Lexicographic refinements in stationary possibilistic Markov Decision Processes

Nahla Ben Amor, Zeineb El Khalfi, Hélène Fargier, Régis Sabbadin

Official URL

DOI : https://doi.org/10.1016/j.ijar.2018.10.011

Lexicographic refinements in stationary possibilistic Markov Decision Processes [☆]

Nahla Ben Amor [a,*], Zeineb El Khalfi [a,b,**], Hélène Fargier [b,*], Régis Sabbadin [c,*]

[a] *LARODEC, University of Tunis, Tunisia*
[b] *IRIT, UPS-CNRS, Université de Toulouse 3, 118 route de Narbonne, F-31062 Toulouse, France*
[c] *MIAT, UR 875, Université de Toulouse, INRA, F-31320 Castanet-Tolosan, France*

Keywords:
Markov Decision Process
Possibility theory
Lexicographic comparisons
Possibilistic qualitative utilities

A B S T R A C T

Possibilistic Markov Decision Processes offer a compact and tractable way to represent and solve problems of sequential decision under qualitative uncertainty. Even though appealing for its ability to handle qualitative problems, this model suffers from the *drowning effect* that is inherent to possibilistic decision theory. The present[1] paper proposes to escape the drowning effect by extending to stationary possibilistic MDPs the lexicographic preference relations defined by Fargier and Sabbadin [13] for non-sequential decision problems. We propose a value iteration algorithm and a policy iteration algorithm to compute policies that are optimal for these new criteria. The practical feasibility of these algorithms is then experimented on different samples of possibilistic MDPs.

1. Introduction

The classical paradigm for sequential decision making under uncertainty is the expected utility-based *Markov Decision Processes* (MDPs) framework [3,21], which assumes that the uncertain effects of actions can be represented by probability distributions and that utilities are additive. But the EU model does no suit problems where uncertainty and preferences are ordinal in essence.

Alternatives to the EU-based model have been proposed to handle ordinal preferences/uncertainty. Remaining within the probabilistic, quantitative, framework while considering ordinal preferences has led to *quantile-based* approaches [15,18,27,29,33]. Purely ordinal approaches to sequential decision under uncertainty have also been considered. In particular, possibilistic MDPs [1,6,22,24] form a purely qualitative decision model with an ordinal evaluation of plausibility and preference. In this model, uncertainty about the consequences of actions is represented by possibility distributions and utilities are also ordinal. The decision criteria are either the pessimistic qualitative utility or its optimistic counterpart [9]. Such degrees can be either elicited from experts, or by automatic learning approaches [23]. However, it is now well known that possibilistic

☆ This paper is part of the Virtual special issue on the 14th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU 2017), edited by Alessandro Antonucci, Laurence Cholvy and Odile Papini.

* Corresponding authors.
** Corresponding author at: IRIT, UPS-CNRS, 118 route de Narbonne, 31062 Toulouse, France.
E-mail addresses: nahla.benamor@gmx.com (N. Ben Amor), zeineb.khalfi@gmail.com (Z. El Khalfi), fargier@irit.fr (H. Fargier), regis.sabbadin@inra.fr (R. Sabbadin).

[1] This paper is an extended and revised version of two conference papers [4,5]. It includes the full proofs of the propositions presented in these preliminary papers, new algorithms (based on policy iteration) and new experiments.

https://doi.org/10.1016/j.ijar.2018.10.011

decision criteria suffer from a *drowning effect* [13]: plausible enough bad or good consequences may completely blur the comparison between policies, that would otherwise be clearly differentiable.

In [13], Fargier and Sabbadin have proposed *lexicographic refinements* of possibilistic criteria for the one-step decision case, in order to remedy the drowning effect. This work has recently been extended for (finite horizon) possibilistic decision trees [4]. In the present paper, we propose to study the interest of the lexicographic preference relations to stationary possibilistic Markov Decision Processes, a model that is more compact than decision trees and not limited to a finite horizon.

The paper is structured as follows: The next Section recalls the background about possibilistic decision theory and stationary possibilistic MDPs, including the drowning effect problem. Section 3 defines the lexicographic comparison of policies and presents a value iteration algorithm which computes a nearly optimal strategy in a limited number of iterations. Then, Section 4 proposes a lexicographic value iteration algorithm and a lexicographic policy iteration algorithm using approximation of utility functions. Lastly, Section 5 presents our experimental results.

2. Background and notations

2.1. Basics of possibilistic decision theory

Most of available decision models refer to probability theory for the representation of uncertainty [20,25]. Despite its success, probability theory is not appropriate when numerical information is not available. When information about uncertainty cannot be quantified in a probabilistic way, possibilistic theory [8,34] is a natural field to consider. The basic component of this theory is the notion of possibility distribution. It is a representation of a state of knowledge of an agent about the state of the world. A possibility distribution π is a mapping from the universe of discourse S (the set of all the possible worlds) to a bounded linearly ordered scale L exemplified (without loss of generality) by the unit interval $[0,1]$, we denote the function by: $\pi : S \to [0,1]$.

For state $s \in S$, $\pi(s) = 1$ means that realization s is totally possible and $\pi(s) = 0$ means that s is an impossible state. It is generally assumed that there exists at least one state s which is totally possible: π is then said to be *normalized*.

In the possibilistic framework, extreme forms of knowledge can be captured, namely:

- Complete knowledge i.e. $\exists\ s$ s.t. $\pi(s) = 1$ and $\forall\ s' \neq s, \pi(s') = 0$.
- Total ignorance i.e. $\forall s \in S, \pi(s) = 1$ (all values in S are possible).

From π one can compute the possibility measure $\Pi(A)$ and the necessity measure $N(A)$ of any event $A \subseteq S$:

$$\Pi(A) = \sup_{s \in A} \pi(s)$$
$$N(A) = 1 - \Pi(\bar{A}) = 1 - \sup_{s \notin A} \pi(s)$$

Measure $\Pi(A)$ evaluates to which extent A is consistent with the knowledge represented by π while $N(A)$ corresponds to the extent to which $\neg A$ is impossible and thus evaluates at which level A is certainly implied by the knowledge.

In decision theory acts are functions $f\colon S \mapsto X$, where X is a finite set of outcomes. In possibilistic decision making, an act f can be viewed as a possibility distribution π_f over X [9], where $\pi_f(x) = \Pi(f^{-1}(x))$. In a single stage decision making problem, a utility function $u : X \mapsto U$ maps outcomes to utility values in a totally ordered scale $U = \{u_1, \ldots, u_n\}$. This function models the attractiveness of each outcome for the decision-maker.

Under the assumption that the utility scale and the possibility scale are commensurate and purely ordinal (i.e. $U = L$), Dubois and Prade [9,7] have proposed pessimistic and optimistic decision criteria.

First, the pessimistic criterion was originally proposed by Whalen [30] and it generalizes the Wald criterion [28]. It suits cautious decision makers who are happy when bad consequences are hardly plausible. It summarizes to what extent it is certain (i.e. necessary according to measure N) that the act reaches a good utility. The definition of the pessimistic criterion is as follows [10]:

Definition 1. Given a possibility distribution π over a set of states S and a utility function u on the set of consequences X, the pessimistic utility of an act f is defined by:

$$\begin{aligned} u_{pes}(f) &= \min_{x_j \in X} \max(u(x_j), 1 - \pi_f(x_j)), \\ &= \min_{s_i \in S} \max(u(f(s_i)), 1 - \pi(s_i)). \end{aligned} \tag{1}$$

Therefore, we can compare two acts f and g on the basis of their pessimistic utilities:

$$f \succeq_{u_{pes}} g \Leftrightarrow u_{pes}(f) \geq u_{pes}(g).$$

The second criterion is the optimistic possibilistic criterion originally proposed by Yager [32,31]. This criterion captures the behavior of an adventurous decision maker who is happy as soon as at least one good consequence is highly plausible. It summarizes to what extent it is possible that an act reaches a good utility. The definition of this criterion is as follows [10]:

Definition 2. Given a possibility distribution π over a set of states S and a utility function u on a set of consequences X, the optimistic utility of an act f is defined by:

$$\begin{aligned} u_{opt}(f) &= \max_{x_j \in X} \min(u(x_j), \pi_f(x_j)), \\ &= \max_{s_i \in S} \min(u(f(s_i)), \pi(s_i)). \end{aligned} \tag{2}$$

Hence, we can compare two acts f and g on the basis of their optimistic utilities:

$$f \succeq_{u_{opt}} g \Leftrightarrow u_{opt}(f) \geq u_{uopt}(g).$$

Example 1. Let $S = \{s_1, s_2\}$ and f and g be two acts whose utilities of consequences in the states s_1 and s_2 are listed in the following table, as well as the degrees of possibility of s_1 and s_2:

	s_1	s_2
$u(f(s))$	0.3	0.5
$u(g(s))$	0.4	0.6
π	1	0.2

Comparing f and g with respect to the pessimistic criterion, we get:

- $u_{pes}(f) = min(max(0.3, 0), max(0.5, 0.8)) = 0.3$,
- $u_{pes}(g) = min(max(0.4, 0), max(0.6, 0.8)) = 0.4$.

Thus, $g \succeq_{u_{pes}} f$.

Let us now compare the two acts with respect to the optimistic criterion:

- $u_{opt}(f) = max(min(0.3, 1), min(0.5, 0.2)) = 0.3$,
- $u_{opt}(g) = max(min(0.4, 1), min(0.6, 0.2)) = 0.4$.

Thus, $g \succeq_{u_{opt}} f$.

It is important to note that while transition probabilities can be estimated through simulations of the process, transition possibilities may not. On the other hand, experts may be involved for the elicitation of the possibility degrees and utilities of transitions. In the possibilistic framework, utility and uncertainty levels can be elicited jointly, by comparison of possibilistic lotteries, for example (e.g. by using certainty equivalents, as in [11]). Simulation can also be used jointly with expert evaluation when the underlying process is too costly to simulate a large number of times: simulation may be used to generate samples on which expert elicitation is applied. Another option is to use possibilistic reinforcement learning procedure (for more details see [23]), in particular model-based reinforcement learning algorithm. The latter uses a uniform simulation of trajectories (with random choice of actions) in order to generate an approximation of the possibilistic decision model.

2.2. Stationary Possibilistic Markov Decision Processes

A stationary Possibilistic Markov Decision Process ($\Pi\mathcal{MDP}$) [22] is defined by:

- A finite set S of **states**;
- A finite set A of **actions**, A_s denotes the set of actions available in state s;
- A possibilistic transition function: for each action $a \in A_s$ and each state $s \in S$ the possibility distribution $\pi(s'|s, a)$ evaluates to what extent each s' is a possible successor of s when action a is applied;
- A utility function μ: $\mu(s)$ is the intermediate satisfaction degree obtained in state s.

The uncertainty about the effect of an action a taken in state s is captured by a possibility distribution $\pi(.|s, a)$. In the present paper, we consider stationary problems, i.e. problems in which the states, the actions and the transition functions do not depend on the stage of the problem. Such a possibilistic MDP may define a graph where states are represented by circles and each state “s” is labeled with a utility degree, and actions are represented by squares. An edge linking an action to a state denotes a possible transition and is labeled by the possibility of that state given the action is executed.

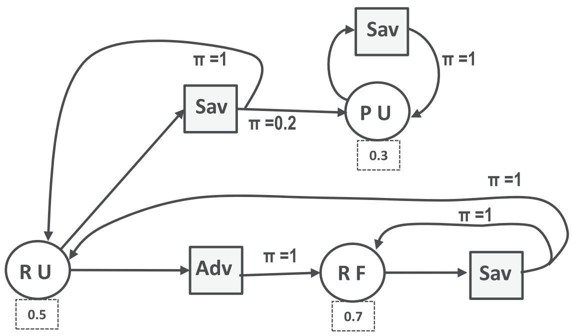

Fig. 1. The stationary $\Pi\mathcal{MDP}$ of Example 2.

Example 2. Let us suppose that a "Rich and Unknown" person runs a startup company. Initially, s/he must choose between Saving money (Sav) or Advertising (Adv) and may then get Rich (R) or Poor (P) and Famous (F) or Unknown (U). In the other states, Sav is the only possible action. Fig. 1 shows the stationary $\Pi\mathcal{MDP}$ that captures this problem, formally described as follows:

$S = \{RU, RF, PU\}$,
$A_{RU} = \{Adv, Sav\}$, $A_{RF} = A_{PU} = \{Sav\}$,
$\pi(PU|RU, Sav) = 0.2$,
$\pi(RU|RU, Sav) = \pi(RF|RU, Adv) = \pi(RF|RF, Sav) = \pi(RU|RF, Sav) = 1$,
$\mu(RU) = 0.5$, $\mu(RF) = 0.7$, $\mu(PU) = 0.3$.

Solving a stationary MDP consists in finding a (stationary) policy, i.e. a function $\delta\colon S \to A$ which is optimal with respect to a decision criterion. In the possibilistic case, as in the probabilistic case, the value of a policy depends on the utility and on the likelihood of its trajectories. Formally, let Δ be the set of all policies that can be built for the $\Pi\mathcal{MDP}$ (the set of all the functions that associate an element of A_s to each s). Each $\delta \in \Delta$ defines a list of scenarios called trajectories. Each trajectory τ is a sequence of states and actions i.e. $\tau = (s_0, a_0, s_1, \ldots, s_{t-1}, \; a_{t-1}, s_t)$.

