IMPROVING AND BENCHMARKING OF ALGORITHMS FOR Γ-MAXIMIN, Γ-MAXIMAX AND INTERVAL DOMINANCE

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

Abstract. Γ-maximin, Γ-maximax and inteval dominance are familiar decision criteria for making decisions under severe uncertainty, when probability distributions can only be partially identified. One can apply these three criteria by solving sequences of linear programs. In this study, we present new algorithms for these criteria and compare their performance to existing standard algorithms. Specifically, we use efficient ways, based on previous work, to find common initial feasible points for these algorithms. Exploiting these initial feasible points, we develop early stopping criteria to determine whether gambles are either Γ-maximin, Γ-maximax or interval dominant. We observe that the primal-dual interior point method benefits considerably from these improvements. In our simulation, we find that our proposed algorithms outperform the standard algorithms when the size of the domain of lower previsions is less or equal to the sizes of decisions and outcomes. However, our proposed algorithms do not outperform the standard algorithms in the case that the size of the domain of lower previsions is much larger than the sizes of decisions and outcomes.

1. Introduction

Consider a situation where a subject must make a decision by selecting any options from a set of possible decisions. Each option leads to an uncertain reward depending on the option and on some uncertain outcome. The reward can be anything, for example, money or casino chips. However, we suppose that rewards can be related to a utility scale, and hence any uncertain reward can be considered as a bounded real-valued function on a set of outcomes. We call such function a *gamble*. In other words, we now consider a situation where a subject would like to choose gambles from a set of possible gambles.

We suppose that, for any set of possible gambles, the subject is able to state gambles that she does not want to select. Once we delete all gambles that she does not choose, the remaining gambles in the set are called *optimal*. In a classical decision theory, the subject should simply select a gamble that yields the maximum expected utility if probabilities for all relevant events are available for her [1]. Unfortunately, she may be unable to specify all probabilities exactly. To handle this situation, Walley [21] suggested that the subject can still model her belief using *lower previsions* which are equivalent to expectation bounds.

There are several well-known decision criteria associated with lower previsions [15]. For example, *E-admissibility*, *maximality* and *interval dominance* will return a set of decisions [15]. In this study, we are interested in interval dominance as well as two other popular criteria which give us a set of equivalent decisions (usually a singleton for generic decision problems), that is, Γ-maximin which maximizes the lower prevision of gambles, and Γ-maximax which maximizes the upper prevision of gambles [5, p. 193].

Key words and phrases. Γ-maximin; Γ-maximax; interval dominance; primal-dual; algorithm; benchmarking.

Algorithms for decision criteria associated with lower previsions have been studied in the literature for quite some time. For example, see [20] for E-admissibility and [6] [8], [14] and [19, p. 336] for maximality. From these, only [14] provides in-depth benchmarking of their algorithms.

Algorithms for Γ-maximin and Γ-maximax have been studied for instance in Kikuti et al. [6] and Troffaes and Hable [19, p. 335]. In those algorithms, to determine whether a gamble is Γ-maximin (or Γ-maximax), one has to evaluate the value of the lower (or the upper) prevision of all gambles in a set of gambles. This can be done by solving linear programs [6, 19]. Therefore, the number of linear programs that we have to solve is the cardinality of the set of gambles. We emphasize that the scope of our study only concerns lower previsions defined on a finite set of gambles over a finite possibility space, and only concerns finite sets of decisions. More general situations where, say, non-linear relationships are imposed to construct the credal set (which can happen under certain independence conditions), and thus where non-linear programming applies, are beyond the scope of this paper.

In previous work, we studied efficient ways to solve linear programs inside algorithms for decision making with lower prevision. Specifically, in [12, 13] we investigated efficient algorithms to solve linear programs for checking avoiding sure loss. We presented an efficient way to find initial feasible points [13, §4.2], and extra stopping criteria to check avoiding sure loss. The results in that work showed that the primal-dual method can exploit these improvements and performs much more better compared to the simplex and the affine scaling methods when working with lower previsions. Since we knew that the primal-dual method is suitable for handling lower previsions, in [14] we improved and benchmarked algorithms for maximality, also using the primal-dual method. Some improvements, such as efficient ways to access common feasible starting points or early stopping criteria, also have clear potential to improve algorithms for Γ-maximin, Γ-maximax and interval dominance. This is what we aim to do in this paper.

The contributions of this paper are as follows. Based on our earlier work [12, 13, 14], we propose new algorithms for Γ-maximin, Γ-maximax and interval dominance. Specifically, we implement a quick way to obtain feasible starting points for the linear programs and early stopping criteria to determine whether a gamble is Γ-maximin (or Γ-maximax). As the primal-dual method solves both the primal and dual at the same time, we show how it can exploit both feasible starting points and early stopping criteria to further improve the efficiency of the algorithms for Γ-maximin and Γ-maximax. We also show that these improvements can be applied to algorithms for interval dominance. In addition, we benchmark these improvements by doing a comparative study of different improved algorithms on random generated sets of gambles for Γ-maximin, Γ-maximax and interval dominance.

We organise the paper as follows. In section 2, we review the theory of lower previsions and natural extension, and we review the definitions of Γ-maximin, Γ-maximax and interval dominance. In section 3, we present algorithms for Γ-maximin, Γ-maximax and interval dominance and propose improvements for these algorithms. For benchmarking these improvements, we use an algorithm from [11] and [12] to generate sets of gambles, and compare the performance of algorithms with different improvements in section 4. Finally, a conclusion is presented in section 5.

2. Decision making with lower previsions

In this section, we briefly state the necessary preliminary definitions and notation used in this work: lower previsions, natural extension (see [21, 9, 10, 17]), and the Γ-maximin, Γ-maximax and interval dominance decision criteria (see [15]).

2.1. **Lower previsions.** The *possibility space* is denoted by Ω and represents the set of states of nature. A *gamble* is a function $f\colon \Omega \to \mathbb{R}$, viewed as an uncertain reward: after the true state of nature ω is revealed, the gamble f will yield the payoff $f(\omega)$. Let $\mathcal{L}$ be the set of all gambles on Ω.

A *lower prevision* $\underline{P}$ is a real-valued function defined on $\operatorname{dom}\underline{P} \subseteq \mathcal{L}$, where $\operatorname{dom}\underline{P}$ denotes the domain of $\underline{P}$. Let f be a gamble in $\operatorname{dom}\underline{P}$. We interpret $\underline{P}(f)$ as the subject's supremum buying price for f. This means that for all $\alpha < \underline{P}(f)$, the gamble $f - \alpha$ is desirable to the subject. Lower previsions are suitable to handle the subject's uncertainty when little information is available, because they do not require a full probability specification [21, 9, 10, 17].

Let us now state a basic consistency requirement for lower previsions.

Definition 1. *A lower prevision* $\underline{P}$ *avoids sure loss if for all* $n \in \mathbb{N}$*, all* $\lambda_1, \ldots, \lambda_n \geq 0$*, and all* $f_1, \ldots, f_n \in \operatorname{dom}\underline{P}$*, it holds that [17, p. 42]:*

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

When eq. (1) does not hold, then there are $f_1, \ldots, f_n \in \operatorname{dom}\underline{P}$ and $\lambda_1, \ldots, \lambda_n \geq 0$ such that:

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

and we say that $\underline{P}$ does not avoid sure loss [17, p. 44]. This implies that the subject agrees to pay more than the maximum possible gain, which is not sensible. All lower previsions further are assumed to avoid sure loss. Note that we could require lower previsions to be coherent which is a stronger rationality requirement, namely, that the highest price a subject is willing to pay for a gamble cannot be increased by considering combinations of other acceptable gambles. However, coherence is not necessary here as all formulas that we will use rely on natural extension, which only requires avoiding sure loss (see [21, §3.1] for further details).

Given any lower prevision $\underline{P}$, we can also consider its conjugate defined by $\overline{P}(f) := -\underline{P}(-f)$ on $\operatorname{dom}\overline{P} := \{-f : f \in \operatorname{dom}\underline{P}\}$. We call $\overline{P}$ an *upper prevision* which can be viewed as the infimum acceptable selling price for f to the subject [17, p. 41].

A lower prevision $\underline{P}$ can be extended to the set of all gambles $\mathcal{L}$ via the following method:

Definition 2. *[17, p. 47] Let* $\underline{P}$ *be a lower prevision. The natural extension of* $\underline{P}$ *is given, for all* $g \in \mathcal{L}$*, by:*

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

This can be interpreted as follows: based on the prices $\underline{P}(f_i)$ for all $f_i \in \operatorname{dom}\underline{P}$, $\underline{E}(g)$ is the subject's supremum price that subject is willing to pay for $g \in \mathcal{L}$ [17, p. 47]. Note that if $\underline{P}$ avoids sure loss, then $\underline{E}$ is finite, and therefore, is a lower prevision [17, p. 68]. Moreover, if both Ω and $\operatorname{dom}\underline{P}$ are finite, then $\underline{E}(g)$ can be computed directly by solving a linear program [19, p. 331]. Therefore, throughout this study, both Ω and $\operatorname{dom}\underline{P}$ are assumed finite.

Similarly, we consider the conjugate of $\underline{E}$ which is denoted by $\overline{E}$ and given by

$$(4) \qquad \overline{E}(g) := -\underline{E}(-g)$$

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

An upper prevision $\overline{E}$ can be interpreted as the subject's infimum selling price for a gamble $g \in \mathcal{L}$ [17, p. 47].

2.2. **Decision criteria.** In this section, we review three decision criteria that are used in the context of lower previsions: Γ-maximin, Γ-maximax and interval dominance.

Definition 3. *[15] Let $\mathcal{K} \subseteq \mathcal{L}$. A gamble is called Γ-maximin in $\mathcal{K}$ if it maximizes the lower natural extension in $\mathcal{K}$, that is, if it belongs to the set*

$$\text{(6)} \qquad \text{opt}_{\underline{E}}(\mathcal{K}) := \{f \in \mathcal{K} \colon (\forall g \in \mathcal{K})(\underline{E}(f) \geq \underline{E}(g))\}$$

$$\text{(7)} \qquad = \arg\max_{f \in \mathcal{K}} \underline{E}(f).$$

In the above, $\arg\max_{f \in \mathcal{K}} \underline{E}(f)$ denotes the set of gambles whose elements attain the lower natural extension's largest value. So, basically, a Γ-maximin gamble is a gamble whose lower prevision (after natural extension) is maximal across all gambles in $\mathcal{K}$. Similarly, we have:

Definition 4. *[15] Let $\mathcal{K} \subseteq \mathcal{L}$. A gamble is called Γ-maximax in $\mathcal{K}$ if it maximizes the upper natural extension in $\mathcal{K}$, that is, if it belongs to the set*

$$\text{(8)} \qquad \text{opt}_{\overline{E}}(\mathcal{K}) := \{f \in \mathcal{K} \colon (\forall g \in \mathcal{K})(\overline{E}(f) \geq \overline{E}(g))\}$$

$$\text{(9)} \qquad = \arg\max_{f \in \mathcal{K}} \overline{E}(f),$$

Here, $\arg\max_{f \in \mathcal{K}} \overline{E}(f)$ denotes the set of gambles whose elements attain the upper natural extension's largest value.

Next, we consider *interval dominance* which is based on a strict partial preference order.

Definition 5. *The set of interval dominant gambles of $\mathcal{K}$ is*

$$\text{(10)} \qquad \text{opt}_{\sqsupset}(\mathcal{K}) := \{f \in \mathcal{K} \colon (\forall g \in \mathcal{K})(g \not\sqsupset f)\}$$

$$\text{(11)} \qquad = \{f \in \mathcal{K} : \overline{E}(f) \geq \max_{g \in \mathcal{K}} \underline{E}(g)\}.$$

where we used the following strict partial preference order:

$$\text{(12)} \qquad f \sqsupset g \text{ if } \underline{E}(f) > \overline{E}(g)$$

Note that [5]:

$$\text{(13)} \qquad \text{opt}_{\underline{E}}(\mathcal{K}) \cup \text{opt}_{\overline{E}}(\mathcal{K}) \subseteq \text{opt}_{\sqsupset}(\mathcal{K}).$$

3. Algorithms

In this section, we will discuss algorithms for Γ-maximin, Γ-maximax and interval dominance and we will propose new algorithms based on improvements in [14].

3.1. **Base Algorithms.** Recall that we only consider the situation where the domain of the lower prevision $\underline{P}$ and the possibility space Ω are finite.

A first straightforward algorithm is simply to calculate $\underline{E}(f)$ for each $f \in \mathcal{K}$, by solving linear programs. Let $k = |\mathcal{K}|$. To find $\underline{E}(f)$, we can either solve (P1) or (D1):

(P1a) (P1) $$\min \quad \sum_{\omega \in \Omega} f(\omega)p(\omega)$$

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

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

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

(D1a) (D1) $$\max \quad \alpha$$

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

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

Note that $\underline{E}(f)$ is exactly the optimal value of these linear programs and (D1) corresponds to the unconditional case of the linear program in [19, p. 331]. Also note that eq. (P1b) is equivalent to requiring that the expectation $E_p(g_i)$ is greater than $\underline{P}(g_i)$, eqs. (P1c) and (P1d) guarantee that p is a probability mass function, and eqs. (P1b), (P1c) and (P1d) are used to compute the feasible region of (P1) which coincides with the credal set of $\underline{P}$, where the credal set of $\underline{P}$ is a closed convex set of probability mass functions over Ω [21, p. 135]. Moreover, (D1) is equivalent to eq. (3).

