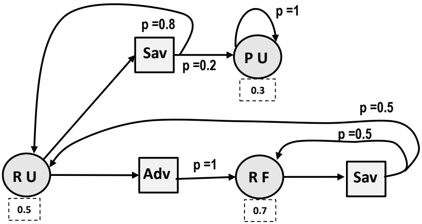
Efficient Policies for Stationary Possibilistic Markov Decision Processes

Nahla Ben Amor[1] **Zeineb El Khalfi**[1,2] Hélène Fargier[2] Régis Sabaddin[4]

[1]LARODEC, Tunis, Tunisia
[2]IRIT, Toulouse, France
[3]INRA, Toulouse, France

Fourteenth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty ECSQARU'2017
Lugano (Switzerland) - July 2017

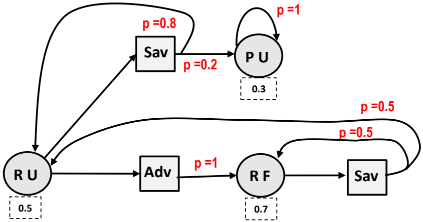
Markov Decision Processes



- $S = \{RU, RF, PU\}$:
  - RU: Rich and Unknown
  - RF: Rich and Famous
  - PU: Poor and Unknown
- $A = \{Sav, Adv\}$:
  - Sav: Saving money
  - Adv: Advertising

- S: finite set of states
- A: finite set of actions, $\rightarrow A_s$: the set of actions available from state s
- μ: utility function, $\rightarrow \mu(s)$: utility obtained in state $s \in S$

Markov Decision Processes



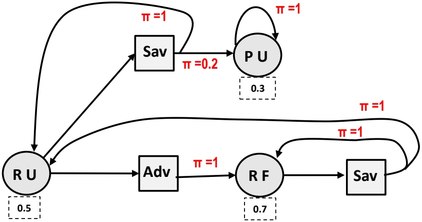
- $S = \{RU, RF, PU\}$:
  - RU: Rich and Unknown
  - RF: Rich and Famous
  - PU: Poor and Unknown
- $A = \{Sav, Adv\}$:
  - Sav: Saving money
  - Adv: Advertising

- S: finite set of states
- A: finite set of actions, $\rightarrow A_s$: the set of actions available from state s
- μ: utility function, $\rightarrow \mu(s)$: utility obtained in state $s \in S$

Uncertainty:

- p: finite set of probability distributions $p(s'|s,a)$: Probabilistic Markov decision process

Markov Decision Processes



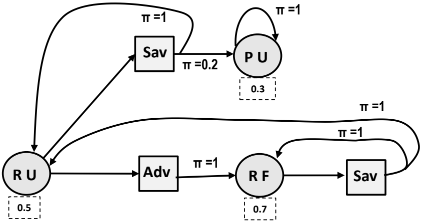
- $S = \{RU, RF, PU\}$:
  - RU: Rich and Unknown
  - RF: Rich and Famous
  - PU: Poor and Unknown
- $A = \{Sav, Adv\}$:
  - Sav: Saving money
  - Adv: Advertising

- S: finite set of states
- A: finite set of actions, $\rightarrow A_s$: the set of actions available from state s
- μ: utility function, $\rightarrow \mu(s)$: utility obtained in state $s \in S$

Uncertainty:

- π: finite set of possibility distributions $\pi(s'|s, a)$: Possibilistic Markov decision process

Markov Decision Processes



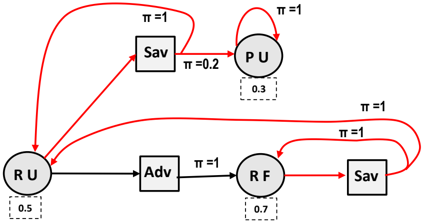
- $S = \{RU, RF, PU\}$:
  - RU: Rich and Unknown
  - RF: Rich and Famous
  - PU: Poor and Unknown
- $A = \{Sav, Adv\}$:
  - Sav: Saving money
  - Adv: Advertising