To simplify notations, we will associate the vector $v_\tau = (\mu_0, \pi_1, \mu_1, \pi_2, \ldots, \pi_{t-1}, \mu_t)$ to each trajectory τ, where $\pi_{i+1} = \pi(s_{i+1}|s_i, a_i)$ is the possibility degree to reach the state s_{i+1} at $t = i + 1$, applying the action a_i at $t = i$ and $\mu_i = \mu(s_i)$ is the utility obtained in the i-th state s_i of the trajectory.

The possibility and the utility of trajectory τ given that δ is applied from s_0 are defined by:

$$\pi(\tau|s_0, \delta) = \min_{i=1\ldots t} \pi(s_i|s_{i-1}, \delta(s_{i-1})) \text{ and } \mu(\tau) = \min_{i=0\ldots t} \mu(s_i). \tag{3}$$

Two criteria, an optimistic and a pessimistic one, can then be used to evaluate δ [24,9]:

$$u_{opt}(\delta, s_0) = \max_\tau \min\{\pi(\tau|s_0, \delta), \mu(\tau)\}, \tag{4}$$

$$u_{pes}(\delta, s_0) = \min_\tau \; max\{1 - \pi(\tau|s_0, \delta), \mu(\tau)\}. \tag{5}$$

The policies optimizing these criteria can be computed by applying, for every state s and time step $i = 0, \ldots, t$, the following counterparts of the Bellman updates [22]:

$$u_{opt}(s, i) \leftarrow \max_{a \in A_s} \min\{\mu(s), \max_{s' \in S} \min(\pi(s'|s, a), u_{opt}(s', i + 1))\}, \tag{6}$$

$$u_{pes}(s, i) \leftarrow \max_{a \in A_s} \min\{\mu(s), \min_{s' \in S} \max(1 - \pi(s'|s, a), u_{pes}(s', i + 1))\}, \tag{7}$$

$$\delta_{opt}(s, i) \leftarrow \arg\max_{a \in A_s} \min\{\mu(s), \max_{s' \in S} \min(\pi(s'|s, a), u_{opt}(s', i + 1))\}, \tag{8}$$

$$\delta_{pes}(s, i) \leftarrow \arg\max_{a \in A_s} \min\{\mu(s), \min_{s' \in S} \max(1 - \pi(s'|s, a), u_{pes}(s', i + 1))\}. \tag{9}$$

There we set, arbitrarily, $u_{opt}(s', t + 1)) = 1$ and $u_{pes}(s', t + 1)) = 1$.

It has allowed the definition of a (possibilistic) *value iteration* algorithm (see Algorithm 1 for the optimistic variant of this algorithm) which converges to an optimal policy in polytime [22].

This algorithm proceeds by iterated modifications of a possibilistic value function $Q(s, a)$ which evaluates the "utility" (pessimistic or optimistic) of performing a in s.

Another algorithm, (possibilistic) Policy Iteration (Algorithm 2 for the optimistic variant) is proposed in [22] for solving possibilistic stationary, infinite horizon MDPs. Policy Iteration alternates steps of evaluation of the current policy with steps of greedy improvement of the current policy.

Algorithm 1: $VI\text{-}MDP$: Possibilistic (Optimistic) Value iteration.

Data: A stationary $\Pi\mathcal{MDP}$
Result: A policy δ optimal for u_{opt}

begin
  foreach $s \in S$ **do** $u_{opt}(s) \leftarrow \mu(s)$;
  repeat
    foreach $s \in S$ **do**
      $u_{old}(s) \leftarrow u_{opt}(s)$;
      foreach $a \in A$ **do**
        $Q(s,a) \leftarrow \min\left\{\mu(s), \max_{s' \in S} \min\{(\pi(s'|s,a), u_{opt}(s')\}\right\}$;
        $u_{opt}(s) \leftarrow \max_a Q(s,a)$;
        $\delta(s) \leftarrow \arg\max_a Q(s,a)$;
  until $u_{opt}(s) == u_{old}(s)$ *for each* s;
  return δ;

Algorithm 2: $PI\text{-}MDP$: Possibilistic (Optimistic) Policy iteration.

Data: A stationary $\Pi\mathcal{MDP}$
Result: A policy δ optimal for u_{opt}

begin
  // Initialization of δ and u_{opt}
  foreach $s \in S$ **do**
    $\delta(s) \leftarrow$ choose any $a_s \in A_s$;
    $u_{opt}(s) \leftarrow \mu(s)$;
  repeat
    // Evaluation of δ until stabilization of u_{opt}
    repeat
      foreach $s \in S$ **do**
        $u_{old}(s) \leftarrow u_{opt}(s)$;
        $u_{opt}(s) \leftarrow \min\left\{\mu(s), \max_{s' \in S} \min\{\pi(s'|s,\delta(s)).u_{old}(s')\}\right\}$;
    until $u_{opt} == u_{old}$;
    // Improvement of δ
    foreach $s \in S$ **do**
      $\delta_{old}(s) \leftarrow \delta(s)$;
      $\delta(s) \leftarrow \arg\max_{a \in A} \min\left\{\mu(s), \max_{s' \in S} \min\{\pi(s'|s,a).u_{opt}(s')\}\right\}$;
  until $\delta(s) == \delta_{old}(s)$ *for each* s;
  // stabilization of δ
  return δ;

2.3. The drowning effect in stationary sequential decision problems

Unfortunately, possibilistic utilities suffer from an important drawback called the *drowning effect*: plausible enough bad or good consequences may completely blur the comparison between acts that would otherwise be clearly differentiated; as a consequence, an optimal policy δ is not necessarily Pareto efficient. Recall that a policy δ is Pareto efficient when no other policy δ' dominates it (i.e. there is no policy δ' such that (i) $\forall\, s \in S, u_{pes}(\delta', s) \succeq u_{pes}(\delta, s)$ and (ii) $\exists\, s \in S$ s.t. $u_{pes}(\delta', s) \succ u_{pes}(\delta, s)$). The following example shows that it can simultaneously happen that δ' dominates δ and $u_{pes}(\delta) = u_{pes}(\delta')$.

Example 3. The $\Pi\mathcal{MDP}$ of Example 2 admits two policies δ and δ':

- $\delta(RU) = Sav$; $\delta(PU) = Sav$; $\delta(RF) = Sav$;
- $\delta'(RU) = Adv$; $\delta'(PU) = Sav$; $\delta'(RF) = Sav$.

Consider a fixed horizon $H = 2$:

- δ has 3 trajectories:
  $\tau_1 = (RU, PU, PU)$ with $v_{\tau_1} = (0.5, 0.2, 0.3, 1, 0.3)$;
  $\tau_2 = (RU, RU, PU)$ with $v_{\tau_2} = (0.5, 1, 0.5, 0.2, 0.3)$;
  $\tau_3 = (RU, RU, RU)$ with $v_{\tau_3} = (0.5, 1, 0.5, 1, 0.5)$.

- δ' has 2 trajectories:
  $\tau_4 = (RU, RF, RF)$ with $v_{\tau_4} = (0.5, 1, 0.7, 1, 0.7)$;
  $\tau_5 = (RU, RF, RU)$ with $v_{\tau_5} = (0.5, 1, 0.7, 1, 0.5)$.

Thus $u_{opt}(\delta, RU) = u_{opt}(\delta', RU) = 0.5$. However δ' seems better than δ since it provides utility 0.5 for sure while δ provides a bad utility (0.3) in some non-impossible trajectories (τ_1 and τ_2). τ_3 which is good and totally possible "drowns" τ_1 and τ_2: δ is considered as good as δ'.

3. Bounded iterations solutions to lexicographic finite horizon $\Pi\mathcal{MDP}$s

Possibilistic decision criteria, especially pessimistic and optimistic utilities, are simple and realistic as illustrated in Section 2, but they have an important shortcoming: the principle of Pareto efficiency is violated since these criteria suffer from the drowning effect. Indeed, one decision may dominate another one while not being strictly preferred. In order to overcome the drowning effect, some refinements of possibilistic utilities have been proposed in the non-sequential case such as lexicographic refinements, proposed by [12,13]. These refinements are fully in accordance with ordinal utility theory and satisfy the principle of Pareto dominance, that is why we have chosen to focus on them.

The present section defines an extension of lexicographic refinements to finite horizon possibilistic Markov decision processes and proposes a value iteration algorithm that looks for policies optimal with respect to these criteria.

3.1. Lexi-refinements of ordinal aggregations

In ordinal (i.e. min-based and max-based) aggregation a solution to the drowning effect based on leximin and leximax comparisons has been proposed by [19]. It has then been extended to non-sequential decision making under uncertainty [13] and, in the sequential case, to decision trees [4]. Let us first recall the basic definition of these two preference relations. For any two vectors t and t' of length m built on the scale L:

$$t \succeq_{lmin} t' \text{ iff } \forall i, t_{\sigma(i)} = t'_{\sigma(i)} \text{ or } \exists i^*, \forall i < i^*, t_{\sigma(i)} = t'_{\sigma(i)} \text{ and } t_{\sigma(i^*)} > t'_{\sigma(i^*)}, \tag{10}$$

$$t \succeq_{lmax} t' \text{ iff } \forall i, t_{\mu(i)} = t'_{\mu(i)} \text{ or } \exists i^*, \forall i < i^*, t_{\mu(i)} = t'_{\mu(i)} \text{ and } t_{\mu(i^*)} > t'_{\mu(i^*)}, \tag{11}$$

where, for any vector v (here, $v = t$ or $v = t'$), $v_{\mu(i)}$ (resp. $v_{\sigma(i)}$) is the i-th best (resp. worst) element of v.

[13,4] have extended these procedures to the comparison of matrices built on L, defining preference relations $\succeq_{lmin(lmax)}$ and $\succeq_{lmax(lmin)}$:

$$\begin{aligned} A \succeq_{lmin(lmax)} B \Leftrightarrow\ & \forall j, a_{(lmax,j)} \cong b_{(lmax,j)} \\ & \textit{or } \exists i \text{ s.t. } \forall j > i, a_{(lmax,j)} \sim_{lmin} b_{(lmax,j)} \text{ and } a_{(lmax,i)} \succ_{lmin} b_{(lmax,i)}, \end{aligned} \tag{12}$$

$$\begin{aligned} A \succeq_{lmax(lmin)} B \Leftrightarrow\ & \forall j, a_{(lmin,j)} \sim_{lmax} b_{(lmin,j)} \\ & \textit{or } \exists i \text{ s.t. } \forall j < i, a_{(lmin,j)} \sim_{lmax} b_{(lmin,j)} \text{ and } a_{(lmin,i)} \succ_{lmax} b_{(lmin,i)}, \end{aligned} \tag{13}$$

where $a_{(\trianglerighteq,i)}$ (resp. $b_{(\trianglerighteq,i)}$) is the i-th largest sub-vector of A (resp. B) according to $\trianglerighteq \in \{lmax, lmin\}$.

Like in (finite-horizon) possibilistic decision trees [4] our idea is to identify the strategies of the MDP with the matrices of their trajectories, and to compare such matrices with a $\succeq_{lmax(lmin)}$ (resp. $\succeq_{lmin(lmax)}$) procedure for the optimistic (resp. pessimistic) case.

3.2. Lexicographic comparisons of policies

Let us first define lexicographic comparisons of policies over a given horizon E.

A trajectory over horizon E being a sequence of states and actions, any stationary policy can be identified with a matrix where each line corresponds to a distinct trajectory of length E. In the optimistic case each line corresponds to a vector $v_\tau = (\mu_0, \pi_1, \mu_1, \pi_2, \ldots, \pi_{E-1}, \mu_E)$ and in the pessimistic case to $w_\tau = (\mu_0, 1-\pi_1, \mu_1, 1-\pi_2, \ldots, 1-\pi_{E-1}, \mu_E)$.

This allows us to define the comparison of trajectories using leximax and leximin as follows:

$$\tau \succeq_{lmin} \tau' \text{ iff } (\mu_0, \pi_1, \ldots, \pi_E, \mu_E) \succeq_{lmin} (\mu'_0, \pi'_2, \ldots, \pi'_E, \mu'_E), \tag{14}$$

$$\tau \succeq_{lmax} \tau' \text{ iff } (\mu_0, 1-\pi_1, \ldots, 1-\pi_E, \mu_E) \succeq_{lmax} (\mu'_0, 1-\pi'_1, \ldots 1-\pi'_E, \mu'_E). \tag{15}$$

Note that the above preference relations implicitly depend on the horizon E and the same holds for stationary policies comparison. We leave aside any reference to E as the dependence will be clear from the context.

Using (14) and (15), we can compare policies by:

$$\delta \succeq_{lmax(lmin)} \delta' \text{ iff } \forall i,\ \tau_{\mu(i)} \sim_{lmin} \tau'_{\mu(i)}$$
$$or\ \exists i^*,\ \forall i < i^*, \tau_{\mu(i)} \sim_{lmin} \tau'_{\mu(i)}\ \ and\ \ \tau_{\mu(i^*)} \succ_{lmin} \tau'_{\mu(i^*)}, \tag{16}$$
$$\delta \succeq_{lmin(lmax)} \delta' \text{ iff } \forall i,\ \tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)}$$
$$or\ \exists i^*,\ \forall i < i^*, \tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)}\ \ and\ \ \tau_{\sigma(i^*)} \succ_{lmax} \tau'_{\sigma(i^*)}, \tag{17}$$

where $\tau_{\mu(i)}$ (resp. $\tau'_{\mu(i)}$) is the i-th best trajectory of δ (resp. δ') according to $\succeq_{lmin}$ and $\tau_{\sigma(i)}$ (resp. $\tau'_{\sigma(i)}$) is the i-th worst trajectory of δ (resp. δ') according to $\succeq_{lmax}$.