To find $\arg\max_{f \in \mathcal{K}} \underline{E}(f)$, we need to solve k linear programs to get their corresponding optimal values; see algorithm 1.

Even though algorithm 1 is straightforward, we present the full algorithm here to introduce the reader to the syntax, and also for ease of comparison with algorithms that will be introduced later.

There are several common linear programming methods, such as the simplex method, and interior-point methods such as the affine scaling and the primal-dual method. These methods are iterative, i.e., given a starting point, the methods will find a next feasible solution that comes closer to, and eventually converges to, an optimal solution (provided that an optimal solution exists).

From earlier work [13] where we performed in-depth analysis of algorithms for working with lower previsions, we know that the primal-dual method, which simultaneously solves both primal and dual problems, performs very well for solving such linear programs. So in the simulation study in section 4, we will solve all linear programs only by the primal-dual method and we do not consider other methods.

What is interesting about the primal-dual method is that it can start from an arbitrary (not necessarily feasible) point. The method will then iterate over points that simultaneously move the solution towards feasibility and optimality. Once feasibility is achieved, all next iterations will retain feasibility, and the solution will eventually converge to an optimal solution [3, §7.3].

We now explain how $\underline{E}(f_i)$ is calculated in algorithm 1 through the primal-dual method. Let (x_i^P, x_i^D) denote a current pair of (potentially infeasible) points, one for (P1), and one for (D1), corresponding to the linear program for f_i. The corresponding (P1) value is denoted by $\underline{e}^*(f_i, x_i^P, x_i^D)$, and the (D1) value by $\underline{e}_*(f_i, x_i^P, x_i^D)$. Let ϕ denote a function that updates this pair of points, representing one iterative step of the algorithm. At each iteration, the method will return an updated

---

Algorithm 1 Γ-maximin (original)

Input: a set of k gambles $\mathcal{K} = \{f_1, \dots, f_k\}$; a small number ϵ; initial states x_i^P and x_i^D for (P1)
    and (D1) corresponding to f_i respectively;
Output: a single Γ-maximin gamble

 1: **procedure** Γ-MAXIMIN1$(\mathcal{K}, \epsilon)$
 2:     $\underline{e} \leftarrow -\infty$
 3:     **for** $i = 1 : k$ **do**
 4:         **repeat**
 5:             $(x_i^P, x_i^D, r_i^P, r_i^D) \leftarrow \phi(f_i, x_i^P, x_i^D)$ $\triangleright$ next iteration to compute $\underline{E}(f_i)$
 6: $\triangleright$ r_i^P is the primal residual
 7: $\triangleright$ r_i^D is the dual residual
 8:             $\ell_i \leftarrow \underline{e}_*(f_i, x_i^P, x_i^D)$ $\triangleright$ the current lower bound for $\underline{E}(f_i)$
 9:             $u_i \leftarrow \underline{e}^*(f_i, x_i^P, x_i^D)$ $\triangleright$ the current upper bound for $\underline{E}(f_i)$
10:             $\gamma_i = u_i - \ell_i$
11:         **until** $\max\{\gamma_i, r_i^P, r_i^D\} < \epsilon$
12:         **if** $\ell_i > \underline{e}$ **then**
13:             $i^* \leftarrow i$; $\underline{e} \leftarrow \ell_i$
14:         **end if**
15:     **end for**
16:     **return** i^* $\triangleright$ i^* is Γ-maximin index
17: **end procedure**

---

point $\phi(f_i, x_i^P, x_i^D)$. The process will be repeated until (x_i^P, x_i^D) is feasible (i.e., $r_i^P, r_i^D < \epsilon$) and
the duality gap $\underline{e}^*(f_i, x_i^P, x_i^D) - \underline{e}_*(f_i, x_i^P, x_i^D)$ is small enough. Once the method terminates, $\underline{E}(f_i)$
can be read from $\underline{e}_*(f_i, x_i^P, x_i^D)$ (or $\underline{e}^*(f_i, x_i^P, x_i^D)$).

Similarly, for Γ-maximax, a straightforward algorithm is to compute $\overline{E}(f)$ for all $f \in \mathcal{K}$. Note
that $\overline{E}(f)$ can be obtained either solving (P2) or (D2):

(P2a)   (P2) $\min \quad \beta$

(P2b)   subject to $\forall \omega \in \Omega \colon \beta - \sum_{i=1}^{k}(g_i(\omega) - \underline{P}(g_i))\lambda_i \geq f(\omega)$

(P2c)   where $\forall i \colon \lambda_i \geq 0 \quad (\beta \text{ free})$,

(D2a)   (D2) $\max \quad \sum_{\omega \in \Omega} f(\omega)p(\omega)$

(D2b)   subject to $\forall g_i \in \operatorname{dom} \underline{P} \colon \sum_{\omega \in \Omega}(g_i(\omega) - \underline{P}(g_i))p(\omega) \geq 0$

(D2c) $\sum_{\omega \in \Omega} p(\omega) = 1$

(P2d)   where $\forall \omega \colon p(\omega) \geq 0$.

To find $\arg\max_{f \in \mathcal{K}} \overline{E}(f)$, again we have to solve $k = |\mathcal{K}|$ linear programs to get their corresponding
optimal values; see algorithm 2. In this case, eq. (D2b) implies that $E_p(g_i) \geq \underline{P}(g_i)$ and eqs. (D2c)

and (P2d) ensures that p is a probability mass function. In addition, eqs. (D2b), (D2c) and (P2d) are used to find the feasible region of (D2) which is exactly the credal set of $\underline{P}$ while (P2) is equivalent to eq. (5).

Note that we set up the linear programs for Γ-maximax differently from the linear programs for Γ-maximin, because we want to match the standard form for the primal-dual method, namely the primal problem is to minimise the objective function and the dual problem is to maximise the objective function. As the pair of problems are dual to each other, we can choose either one of the pair as the primal problem and the other one then becomes its dual problem [4, p. 174].

Algorithm 2 Γ-maximax (original)

Input: a set of k gambles $\mathcal{K} = \{f_1, \ldots, f_k\}$; a small number ϵ; initial states y_i^P and y_i^D for (P2)
    and (D2) corresponding to f_i respectively;
Output: a single Γ-maximax gamble
 1: **procedure** Γ-MAXIMAX1$(\mathcal{K}, \epsilon)$
 2:    $\bar{e} \leftarrow -\infty$
 3:    **for** $i = 1 : k$ **do**
 4:        **repeat**
 5:            $(y_i^P, y_i^D, r_i^P, r_i^D) \leftarrow \psi(f_i, y_i^P, y_i^D)$ ▷ next iteration to compute $\overline{E}(f_i)$
 6:   ▷ r_i^P is the primal residual
 7:   ▷ r_i^D is the dual residual
 8:            $\ell_i \leftarrow \bar{e}_*(f_i, y_i^P, y_i^D)$ ▷ the current lower bound for $\overline{E}(f_i)$
 9:            $u_i \leftarrow \bar{e}^*(f_i, y_i^P, y_i^D)$ ▷ the current upper bound for $\overline{E}(f_i)$
10:            $\gamma_i = u_i - \ell_i$
11:        **until** $\max\{\gamma_i, r_i^P, r_i^D\} < \epsilon$
12:        **if** $u_i > \bar{e}$ **then**
13:            $i^* \leftarrow i; \bar{e} \leftarrow u_i$
14:        **end if**
15:    **end for**
16:    **return** i^* ▷ i^* is a Γ-maximax index
17: **end procedure**

The explanation of algorithm 2 is similar as for algorithm 1. Here, (y_i^P, y_i^D) denotes a current pair of (potentially infeasible) points for (P2) and (D2) respectively, corresponding to f_i. The corresponding (P2) value is denoted by $\bar{e}^*(f_i, y_i^P, y_i^D)$, and the (D2) value by $\bar{e}_*(f_i, y_i^P, y_i^D)$. At each iteration, the method will return an updated point $\psi(f_i, y_i^P, y_i^D)$. This process is repeated until (y_i^P, y_i^D) is feasible (i.e., $r_i^P, r_i^D < \epsilon$) and the duality gap $\bar{e}^*(f_i, y_i^P, y_i^D) - \bar{e}_*(f_i, y_i^P, y_i^D)$ is small enough. When the method terminates, $\overline{E}(f_i)$ can be read from $\bar{e}_*(f_i, y_i^P, y_i^D)$ (or $\bar{e}^*(f_i, y_i^P, y_i^D)$).

Next, we consider an algorithm for interval dominance. Let $\mathcal{K}$ be a set of k gambles. We are going to find a set of interval dominant gambles in $\mathcal{K}$. To do so, for each f_i, we are going to check whether f_i is interval dominant or not. Note that f_i is interval dominant in the set $\mathcal{K}$ if

$$(14) \qquad \overline{E}(f_i) \geq \max_{f_j \in \mathcal{K}} \underline{E}(f_j),$$

where $\arg\max_{f_j \in \mathcal{K}} \underline{E}(f_j)$ is the Γ-maximin gamble. In other words, for each f, we compare it against the Γ-maximin gamble. We can calculate $\underline{E}(f_i)$ by either solving (P1) or (D1) and calculate $\overline{E}(f_i)$ by either solving (P2) or (D2). Overall, to find out a set of interval dominant gambles from a set

of k gambles, we have to solve $2k - 1$ linear programs (as $f_\ell := \arg\max_{f_j \in \mathcal{K}} \underline{E}(f_j)$ is immediately interval dominant) [19, p. 337]. An algorithm for finding interval dominant gambles is given in algorithm 3.

Algorithm 3 Interval dominance (original)

Input: a set of k gambles $\mathcal{K} = \{f_1, \ldots, f_k\}$; a small number ϵ; initial states x_i^P and x_i^D for (P1) and (D1) corresponding to f_i respectively; initial states y_i^P and y_i^D for (P2) and (D2) corresponding to f_i respectively;
Output: a single Γ-maximin gamble; an index set of $\text{opt}_\sqsupset(\mathcal{K})$

1: **procedure** INTERVALDOMINANT1$(\mathcal{K}, \epsilon)$
2:     **Stage I:** Apply Γ-MAXIMIN1 to obtain i^* and $\underline{e}$ $\triangleright$ i^* is an index of the Γ-maximin and $\underline{e} = \max_{f_j \in \mathcal{K}} \underline{E}(f_j)$

3:     **Stage II: Interval dominance**
4:     $I \leftarrow \{i^*\}$
5:     **for** $i \in \{1, 2, \ldots, k\} \setminus \{i^*\}$ **do**
6:         **repeat**
7:             $(y_i^P, y_i^D, r_i^P, r_i^D) \leftarrow \psi(f_i, y_i^P, y_i^D)$ $\triangleright$ next iteration to compute $\overline{E}(f_i)$
8:              $\triangleright$ r_i^P is the primal residual
9:              $\triangleright$ r_i^D is the dual residual
10:             $\ell_i \leftarrow \overline{e}_*(f_i, y_i^P, y_i^D)$ $\triangleright$ a lower bound for $\overline{E}(f_i)$
11:             $u_i \leftarrow \overline{e}^*(f_i, y_i^P, y_i^D)$ $\triangleright$ an upper bound for $\overline{E}(f_i)$
12:             $\gamma_i = u_i - \ell_i$
13:         **until** $\max\{\gamma_i, r_i^P, r_i^D\} < \epsilon$
14:         **if** $u_i \geq \underline{e}$ **then**
15:             $I \leftarrow I \cup \{i\}$ $\triangleright$ f_i is interval dominant
16:         **end if**
17:     **end for**
18:     **return** (i^*, I) $\triangleright$ I is an index set of $\text{opt}_\sqsupset(\mathcal{K})$
19: **end procedure**

3.2. **Improvements.** We now discuss how to improve algorithms 1, 2 and 3, by exploiting several features of the primal-dual method itself.

Recall that we only need to verify whether $\underline{E}(f_i) > \underline{e}$ or not. Note that for any (x_i^P, x_i^D),

$$(15) \qquad \underline{e}_*(f_i, x_i^P, x_i^D) \leq \underline{E}(f_i) \leq \underline{e}^*(f_i, x_i^P, x_i^D).$$

Therefore, provided that the current solution is feasible, we can terminate as soon as we find upper bound $\underline{e}^*(f_i, x_i^P, x_i^D)$ that is less than $\underline{e}$ since we immediately know that the optimal value of (P1) (or $\underline{E}(f_i)$) is also less than $\underline{e}$ (see fig. 1). Hence, it is not necessary to find the optimal value. This gives us an early stopping criterion for Γ-maximin.

