:10.1016/j.ijar.2007.12.001.

Enrique Miranda and Gert de Cooman. *Introduction to Imprecise Probabilities*, chapter Lower prevision, pages 28–55. Wiley, 2014. doi:10.1002/9781118763117.ch2.

N. Nakharutai, M. C. M. Troffaes, and C. C. S. Caiado. Improved linear programming methods for checking avoiding sure loss. *International Journal of Approximate Reasoning*, 101:293–310, 2018. doi:10.1016/j.ijar.2018.07.013.

Matthias C. M. Troffaes. Decision making under uncertainty using imprecise probabilities. *International Journal of Approximate Reasoning*, 45(1):17–29, may 2007. doi:10.1016/j.ijar.2006.06.001.

Matthias C. M. Troffaes and Gert de Cooman. *Lower Previsions*. Wiley Series in Probability and Statistics. Wiley, 2014. ISBN 978-0-470-72377-7. URL `http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470723777.html`.

Matthias C. M. Troffaes and Robert Hable. *Introduction to Imprecise Probabilities*, chapter Computation, pages 329–337. Wiley, 2014. doi:10.1002/9781118763117.ch16.

Lev V. Utkin and Thomas Augustin. Powerful algorithms for decision making under partial prior information and general ambiguity attitudes. In *ISIPTA*, 2005.

Peter Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, London, 1991.

Peter M. Williams. Notes on conditional previsions. Technical report, School of Math. and Phys. Sci., Univ. of Sussex, 1975.

Peter M. Williams. Notes on conditional previsions. *International Journal of Approximate Reasoning*, 44(3):366–383, 2007. doi:10.1016/j.ijar.2006.07.019.

Marco Zaffalon, Keith Wesnes, and Orlando Petrini. Reliable diagnoses of dementia by the naive credal classifier inferred from incomplete cognitive data. *Artificial Intelligence in Medicine*, 29(1–2):61–79, 2003.

Appendix A. Proof of technical results in section 4

Proof of Lemma 1. By the definition, we have

$$h-\alpha \in \operatorname{opt}_{\succ}(\mathcal{K}\cup\{h-\alpha\}) \Leftrightarrow \forall g\in\mathcal{K}\colon \overline{E}(h-g-\alpha)\geq 0, \tag{38}$$

$$\Leftrightarrow \forall g\in\mathcal{K}\colon \overline{E}(h-g)\geq \alpha, \tag{39}$$

$$\Leftrightarrow \min_{g\in\mathcal{K}} \overline{E}(h-g)\geq \alpha. \tag{40}$$

Note that

$$\overline{E}(h-g)\geq \overline{E}(h-f)-\overline{E}(g-f). \tag{41}$$

Suppose that if $g\notin \operatorname{opt}_{\succ}(\mathcal{K})$, then we have $\overline{E}(g-f)<0$ for some $f\in\operatorname{opt}_{\succ}(\mathcal{K})$, as g is dominated by at least one maximal gamble in $\mathcal{K}$ [Troffaes and Hable, 2014, p. 336]. Therefore, for all $g\notin\operatorname{opt}_{\succ}(\mathcal{K})$:

$$\exists f\in\operatorname{opt}_{\succ}(\mathcal{K}),\ \overline{E}(h-g)\geq\overline{E}(h-f). \tag{42}$$

Consequently,

$$h-\alpha\in\operatorname{opt}_{\succ}(\mathcal{K}\cup\{h-\alpha\}) \Leftrightarrow \min_{f\in\operatorname{opt}_{\succ}(\mathcal{K})}\overline{E}(h-f)\geq\alpha. \tag{43}$$

□

Proof of lemma 2. Let f be a maximal gamble in $\mathcal{K}$. We first show that f is a maximal gamble in $\mathcal{K} \cup \{h - \alpha\}$ if and only if $\underline{E}(h - f) \leq \alpha$. We see that

$$f \in \mathrm{opt}_{\succ}(\mathcal{K} \cup \{h - \alpha\}) \Leftrightarrow \forall g \in \mathcal{K} \cup \{h - \alpha\} \colon \overline{E}(f - g) \geq 0 \tag{44}$$

and because $\overline{E}(f - g) \geq 0$ for all $g \in \mathcal{K}$,

$$\Leftrightarrow \overline{E}(f - h + \alpha) \geq 0 \tag{45}$$

$$\Leftrightarrow \overline{E}(f - h) \geq -\alpha \tag{46}$$

$$\Leftrightarrow \underline{E}(h - f) \leq \alpha. \tag{47}$$

Consequently, all maximal gambles in $\mathcal{K}$ are still maximal in $\mathcal{K} \cup \{h - \alpha\}$ if and only if

$$\max_{f \in \mathrm{opt}_{\succ}(\mathcal{K})} \underline{E}(h - f) \leq \alpha. \tag{48}$$

□

Proof of Lemma 3. We first show that

$$\forall f, g \in \mathrm{opt}_{\succ}(\mathcal{K}) \colon \underline{E}(h - f) \leq \overline{E}(h - g). \tag{49}$$

holds. Let $\mathcal{K}$ be a set of gambles and let $\mathrm{opt}_{\succ}(\mathcal{K})$ be the set of maximal gambles. Suppose that h is another gamble. Then, for any $f, g \in \mathrm{opt}_{\succ}(\mathcal{K})$, we have

$$0 \leq \overline{E}(f - g) \tag{50}$$

$$= \overline{E}(f - h + h - g) \tag{51}$$

$$\leq \overline{E}(f - h) + \overline{E}(h - g) \tag{52}$$

$$= -\underline{E}(h - f) + \overline{E}(h - g). \tag{53}$$

Therefore, by eq. (53), for any $f, g \in \mathrm{opt}_{\succ}(\mathcal{K})$:

$$\underline{E}(h - f) \leq \overline{E}(h - g). \tag{54}$$

Consequently,

$$\max_{f \in \mathrm{opt}_{\succ}(\mathcal{K})} \underline{E}(h - f) = \underline{E}(h - f^*) \quad \text{for some } f^* \in \mathrm{opt}_{\succ}(\mathcal{K}) \tag{55}$$

$$\leq \overline{E}(h - g) \quad \text{for all } g \in \mathrm{opt}_{\succ}(\mathcal{K}) \quad \text{(by eq. (49))} \tag{56}$$

$$\leq \min_{g \in \mathrm{opt}_{\succ}(\mathcal{K})} \overline{E}(h - g). \tag{57}$$

Next, we show that

$$\min_{f \in \mathrm{opt}_{\succ}(\mathcal{K})} \overline{E}(h - f) \leq \overline{E}(h) - \max_{g \in \mathcal{K}} \underline{E}(g). \tag{58}$$

We see that

$$\min_{f \in \mathrm{opt}_{\succ}(\mathcal{K})} \overline{E}(h - f) = \min_{g \in \mathcal{K}} \overline{E}(h - g) \quad \text{(by eq. (42))} \tag{59}$$

$$\leq \min_{g \in \mathcal{K}} \left(\overline{E}(h) - \underline{E}(g)\right) \tag{60}$$

$$= \overline{E}(h) - \max_{g \in \mathcal{K}} \underline{E}(g). \tag{61}$$

□

Durham University, Department of Mathematical Sciences, UK
Email address: `nawapon.nakharutai@gmail.com`

Durham University, Department of Mathematical Sciences, UK
Email address: `matthias.troffaes@durham.ac.uk`

Durham University, Department of Mathematical Sciences, UK
Email address: `c.c.d.s.caiado@durham.ac.uk`