Hence, the utility degree of a policy δ can be represented by a matrix U_δ with n lines, s.t. n is the number of trajectories, and $m = 2E + 1$ columns. Indeed, comparing two policies w.r.t. $\succeq_{lmax(lmin)}$ (resp. $\succeq_{lmin(lmax)}$) consists in first ordering the two corresponding matrices of trajectories as follows:

- order the elements of each trajectory (i.e. the elements of each line) in increasing order w.r.t. $\succeq_{lmin}$ (resp. in decreasing order w.r.t. $\succeq_{lmax}$),
- then order all the trajectories. The lines of each policy are arranged lexicographically top-down in decreasing order (resp. top-down in increasing order).

Then, it is enough to lexicographically compare the two new matrices of trajectories, denoted U_δ (resp. $U_{\delta'}$), element by element. The first pair of different elements determines the best matrix/policy. Note that the ordered matrix U_δ (resp. $U_{\delta'}$) can be seen as the utility of applying policy δ (resp. δ') over a length E horizon.

Example 4. Let us consider the Counter-Example 3 with the same $\Pi\mathcal{MDP}$ of Example 2. We consider, once again, the policies δ and δ' defined by:

- $\delta(RU) = Sav; \delta(PU) = Sav; \delta(RF) = Sav;$
- $\delta'(RU) = Adv; \delta'(PU) = Sav; \delta'(RF) = Sav.$

For horizon $H = 2$:

- δ has 3 trajectories:
  $\tau_1 = (RU, PU, PU)$ with $v_{\tau_1} = (0.5, 0.2, 0.3, 1, 0.3)$;
  $\tau_2 = (RU, RU, PU)$ with $v_{\tau_2} = (0.5, 1, 0.5, 0.2, 0.3)$;
  $\tau_3 = (RU, RU, RU)$ with $v_{\tau_3} = (0.5, 1, 0.5, 1, 0.5)$.
  The matrix of trajectories is: $U_\delta = \begin{bmatrix} 0.5 & 0.2 & 0.3 & 1 & 0.3 \\ 0.5 & 1 & 0.5 & 0.2 & 0.3 \\ 0.5 & 1 & 0.5 & 1 & 0.5 \end{bmatrix} \sim \begin{bmatrix} 0.2 & 0.3 & 0.3 & 0.5 & 1 \\ 0.2 & 0.3 & 0.5 & 0.5 & 1 \\ 0.5 & 0.5 & 0.5 & 1 & 1 \end{bmatrix}$.
  So, the ordered matrix of trajectories is: $U_\delta = \begin{bmatrix} 0.5 & 0.5 & 0.5 & 1 & 1 \\ 0.2 & 0.3 & 0.3 & 0.5 & 1 \\ 0.2 & 0.3 & 0.5 & 0.5 & 1 \end{bmatrix}$.
- δ' has 2 trajectories:
  $\tau_4 = (RU, RF, RF)$ with $v_{\tau_4} = (0.5, 1, 0.7, 1, 0.7)$;
  $\tau_5 = (RU, RF, RU)$ with $v_{\tau_5} = (0.5, 1, 0.7, 1, 0.5)$.
  The ordered matrix of trajectories is: $U_{\delta'} = \begin{bmatrix} 0.5 & 0.7 & 0.7 & 1 & 1 \\ 0.5 & 0.5 & 0.7 & 1 & 1 \end{bmatrix}$.

Given the two ordered matrices U_δ and $U_{\delta'}$, δ and δ' are indifferent for optimistic utility since the two first (i.e. top-left) elements of the matrices are equal i.e. $u_{opt}(\delta) = u_{opt}(\delta') = 0.5$. For *lmax*(*lmin*) we compare successively the next elements (left to right then top to bottom) until we find a pair of different values. In particular, we have the second element of the first (i.e. the best) trajectory of δ' is strictly greater than the second element of the first trajectory of δ ($0.7 > 0.5$). So, the first trajectory of δ' is strictly preferred to the first trajectory of δ according to $\succeq_{lmin}$. We deduce that δ' is strictly preferred to δ:

$$\delta' \succ_{lmax(lmin)} \delta \text{ since } (0.5, 0.7, 0.7, 1, 1) \succ_{lmin} (0.5, 0.5, 0.5, 1, 1).$$

The following propositions can be shown, concerning the fixed horizon comparison of stationary policies. Note again that the dependence on E is left implicit.

Proposition 1.

If $u_{opt}(\delta) > u_{opt}(\delta')$ *then* $\delta \succ_{lmax(lmin)} \delta'$.
If $u_{pes}(\delta) > u_{pes}(\delta')$ *then* $\delta \succ_{lmin(lmax)} \delta'$.

Proposition 2. $\succeq_{lmax(lmin)}$ *and* $\succeq_{lmin(lmax)}$ *satisfy the principle of Pareto efficiency.*

Now, in order to design dynamic programming algorithms, i.e. to extend the value iteration algorithm to lexi comparison, we show that the comparison of policies is a preorder and satisfies the principle of strict monotonicity defined as follows for any optimization criterion O by: $\forall \delta, \delta', \delta'' \in \Delta$,

$$\delta \succeq_O \delta' \iff \delta + \delta'' \succeq_O \delta' + \delta'',$$

where δ (resp. δ') and δ'' denote two disjoint sets of trajectories and $\delta + \delta''$ (resp. $\delta' + \delta''$) is the set of trajectories that gathers the ones of δ (resp. δ') and the ones of δ''.

Then, adding or removing identical trajectories to two sets of trajectories does not change their comparison by $\succeq_{lmax(lmin)}$ (resp. $\succeq_{lmin(lmax)}$).

Proposition 3. *Relations* $\succeq_{lmin(lmax)}$ *and* $\succeq_{lmax(lmin)}$ *are complete, transitive and satisfy the principle of strict monotonicity.*

Note that u_{opt} and u_{pes} satisfy only a weak form of monotonicity since the addition or the removal of trajectories may transform a strict preference into an indifference if u_{opt} or u_{pes} is used.

Let us define the *complementary* MDP $(S, A, \pi, \bar{\mu})$ of a given $\Pi\mathcal{MDP}$ (S, A, π, μ) where $\bar{\mu}(s) = 1 - \mu(s), \forall s \in S$. The complementary MDP simply gives complementary utilities. From the definitions of $\succeq_{lmax}$ and $\succeq_{lmin}$, we can check that:

Proposition 4. $\tau \succeq_{lmax} \tau' \Leftrightarrow \bar{\tau}' \succeq_{lmin} \bar{\tau}$ *and* $\delta \succeq_{lmin(lmax)} \delta' \Leftrightarrow \bar{\delta}' \succeq_{lmax(lmin)} \bar{\delta}$.

There $\bar{\tau}$ and $\bar{\delta}$ are obtained by replacing μ with $\bar{\mu}$ in the trajectory/$\Pi\mathcal{MDP}$.

Therefore, all the results which we will prove for $\succeq_{lmax(lmin)}$ also hold for $\succeq_{lmin(lmax)}$, if we take care to apply them to complementary policies. Since considering $\succeq_{lmax(lmin)}$ involves less cumbersome expressions (no $1 - \cdot$), we will give the results for this criterion. A consequence of Proposition 4 is that the results hold for the pessimistic criterion as well.

This monotonicity of the *lmin*(*lmax*) and *lmax*(*lmin*) criteria is sufficient to allow us to use a dynamic programming algorithm such as value iteration or policy iteration [2]. The algorithms we propose in the present paper perform explicit Bellman updates in the lexicographic framework (lines 12–13 of Algorithms 3 and 4, line 11 of Algorithm 5); the correctness of their use is proved in Propositions 6 to 10.

3.3. Basic operations on matrices of trajectories

Before going further, in order to give more explicit and compact descriptions of the algorithms and the proofs, let us introduce the following notations and some basic operations on matrices (typically, on the matrix $U(s)$ representing trajectories issued from state s). Abusing notations slightly, we identify trajectories τ (resp. policies) with their v_τ vectors (resp. matrices of v_τ vectors) when there is no ambiguity. For any matrix U, $[U]_{l,c}$ denotes the restriction of U to its first l lines and first c columns and $U_{i,j}$ denotes the element at line i and column j.

- **Composition**: Let U be a $a \times b$ matrix and $N_1, \ldots, N_a$ be a series of a matrices of dimension $n_i \times c$ (they all share the same number of columns). The composition of U with $(N_1, \ldots, N_a)$ denoted $U \times (N_1, \ldots, N_a)$ is a matrix of dimension $(\sum\limits_{1 \le i \le a} n_i) \times (b + c)$. For any $i \le a$, $j \le n_j$, the $((\Sigma_{i' < i} n_{i'}) + j)$-th line of $U \times (N_1, \ldots, N_a)$ is the concatenation of the i-th line of U and the j-th line of N_i.
The composition of $U \times (N_1, \ldots, N_a)$ is done in $O(n \cdot m)$ operations, where $n = \sum\limits_{1 \le i \le a} n_i$ and $m = b + c$. The matrix $U(s)$, matrix of trajectories out of state s when making decision a, is typically the concatenation of the matrix $U = ((\pi(s'|s, a), \mu(s')), s' \in succ(s, a))$ with the matrices $N_{s'} = U(s')$. This procedure adds two columns to each matrix $U(s')$, filled with $\pi(s'|s, a)$ and $\mu(s')$ the possibility degrees and the utility of reaching s'; then the matrices are vertically concatenated to get the matrix $U(s)$ when making decision a. Then it is possible to lexicographically compare the resulting matrices in order to get the optimal action in state s.
- **Ordering matrices**: Let U be a $n \times m$ matrix, $U^{lmaxlmin}$ is the matrix obtained by ordering the elements of the lines of U in increasing order and the lines of U according to *lmax* in decreasing order (see Example 4). This operation allows to compare matrices of trajectories $Q(s, a)$ of every action in order to compare them and choose the optimal decision.
The complexity of the operation depends on the sorting algorithm: if we use QuickSort then ordering the elements within a line is performed in $O(m \cdot log(m))$, and the inter-ranking of the lines is done in $O(n \cdot log(n) \cdot m)$ operations. Hence, the overall complexity is $O(n \cdot m \cdot log(n \cdot m))$.

- **Comparison of ordered matrices**: Given two ordered matrices $U^{lmaxlmin}$ and $V^{lmaxlmin}$, we say that $U^{lmaxlmin} > V^{lmaxlmin}$ iff $\exists i, j$ such that $\forall i' < i, \forall j', U^{lmaxlmin}_{i',j'} = V^{lmaxlmin}_{i',j'}$ and $\forall j' < j$, $U^{lmaxlmin}_{i,j'} = V^{lmaxlmin}_{i,j'}$ and $U^{lmaxlmin}_{i,j} > V^{lmaxlmin}_{i,j}$. $U^{lmaxlmin} \sim V^{lmaxlmin}$ iff they are identical (comparison complexity: $O(n \cdot m)$). Once matrices $Q(s,a)$ are ordered, the lexicographic comparison of two decisions is performed by scanning the elements of their matrices, line by line from the first one. The first pair of different values determines the best matrix and the best corresponding action a is selected (see Example 4).

If the policies (or sub-policies) have different numbers of trajectories, the comparison of two matrices is based on the number of trajectories of the shortest matrix. Two cases may arise:

- If we have a strict preference between the two matrices before reaching the last line of the shortest matrix, we get a strict preference between the policies (or between the sub-policies).
- If we have an indifference up to the last line, the shortest matrix is the best for the lexicographic criterion, since it expresses less uncertainty in the corresponding policy (or in the sub-policy).

3.4. Bounded iterations lexicographic value iteration

In this section, we propose an iterative value iteration-type algorithm (Algorithm 3). This algorithm follows the same principle as in the possibilistic case (Eqs. (6)–(9)). Repeated Bellman updates are performed successively E times. This algorithm will provide an approximation of a lexi optimal strategy in the infinite horizon case (by considering the policy returned for the first time step). This algorithm is sub-optimal for any fixed E, but we will see in Section 4 that letting E grow, an optimal lexicographic policy will be obtained for finite E.

We propose two versions of the value iteration algorithm: The first one computes the optimal policy with respect to the lmax(lmin) criterion and the second one provides the optimal policy with respect to the lmin(lmax) criterion. In this paper, we present and detail only the first algorithm, since the second is very similar.[2]

Algorithm 3: Bounded iterations lmax(lmin)-value iteration (BI-VI).