For this to work best, ideally, we start the primal-dual method with feasible points. To obtain such feasible points, we can reuse the fast techniques proposed in [12, 13, 14]. Specifically, to find a starting feasible solution $p(\omega)$ for (P1) and (D2), we could employ the first phase of the two-phase method, similar to earlier work [13, §4.2]. Note that the constraints of each (P1) and (D2) do not depend on the gamble $f \in \mathcal{K}$. If we find their feasible starting points, then these feasible solutions
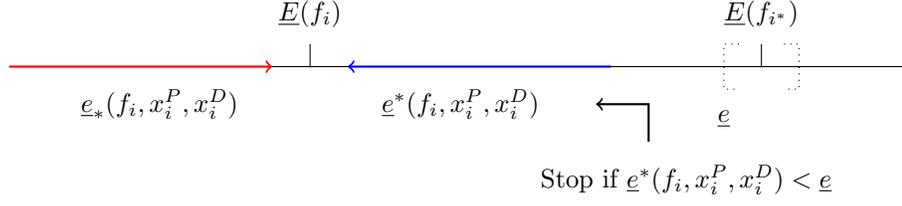
FIGURE 1. Early stopping criterion for Γ-maximin

can be reused for (P1) and (D2) across all $f \in \mathcal{K}$. In other words, for any given lower prevision, we only have to find their corresponding starting points once at the beginning. On the other hand, to obtain initial feasible solutions for (D1) and (P2), there is no need to solve any linear program as we can use a result from [13, Theorem 7] to directly calculate feasible starting points.

Also note that if we can identify a Γ-maximin gamble early, then fewer iterations are required. Fortunately, if we sort all gambles in $\mathcal{K}$ as $f_1, \ldots, f_k$ such that for some probability mass function p (which we have to find anyway as we need an initial feasible solution), for all $i < j$:

$$(16) \qquad E_p(f_i) \geq E_p(f_j),$$

then it is more likely that f_i is Γ-maximin for a lower index i, and we have to do fewer computations as a result. Note that this technique helps in most cases, but not always. In [14, Alg 2], we showed that this technique did help in the most cases even though it is just heuristic and may not always help. An algorithm that implements this ordering trick, along with the early stopping criterion, is presented in algorithm 4. Note that we do not need to check feasibility in this algorithm as we start with feasible points [4, §10.7].

Algorithm 4 Γ-maximin

Input: a set of k gambles $\mathcal{K} = \{f_1, \ldots, f_k\}$ such that for some probability mass function p, $E_p(f_1) \geq E_p(f_2) \geq \cdots \geq E_p(f_k)$; a small number ϵ; initial feasible states x_i^P and x_i^D for (P1) and (D1) corresponding to f_i respectively;

Output: a single Γ-maximin gamble

 1: **procedure** Γ-MAXIMIN2$(\mathcal{K}, \epsilon)$
 2: $\quad \underline{e} \leftarrow -\infty$
 3: $\quad$ **for** $i = 1 : k$ **do**
 4: $\quad\quad$ **repeat**
 5: $\quad\quad\quad (x_i^P, x_i^D) \leftarrow \phi(f_i, x_i^P, x_i^D)$ $\qquad\qquad\qquad\qquad$ ▷ next iteration to compute $\underline{E}(f_i)$
 6: $\quad\quad\quad \ell_i \leftarrow \underline{e}_*(f_i, x_i^P, x_i^D)$ $\qquad\qquad\qquad\qquad\qquad$ ▷ the current lower bound for $\underline{E}(f_i)$
 7: $\quad\quad\quad u_i \leftarrow \underline{e}^*(f_i, x_i^P, x_i^D)$ $\qquad\qquad\qquad\qquad\qquad$ ▷ the current upper bound for $\underline{E}(f_i)$
 8: $\quad\quad\quad \gamma_i = u_i - \ell_i$
 9: $\quad\quad$ **until** $u_i < \underline{e}$ **or** $\gamma_i < \epsilon$ $\qquad\qquad\qquad\qquad\qquad$ ▷ an early stopping criterion
10: $\quad\quad$ **if** $\ell_i > \underline{e}$ **then**
11: $\quad\quad\quad \underline{e} \leftarrow \ell_i; \ i^* \leftarrow i$
12: $\quad\quad$ **end if**
13: $\quad$ **end for**
14: $\quad$ **return** i^* $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ ▷ i^* is a Γ-maximin index
15: **end procedure**
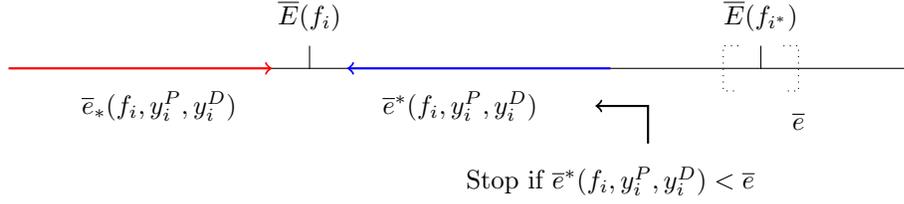
Figure 2. Early stopping criterion for Γ-maximax

These improvements can be applied to Γ-maximax as well. Specifically, we exploit the fact that we only have to verify whether $\overline{E}(f_i) > \overline{e}$ or not. Since for any (y_i^P, y_i^D),

$$\overline{e}_*(f_i, y_i^P, y_i^D) \leq \overline{E}(f_i) \leq \overline{e}^*(f_i, y_i^P, y_i^D), \tag{17}$$

we can terminate as soon as we find $\overline{e}^*(f_i, y_i^P, y_i^D)$ that is less than $\overline{e}$ because then we know that the optimal value of (P2), i.e. $\overline{E}(f_i)$, is less than $\overline{e}$ (see fig. 2).

Again, if we sort the gambles in $\mathcal{K} = \{f_1, \ldots, f_k\}$ such that eq. (16) is satisfied, then it is more likely that f_i is Γ-maximax for a lower index i, and we can save computations. See algorithm 5 for a description of an improved algorithm for Γ-maximax.

Algorithm 5 Γ-maximax

Input: a set of k gambles $\mathcal{K} = \{f_1, \ldots, f_k\}$ such that for some probability mass function p, $E_p(f_1) \geq E_p(f_2) \geq \cdots \geq E_p(f_k)$; a small number ϵ; initial feasible states y_i^P and y_i^D for (P2) and (D2) corresponding to f_i respectively;

Output: a single Γ-maximax gamble

1: **procedure** Γ-MAXIMAX2$(\mathcal{K}, \epsilon)$
2:      $\overline{e} \leftarrow -\infty$
3:      **for** $i = 1 : k$ **do**
4:          **repeat**
5:              $(y_i^P, y_i^D) \leftarrow \psi(f_i, y_i^P, y_i^D)$ $\triangleright$ next iteration to compute $\overline{E}(f_i)$
6:              $\ell_i \leftarrow \overline{e}_*(f_i, y_i^P, y_i^D)$ $\triangleright$ the current lower bound for $\overline{E}(f_i)$
7:              $u_i \leftarrow \overline{e}^*(f_i, y_i^P, y_i^D)$ $\triangleright$ the current upper bound for $\overline{E}(f_i)$
8:              $\gamma_i = u_i - \ell_i$
9:          **until** $u_i < \overline{e}$ **or** $\gamma_i < \epsilon$ $\triangleright$ an early stopping criterion
10:          **if** $u_i > \overline{e}$ **then**
11:              $\overline{e} \leftarrow u_i; i^* \leftarrow i$
12:          **end if**
13:      **end for**
14:      **return** i^* $\triangleright$ i^* is a Γ-maximax index
15: **end procedure**

For interval dominance, even though we have to solve $2k - 1$ linear programs, we do not need to find all these optimal values since we can apply these improvements for Γ-maximin and Γ-maximax. Specifically, in the stage I of algorithm 3, we can apply algorithm 4 to quickly obtain the Γ-maximin gamble, say f_{i^*}, and $\underline{e} = \max_{f_j \in \mathcal{K}} \underline{E}(f_j)$. In the stage II of of algorithm 3, we can speed up the process of evaluating $\overline{E}(f)$ through solving both (P2) and (D2) similarly to what we have done in

Stop if $\overline{e}_*(f_i, y_i^P, y^D) \geq \underline{e}$,
then f is interval dominant

Stop if $\overline{e}^*(f_i, y_i^P, y^D) < \underline{e}$
then f is not interval dominant

$\overline{e}_*(f_i, y_i^P, y^D)$ $\overline{E}(f_i)$ $\overline{e}^*(f_i, y_i^P, y^D)$
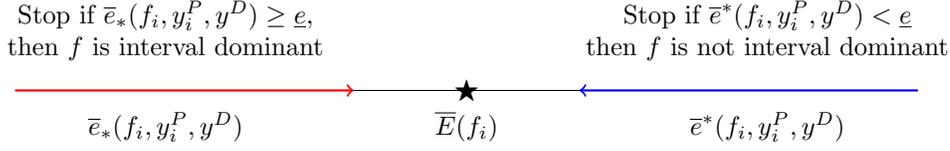
FIGURE 3. Early stopping criterion for interval dominance

algorithm 5, exploiting the fact that we only need to verify whether or not $\overline{E}(f_i) \geq \underline{e}$ (in this stage, $\underline{e}$ is $\max_{f_j \in \mathcal{K}} \underline{E}(f_j)$). Note that for any (y_i^P, y_i^D),

$$(18) \qquad \overline{e}_*(f_i, y_i^P, y_i^D) \leq \overline{E}(f_i) \leq \overline{e}^*(f_i, y_i^P, y_i^D).$$

So, we can stop as soon as we find $\overline{e}^*(f_i, y_i^P, y_i^D)$ that is less than $\underline{e}$ because then we know that the optimal value of (P2), i.e. $\overline{E}(f_i)$, is immediately less than $\underline{e}$. In this case, f is not interval dominant in $\mathcal{K}$. Meanwhile, we can stop as soon as we find $\overline{e}_*(f_i, y_i^P, y_i^D)$ that is larger or equal to $\underline{e}$ since we know that the optimal value of (D2), i.e. $\overline{E}(f_i)$, will be larger or equal to $\underline{e}$ as well. In this case, f is interval dominant in $\mathcal{K}$. Figure 3 summarises these early stopping criteria. An algorithm for interval dominance implemented the early stopping criteria is presented in algorithm 6.

Algorithm 6 Interval dominance

Input: a set of k gambles $\mathcal{K} = \{f_1, \ldots, f_k\}$ such that for some probability mass function p, $E_p(f_1) \geq E_p(f_2) \geq \cdots \geq E_p(f_k)$; a small number ϵ; initial feasible states x_i^P and x_i^D for (P1) and (D1) corresponding to f_i respectively; initial feasible states y_i^P and y_i^D for (P2) and (D2) corresponding to f_i respectively;
Output: a single Γ-maximin gamble; an index set of $\text{opt}_{\sqsupset}(\mathcal{K})$
  1: **procedure** INTERVALDOMINANT2($\mathcal{K}, \epsilon$)
  2:     **Stage I:** Apply Γ-MAXIMIN2 to obtain i^* and $\underline{e}$ ▷ i^* is an index of the Γ-maximin gamble and $\underline{e} = \max_{f_j \in \mathcal{K}} \underline{E}(f_j)$

  3:     **Stage II: Interval dominance**
  4:     $I \leftarrow \{i^*\}$ ▷ i^* is an index of the Γ-maximin gamble
  5:     **for** $i \in \{1, 2, \ldots, k\} \setminus \{i^*\}$ **do**
  6:       **repeat**
  7:         $(y_i^P, y_i^D) \leftarrow \psi(f_i, y_i^P, y_i^D)$ ▷ next iteration to compute $\overline{E}(f_i)$
  8:         $\ell_i \leftarrow \overline{e}_*(f_i, y_i^P, y_i^D)$ ▷ a lower bound for $\overline{E}(f_i)$
  9:         $u_i \leftarrow \overline{e}^*(f_i, y_i^P, y_i^D)$ ▷ an upper bound for $\overline{E}(f_i)$
 10:         $\gamma_i = u_i - \ell_i$
 11:         **if** $u_i < \underline{e}$ **then break**   ▷ f_i is not interval dominant
 12:         **else if** $\ell_i \geq \underline{e}$ **or** $\gamma_i < \epsilon$ **then**   ▷ f_i is interval dominant
 13:            $I \leftarrow I \cup \{i\}$; **break**
 14:         **end if**
 15:       **until False**
 16:     **end for**
 17:     **return** (i^*, I) ▷ I is an index set of $\text{opt}_{\sqsupset}(\mathcal{K})$
 18: **end procedure**

3.3. **Further Improvements.** Note that in order to find a Γ-maximin gamble by algorithm 4, in the first stage, we have to iterate until we obtain the optimal value for f_1. Only then can we start applying the early stopping criteria. In addition, the output of algorithms 1 and 4 is just a single gamble even though there can be more than one gamble that is Γ-maximin. We will consider another algorithm that can find Γ-maximin gambles without having to obtain the optimal value even of the first gamble, and that can find all Γ-maximin gambles.