Data: A possibilistic MDP and maximum number of iterations E
Result: The δ_E strategy obtained after E iterations

begin
  $e \leftarrow 0$;
  foreach $s \in S$ **do** $U(s) \leftarrow ((\mu(s)))$;
  foreach $s \in S, a \in A$ **do**
  $TU_{s,a} \leftarrow T_{s,a} \times ((\mu(s')), s' \in succ(s,a))$;
  repeat
    $e \leftarrow e + 1$;
    foreach $s \in S$ **do**
      $U^{old}(s) = U(s)$;
      $Q^* \leftarrow ((0))$;
      foreach $a \in A$ **do**
        $Future \leftarrow (U^{old}(s'), s' \in succ(s,a))$; `// Gather the matrices provided by the successors of` s;
        $Q(s,a) \leftarrow \left(TU_{s,a} \times Future\right)^{lmaxlmin}$;
        if $Q^* \leq_{lmaxlmin} Q(s,a)$ **then**
          $Q^* \leftarrow Q(s,a)$;
          $\delta(s) \leftarrow a$
      $U(s) \leftarrow Q^*(s, \delta(s))$
  until $e == E$;
  $\delta(s) \leftarrow argmax_a Q(s,a)$
  return $\delta_E = \delta$;

This algorithm is an iterative procedure that performs a prescribed number of updates, E, of the utility of each state, represented by a finite matrix of trajectories, using the utilities of the neighboring states.

At stage $1 \leq e \leq E$, the procedure updates the utility of every state $s \in S$ as follows:

- For each action $a \in A$, a matrix $Q(s,a)$ is built to evaluate the "utility" of performing a in s at stage e: this is done by combining $TU_{s,a}$ (combination of the transition matrix $T_{s,a} = \pi(\cdot|s,a)$ and the utilities $\mu(s')$ of the states s' that may

[2] Indeed, it just amounts to order and compare the matrices (of the sub-policies) using lmin(lmax) instead of lmax(lmin).

follow s when a is executed) with the matrices $U^{old}(s')$ of trajectories provided by these s' at the previous stage. The matrix $Q(s,a)$ is then ordered (the operation is made less complex by the fact that the matrices $U^{old}(s')$ have already been ordered at $e-1$).
- The *lmax*(*lmin*) comparison is performed on the fly to memorize the best $Q(s,a)$.
- The value of state s at stage e, $U(s)$, is the one given by the action a which provides the best $Q(s,a)$. δ is updated, U is memorized (and U^{old} can be discarded).

Time and space complexities of this algorithm are nevertheless expensive, since it eventually memorizes all the trajectories. At each step e its size may grow to $b^e \cdot (2 \cdot e + 1)$, where b is the maximal number of possible successors of an action; the overall complexity of the algorithm is $O(|S| \cdot |A| \cdot E \cdot b^E)$, which is a problem.

Algorithm 3 is provided with a number of iterations, E. Does it converge when E tends to infinity? That is, are the returned policies identical for any E exceeding a given threshold? Before answering (positively) this question in Section 4.4, we are going to define *bounded utility matrices* solutions to lexicographic possibilistic MDPs. These solution concepts will be useful to answer the above question.

4. Bounded utility solutions to lexicographic $\Pi\mathcal{MDP}$s

We have just proposed a lexicographic value iteration algorithm for the computation of lexicographic policies based on the whole matrices of trajectories. As a consequence, the spatial/temporal complexity of the algorithm is exponential in the number of iterations. This section presents an alternative way to get lexicographic policies. Rather than limiting the size of the matrices of trajectories by limiting the number of iterations, we propose to “forget” the less significant part of the matrices of utility and to decide only based on the most significant (l,c) sub-matrices – we “bound” the utility matrices. We propose in the present section two algorithms based on this idea, namely a value iteration and a policy iteration algorithms.

4.1. Bounded lexicographic comparisons of utility matrices

Recall that, for any matrix U, $[U]_{l,c}$ denotes the restriction of U to its first l lines and first c columns. Notice now that, at any stage e and for any state s $[U(s)]_{1,1}$ (i.e. the top left value in $U(s)$) is precisely equal to $u_{opt}(s)$. We have seen that making the choices on this basis is not discriminant enough. On the other hand, taking the whole matrix into account is discriminant, but exponentially costly. Hence the idea of considering more than one line and one column, but less than the whole matrix – namely the first l lines and c columns of $U^t(s)^{lmaxlmin}$; hence the definition of the following preference:

$$\delta \geq_{lmaxlmin,l,c} \delta' \text{ iff } [\delta^{lmaxlmin}]_{l,c} \geq [\delta'^{\,lmaxlmin}]_{l,c}. \tag{18}$$

$\geq_{lmaxlmin,1,1}$ corresponds to $\succeq_{opt}$ and $\geq_{lmaxlmin,+\infty,+\infty}$ corresponds to $\geq_{lmaxlmin}$.

The following proposition shows that this approach is sound and that $\succ_{lmaxlmin,l,c}$ refines u_{opt}:

Proposition 5.

- *For any l, l', c such that $l' > l$, $\delta \succ_{lmaxlmin,l,c} \delta' \Rightarrow \delta \succ_{lmaxlmin,l',c} \delta'$.*
- *For any l, c $\delta \succ_{opt} \delta' \Rightarrow \delta \succ_{lmaxlmin,l,c} \delta'$.*

In other words, the order over the policies is refined for a fixed c when l increases. It tends to $\succ_{lmaxlmin}$ when $c = 2.E+1$ and l tends to b^E.

Notice that the combinatorial explosion is due to the number of lines (the number of columns is bounded by $2 \cdot E + 1$), hence we shall bound the number of considered lines only.

Up to this point, the comparison by $\geq_{lmaxlmin,l,c}$ is made on the basis of the first l lines and c columns of the *full* matrices of trajectories. This does obviously not reduce their size. The important following Proposition allows us to make the l,c reduction of the ordered matrices *at each step* (after each composition), and not only at the very end, thus keeping space and time complexities polynomial.

Proposition 6. *Let U be a $a \times b$ matrix and $N_1, \ldots, N_a$ be a series of a matrices of dimension $a_i \times c$. It holds that:*

$$[(U \times (N_1, \ldots, N_a))^{lmaxlmin}]_{l,c} = [(U \times ([N_1^{lmaxlmin}]_{l,c}, \ldots, [N_a^{lmaxlmin}]_{l,c}))^{lmaxlmin)}]_{l,c}.$$

4.2. Bounded utility lexicographic value iteration

It is now easy to design a generalization of the possibilistic algorithm of value iteration (Algorithm 1) by keeping a submatrix of each current value matrix – namely the first l lines and c columns. We call this algorithm *Bounded Utility Value Iteration* $(BU\text{-}VI)$ (see Algorithm 4).

Algorithm 4: Bounded Utility Lmax(lmin) Value Iteration (BU-VI).

Data: A possibilistic MDP, bounds (l, c); δ, the policy built by the algorithm, is a global variable
Result: A policy δ optimal for $\succeq_{lmaxlmin,l,c}$

begin
  foreach $s \in S$ **do** $U(s) \leftarrow ((\mu(s)))$;
  foreach $s \in S, a \in A$ **do**
  $TU_{s,a} \leftarrow T_{s,a} \times ((\mu(s')), s' \in succ(s,a))$;
  repeat
  until $U(s) == U^{old}(s)$ *for each* s;
  foreach $s \in S$ **do**
    $U^{old}(s) \leftarrow U(s)$;
    $Q^* \leftarrow ((0))$;
    foreach $a \in A$ **do**
      $Future \leftarrow (U^{old}(s'), s' \in succ(s,a))$; // Gather the matrices provided by the successors of s;
      $Q(s,a) \leftarrow [(TU_{s,a} \times Future)^{lmaxlmin}]_{l,c}$;
      if $Q^* \le_{lmaxlmin} Q(s,a)$ **then**
        $Q^* \leftarrow Q(s,a)$;
        $\delta(s) \leftarrow a$
    $U(s) \leftarrow Q^*(s, \delta(s))$
  $\delta(s) \leftarrow argmax_a Q(s,a)$;
  $U(s) \leftarrow \max_a Q(s,a)$
  return δ;

When the horizon of the MDP is finite this algorithm provides in polynomial time a policy that is always at least as good as the one provided by u_{opt} (according to $lmax(lmin)$) and tends to lexicographic optimality when $c = 2 \cdot E + 1$ and l tends to b^E.

Let us now study the time complexity. The number of iterations is bounded by the size of the set of possible matrices of trajectories which is in $O(|S| \cdot |A| \cdot E)$. One iteration of the algorithm requires composition, ordering and comparing operations on b matrices of size (l, c). Since the composition and comparison of matrices are linear operations, the complexity of one iteration in worst case is in $b \cdot (l \cdot c) \cdot log(l \cdot c)$. Therefore, the complexity of the algorithm is in $O(|S| \cdot |A| \cdot E \cdot b \cdot (l \cdot c) \cdot log(l \cdot c))$.

When the horizon of the MDP is not finite, equations (16) and (17) are not enough to rank-order the policies. The length of the trajectories may be infinite, as well as their number. This problem is well known in classical probabilistic $\mathcal{MDP}$s where a discount factor is used to attenuate the influence of later utility degrees – thus allowing the convergence of the algorithm [21]. On the contrary, classical $\Pi\mathcal{MDP}$s do not need any discount factor and Value Iteration, based on the evaluation for $l = c = 1$, converges for infinite horizon case [22]. In a sense, this limitation to $l = c = 1$ plays the role of a discount factor – but a very drastic one. Extending the comparison by using $\succeq_{lmaxlmin,l,c}$ with larger (l, c) as shown below allows to use a less drastic discount.

In other terms, $\succeq_{lmaxlmin,l,c}$ can be used in the infinite case, as shown by the following proposition.

Proposition 7 *(Bounded utility lmax(lmin)-policy evaluation converges). Let $U^t(s)$ be the matrix issued from s at instant t when a strategy δ is executed. It holds that:*

$$\forall\, l, c, \exists\, t,\ \textit{such that}\ \forall\, t' \ge t,\ (U^t)^{lmaxlmin}_{l,c}(s) = (U^{t'})^{lmaxlmin}_{l,c}(s)\ \ \forall s.$$

Hence there exists a stage t, where the value of a policy becomes stable if computed with the bounded utility lmax(lmin) evaluation algorithm. This criterion is thus soundly defined and can be used in the infinite horizon case (and of course in the finite horizon case).

The number of iterations of Algorithm 4 is not explicitly bounded but the convergence of the algorithm is guaranteed – this is a direct consequence of Proposition 7.

Corollary 1 *(Bounded utility lmax(lmin)-value iteration converges). $\forall\, l, c, \exists\, t$ such that, $\forall\, t' \ge t$, $(U^t)^{lmaxlmin}_{l,c}(s) = (U^{t'})^{lmaxlmin}_{l,c}(s)$ $\forall s$.*

The overall complexity of *bounded utility lmax(lmin)-value iteration* (Algorithm 4) is bounded by $O(|S| \cdot |A| \cdot |L| \cdot b \cdot (l \cdot c) \cdot log(l \cdot c))$ since the number of iterations is $O(|S| \cdot |A| \cdot |L|$ and all matrices are of size (l, c).

4.3. *Bounded utility lexicographic-policy iteration*

In Ref. [17], Howard shows that a policy often becomes optimal long before the convergence of the value estimates. That is why Puterman [21] has proposed a policy iteration algorithm. This algorithm has been adapted to possibilistic MDPs by [22].

Likewise, we propose a (bounded utility) *lexicographic policy iteration* algorithm (Algorithm 5), denoted here $BU\text{-}PI$ that alternates improvement and evaluation phases, as any policy iteration algorithm.

Algorithm 5: Lmax(lmin)-Bounded Utility Policy Iteration.

Data: A possibilistic MDP, bounds (l, c)
Result: A policy δ^* optimal when l, c grows

1 **begin**
2   // Arbitrary initialization of δ on S
3   **foreach** $s \in S$ **do** $\delta(s) \leftarrow$ choose any $a_s \in A_s$;
4   **repeat**
5   // Evaluation of δ
6   **foreach** $s \in S$ **do** $U(s) \leftarrow \mu(s)$;
7   **repeat**
8   **foreach** $s \in S$ **do**
9         $U^{old}(s) \leftarrow U(s)$;
10   // Gather the matrices of the successors of s given δ
11         $Future \leftarrow (U(s'), s' \in succ(s, \delta(s)))$;
          $U(s) \leftarrow \left[\left(TU_{s,\delta(s)} \times Future\right)^{lmaxlmin}\right]_{l,c}$;
12   **until** $U(s) == U^{old}(s)$ *for each* s;
13     $\delta^{old} \leftarrow \delta$;
14   // Improvement of δ
15   **foreach** $s \in S$ **do**
16   // Compute the utility of the strategy playing a (for each a), given what was chosen for the other states
17   **foreach** $a \in A$ **do**
18         $Future \leftarrow (U(s'), s' \in succ(s, \delta^{old}(s)))$;
           $Q(s, a) \leftarrow \left[\left(TU_{s,a} \times Future\right)\right]^{lmaxlmin}_{l,c}$
19   // Update the choice of an action for S
20       $\delta(s) \leftarrow \arg\max^{lmax(lmin)}_{a \in A} Q(s, a)$
21   **until** $\delta == \delta^{old}$;
22   **return** δ;

In line 3 of Algorithm 5, an arbitrary initial policy is chosen. The algorithm then proceeds by evaluating the current policy, through successive updates of the value function (lines 8 to 11); the convergence of this evaluation is easily derived from that of the *bounded utility lmax(Lmin)-value iteration* algorithm. Then the algorithm enters the improvement phase: Lines 17–18 compute $Q(s, a)$, the (bounded lexicographic) utility of playing action a in state s and then applying policy δ^{old} in subsequent states (the policy computed during the last iteration); as usual in Policy Iteration style algorithms, the updated policy (δ) is then obtained by greedily improving the current action, which is done in line 20. Since the actions considered at line 20 do include the one prescribed by δ^{old}, either nothing is changed, and the algorithm stops, or the new policy, δ, is better than the previous one δ^{old}.