As an example, consider a set of gambles $\mathcal{K} = \{f_1, f_2, f_3, f_4\}$. Suppose that we would like to find Γ-maximin gambles in $\mathcal{K}$. To do so, we might start by calculating $\underline{e}_*(f_i, x_i^P, x_i^D)$ and $\underline{e}^*(f_i, x_i^P, x_i^D)$ for each f_i in $\mathcal{K}$ where (x_i^P, x_i^D) is initial states for the linear programs associated with f_i. Assume that the result is illustrated as in fig. 4 (a). Note that $\underline{e}_*(f_1, x_1^P, x_1^D) = \max_i \underline{e}_*(f_i, x_i^P, x_i^D)$ and both $\underline{e}^*(f_2, x_2^P, x_2^D)$ and $\underline{e}^*(f_3, x_3^P, x_3^D)$ are smaller than $\underline{e}_*(f_1, x_3^P, x_3^D)$. At this point, we already know that f_2 and f_3 cannot be Γ-maximin since $\underline{E}(f_2)$ and $\underline{E}(f_3)$ must be less than $\underline{e}_*(f_1, x_1^P, x_1^D)$. From this, we can see that we can quickly eliminate gambles that are not Γ-maximin, before obtaining the optimal value of the first linear program.

We can translate this argument into an algorithm that can sequentially narrow the set of potentially Γ-maximin gambles. Specifically, for each gamble f_i in $\mathcal{K}$, we evaluate $\underline{e}_*(f_i, x_i^P, x_i^D)$ and $\underline{e}^*(f_i, x_i^P, x_i^D)$ which are a lower bound and an upper bound for $\underline{E}(f_i)$. Next, we calculate $M_* := \max_i \underline{e}_*(f_i, x_i^P, x_i^D)$ and $M^* := \max_i \underline{e}^*(f_i, x_i^P, x_i^D)$. Then, any gamble f_i for which $\underline{e}^*(f_i, x_i^P, x_i^D) < M_*$ can not be Γ-maximin and therefore can be eliminated. After that, we update the states (x_i^P, x_i^D) for all gambles f_i that are still potentially Γ-maximin. We repeat this process until either only one gamble is left in a set of potentially Γ-maximin gambles, or until the difference between M_* and M^* is less than a given tolerance, in which case we have found multiple Γ-maximin gambles. An algorithm that implements these arguments is presented in algorithm 7.

Algorithm 7 Γ-maximin

Input: a set of k gambles $\mathcal{K} = \{f_1, \ldots, f_k\}$; a small number ϵ; initial feasible states x_i^P and x_i^D for (P1) and (D1) corresponding to f_i respectively;
Output: a set of Γ-maximin gambles
 1: **procedure** Γ-MAXIMIN3$(\mathcal{K}, \epsilon)$
 2:    $R \leftarrow \{1, 2, \ldots, k\}$ $\triangleright$ an index set of potentially Γ-maximin
 3:    **repeat**
 4:        $\forall i \in R : (x_i^P, x_i^D) \leftarrow \phi(f_i, x_i^P, x_i^D)$ $\triangleright$ the next iteration to compute $\underline{E}(f_i)$
 5:        $\forall i \in R : \ell_i \leftarrow \underline{e}_*(f_i, x_i^P, x_i^D)$ $\triangleright$ the current lower bound for $\underline{E}(f_i)$
 6:        $\forall i \in R : u_i \leftarrow \underline{e}^*(f_i, x_i^P, x_i^D)$ $\triangleright$ the current upper bound for $\underline{E}(f_i)$
 7:        $M_* = \max_{i \in R} \ell_i$ $\triangleright$ a lower bound for Γ-maximin value
 8:        $M^* = \max_{i \in R} u_i$ $\triangleright$ an upper bound for Γ-maximin value
 9:        $R \leftarrow \{i \in R : (u_i \geq M_*)\}$
10:    **until** $(|R| = 1)$ **or** $(M^* - M_* < \epsilon)$
11:    **return** R
12: **end procedure**

A key difference between algorithm 7 and the previous two algorithms for Γ-maximin (algorithms 1 and 4) lies in the output of the algorithms. When there is more than one Γ-maximin gamble, algorithm 7 will return all Γ-maximin gambles, while the other two algorithms will return only one of the Γ-maximin gambles. Also note that algorithms 1 and 7 can be parallelized but not
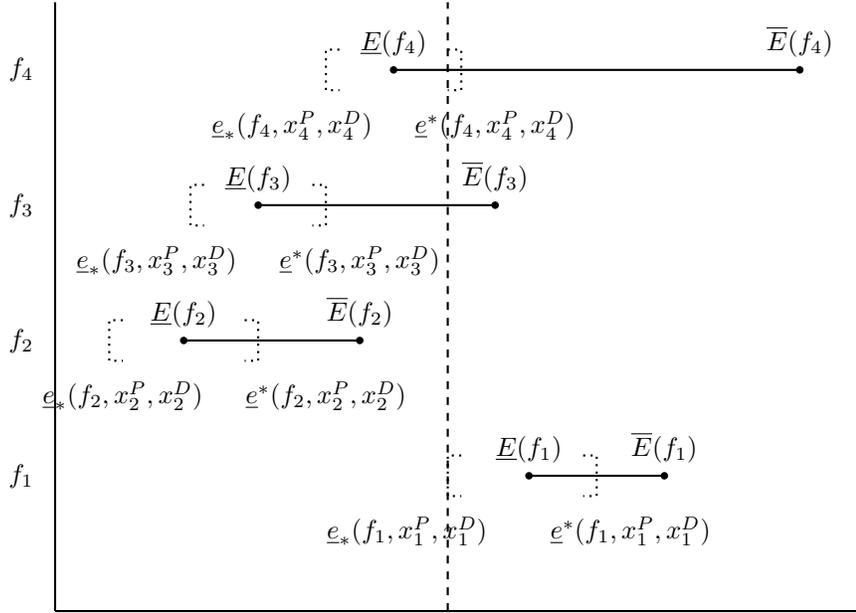
(a) Elimination of non Γ-maximin gambles.
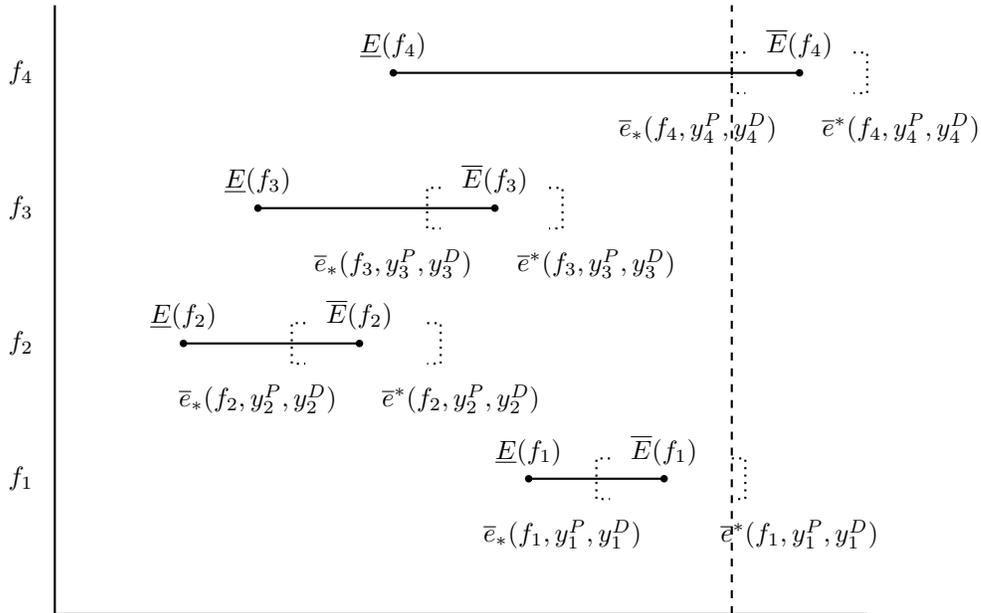
(b) Elimination of non Γ-maximax gambles.

FIGURE 4. Elimination of many gambles.

algorithm 4. In particular, algorithm 7 is basically a parallel version of algorithm 1 with an extra step to avoid unnecessary calculations.

The ideas in algorithm 7 can also be applied to eliminate non Γ-maximax gambles; see fig. 4 (b) and algorithm 8.

Algorithm 8 Γ-maximax

Input: a set of k gambles $\mathcal{K} = \{f_1, \ldots, f_k\}$; a small number ϵ; initial feasible state y_i^P and y_i^D for (P2) and (D2) corresponding to f_i;
Output: a set of Γ-maximax gambles

1: **procedure** Γ-MAXIMAX3$(\mathcal{K}, \epsilon)$
2:     $R \leftarrow \{1, 2, \ldots, k\}$ ▷ an index set of potentially Γ-maximax
3:     **repeat**
4:         $\forall i \in R : (y_i^P, y_i^D) \leftarrow \psi(f_i, y_i^P, y_i^D)$ ▷ the next iteration to compute $\overline{E}(f_i)$
5:         $\forall i \in R : \ell_i \leftarrow \overline{e}_*(f_i, y_i^P, y_i^D)$ ▷ the current lower bound for $\overline{E}(f_i)$
6:         $\forall i \in R : u_i \leftarrow \overline{e}^*(f_i, y_i^P, y_i^D)$ ▷ the current upper bound for $\overline{E}(f_i)$
7:         $M_* = \max_{i \in R} \ell_i$ ▷ a lower bound for Γ-maximax value
8:         $M^* = \max_{i \in R} u_i$ ▷ an upper bound for Γ-maximax value
9:         $R \leftarrow \{i \in R : (u_i \geq M_*)\}$
10:     **until** $(|R| = 1)$ **or** $(M^* - M_* < \epsilon)$
11:     **return** R
12: **end procedure**

Similarly, the ideas in algorithms 7 and 8 can be applied to interval dominance by simultaneously computing both upper and lower bounds for the lower and upper natural extensions. We have found however that, perhaps counterintuitively at first, this does not necessarily lead to improved algorithms for interval dominance. Understanding why this happens requires a somewhat detailed analysis, to which we turn next.

First, to bound the Γ-maximin value, for each f_i, we compute $\underline{e}_*(f_i, x_i^P, x_i^D)$ and $\underline{e}^*(f_i, x_i^P, x_i^D)$, which bound $\underline{E}(f_i)$. We then compute $M_* := \max_i \underline{e}_*(f_i, x_i^P, x_i^D)$ and $M^* := \max_i \underline{e}^*(f_i, x_i^P, x_i^D)$ to bound the Γ-maximin value $\max_i \underline{E}(f_i)$. We know that any f_i such that $\underline{e}^*(f_i, x_i^P, x_i^D) < M^*$ is definitely not Γ-maximin, and thus can be removed from future iterations for improving M^* and M_*. Now, we do not have to wait for M_* and M^* to have converged. Indeed, we can already immediately calculate $\overline{e}_*(f_i, y_i^P, y_i^D)$ and $\overline{e}^*(f_i, y_i^P, y_i^D)$, which bound $\overline{E}(f_i)$, and also compute the duality gap $\gamma_i = \overline{e}^*(f_i, y_i^P, y_i^D) - \overline{e}_*(f_i, y_i^P, y_i^D)$. We then know that any f_i such that $\overline{e}_*(f_j, y_j^P, y_j^D) > M^*$ will be interval dominant while any f_i such that $\overline{e}^*(f_i, y_j^P, y_j^D) < M_*$ will not be interval dominant. The entire process is repeated until there are no more undetermined gambles or until all γ_i and $M^* - M_*$ are less than a given tolerance. Additionally, we can also stop computations in the first part of the algorithm as soon as $M^* - M_*$ is small enough, i.e. as soon as we know the Γ-maximin value to sufficient precision. An algorithm that implements all of these ideas is shown in algorithm 9.

At first sight, this algorithm has potential to be faster because it provides an opportunity to eliminate interval dominated gambles before the Γ-maximin value is known to full precision. However, the algorithm may need to do more iterations to calculate the upper previsions to sufficient precision as long as the bounds on the Γ-maximin value remain too wide. Additionally, prior sorting of gambles does not help here, because all gambles are treated simultaneously, and not in sequence, as in algorithm 6. Our numerical experiments have found that algorithm 6 beats algorithm 9. Basically, by postponing our evaluation of the Γ-maximin value, the algorithm cannot eliminate

---

Algorithm 9 Interval dominance

Input: a set of k gambles $\mathcal{K} = \{f_1, \ldots, f_k\}$; a small number ϵ; initial feasible states x_i^P and x_i^D for (P1) and (D1) corresponding to f_i respectively; initial feasible states y_i^P and y_i^D for (P2) and (D2) corresponding to f_i respectively;
Output: an index set of $\mathrm{opt}_{\sqsupset}(\mathcal{K})$

1: **procedure** INTERVALDOMINANT3($\mathcal{K}, \epsilon$)
2:     $R \leftarrow \{1, 2 \ldots, k\}$ ▷ index set of potentially Γ-maximin gambles
3:     $J \leftarrow \{1, 2 \ldots, k\}$ ▷ index set of potentially interval dominant gambles
4:     $I \leftarrow \emptyset$ ▷ index set of known interval dominant gambles
5:     $N \leftarrow \emptyset$ ▷ index set of known non interval dominant gambles
6:     $(M_*, M^*) \leftarrow (-\infty, +\infty)$ ▷ bounds on Γ-maximin value
7:     **repeat**
8:         **if** $(M^* - M_* \geq \epsilon)$ **then**
9:             $\forall i \in R$:
10:                 $(x_i^P, x_i^D) \leftarrow \phi(f_i, x_i^P, x_i^D)$ ▷ next iteration to compute $\underline{E}(f_i)$
11:                 $(\ell_i, \underline{u}_i) \leftarrow (e_*(f_i, x_i^P, x_i^D), e^*(f_i, x_i^P, x_i^D))$ ▷ bounds on $\underline{E}(f_i)$
12:             $M_* = \max_{i \in R} \ell_i$ ▷ lower bound on Γ-maximin value
13:             $M^* = \max_{i \in R} \underline{u}_i$ ▷ upper bound on Γ-maximin value
14:             $R \leftarrow \{i \in R : \underline{u}_i \geq M_*\}$
15:         **end if**
16:         $\forall j \in J$:
17:             $(y_j^P, y_j^D) \leftarrow \psi(f_j, y_j^P, y_j^D)$ ▷ next iteration to compute $\overline{E}(f_j)$
18:             $(\overline{\ell}_j, \overline{u}_j) \leftarrow (\overline{e}_*(f_j, y_j^P, y_j^D), \overline{e}^*(f_i, y_j^P, y_j^D))$ ▷ bounds on $\overline{E}(f_j)$
19:             $\gamma_j = \overline{u}_j - \overline{\ell}_j$ ▷ duality gap
20:         $I \leftarrow I \cup \{j \in J : \overline{\ell}_j \geq M^*\}$ ▷ f_j is interval dominant
21:         $N \leftarrow N \cup \{j \in J : \overline{u}_j < M_*\}$ ▷ f_j is not interval dominant
22:         $J = (I \cup N)^{\mathsf{c}}$
23:     **until** $(J = \emptyset)$ **or** $\left((\forall j \in J : \gamma_j < \epsilon) \text{ and } M^* - M_* < \epsilon\right)$
24:     $I \leftarrow I \cup J$
25:     **return** I
26: **end procedure**

---

gambles early on, and needs to do more overall iterations in the end, for most of the benchmarking problems considered.

In our numerical experiments, we observed that in the second part of algorithm 9, when we compare the lower bound of $\overline{E}(f)$ against M^* and compare the upper bound of $\underline{E}(f)$ against M_*, we rarely determine whether f is interval dominant or not before we can identify the (usually unique) Γ-maximin gamble. In contrast, once we know the Γ-maximin gamble, then we observed that we can very quickly determine whether f is interval dominant or not with much fewer iterations. This leads us to try one more final idea: use algorithm 7 until a Γ-maximin gamble is identified but still allow $M^* - M_* \geq \epsilon$, and then proceed as with algorithm 9. This is shown in algorithm 10. This version of the algorithm is simpler and easier to implement. However, we found that the performance of algorithms 9 and 10 is still nearly identical. Therefore, algorithm 10 still does not beat algorithm 6, as we shall see in section 4.

Algorithm 10 Interval dominance

Input: a set of k gambles $\mathcal{K} = \{f_1, \ldots, f_k\}$; a small number ϵ; initial feasible states x_i^P and x_i^D for (P1) and (D1) corresponding to f_i respectively; initial feasible states y_i^P and y_i^D for (P2) and (D2) corresponding to f_i respectively;

Output: a set of Γ-maximin gambles; an index set of $\text{opt}_{\sqsupseteq}(\mathcal{K})$

1: **procedure** INTERVALDOMINANT3$(\mathcal{K}, \epsilon)$
2:     **Stage I:** Apply Γ-MAXIMIN3 to obtain R, M_* and M^*
             $\triangleright$ R is an index set of the Γ-maximin gambles, M_* is a lower bound for the Γ-maximin value and M^* is an upper bound for the Γ-maximin value
3:     **Stage II: Interval dominance**
4:       $I \leftarrow R$ $\triangleright$ index set of known interval dominant gambles
5:       $N \leftarrow \emptyset$ $\triangleright$ index set of known non interval dominant gambles
6:       $J \leftarrow R^{\mathsf{c}}$ $\triangleright$ index set of potentially interval dominant gambles
7:       **repeat**
8:         $\forall j \in J$:
9:           $(y_j^P, y_j^D) \leftarrow \psi(f_j, y_j^P, y_j^D)$ $\triangleright$ next iteration to compute $\overline{E}(f_j)$
10:           $(\ell_j, u_j) \leftarrow (\overline{e}_*(f_j, y_j^P, y_j^D), \overline{e}^*(f_j, y_j^P, y_j^D))$ $\triangleright$ bounds on $\overline{E}(f_j)$
11:           $\gamma_j = u_j - \ell_j$ $\triangleright$ duality gap
12:         **if** $M^* - M_* \geq \epsilon$ **then**
13:           $\forall r \in R$:
14:             $(x_r^P, x_r^D) \leftarrow \phi(f_r, x_r^P, x_r^D)$ $\triangleright$ continue compute $\underline{E}(f_r)$
15:             $(\ell_r, u_r) \leftarrow (\underline{e}_*(f_r, x_r^P, x_r^D), \underline{e}^*(f_r, x_r^P, x_r^D))$
16:           $M_* = \max_{r \in R} \ell_r$
17:           $M^* = \max_{r \in R} u_r$
18:         **end if**
19:         $I \leftarrow I \cup \{j \in J : \ell_j \geq M^*\}$ $\triangleright$ f_j is interval dominant
20:         $N \leftarrow N \cup \{j \in J : u_j < M_*\}$ $\triangleright$ f_j is not interval dominant
21:         $J = (I \cup N)^{\mathsf{c}}$
22:       **until** $(J = \emptyset)$ **or** $\big((\forall j \in J : \gamma_j < \epsilon)$ **and** $M^* - M_* < \epsilon\big)$
23:       $I \leftarrow I \cup J$
24:       **return** (R, I)
25: **end procedure**

We note that many of the computations can be parallelized, which could be exploited to reduce the running time of the algorithm. In particular, the parts under the universal quantifiers (i.e. $\forall i \in R$ and $\forall j \in J$) are independent, and can therefore in principle run simultaneously. However, after each parallel step, the outcomes must be combined to continue with the next step, to determine the updated R and J, as well as the updated bounds M_* and M^* on the Γ-maximin value, so some communication between the parallel processes is needed between iterations. From a runtime perspective, algorithm 9 might still be an attractive option to consider even though it takes more computational effort. For this paper however, we have focused our implementations and benchmarking on the serial running time to reflect the overall computational effort, and leave a study of parallel algorithms, where interesting trade-offs are to be had compared to serial algorithms, to future work.

To summarise, algorithms 1, 4 and 7 are for finding Γ-maximin gambles, algorithms 2, 5 and 8 are for finding Γ-maximax gambles and algorithms 3, 6, 9 and 10 are for finding interval dominant gambles. We will benchmark these algorithms in the next section.

4. Benchmarking results

4.1. **Γ-maximin and Γ-maximax.** In this section, we will compare our algorithms for Γ-maximin and Γ-maximax for the case that $|\mathcal{K}| = 2^i$ for $i \in \{1, 2, \ldots, 10\}$ and $|\Omega| = 2^j$ for $j \in \{1, 2, \ldots, 10\}$. This covers a good range of the experiment sizes that we expect to see in real-world applications of decision making (for example [18, 16, 2]), where 2^{10} gambles (or decisions) and 2^{10} outcomes should be plenty big enough for real-world decision making.

For each case, we first generate a lower prevision $\underline{P}$, where $|\operatorname{dom} \underline{P}| = 2^4$, on a finite domain which avoids sure loss as follows. To do so, we use [13, 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 [13, 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. We also consider $|\operatorname{dom} \underline{P}| = 2^n$ for $n \in \{6, 8, 10\}$. Next, to generate gambles $f_i \in \mathcal{K}$, for each ω and i, we sample $f_i(\omega)$ uniformly from interval $[0, 1]$.

For each generated set of gambles $\mathcal{K}$, we perform algorithms 1, 4 and 7 to find Γ-maximin gambles and algorithms 2, 5 and 8 to find Γ-maximax gambles. We apply [13, §4.2] to find feasible starting points and implement early stopping criteria in the primal-dual method corresponding to different cases. Specifically, the improved primal-dual method solves linear programs (P1) and (D1) inside algorithms 4 and 7, and solves linear programs (P2) and (D2) inside algorithms 5 and 8. Based on the implementation used in [13], we wrote our own implementation of the improved primal-dual method in MATLAB (R2019b) [7] rather than using MATLAB's existing library. This is because the existing library does not allow us to inspect the algorithms at different iterations easily. Since we want to compare different algorithms that work in different ways, we need to ensure that the only difference is in our part of the algorithms. For algorithms 1 and 2 where we solve linear programs inside these algorithms by the standard primal-dual method, to have a same advantage, we also wrote our own implementation of the standard primal-dual method instead of using the primal-dual method in MATLAB's toolbox.

For each algorithm, we record the total running time spent. In particular, for algorithms 4 and 5 this includes the time to sort gambles with respect to their expectations as in eq. (16). To sort gambles, we use quicksort which is a built-in function available in MATLAB [7]. Even though quicksort is an obvious choice for sorting, we state it for completeness, as there are other available functions for sorting. This process is repeated 1000 times. A summary of the results is presented in figs. 5 and 6.

Figures 5 and 6 present the average total running time of different algorithms for Γ-maximin in the left column and Γ-maximax in the right column. In fig. 5, each row presents a different number of gambles and the horizontal axis presents the number of outcomes. In fig. 6, each row presents a different number of outcomes and the horizontal axis presents the number of gambles. In both figures, the vertical axis shows the running time which is averaged over 1000 random generated decision problems. The error bars on the figures show approximate 95% confidence intervals on the average running time. They are barely visible except in few cases, e.g. in fig. 7, because of the large sample size.