Proposition 8. *Bounded utility lmax(lmin)-policy iteration converges to an optimal policy for* $\succeq_{lmaxlmin,l,c}$ *in finite time.*

Policy iteration (Algorithm 5) converges and is guaranteed to find a policy optimal for the (l, c) lexicographic criterion in finite time and usually in a few iterations. As for the algorithmic complexity of the classical, stochastic, policy iteration algorithm (which is still not well understood [16]), a tight bound worst-case complexity of *lexicographic policy iteration* is hard to obtain. Therefore, we provide an upper-bound of this complexity.

The *policy iteration* algorithm never visits a policy twice: in the worst case, the number of trial iterations before convergence is exponential but it is dominated by the number of distinct policies. So, the complexity of this algorithm is dominated by $(|A|^{|S|})$. Besides, each iteration has a cost, the evaluation phase relying on a bounded utility value iteration algorithm that costs $O(|S| \cdot |A| \cdot |L| \cdot b \cdot (l \cdot c) \cdot b \cdot log(l \cdot c))$ when many actions are possible at a given step, and cost $O(|S| \cdot |L| \cdot b \cdot (l \cdot c) \cdot b \cdot log(l \cdot c))$ here because one action is selected (by the current policy) for each state. Thus, the overall complexity of the algorithm is in $O(|A|^{|S|} \cdot |S| \cdot |L| \cdot b \cdot (l \cdot c) \cdot b \cdot log(l \cdot c))$.

Thus, the overall complexity of the algorithm is in $O(|A|^{|S|} \cdot |L| \cdot (l \cdot c)^2 \cdot |S| \cdot b \cdot log(l \cdot c))$.

4.4. Back to lexicographic-value iteration: from finite to infinite horizon Π-MDPs

The bounded iterations algorithm defined in section 3 (Algorithm 3, $(BI\text{-}VI)$) can be used for both finite horizon and infinite horizon MDPs, because it fixes a number of iterations E; if E is low, the policy reached in not necessarily optimal – the algorithm is an approximation algorithm.

Now, exploiting the above propositions, we are able to show that the bounded iterations Lmax(lmin) value iteration algorithm (Algorithm 3) converges when E tends to infinity. To do so, we first prove the following proposition:

Proposition 9. *Let an arbitrary stationary $\Pi\mathcal{MDP}$ be given. Then, there exist two positive natural numbers (l^*, c^*), such that for any pair (δ, δ') of arbitrary policies and any state $s \in S$, and for any pair (l, c) such that $l \geq l^*$ and $c \geq c^*$,*

$$\delta(s) \succ_{lmaxlmin,l^*,c^*} \delta'(s) \Leftrightarrow \delta(s) \succ_{lmaxlmin,l,c} \delta'(s)$$

Now, this proposition can be used to prove the convergence of the *bounded iterations Lmax(lmin)-value iteration* algorithm. For this, let us define $\succ_{lmaxlmin} =_{def} \succ_{lmaxlmin,l^*,c^*}$, the unique preference relation between policies that results from Proposition 9.

Proposition 10. *If we let δ_E be the policy returned by Algorithm 3 for any fixed E, we can show that the sequence (δ_E) converges and that there exists a finite E^*, such that:*

$$\lim_{E \to \infty} \delta_E = \delta_{E^*}.$$

Furthermore, δ_{E^*} is optimal with respect to $\succ_{lmaxlmin}$.

The sequence of policies obtained for $(BI\text{-}VI)$ (Algorithm 3) when E tends to infinity converges. Furthermore, the limit is attained for a finite (but unknown in advance) E^*. Alternately, it is also attained for the $(BU\text{-}VI)$ and $(BU\text{-}PI)$ algorithms, with finite but unknown (l^*, c^*).

Now, let us summarize the theoretical results that we have obtained so far. We have shown that possibilistic utilities (optimistic and pessimistic) are special cases of bounded lexicographic utilities, which can be represented by matrices. Possibilistic utilities are obtained when $l = c = 1$.

The possibilistic value iteration and policy iteration algorithms can be extended to compute policies which are optimal according to $\succ_{lmaxlmin,l,c}$.

Finally, if infinite horizon lexicographic optimal policies are defined as the limiting policies obtained from a non-bounded lexicographic value-iteration algorithm, we have shown that such policies can be computed by applying our bounded utility lmax(lmin) value iteration algorithm and that only a finite number of iterations (even though not known in advance) is required.

5. Experiments

In order to evaluate the previous algorithms, we propose, in the following, two experimental analyses: in the first one we will compare the bounded iterations algorithm of value iteration (Algorithm 3) with the bounded utility one and in the second we propose to compare the bounded utility lexicographic policy iteration algorithm with the bounded utility lexicographic value iteration one. The algorithms have been implemented in Java and the experiments have been performed on an Intel Core i5 processor computer (1.70 GHz) with 8GB DDR3L of RAM.

5.1. Bounded utility vs bounded iterations value iteration

Experimental protocol. We now compare the performance of *bounded utility lexicographic value iteration* $(BU\text{-}VI)$ as an approximation of *lexicographic value iteration* $(BI\text{-}VI)$ for finite horizon problems, in the *Lmax*(*lmin*) variant. Because the horizon is finite, the number of steps of $(BI\text{-}VI)$ can be set equal to the horizon and the algorithm provides a solution optimal according to *Lmax*(*lmin*). $(BU\text{-}VI)$ on the other side limits the size on the matrices, and can lead to sub-optimal solutions.

We evaluate the performance of the algorithms by carrying out simulations on randomly generated finite horizon $\Pi\mathcal{MDP}$s with 25 states – we generate five series of problems, letting E varying form 5 to 25. The number of actions in each state is equal to 4. The output of each action is a distribution on two states randomly sampled (i.e. the branching factor is equal to 2). The utility values are uniformly randomly sampled in the set $L = \{0.1, 0.3, 0.5, 0.7, 1\}$. Conditional possibilities relative to decisions should be normalized. To this end, one choice is fixed to possibility degree 1 and the possibility degree of the other one is uniformly sampled in L. For each experience, 100 $\Pi\mathcal{MDP}$s are generated. The two algorithms are compared w.r.t. two measures: (i) CPU time and (ii) Pairwise success rate (*Success*) i.e. the percentage of optimal solutions provided by $BU\text{-}VI$ with fixed (l, c) w.r.t. the lmax(lmin) criterion in its full generality. The higher *Success*,

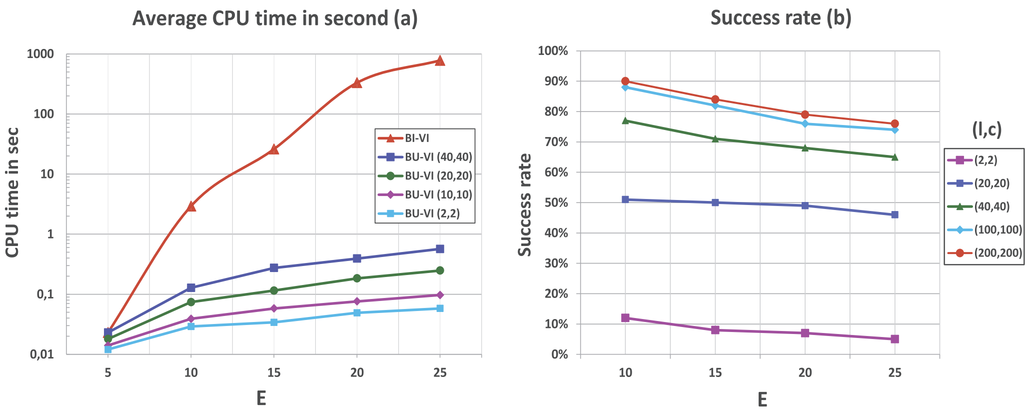

Fig. 2. Bounded utility lexicographic value iteration vs lexicographic value iteration.

Table 1
Average CPU time (in seconds) and average number of iterations.

Bounded utility policy iteration				
(l, c)	$(2, 2)$	$(4, 4)$	$(6, 6)$	$(10, 10)$
CPU time (s)	0.029	0.042	0.064	0.091
Average number of iterations	3.2	4.33	5.6	9.7
Bounded utility value iteration				
(l, c)	$(2, 2)$	$(4, 4)$	$(6, 6)$	$(10, 10)$
CPU time (s)	0.03	0.052	0.082	0.1
Average number of iterations	6.75	9.25	16.11	20.2

the more important the effectiveness of cutting matrices with $BU\text{-}VI$; the lower this rate, the more important the drowning effect.

Results. Fig. 2(a) presents the average execution CPU time for the two algorithms. Obviously, for both $BI\text{-}VI$ and $BU\text{-}VI$, the execution time increases with the horizon. Also, we observe that the CPU time of $BU\text{-}VI$ increases according to the values of (l, c) but it remains affordable, as the maximal CPU time is lower than 1 s for MDPs with 25 states and 4 actions when $(l, c) = (40, 40)$ and $E = 25$. Unsurprisingly, we can check that the $BU\text{-}VI$ (regardless of the values of (l, c)) is faster than $BI\text{-}VI$ especially when the horizon increases: the manipulation of l, c-matrices is obviously less expensive than the one of full matrices. The saving increases with the horizon.

As with the success rate, the results are described in Fig. 2(b). It appears that $BU\text{-}VI$ provides a very good approximation especially when increasing (l, c). It provides the same optimal solution as the $BI\text{-}VI$ in about 90% of cases, with an $(l, c) = (200, 200)$. Moreover, even when the success rate of $BU\text{-}VI$ decreases (when E increases), the quality of approximation is still good: never less than 70% of optimal actions returned, with $E = 25$. These experiments conclude in favor of bounded value iteration: the quality of its approximated solutions are comparable with those of the unbounded version for high (l, c) and increases when (l, c) increase, while it is much faster.

5.2. *Bounded utility lexicographic policy iteration vs bounded utility lexicographic value iteration*

Experimental protocol. In what follows we evaluate the performances of *bounded utility lexicographic policy iteration* $(BU\text{-}PI)$ and *bounded lexicographic value iteration* $(BU\text{-}VI)$, in the lmax(lmin) variant. We evaluate the performance of the algorithms on randomly generated $\Pi\mathcal{MDP}$s as those of Section 5.1 with $|S| = 25$ and $|A_s| = 4$, $\forall s$.

We ran the two algorithms for different values of (l, c) (100 $\Pi\mathcal{MDP}$s are considered in each sample). For each of the two algorithms we measure the CPU time needed to converge. We also measure the average number of value iterations for $(BU\text{-}VI)$ and the average number of policy iterations for $(BU\text{-}PI)$.

Results. Table 1 presents the average execution CPU time and the average number of iterations for the two algorithms.

Obviously, for both $BU\text{-}PI$ and $BU\text{-}VI$, the execution time increases according to the values of (l, c) but it remains affordable, as the maximal CPU time is lower than 0.1 s for MDPs with 25 states and 4 actions when $(l, c) = (10, 10)$. It appears that $BU\text{-}PI$ (regardless of the values of (l, c)) is slightly faster than $BU\text{-}VI$.

Consider now the number of iterations. At each iteration, $BU\text{-}PI$ considers one policy, explicitly, and updates it at line 20. And so does value iteration: for each state, the current policy is updated at line 15. Table 1 shows that $BU\text{-}PI$ always considers fewer policies than $BU\text{-}VI$. This experiment provides an empirical evidence in favor of *policy iteration* over *value iteration*, as the former converges to the approximate solution faster. However, this conclusion may vary with the experiments, so both algorithms are worth considering when tackling a given problem.

6. Conclusion

In order to overcome the drowning effect in possibilistic stationary sequential decision problems, we have proposed two lexicographic criteria, initially introduced by Fargier and Sabbadin in [13] for non-sequential problems, that compare policies based on their corresponding matrices of trajectories. We have shown that these decision criteria satisfy the principle of efficiency and strict monotonicity.

Because the sizes of the matrices of trajectories grow exponentially with the number of iterations, such comparisons cannot lead to a finite time algorithm except if the horizon of the problem is finite. That is why we have proposed bounded utility lexicographic algorithms as approximations of the full lexicographic ones.

First, we have proposed a *Lexicographic Value Iteration* algorithm for stationary $\Pi\mathcal{MDP}$s with two variants: (i) an algorithm that compare full matrices but halts in a finite number of steps, (ii) an algorithm which bounds the size of the saved matrices and refines the possibilistic criteria, whatever the choice of the bounds.

Then, we have proposed a bounded *Lexicographic Policy iteration* algorithm and we have shown that this algorithm converges in a finite number of iterations.

The convergence of these algorithms has been shown and their efficiency has been assessed experimentally. In particular, the algorithm of Policy Iteration appears to be experimentally faster than the algorithm of Bounded Utility Value Iteration (which is also usual in the stochastic case [16]).

Future work includes two research tracks. First, as far as the infinite horizon case is concerned, other types of lexicographic refinements could be proposed. One of these options could be to avoid the duplication of the set of transitions that occur several times in a single trajectory and only record once the observed transitions.