We also consider the effect of the size of dom $\underline{P}$. We ensure that the generated lower prevision has no special properties, for example, 2-monotonicity [17, Chapter 6]) by considering $|\operatorname{dom} \underline{P}| = 2^k$ for $k \in \{4, 6, 8, 10\}$ and find that different size of k has an effect on the results (see fig. 7). To
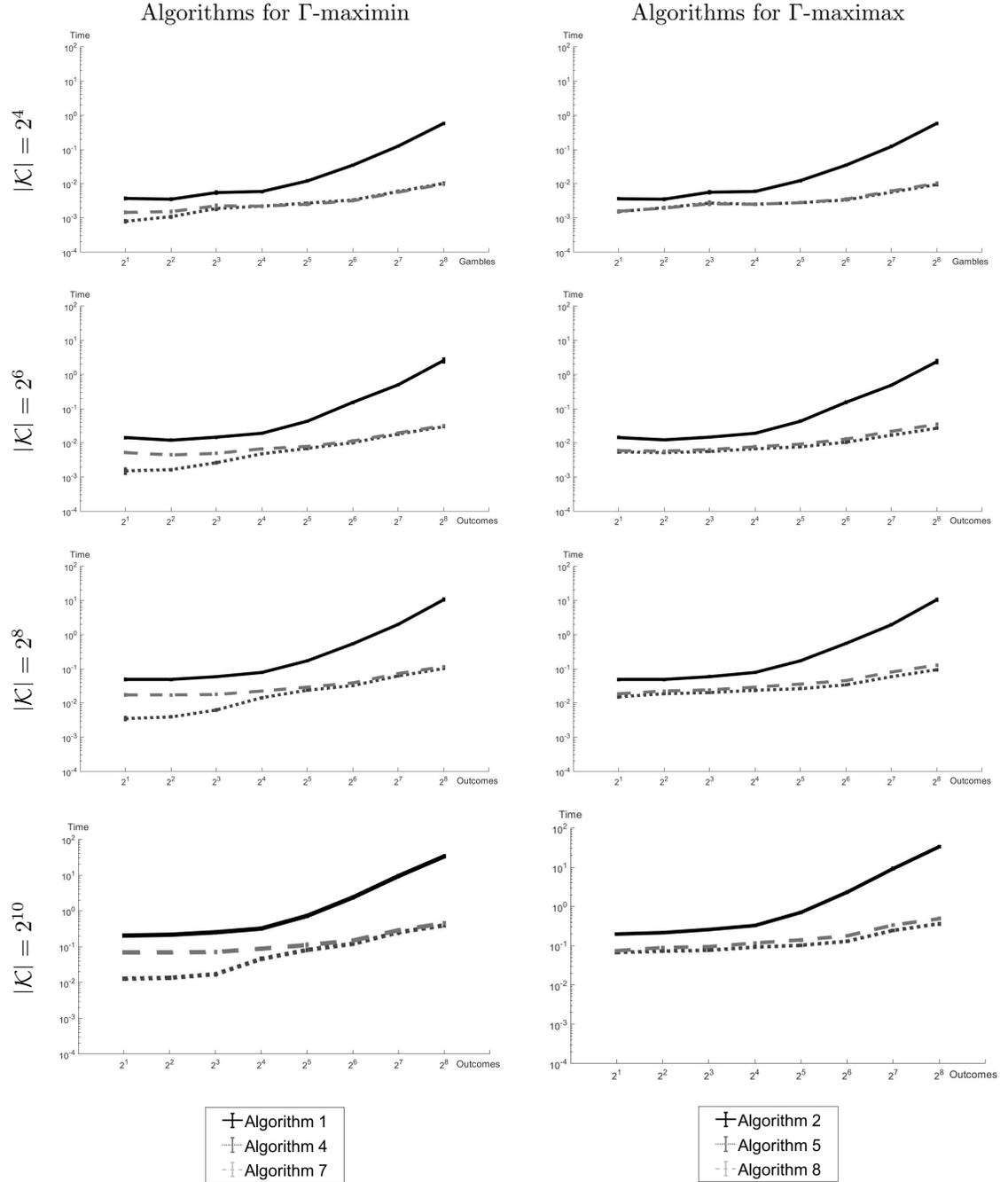
FIGURE 5. Comparison plots of the average total running time for different algorithms for Γ-maximin (the left column) and Γ-maximax (the right column). Each row shows a different number of gambles with varying numbers of outcomes and fixed $|\operatorname{dom}\underline{P}| = 2^4$. The labels indicate different algorithms.
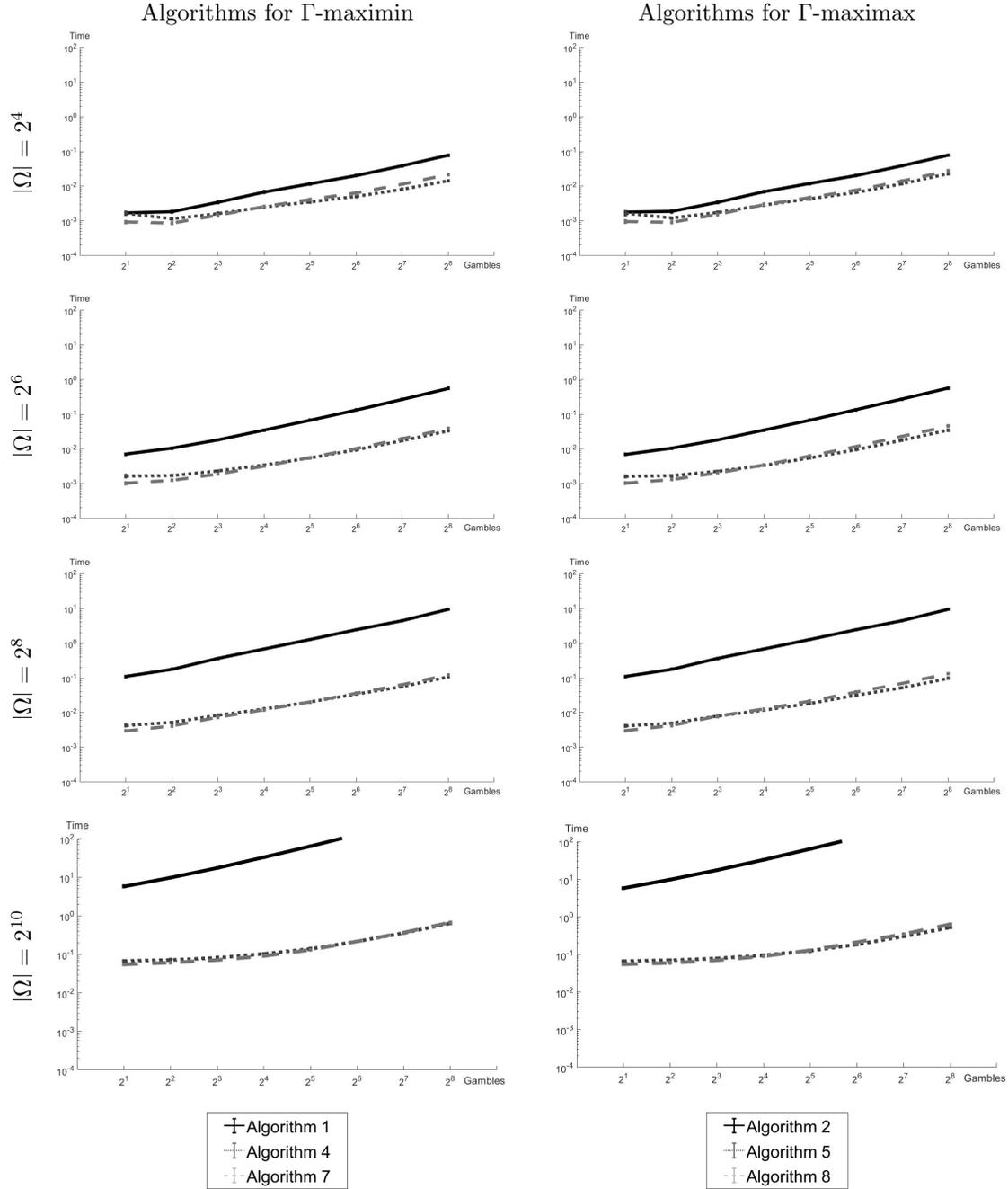
FIGURE 6. Comparison plots of the average total running time of different algorithms for Γ-maximin (the left column) and Γ-maximax (the right column). Each row shows a different number of outcomes with varying numbers of gambles and fixed $|\operatorname{dom} \underline{P}| = 2^4$. The labels indicate different algorithms.

investigate this, we also plot the computational time spent in the setup stage of the proposed algorithms (algorithms 4, 7, 5 and 8), i.e. finding feasible starting points and sorting gambles.

We find that in figs. 5, 6 and 7, the running times for finding Γ-maximin by algorithms 4 and 7 are nearly identical. Similarly, for algorithms 5 and 8, we see that the running time for finding Γ-maximax gambles is very similar. In addition to presenting average running times and error bars, we also provide some scatter plots of running time for finding Γ-maximin and Γ-maximax gambles in the case that they are very closed, i.e. comparing algorithm 4 against algorithm 7 and comparing algorithm 5 against algorithm 8 in figs. 8 and 9, respectively. This shows any simulations where one algorithm has performed better than another or perhaps whether the running times under the two algorithms are related to each other or not.

Figures 8 and 9 show scatter plots of running time for different pairs of algorithms for finding Γ-maximin gambles in the left column and Γ-maximax gambles in the right column. In fig. 8, each row shows a different number of gambles and in fig. 9, each row presents a different number of outcomes. In the left column where we compare running time taken for finding Γ-maximin gamble. The horizontal axis presents the running time of algorithm 4 and the vertical axis presents the running time of algorithm 7. In the right column, we compare running time for finding Γ-maximax gambles, where the horizontal axis presents the running time of algorithm 5 and the vertical axis presents the running time of algorithm 8.

4.2. **Interval dominance.** To benchmark the algorithms for interval dominance (algorithms 3, 6, 9 and 10), we will generate random sets of gambles that have a precise number of Γ-maximin and interval dominant gambles by using [11, algorithm 3]. We set the number of outcomes to be $|\Omega| = 2^4$ and $|\Omega| = 2^6$, the number of gambles in $\mathcal{K}$ to be $2^4, 2^6$ and 2^8, and the size of dom $\underline{P}$ to be 2^4 and 2^6.

To do so, we first generate $\underline{P}$ with size of dom $\underline{P}$ as we did before in benchmarking for Γ-maximin and Γ-maximax. Next, we apply [11, algorithm 3] to generate a random set $\mathcal{K}$ for $|\mathcal{K}| = k \in \{2^4, 2^6, 2^8\}$, $|\operatorname{opt}_{\underline{E}}(\mathcal{K})| = \ell$ and $|\operatorname{opt}_{\sqsupset}(\mathcal{K})| = n$ where $\ell \leq n \leq k$ by using the previous $\underline{P}$ to evaluate $\underline{E}$ and $\overline{E}$. To cover a range of possible options of ℓ, n, and k that satisfy $\ell \leq n \leq k$, we follows different options with respect to the size of $\mathcal{K}$ as suggested in [14, Table 1]; see table 1 and fig. 10 that illustrate these cases. Specifically, options a to d represent the sets of gambles with $\ell = 1$ while we increase n from 1 to k. Options d, g, i, and j represent the sets of gambles with $n = k$ while we increase ℓ from 1 to k. Options a, e, h, and j represent the sets of gambles that $\ell = n$ while we increase them jointly from 1 to k. Option f represents a set of gambles where $\ell < n < k$.

For each generated set of gambles $\mathcal{K}$, we apply algorithms 3, 6, 9 and 10 and record the total running time. For algorithm 3, we solve linear programs by the standard primal-dual method while for algorithms 6, 9 and 10, we solve the linear programs by the improved primal-dual method which has all improvements from section 3.2. Again, to ensure a fair comparison, these algorithms are written in MATLAB, and use identical implementations for solving the linear programs. For sorting gambles we used the built-in quicksort function. For each case, we repeat the process 500 times and present summaries in figs. 11 and 12.

Algorithm 9 is not depicted in the plots, as we have found it to perform nearly identically to algorithm 10. This is because there is a trade-off between effort spent on bounding the Γ-maximin value, and the number of comparisons required against these bounds, as discussed in section 3.3.

Figures 11 and 12 present the average total running time of three algorithms for interval dominance. In the left column, the number of outcomes is 2^4 while the right column, the number of outcomes is 2^6. Each row represents a different number of gambles in the set $\mathcal{K}$. The vertical
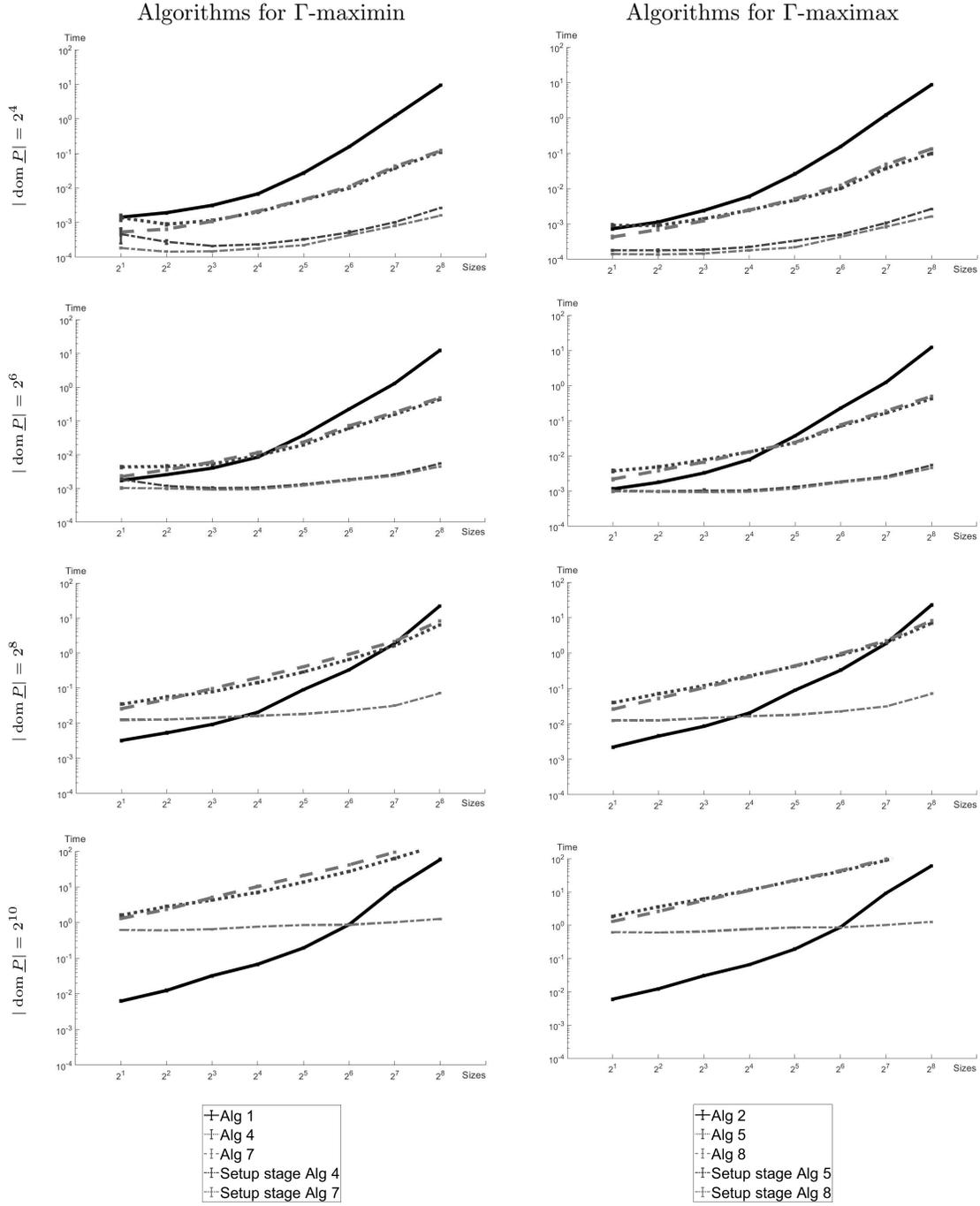
FIGURE 7. Comparison plots of the average running time for different improved algorithms for Γ-maximin (the left column) and Γ-maximax (the right column). Each row shows a different number of $|\operatorname{dom}\underline{P}|$ with varying numbers of gambles and outcomes, i.e. $|\Omega| = |\mathcal{K}| = 2^i$, for $i \in \{1, 2, \ldots, 8\}$. The labels indicate different algorithms.