A second line of research of this work will be to define *reinforcement learning* [26] type algorithms for $\Pi\mathcal{MDP}$s. Such algorithms would use observed samples (s, a, s', π, μ) of the trajectories to compute an approximate lexicographic policy, instead of using full dynamic programming over all possible trajectories. This would offer an alternative to existing quantile-based reinforcement learning approaches [14] for ordinal MDPs.

Appendix A. Proofs of propositions

Proof of Proposition 1.

- We prove that $\succeq_{lmax(lmin)}$ refines $\succeq_{u_{opt}}$ in stationary $\Pi\mathcal{MDP}$. We consider two policies δ and δ'.
  If $u_{opt}(\delta, s_0) > u_{opt}(\delta', s_0)$
  $\Leftrightarrow \max_{\tau\in\delta} \min\{\pi(\tau|s_0,\delta), u(\tau)\} > \max_{\tau'\in\delta'} \min\{\pi(\tau'|s_0,\delta'), u(\tau')\}$
  $\Rightarrow \max_{\tau\in\delta} \min\{\min_{i=1..h} \pi(s_i|s_{i-1},\delta(s_{i-1})), \min_{i=1..h} u(s_i)\} > \max_{\tau'\in\delta'} \min\{\min_{i=1..h} \pi'(s_i|s_{i-1},\delta'(s_{i-1})), \min_{i=1..h} u'(s_i)\}$
  $\Rightarrow \max_{\tau\in\delta} \min(\mu_0, \pi_1, \mu_1, \pi_2, \ldots, \pi_{h-1}, \mu_h) > \max_{\tau'\in\delta'} \min(\mu'_0, \pi'_1, \mu'_1, \pi'_2, \ldots, \pi'_{h-1}, \mu'_h)$.
  Since $\min(\mu_0, \pi_1, \mu_1, \pi_2, \ldots, \pi_{h-1}, \mu_h) > \min(\mu'_0, \pi'_1, \mu'_1, \pi'_2, \ldots, \pi'_{h-1}, \mu'_h)$
  $\Rightarrow (\mu_0, \pi_1, \mu_1, \pi_2, \ldots, \pi_{h-1}, \mu_h) \succ_{lmin} (\mu'_0, \pi'_1, \mu'_1, \pi'_2, \ldots, \pi'_{h-1}, \mu'_h)$ (as leximin ordering refines min ordering), then $\tau_{\lambda(1)} \succ_{lmin} \tau'_{\lambda(1)} \Rightarrow \delta \succ_{lmax(lmin)} \delta'$ where $\tau_{\lambda(1)}$ (resp. $\tau'_{\lambda(1)}$) is the best trajectory of δ (resp. δ') according to $\succeq_{lmin}$.
  Using the definition of $\succeq_{lmax(lmin)}$ (Eq. (16)) we have $\delta \succ_{lmax(lmin)} \delta'$. Thus, $\succeq_{lmax(lmin)}$ refines $\succeq_{u_{opt}}$.
- We prove in the same way that $\succeq_{lmin(lmax)}$ refines $\succeq_{u_{pes}}$. Considering two policies δ and δ' s.t. $u_{pes}(\delta, s_0) > u_{pes}(\delta', s_0)$
  $\Leftrightarrow \min_{\tau\in\delta} \max\{1-\pi(\tau|s_0,\delta), u(\tau)\} > \min_{\tau'\in\delta'} \max\{1-\pi(\tau'|s_0,\delta'), u(\tau')\}$
  $\Leftrightarrow \min_{\tau\in\delta} \max\{\min_{i=1..h} (1-\pi(s_i|s_{i-1},\delta(s_{i-1}))), \min_{i=1..h} u(s_i)\} > \min_{\tau'\in\delta'} \max\{\min_{i=1..h} (1-\pi'(s_i|s_{i-1},\delta'(s_{i-1}))), \min_{i=1..h} u'(s_i)\}$
  $\Leftrightarrow \min_{\tau\in\delta} \max\{\min(\mu_0, 1-\pi_1, \mu_1, 1-\pi_2, \ldots, 1-\pi_{h-1}, \mu_h)\} > \min_{\tau'\in\delta'} \max\{(\mu'_0, 1-\pi'_1, \mu'_1, 1-\pi'_2, \ldots, 1-\pi'_{h-1}, \mu'_h)\}$.
  Since $\max(\mu_0, 1-\pi_1, \mu_1, 1-\pi_2, \ldots, 1-\pi_{h-1}, \mu_h) > \max(\mu'_0, 1-\pi'_1, \mu'_1, 1-\pi'_2, \ldots, 1-\pi'_{h-1}, \mu'_h)$
  $\Rightarrow (\mu_0, 1-\pi_1, \mu_1, 1-\pi_2, \ldots, 1-\pi_{h-1}, \mu_h) \succ_{lmax} (\mu'_0, 1-\pi'_1, \mu'_1, 1-\pi'_2, \ldots, 1-\pi'_{h-1}, \mu'_h)$ (as leximax ordering refines max ordering).
  Then $\tau_{\sigma(1)} \succ_{lmax} \tau'_{\sigma(1)} \Rightarrow \delta \succ_{lmin(lmax)} \delta'$ where $\tau_{\sigma(1)}$ (resp. $\tau'_{\sigma(1)}$) is the worst trajectory of δ (resp. δ') according to $\succeq_{lmax}$. So, by definition of $\succeq_{lmin(lmax)}$ (Eq. (17)) we have $\delta \succ_{lmin(lmax)} \delta'$. We deduce that $\succeq_{lmin(lmax)}$ refines $\succeq_{u_{pes}}$. $\square$

Proof of Proposition 2. (i) We prove that $\succeq_{lmax(lmin)}$ satisfy the principle of Pareto efficiency. So, suppose that $\delta \succeq_{lmax(lmin)} \delta'$. Two cases arise:

- if $\forall N$, $\delta_N \sim_{lmax(lmin)} \delta'_N$ i.e. $\forall i$, $\tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)}$ and then $\delta \sim_{lmax(lmin)} \delta'$,
- if $\forall N$, $\delta_N \succeq_{lmax(lmin)} \delta'_N$ and $\exists N^*, \delta_{N^*} \succ_{lmax(lmin)} \delta'_{N^*}$ i.e. if $\exists i^*$, s.t. $\forall i < i^*$, $\tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)}$ and $\tau_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i^*)}$. Then, $\tau_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i^*)}$ implies that there exist a pair of different $(\beta_{i^*,k}, \beta'_{i^*,k})$, where $\beta_{i^*,k}$ (resp. $\beta'_{i^*,k}$) is an element of $\tau_{\lambda(i^*)}$ (resp. $\tau'_{\lambda(i^*)}$), that determines the best policy. Here we get $\beta_{i^*,k} > \beta'_{i^*,k}$ i.e. $\tau_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i^*)}$ and thus $\delta_{N^*} \succ_{lmax(lmin)} \delta'_{N^*}$ then $\delta \succ_{lmax(lmin)} \delta'$.

In summary, if we have $\delta \succeq_{lmax(lmin)} \delta'$ and $\exists i^*$, s.t. $\tau_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i^*)}$ we get $\delta \succ_{lmax(lmin)} \delta'$ which expresses exactly the principle of Pareto efficiency in the case $\succeq_{lmax(lmin)}$.

(ii) Let us prove $\succeq_{lmin(lmax)}$ satisfy the principle of Pareto efficiency. When considering the $\succeq_{lmin(lmax)}$ order, the same kind of result can be obtained.

So, suppose that $\delta \succeq_{lmin(lmax)} \delta'$. Two cases arise:

- if $\forall N$, $\delta_N \sim_{lmin(lmax)}$ i.e. $\forall i$, $\tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)}$ and then $\delta \sim_{lmin(lmax)} \delta'$,
- if $\forall N$, $\delta_N \succeq_{lmin(lmax)} \delta'_N$ and $\exists N^*, \delta_{N^*} \succ_{lmax(lmin)} \delta'_{N^*}$ i.e. $\exists i^*$, s.t. $\forall i < i^*$, $\tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)}$ and $\tau_{\sigma(i^*)} \succ_{lmax} \tau'_{\sigma(i^*)}$. Then, $\tau_{\sigma(i^*)} \succ_{lmax} \tau'_{\sigma(i^*)}$ implies that there exist a pair of different $(\beta_{i^*,k}, \beta'_{i^*,k})$, where $\beta_{i^*,k}$ (resp. $\beta'_{i^*,k}$) is an element of $\tau_{\sigma(i^*)}$ (resp. $\tau'_{\sigma(i^*)}$), determining the best policy. We get $\beta_{i^*,k} > \beta'_{i^*,k}$ i.e. $\tau_{\sigma(i^*)} \succ_{lmax} \tau'_{\sigma(i^*)}$ and thus $\delta \succ_{lmin(lmax)} \delta'$.

In summary, if we have $\delta \succeq_{lmin(lmax)} \delta'$ and $\exists i^*$, s.t. $\tau_{\sigma(i^*)} \succ_{lmax} \tau'_{\sigma(i^*)}$ we get $\delta \succ_{lmin(lmax)} \delta'$ which expresses exactly the principle of Pareto efficiency in the case of $\succeq_{lmin(lmax)}$. $\square$

Proof of Proposition 3.

- **Completeness**. It is a consequence of the completeness of $\succeq_{lmax}$ and $\succeq_{lmin}$.
- **Transitivity**. We prove that $\succeq_{lmax(lmin)}$ is transitive. The proof relies on the transitivity of $\succeq_{lmin}$. Let us consider three policies, δ, δ' and δ'' and assume $\delta \succeq_{lmax(lmin)} \delta'$ and $\delta' \succeq_{lmax(lmin)} \delta''$. Since $\delta \succeq_{lmax(lmin)} \delta'$ and $\delta' \succeq_{lmax(lmin)} \delta''$, then we are in either following cases:
  1. $\forall i$, $\tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)} \sim_{lmin} \tau''_{\lambda(i)}$. This happens when $\delta \sim_{lmax(lmin)} \delta' \sim_{lmax(lmin)} \delta''$. And then, by transitivity of $\succeq_{lmin}$, we have $\forall i$, $\tau_{\lambda(i)} \sim_{lmin} \tau''_{\lambda(i)} \Leftrightarrow \delta \sim_{lmax(lmin)} \delta''$.
  2. When either $\delta \succ_{lmax(lmin)} \delta'$ or $\delta' \succ_{lmax(lmin)} \delta''$, then, by definition of $\succeq_{lmax(lmin)}$, there exists i^*, such that:
     (a) $\forall i < i^*$, $\tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)} \sim_{lmin} \tau''_{\lambda(i)}$,
     (b) $\tau_{\lambda(i^*)} \succeq_{lmin} \tau'_{\lambda(i^*)} \succeq_{lmin} \tau''_{\lambda(i^*)}$ and
     (c) either $\tau_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i^*)}$ or $\tau'_{\lambda(i^*)} \succ_{lmin} \tau''_{\lambda(i^*)}$, or both.
     Then, once again by transitivity of $\succeq_{lmin}$, $\tau_{\lambda(i^*)} \succ_{lmin} \tau''_{\lambda(i^*)}$.
     So, $\delta \succ_{lmax(lmin)} \delta''$.

  So, points 1 and 2 imply, together, that $\delta \succeq_{lmax(lmin)} \delta'$ and $\delta' \succeq_{lmax(lmin)} \delta''$ imply $\delta \succeq_{lmax(lmin)} \delta''$.
- Similarly, it can be checked that $\succeq_{lmin(lmax)}$ is transitive. Let us consider three policies, δ, δ' and δ'' and assume $\delta \succeq_{lmin(lmax)} \delta'$ and $\delta' \succeq_{lmin(lmax)} \delta''$. Since $\delta \succeq_{lmin(lmax)} \delta'$ and $\delta' \succeq_{lmin(lmax)} \delta''$, then we are in either following cases:
  1. $\forall i$, $\tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)} \sim_{lmax} \tau''_{\sigma(i)}$.
     This happens when $\delta \sim_{lmin(lmax)} \delta' \sim_{lmin(lmax)} \delta''$. And then, by transitivity of $\succeq_{lmax}$, we have $\forall i$, $\tau_{\sigma(i)} \sim_{lmax} \tau''_{\sigma(i)} \Leftrightarrow \delta \sim_{lmin(lmax)} \delta''$.
  2. When either $\delta \succ_{lmin(lmax)} \delta'$ or $\delta' \succ_{lmin(lmax)} \delta''$, then, by definition of $\succeq_{lmin(lmax)}$, there exists i^*, such that:
     (a) $\forall i < i^*$, $\tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)} \sim_{lmax} \tau''_{\sigma(i)}$,
     (b) $\tau_{\sigma(i^*)} \succeq_{lmax} \tau'_{\sigma(i^*)} \succeq_{lmax} \tau''_{\sigma(i^*)}$ and
     (c) either $\tau_{\sigma(i^*)} \succ_{lmax} \tau'_{\sigma(i^*)}$ or $\tau'_{\sigma(i^*)} \succ_{lmax} \tau''_{\sigma(i^*)}$, or both.
     Then, once again by transitivity of $\succeq_{lmax}$, $\tau_{\sigma(i^*)} \succ_{lmax} \tau''_{\sigma(i^*)}$.
     So, $\delta \succ_{lmin(lmax)} \delta''$.

  So, points 1 and 2 imply, together, that $\delta \succeq_{lmin(lmax)} \delta'$ and $\delta' \succeq_{lmin(lmax)} \delta''$ imply $\delta \succeq_{lmin(lmax)} \delta''$.
- **Monotonicity**. $\delta + \delta''$ contains two disjoint sets of trajectories (i.e. vectors): the ones of δ and the ones of δ'' (and similarly for $\delta' + \delta''$).
  Then, adding or removing identical trajectories (i.e. vectors) to two sets of trajectories does not change their comparison by $\succeq_{lmax(lmin)}$ (resp. $\succeq_{lmin(lmax)}$) – while it may transform a strict preference into an indifference if u_{opt} (resp. u_{pes}) were used.
  To be more precise, assume, for example, that $\delta \succ_{lmax(lmin)} \delta'$.
  Then, $\exists i^*$, $\forall i < i^*$, $\tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)}$ and $\tau_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i^*)}$. The trajectories corresponding to δ'' are composed of trajectories which rank before $\tau_{\lambda(i^*)}$, and after $\tau_{\lambda(i^*)}$. Obviously, the ones that rank before $\tau_{\lambda(i^*)}$ are added to both lists of trajectories, and thus simply delay i^* while not inducing a new preference. And the ones that rank after $\tau_{\lambda(i^*)}$ are not taken into consideration in the comparison of $\delta + \delta''$ and $\delta' + \delta''$. In the same way, by definition of $\succeq_{lmin(lmax)}$ we get $\delta \succ_{lmin(lmax)} \delta'$ i.e. $\exists i^*$, $\forall i < i^*$, $\tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)}$ and $\tau_{\sigma(i^*)} \succ_{lmax} \tau'_{\sigma(i^*)}$. The same result as for $\tau_{\lambda(i^*)}$ handles for $\tau_{\sigma(i^*)}$. Thus, $\succeq_{lmax(lmin)}$ and $\succeq_{lmin(lmax)}$ are strictly monotonic. $\square$

Proof of Proposition 4. It is sufficient to show that:

$$\delta \succeq^{\mathcal{MDP}}_{lmax(lmin)} \delta' \Leftrightarrow \delta' \succeq^{\mathcal{MDP}^{inv}}_{lmin(lmax)} \delta, \tag{19}$$

where $\succeq^{\mathcal{MDP}^{inv}}_{lmin(lmax)}$ is the pessimistic lexicographic comparison of policies in the MDP where the utilities of all states have been reversed ($u'(N) = 1 - u(N)$).

For any two policies δ and δ':

- If $\delta \succ^{\mathcal{MDP}}_{lmax(lmin)} \delta'$ then $\exists i^*, \forall i \leq i*, \tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)}$ and $\tau_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i^*)} \Leftrightarrow (\mu_0, \pi_1, \mu_1, \pi_2, \ldots, \pi_{h-1}, \mu_h) \succ_{lmin} (u'_0, \pi'_1, \mu'_1, \pi'_2, \ldots, \pi'_{h-1}, \mu'_h)$.
Now let inverse each degree in both trajectories, we get:

$$((1-\mu_0),(1-\pi_1),(1-\mu_1),(1-\pi_2),\ldots,(1-\pi_{h-1}),(1-\mu_h)) \prec_{lmax}$$
$$((1-\mu'_0),(1-\pi'_1),(1-\mu'_1),(1-\pi'_2),\ldots,(1-\pi'_{h-1}),(1-\mu'_h))$$
$$\Leftrightarrow (\mu_0,(1-\pi_1),\mu_1,(1-\pi_2),\ldots,(1-\pi_{h-1}),\mu_h) \prec_{lmax} (\mu'_0,(1-\pi'_1),\mu'_1,(1-\pi'_2),\ldots,(1-\pi'_{h-1}),\mu'_h)$$

i.e. $\tau_{\sigma(i^*)} \prec_{lmax} \tau'_{\sigma(i^*)}$ which is the $\succeq_{lmax}$ relation when considering $\mathcal{MDP}^{inv}$. We get $\delta' \succeq^{\mathcal{MDP}^{inv}}_{lmin(lmax)} \delta$.
- If $\delta \sim^{\mathcal{MDP}}_{lmax(lmin)} \delta'$ then $\forall i,\ \tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)} \Leftrightarrow (\mu_0, \pi_1, \mu_1, \pi_2, \ldots, \pi_{h-1}, \mu_h) \sim_{lmin} (\mu'_0, \pi'_1, \mu'_1, \pi'_2, \ldots, \pi'_{h-1}, \mu'_h)$.
If we inverse each degree in both trajectories, we get:

$$((1-\mu_0),(1-\pi_1),(1-\mu_1),(1-\pi_2),\ldots,(1-\pi_{h-1}),(1-\mu_h)) \sim_{lmax}$$
$$((1-\mu'_0),(1-\pi'_1),(1-\mu'_1),(1-\pi'_2),\ldots,(1-\pi'_{h-1}),(1-\mu'_h))$$
$$\Leftrightarrow (\mu_0,(1-\pi_1),\mu_1,(1-\pi_2),\ldots,(1-\pi_{h-1}),\mu_h) \sim_{lmax} ((1-\pi'_1),\ldots,(1-\pi'_h),\mu'_h)$$

i.e. $\tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)}$. We get $\delta' \sim^{\mathcal{MDP}^{inv}}_{lmin(lmax)} \delta$. $\square$

Proof of Proposition 5.

- Note that, for any t, s, we have:

$$\left[U^t(s)^{lmaxlmin}\right]_{l,c} \text{ has the form } \left[\left(\begin{array}{c|c} \multicolumn{2}{c}{\ldots} \\ \hline \pi(s'_i|s,a), u(s'_i) & \\ \ldots & U^{t-1}(s'_i)^{lmaxlmin} \\ \pi(s'_i|s,a), u(s'_i) & \\ \hline \multicolumn{2}{c}{\ldots} \end{array}\right)^{lmaxlmin}\right]_{l,c}$$

Formally,

$$\left[U^t(s)^{lmaxlmin}\right]_{l,c} = [((\pi(s'_1|s,a), u(s'_1)) \otimes U^{t-1}(s'_1)^{lmaxlmin}) \oplus ((\pi(s'_2|s,a), u(s'_2)) \otimes U^{t-1}(s'_2)^{lmaxlmin})$$
$$\oplus \ldots \oplus ((\pi(s'_k|s,a), u(s'_k)) \otimes U^{t-1}(s'_k)^{lmaxlmin})].$$

Let A be a $n \times m$ matrix, $A^{lmaxlmin}_{(i,x:y)}$ denote the part of the line i, of A, having $y - x$ elements from column x to y s.t. $x < y \leq m$.
Now, note that, if A and B are two matrices with exactly c columns:
$\left[A^{lmaxlmin}\right]_{l,c} \succ_{lmaxlmin} \left[B^{lmaxlmin}\right]_{l,c}$ if and only if $\exists i^* \leq l$, such that $\forall i < i^*$, $A^{lmaxlmin}_{(i)} =_{lmin} B^{lmaxlmin}_{(i)}$ and $A^{lmaxlmin}_{(i^*)} >_{lmin} B^{lmaxlmin}_{(i^*)}$.
Clearly, in this case replacing A and B with $U^t(s)^{lmaxlmin}$ when considering δ and $U'^t(s)^{lmaxlmin}$ when considering δ', if such a $i^* \leq l$ exists for a given l, the same i^* works for $l' > l$.
Thus, $\succ_{lmaxlmin,l',c}$ refines $\succ_{lmaxlmin,l,c}$.

Remark that the property does not hold for c. Increasing c does not refine the order $\succ_{lmaxlmin,l,c,t,s}$. Indeed, given $c < c'$, we can find a pair of matrices for which it holds all together that:
– $A^{lmaxlmin}_{(1,1:c)} =_{lmin} B^{lmaxlmin}_{(1,1:c)}$,
– $A^{lmaxlmin}_{(2,1:c)} >_{lmin} B^{lmaxlmin}_{(2,1:c)}$ and
– $A^{lmaxlmin}_{(1,1:c')} <_{lmin} B^{lmaxlmin}_{(i^*,1:c')}$.
Thus, $A^{lmaxlmin} \succ_{lmaxlmin,l=2,c} B^{lmaxlmin}$ and $B^{lmaxlmin} \succ_{lmaxlmin,l=2,c'} A^{lmaxlmin}$. One can easily build a decision problem and two policies corresponding to matrices A and B satisfying the above. Thus, increasing c does not lead to a more refined order.

- From the first point, above, it holds that

$$\succ_{lmaxlmin,l=1,c} \text{ refines } \succ_{lmaxlmin,l=1,c=1} \text{ and that } \succ_{lmaxlmin,l,c} \text{ refines } \succ_{lmaxlmin,l=1,c}.$$

  So, $\succ_{lmaxlmin,l,c}$ refines $\succ_{lmaxlmin,l=1,c=1}$, which is equivalent to the order induced by u_{opt}. Thus, optimal solutions of $\succ_{lmaxlmin,l,c}$ are also optimal for u_{opt}, in all steps of the stationary $\Pi\mathcal{MDP}$. $\square$

Proof of Proposition 6. Note that

$$\left[(U \otimes (N_1, \ldots, N_a))^{lmaxlmin}\right]_{l,c} = [(U_1 \otimes N_1) \oplus (U_2 \otimes N_2) \oplus \ldots \oplus (U_k \otimes N_k)]^{lmaxlmin}.$$

Now, note the two following facts:

Fact 1. $(A \oplus B)^{lmaxlmin} = \left((A)^{lmaxlmin} \oplus (B)^{lmaxlmin}\right)^{lmaxlmin}$.

The reason is that: $A^{lmaxlmin}$ first reorders each line in *lmin* order, which can be done independently for each line and then all ordered lines are ordered through *lmax*. This second step can be done separately for each submatrix, provided that the lines are leximax-reordered once more, which is done by the external $lmax(lmin)$ operator.

Fact 2. $\left(U_{(i)} \otimes A\right)^{lmaxlmin} = \left(U_{(i)} \otimes (A)^{lmaxlmin}\right)^{lmaxlmin}$.

This second fact holds since adding identical elements to each line of a matrix does not modify the leximin ordering of the lines. In the right hand term of the equality, the outer *lmaxlmin* operator only inserts the terms of $U_{(i)}$ in all lines of matrix $A^{lmaxlmin}$.

Now, from Fact 1, we get:

$$(U \otimes (N_1, \ldots, N_a))^{lmaxlmin} = [(U_1 \otimes N_1)^{lmaxlmin} \oplus (U_2 \otimes N_2)^{lmaxlmin} \oplus \ldots \oplus (U_k \otimes N_k)^{lmaxlmin}]^{lmaxlmin}.$$

And then, from Fact 2:

$$(U \otimes (N_1, \ldots, N_a))^{lmaxlmin} = [(U_1 \otimes (N_1)^{lmaxlmin})^{lmaxlmin} \oplus (U_2 \otimes (N_2)^{lmaxlmin})^{lmaxlmin} \oplus \ldots \oplus (U_k \otimes (N_k)^{lmaxlmin})^{lmaxlmin}]^{lmaxlmin}.$$

Now, using Fact 1 again, in the other direction of the equality:

$$\begin{aligned}(U \otimes (N_1, \ldots, N_a))^{lmaxlmin} &= [(U_1 \otimes (N_1)^{lmaxlmin})^{lmaxlmin} \oplus (U_2 \otimes (N_2)^{lmaxlmin})^{lmaxlmin} \\ &\quad \oplus \ldots \oplus (U_k \otimes (N_k)^{lmaxlmin})^{lmaxlmin}]^{lmaxlmin} \\ &= \left(U \otimes ((N_1)^{lmaxlmin}, \ldots, (N_a)^{lmaxlmin})\right)^{lmaxlmin} \end{aligned} \tag{20}$$

From (20), we have, of course,

$$\left[(U \otimes (N_1, \ldots, N_a))^{lmaxlmin}\right]_{l,c} = \left[\left(U \otimes ((N_1)^{lmaxlmin}, \ldots, (N_a)^{lmaxlmin})\right)^{lmaxlmin}\right]_{l,c}.$$

$$\left[(U \times (N_1, \ldots, N_a))^{lmaxlmin}\right]_{l,c} = \left[\left(U \times (N_1^{lmaxlmin}, \ldots, N_a^{lmaxlmin})\right)^{lmaxlmin}\right]_{l,c}.$$

Now, notice that:

- the $N_i^{lmaxlmin}$ are lmax(min)-ordered, as well as the lines $\left(U_{(i)}\right)$,
- the $N_i^{lmaxlmin}$ have exactly c columns and
- since the resulting (reordered) matrix has l lines, it cannot contain more than l lines of any of the $N_i^{lmaxlmin}$ matrices.

We can safely replace the inner $N_i^{lmaxlmin}$ matrices with their sub-matrices $\left[N_i^{lmaxlmin}\right]_{l,c}$ and get the result. $\square$

Proof of Proposition 7. In bounded utility lexicographic value iteration, at each time step, $U^t(s)$ is composed of a set of trajectories $\tau_t^i = \langle s_0^i, a_0^i, s_1^i, \ldots, s_t^i, a_t^i, s_{t+1}^i \rangle$, which can be identified with the set of possibilities/utilities of each transition

$(s^i_{t'}, a^i_{t'}, s'^i_{t'+1})$ s.t. $t \geq t' \geq 0$, these are obtained from $\{\langle \pi^i_{t'}, u^i_{t'} \rangle\}_{t'=0,t}$. Thus, τ^i_t has $2t$ elements. Let $v^{i,\alpha}_t$ count the number of times a level $\alpha \in V$ has been obtained by $\pi^i_{t'}$ or $u^i_{t'}$ during the trajectory τ^i_t. Statistics $< v^{i,1}_t, \ldots, v^{i,|V|}_t >$ can be maintained for every trajectory.