Algorithms for Γ-maximin   Algorithms for Γ-maximax
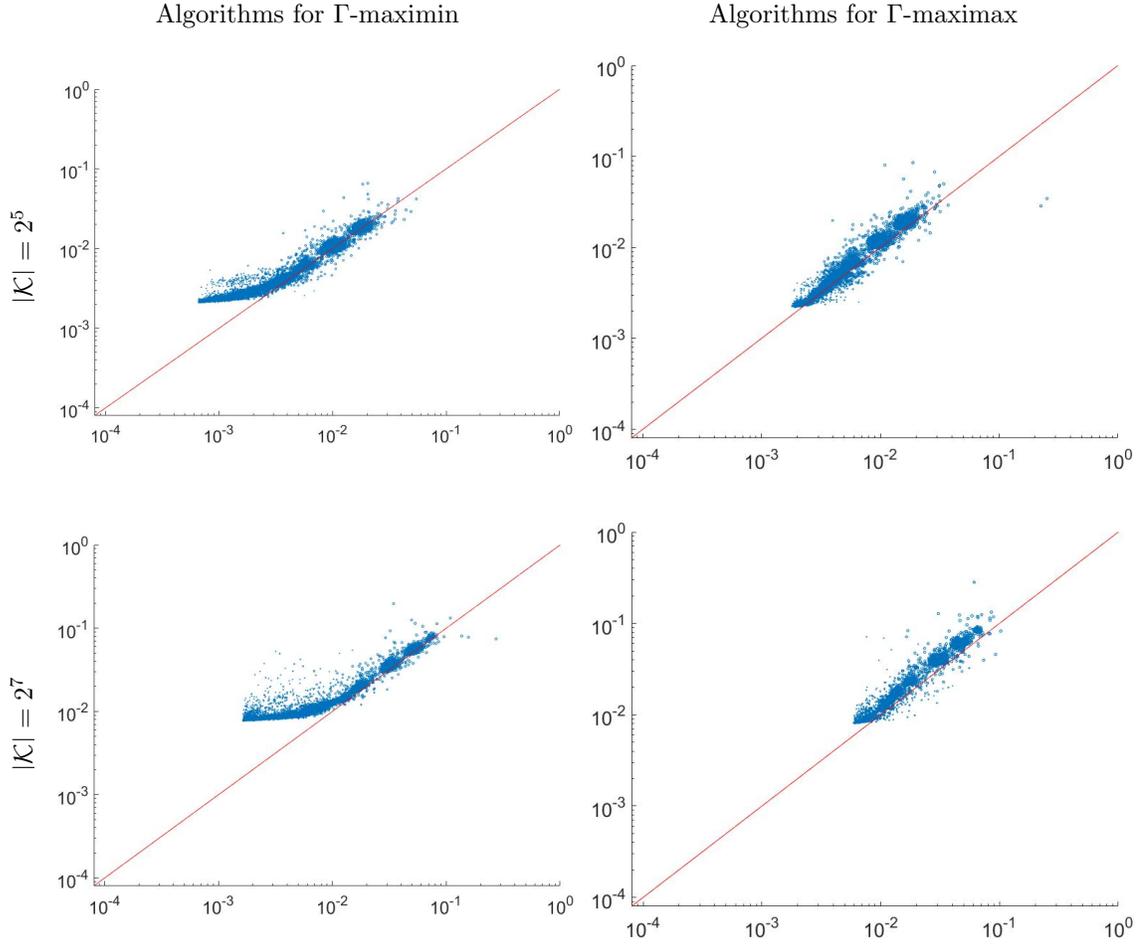
FIGURE 8. Comparison scatter plots of the running time for different pairs of algorithms for Γ-maximin (the left column) and Γ-maximax (the right column). Each represent a different number of gambles with varying numbers of outcomes and fixed $|\operatorname{dom} \underline{P}| = 2^4$. In the left column, the horizontal axis shows running time for algorithm 4 and the vertical axis shows running time for algorithm 7. In the right column, the horizontal axis shows running time for algorithm 5 and the vertical axis shows running time for algorithm 8. The red line is the identity line.

axis shows the average computational time over 500 random generated sets and the horizontal axis shows the different ten options of ℓ, n and k that we gave in table 1. The error bars representing 95% confidence intervals on the total running time are barely visible here.

5. DISCUSSION AND CONCLUSION

In this study, we proposed new algorithms for Γ-maximin (algorithms 4 and 7), for Γ-maximax (algorithms 5 and 8) and for interval dominance (algorithms 6, 9 and 10) and compared their

Algorithms for Γ-maximin   Algorithms for Γ-maximax
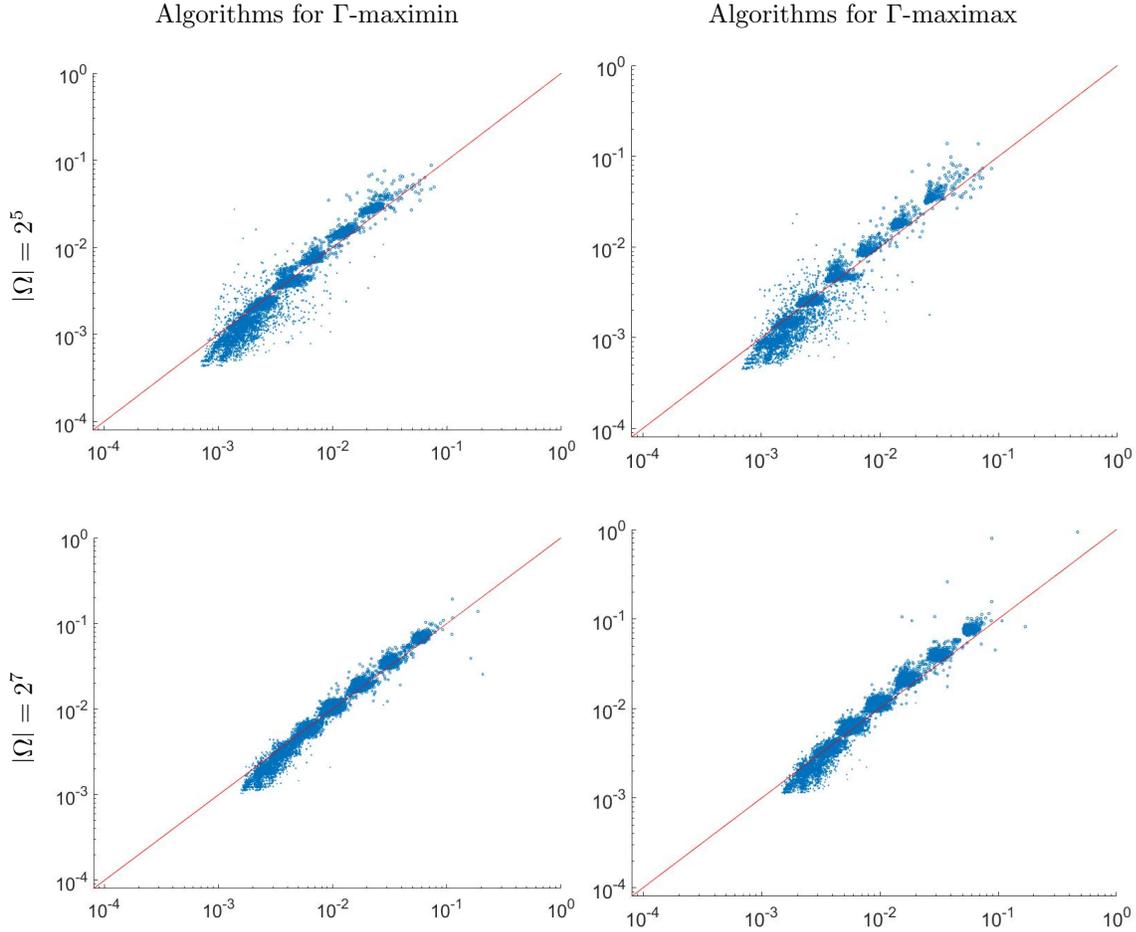
FIGURE 9. Comparison scatter plots of the running time for different pairs of algorithms for Γ-maximin (the left column) and Γ-maximax (the right column). Each presents a different number of outcomes with varying numbers of gambles and fixed $|\operatorname{dom} \underline{P}| = 2^4$. In the left column, the horizontal axis shows running time for algorithm 4 and the vertical axis shows running time for algorithm 7. In the right column, the horizontal axis shows running time for algorithm 5 and the vertical axis shows running time for algorithm 8. The red line is the identity line.

performance with the standard algorithms (algorithms 1, 2 and 3 for Γ-maximin, for Γ-maximax and for interval dominance respectively).

To find Γ-maximin and Γ-maximax gambles, the algorithms solve linear programs for each gamble in a set of gambles. For finding interval dominant gambles, the algorithms first need to solve a sequence of linear programs and then solve linear programs for each gamble in a set of gambles. Applying improvements from [14], for each case, we gave early stopping criteria and a quick way to compute feasible starting points for linear programs. We found that the primal-dual method can
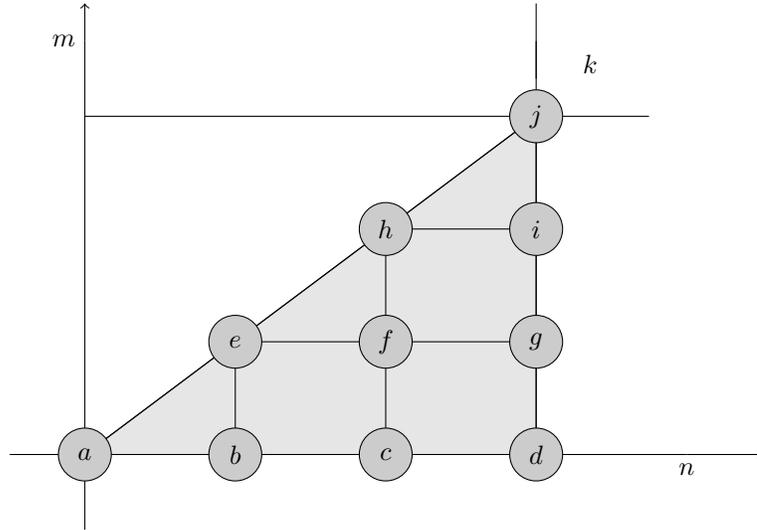
FIGURE 10. The area of $\ell \leq n \leq k$ and 10 options label the different ℓ and n that we consider in the simulation (see table 1)

Options	$|\mathcal{K}| = 2^4$		$|\mathcal{K}| = 2^6$		$|\mathcal{K}| = 2^8$	
	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 varying number of Γ-maximin gambles ℓ and number of interval dominant gambles n in $\mathcal{K}$
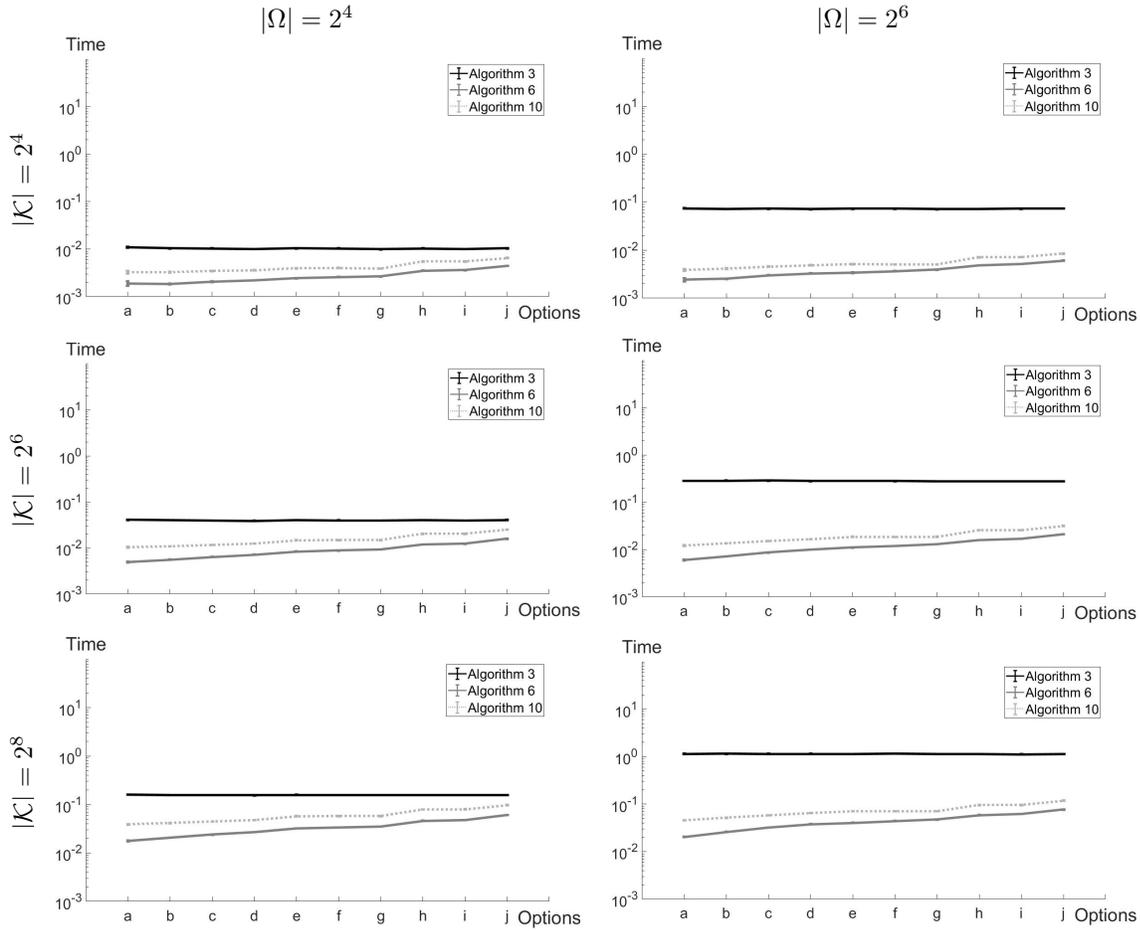
FIGURE 11. Comparison plots of the average computational time for different improved algorithms for finding interval dominant gambles where we fix $|\operatorname{dom} \underline{P}| = 2^4$. The number of outcomes in left column is 2^4 and 2^6 in the right column. Each row represents a different number of gambles with varying options of the numbers of Γ-maximin gambles and interval dominant gambles in the set (see table 1 for each option). The labels indicate different algorithms.

benefit from these improvements, as suggested in [14]. Our proposed algorithms for Γ-maximin, Γ-maximax and interval dominance implemented these improvements.