As t increases all $(v^{i,\alpha}_t)$, which are non-decreasing sequences, converge toward a finite or infinite limit. We let

$$\lim(i,\alpha) = \lim_{t\to\infty} (v^{i,\alpha}_t), \text{ s.t. } \lim(i,\alpha) \in \mathcal{N} \cup \{+\infty\}, \ \forall\, \tau^i_t, \alpha.$$

Thus, if we denote U_i, the limit of the line vector in $U^t(s)$ corresponding to trajectory τ^i_t when t tends to infinity, we have:

$$U_i = \left[\underbrace{\alpha_1, \ldots, \alpha_1}_{\lim(i,1)}, \underbrace{\alpha_2, \ldots, \alpha_2}_{\lim(i,2)}, \ldots \right].$$

Let us now consider $U^*_{l,c}$, the $l \times c$ matrix made from the l first (in leximax order) line vectors $[U_i]_{1,c}$.

Then, obviously,

$$\lim_{t\to\infty} \left[U^t(s)^{lmaxlmin} \right]_{l,c} = U^*_{l,c}.$$

Thus, bounded utility lexicographic value iteration algorithm converges.

Besides, we show now that if $[U^t(s)^{lmaxlmin}]_{l,c} = [U^{t-1}(s)^{lmaxlmin}]_{l,c}$, thus, $\forall\ t' \geq t$, we have $[U^{t'}(s)^{lmaxlmin}]_{l,c} = [U^t(s)^{lmaxlmin}]_{l,c}$:

Let us consider the following hypothesis:

$$H:\ [U^t(s)^{lmaxlmin}]_{l,c} = [U^{t-1}(s)^{lmaxlmin}]_{l,c}.$$

Considering H, let us calculate $[U^{t+1}(s)^{lmaxlmin}]_{l,c}$:

$$Future^{t+1} = (U^t(s'), s' \in succ(s,a)) = Future^t,$$

$$\to\ Q^{t+1}(s,a) = [\left(TU_{s,a} \otimes Future^t\right)^{lmaxlmin}]_{l,c} = [\left(TU_{s,a} \otimes Future^{t-1}\right)^{lmaxlmin}]_{l,c} = Q^t(s,a).$$

We deduce that, $\left(U^{t+1}\right)^{lmaxlmin}_{l,c} = \left(U^{t-1}\right)^{lmaxlmin}_{l,c}$. $\square$

Proof of Proposition 8. We prove the convergence in two steps:

- We show that for any pair of successive policies (δ^{old}, δ) computed by the algorithm, we have:

  $$U^{\delta}_{l,c}(s) \geq^{lmaxlmin} U^{\delta^{old}}_{l,c}(s), \forall s \in S.$$

  As a consequence, and since the set of matrices $U_{l,c}$ is finite, the algorithm converges.

  To prove this, just notice that since $\delta(s) \leftarrow \arg\max^{lmax(lmin)}_a Q^{\delta^{old}}_{l,c}(s,a)$, $U^{\delta}_{l,c}(s) \geq^{lmaxlmin} Q^{\delta^{old}}_{l,c}(s,a), \forall a$. Thus,

  $$U^{\delta}_{l,c}(s) \geq^{lmaxlmin} Q^{\delta^{old}}_{l,c}(s, \delta^{old}(s)) = U^{\delta^{old}}_{l,c}(s).$$

- In a second step, we show that the fixed-points of $Lmax(lmin)$-BU-PI are also fixed points of $Lmax(lmin)$-BU-VI, which is enough to prove the correctness of the $Lmax(lmin)$-BU-PI algorithm (we have already shown the correctness of $Lmax(lmin)$-BU-VI).

  Let δ be a fixed-point of $Lmax(lmin)$-BU-PI. We have

  $$\delta(s) = \text{argmax}^{lmax(lmin)}_a Q^{\delta}_{l,c}(s,a), \forall s \text{ and}$$
  $$U^{\delta}_{l,c}(s) = \max^{lmax(lmin)}_a Q^{\delta}_{l,c}(s,a), \forall s.$$

  Let us now set $U^{t-1}_{l,c} = U^{\delta}_{l,c}$ in the $Lmax(lmin)$-BU-VI algorithm, and let us compute $U^t_{l,c}$:

  $$U^t_{l,c}(s) = \max^{lmax(lmin)}_a [(TU_{s,a} \times Future)^{lmaxlmin}]_{l,c}, \forall s.$$

  $$\begin{aligned}\text{But, } Future &=_{def} \left(U^{t-1}_{l,c}(s'), s' \in succ(s,a)\right) \\ &= \left(U^{\delta}_{l,c}(s'), s' \in succ(s,a)\right) \\ &= Future^{\delta}.\end{aligned}$$

Thus,

$$\begin{aligned} U^{t}_{l,c}(s) &= \max_a^{lmax(lmin)} [(TU_{s,a} \times Future^{\delta})^{lmaxlmin}]_{l,c} \\ &= \max_a^{lmax(lmin)} Q^{\delta}_{l,c}(s,a), \forall s \\ &= U^{\delta}_{l,c}(s), \forall s. \end{aligned}$$

Thus, $U^{\delta}_{l,c}(s)$ is a fixed-point of Lmax(lmin)-bounded Value Iteration. $\square$

Proof of Proposition 9. Note that Proposition 5 proves the right implication ($\Rightarrow$) for any $l > l^*$, $c > c^*$. Now, since for any possibilistic MDP the number of stationary policies is bounded, so is the number of possible different preorderings $\succeq$ over the set of bounded matrix utilities of policies in any state.

Thus, for any sequence $(\succeq_t)$ such that $\succeq_{t+1}$ (strictly) refines $\succeq_t$ converges in a finite number of iterations. This is the case for any sequence $\left(\succeq_t \equiv \succeq_{lmaxlmin,l_t,c_t}\right)$ with strictly increasing sequences (c_t), (l_t). Proposition 9 results. $\square$

Proof of Proposition 10. From Proposition 9, there exist a pair (l^*, c^*) of finite positive integers, such that, $\forall l > l^*, c > c^*$, $\forall \delta, \delta', s$,

$$\delta(s) \succeq_{lmaxlmin,l,c} \delta'(s) \Leftrightarrow \delta(s) \succeq_{lmaxlmin,l^*,c^*} \delta'(s).$$

Now, if we focus on the *bounded iterations lmax(lmin) Value iteration* algorithm, then for any $E > 0$, there obviously exist (l_E, c_E) such that

$$\delta(s) \succeq_{lmaxlmin,E} \delta'(s) \Leftrightarrow \delta(s) \succeq_{lmaxlmin,l_E,c_E} \delta'(s).$$

It is enough to take l_E as the maximum number of possible trajectories of length E for any policy and starting state, and to choose $c_E = 2E + 1$, as the length of trajectories.

Note that (l_E) and (c_E) can be chosen as strictly increasing functions of E. Thus, for the above defined pair (l^*, c^*), there exist E^* such that, $\forall E > E^*$, $l_E > l_{E^*} > l^*$ and $c_E > c_{E^*} > c^*$.

Thanks to Proposition 9, we get that, $\forall E > E^*$,

$$\delta(s) \succeq_{lmaxlmin,E} \delta'(s) \Leftrightarrow \delta(s) \succeq_{lmaxlmin,l^*,c^*} \delta'(s).$$

This means that the *bounded iterations lmax(lmin) value iteration* algorithm converges after a finite number of steps (E^*, here). $\square$

References

[1] K. Bauters, W. Liu, L. Godo, Anytime algorithms for solving possibilistic MDPs and hybridMDPs, in: Proceedings of 9th International Symposium on Foundations of Information and Knowledge Systems, FoIKS 2016, 2016, pp. 24–41.
[2] R. Bellman, Dynamic Programming, Princeton University Press, 1957.
[3] R. Bellman, A Markovian decision process, J. Math. Mech. 6 (1957).
[4] N. Ben Amor, Z. El Khalfi, H. Fargier, R. Sabbadin, Lexicographic refinements in possibilistic decision trees, in: Proceedings of the 22nd European Conference on Artificial Intelligence, ECAI 2016, 2016, pp. 202–208.
[5] N. Ben Amor, Z. El Khalfi, H. Fargier, R. Sabbadin, Efficient policies for stationary possibilistic Markov decision processes, in: Proceedings of 14th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU 2017, 2017, pp. 306–317.
[6] N. Drougard, F. Teichteil-Königsbuch, J.-L. Farges, D. Dubois, Qualitative possibilistic mixed-observable MDPs, in: Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence, UAI 2013, 2013, pp. 192–201.
[7] D. Dubois, L. Godo, H. Prade, A. Zapico, On the possibilistic decision model: from decision under uncertainty to case-based decision, Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 7 (6) (1999) 631–670.
[8] D. Dubois, H. Prade, Possibility Theory: An Approach to Computerized Processing of Uncertainty, Plenum Press, New York, 1988.
[9] D. Dubois, H. Prade, Possibility theory as a basis for qualitative decision theory, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI 1995, 1995, pp. 1925–1930.
[10] D. Dubois, H. Prade, R. Sabbadin, Qualitative decision theory with Sugeno integrals, in: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, UAI 1998, 1998, pp. 121–128.
[11] D. Dubois, H. Prade, R. Sabbadin, Decision-theoretic foundations of qualitative possibility theory, Eur. J. Oper. Res. 128 (3) (2001) 459–478.
[12] H. Fargier, R. Sabbadin, Qualitative decision under uncertainty: back to expected utility, in: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, IJCAI 2003, 2003, pp. 303–308.
[13] H. Fargier, R. Sabbadin, Qualitative decision under uncertainty: back to expected utility, Artif. Intell. 164 (2005) 245–280.
[14] H. Gilbert, P. Weng, Quantile reinforcement learning, Computing Research Repository, 2016.
[15] H. Gilbert, P. Weng, Y. Xu, Optimizing quantiles in preference-based Markov decision processes, in: Proceedings of the 31st Conference on Artificial Intelligence, AAAI 2017, 2017, pp. 3569–3575.
[16] A. Gupta, S. Kalyanakrishnan, Improved strong worst-case upper bounds for MDP planning, in: Proceedings of 26th International Joint Conference on Artificial Intelligence, MDAI 2016, 2017, pp. 1788–1794.
[17] R.A. Howard, Dynamic Programming and Markov Processes, MIT Press, Cambridge, MA, 1960.
[18] I. Montes, E. Miranda, S. Montes, Decision making with imprecise probabilities and utilities by means of statistical preference and stochastic dominance, Eur. J. Oper. Res. 234 (2014) 209–220.

[19] H. Moulin, Axioms of Cooperative Decision Making, Cambridge University Press, 1988.
[20] J.V. Neumann, O. Morgenstern, Theory of Games and Economic Behavior, 1944.
[21] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st edition, John Wiley & Sons, Inc., New York, NY, USA, 1994.
[22] R. Sabbadin, Possibilistic Markov decision processes, Eng. Appl. Artif. Intell. 14 (2001) 287–300.
[23] R. Sabbadin, Towards possibilistic reinforcement learning algorithms, in: 10th IEEE International Conference on Fuzzy Systems, vol. 1, 2001, pp. 404–407.
[24] R. Sabbadin, H. Fargier, J. Lang, Towards qualitative approaches to multi-stage decision making, Int. J. Approx. Reason. 19 (1998) 441–471.
[25] L.J. Savage, The Foundations of Statistics, J. Wiley, New York, 1954; second revised edition, 1972.
[26] R. Sutton, A. Barto, Introduction to Reinforcement Learning, MIT Press, 1998.
[27] B. Szörényi, R. Busa-Fekete, P. Weng, E. Hüllermeier, Qualitative multi-armed bandits: a quantile-based approach, in: Proceedings of the 32nd Interna-tional Conference on Machine Learning, ICML 2015, 2015, pp. 1660–1668.
[28] A. Wald, Statistical Decision Functions, Chelsea Pub. Co., 1971.
[29] P. Weng, Markov decision processes with ordinal rewards: reference point-based preferences, in: Proceedings of the 21st International Conference on Automated Planning and Scheduling, ICAPS 2011, 2011, pp. 282–289.
[30] T. Whalen, Decision making under uncertainty with various assumptions about available information, IEEE Trans. Syst. Man Cybern. 14 (1984) 888–900.
[31] R.R. Yager, J. Kacprzyk (Eds.), The Ordered Weighted Averaging Operators: Theory and Applications, Kluwer Academic Publishers, Norwell, MA, USA, 1997.
[32] R.R. Yager, Possibilistic decision making, IEEE Trans. Syst. Man Cybern. 9 (1979) 388–392.
[33] Y. Yue, J. Broder, R. Kleinberg, T. Joachims, The k-armed dueling bandits problem, J. Comput. Syst. Sci. 78 (5) (2012) 1538–1556.
[34] L. Zadeh, Fuzzy sets as a basis for theory of possibility, Fuzzy Sets Syst. 1 (1978) 3–28.