For benchmarking these improvements, we used [13, algorithms 2 and 4] to generate sets of gambles and applied algorithms 1, 4 and 7 for Γ-maximin and algorithms 2, 5 and 8 for Γ-maximax on these generated sets. For benchmarking algorithms for interval dominance, we used [11, algorithm 3] to generate sets of gambles with a precise number of Γ-maximin and interval dominant gambles and applied algorithms 3, 6, 9 and 10. According to our simulation, the numbers of outcomes and gambles in the set have an impact on the performance of these different algorithms. Specifically, the mean running time spent on the algorithm is larger if the numbers of outcomes or the number
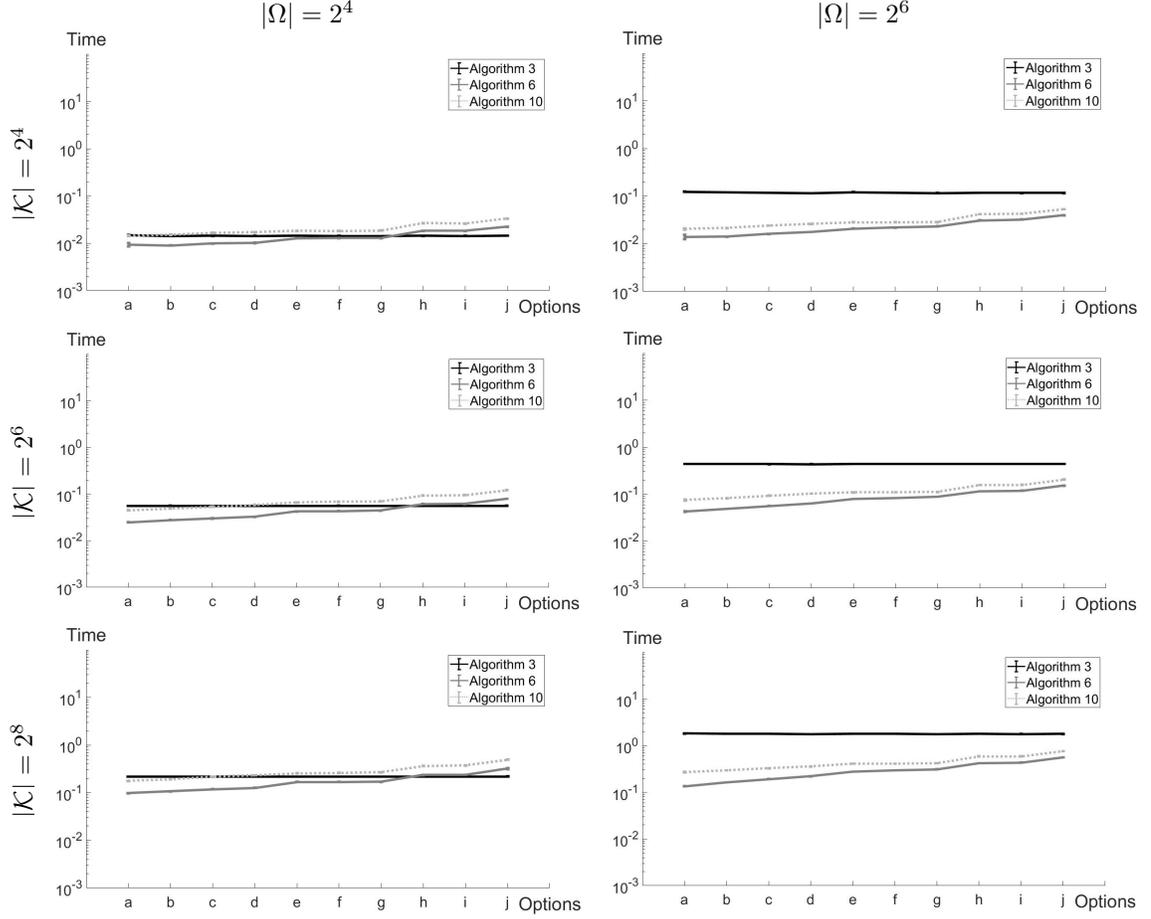
FIGURE 12. Comparison plots of the average computational time for different improved algorithms for finding interval dominant gambles where we fix $|\operatorname{dom} \underline{P}| = 2^6$. The number of outcomes in left column is 2^4 and 2^6 in the right column. Each row represents a different number of gambles with varying options of the numbers of Γ-maximin gambles and interval dominant gambles in the set (see table 1 for each option). The labels indicate different algorithms.

of gambles is increasing. The mean running time also depends on the size of the domain of the lower previsions.

Overall, both algorithms 4 and 7 outperform algorithm 1 for finding Γ-maximin gambles and both algorithms 5 and 8 outperform algorithm 2 for finding Γ-maximax gambles especially when either the number of gambles or the number of outcomes is large. There is no big difference in the time taken on algorithms 5 and 8 for finding Γ-maximax gambles. For the time taken on algorithms 4 and 7, there is also no big difference except when the number of gambles is large and the number of outcome is small (see fig. 5). In this specific case, algorithm 4 performs slightly better than algorithm 7.

Nevertheless, when we increased the size of the domain of lower previsions (see fig. 7), the result showed that the size of the domain has an impact on the setup stage of the proposed algorithms. We found that if the size of the domain is much larger than the sizes of gambles and outcomes, then algorithm 1 performs better than algorithms 4 and 7 while algorithm 2 performs better than algorithms 5 and 8. This is because algorithms 4, 7, 5 and 8 spent longer time in the setup stage in order to get feasible starting points. However, in the case that the size of the domain is less or equal to the sizes of gambles and outcomes, our proposed algorithms 4 and 7 outperform algorithm 1 and our proposed algorithms 5 and 8 outperform algorithm 2 as there is little effect from the setup stage in these proposed algorithms.

We note that algorithm 7 is the only algorithm that can return all Γ-maximin gambles while the other algorithms return just one. Therefore, if we suspect that there might be more than one Γ-maximin gamble in the set, and we are interested in finding them all, then we should apply algorithm 7.

For the result of algorithms for interval dominance, we find that our improved algorithms algorithms 6, 9 and 10 outperform algorithm 3 in most cases except when the size of the domain of lower previsions is much larger than the sizes of gambles and outcomes where our improved algorithms need to take time for the setup stage. If either the number of Γ-maximin or interval dominant gambles is increasing, then the computational time on algorithms 6, 9 and 10 is slightly longer while algorithm 3 remains the same. Algorithm 6 slightly performs better than algorithms 9 and 10.

Among the three proposed algorithms for interval dominance, we found that there was an interesting trade-off between being precise and the number of comparisons. In particular, if we aim to be precise at the beginning, that is, if we obtain a precise Γ-maximin value, then we require a smaller number of comparisons later on. On the other hand, if we allow being less precise at the beginning and start comparison immediately, then we could save computational effort as we may not need to identify the precise Γ-maximin value, at the expense of having to do more comparisons. It turns out that our new algorithm for finding the Γ-maximin value (algorithm 4) is fast enough to tip the balance in clear favour of algorithm 6.

Based on our benchmarking results from all scenarios considered, both of our proposed algorithms are good choices for implementation when the size of the domain is less or equal to the sizes of gambles and outcomes, as they outperform the standard algorithms, and otherwise perform quite similarly for the most part. However, when the size of the domain is much larger than the sizes of gambles and outcomes, the proposed algorithms may not be recommended due to the effect from the setup stage.

Acknowledgements

References

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

[2] Thierry Denoeux. Decision-making with belief functions: A review. *International Journal of Approximate Reasoning*, 109:87–110, 2019. ISSN 0888-613X. doi:10.1016/j.ijar.2019.03.009.

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

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

[5] 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.

[6] Daniel Kikuti, Fabio G. Cozman, and Ricardo S. Filho. Sequential decision making with partially ordered preferences. *Artificial Intelligence*, 175(7):1346–1365, 2011. ISSN 0004-3702. doi:10.1016/j.artint.2010.11.017. Representing, Processing, and Learning Preferences: Theoretical and Practical Challenges.

[7] MATLAB. *version 9.7.0.1190202 (R2019b)*. The MathWorks Inc., Natick, Massachusetts, 2019.

[8] Paul-Amaury Matt. Uses and computation of imprecise probabilities from statistical data and expert arguments. *International Journal of Approximate Reasoning*, 81:63–86, 2017. ISSN 0888-613X. doi:10.1016/j.ijar.2016.11.003.

[9] 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.

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

[11] Nawapon Nakharutai. Algorithms for generating sets of gambles for decision making with lower previsions. In Van-Nam Huynh, Tomoe Entani, Chawalit Jeenanunta, Masahiro Inuiguchi, and Pisal Yenradee, editors, *Integrated Uncertainty in Knowledge Modelling and Decision Making*, pages 62–71, Cham, 2020. Springer International Publishing.

[12] Nawapon Nakharutai, Matthias C. M. Troffaes, and Camila C. S. Caiado. Efficient algorithms for checking avoiding sure loss. In Alessandro Antonucci, Giorgio Corani, Inés Couso, and Sébastien Destercke, editors, *Proceedings of the Tenth International Symposium on Imprecise Probability: Theories and Applications*, volume 62 of *Proceedings of Machine Learning Research*, pages 241–252. PMLR, July 2017. URL http://proceedings.mlr.press/v62/nakharutai17a.html.

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

[14] Nawapon Nakharutai, Matthias C. M. Troffaes, and Camila C. S. Caiado. Improving and benchmarking of algorithms for decision making with lower previsions. *International Journal of Approximate Reasoning*, 113:91–105, October 2019. doi:10.1016/j.ijar.2019.06.008.

[15] 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.

[16] Matthias C. M. Troffaes. The crayfish problem. 11th Workshop on Principles and Methods of Statistical Inference with Interval Probability (WPMSIIP), 2018. URL http://bellman.ciencias.uniovi.es/~ssipta18/Slides/matthias-crayfish.pdf.

[17] Matthias C. M. Troffaes and Gert de Cooman. *Lower Previsions*. Wiley Series in Probability and Statistics. Wiley, 2014. ISBN 978-0-470-72377-7.

[18] Matthias C. M. Troffaes and John Paul Gosling. Robust detection of exotic infectious diseases in animal herds: A comparative study of three decision methodologies under severe uncertainty. *International Journal of Approximate Reasoning*, 53(8):1271–1281, November 2012. doi:10.1016/j.ijar.2012.06.020.

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

[20] Lev V. Utkin and Thomas Augustin. Powerful algorithms for decision making under partial prior information and general ambiguity attitudes. In Fabio G. Cozman, Robert Nau, and Teddy Seidenfeld, editors, *ISIPTA'05: Proceedings of the Fourth International Symposium on Imprecise Probabilities and Their Applications*, pages 349–358, July 2005.

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

Chiang Mai University, 239 Huaykaew Rd, Suthep, Mueang District, Chiang Mai 50200, Thailand
Email address: nawapon.nakharutai@cmu.ac.th

Durham University, Stockton Rd, Durham DH1 3LE, United Kingdom
Email address: matthias.troffaes@durham.ac.uk

Durham University, Stockton Rd, Durham DH1 3LE, United Kingdom
Email address: c.c.s.caiado@durham.ac.uk