Policy

- $\delta \in \Delta : S \rightarrow A_s$

Δ	RU	PU	RF
δ_1	Sav	$Stay$	Sav
δ_2	Adv	$Stay$	Sav

Markov Decision Processes



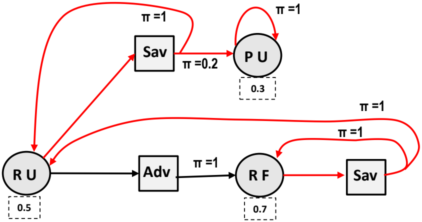
- $S = \{RU, RF, PU\}$:
  - RU: Rich and Unknown
  - RF: Rich and Famous
  - PU: Poor and Unknown
- $A = \{Sav, Adv\}$:
  - Sav: Saving money
  - Adv: Advertising

Policy

- $\delta \in \Delta : S \rightarrow A_s$

Δ	RU	PU	RF
δ_1	Sav	$Stay$	Sav
δ_2	Adv	$Stay$	Sav

Finite possibilistic Markov decision processes

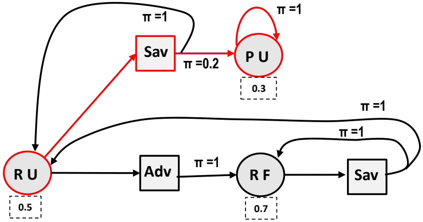


Policy

Δ	*RU*	*PU*	*RF*
δ_1	*Sav*	*Stay*	*Sav*
δ_2	*Adv*	*Stay*	*Sav*

- δ: set of trajectories $\tau \in \delta$
- With $E = 2$, δ_1 has 3 trajectories

Finite possibilistic Markov decision processes



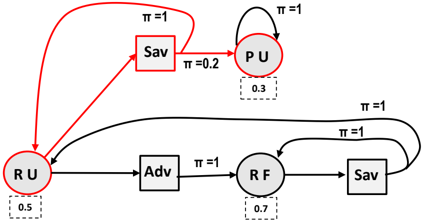
$$\tau_1 \begin{pmatrix} 0.5 & 0.2 & 0.3 & 1 & 0.3 \end{pmatrix}$$

Policy

Δ	*RU*	*PU*	*RF*
δ_1	*Sav*	*Stay*	*Sav*
δ_2	*Adv*	*Stay*	*Sav*

- With $E = 2$, δ_1 has 3 trajectories:
  - τ_1=(RU,Sav,PU,Stay,PU)

Finite possibilistic Markov decision processes



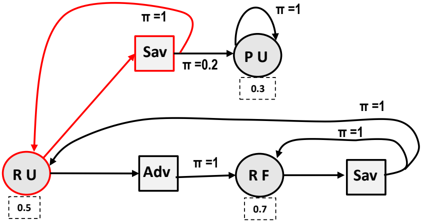
τ_1	0.5	0.2	0.3	1	0.3
τ_2	0.5	1	0.5	0.2	0.3

Policy

Δ	*RU*	*PU*	*RF*
δ_1	*Sav*	*Stay*	*Sav*
δ_2	*Adv*	*Stay*	*Sav*

- With $E = 2$, δ_1 has 3 trajectories:
  - τ_1=(RU,Sav,PU,Stay,PU)
  - τ_2=(RU,Sav,RU,Sav,PU)

Finite possibilistic Markov decision processes



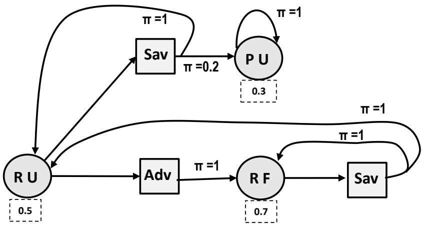
τ_1	0.5	0.2	0.3	1	0.3
τ_2	0.5	1	0.5	0.2	0.3
τ_3	0.5	1	0.5	1	0.5

Policy

Δ	*RU*	*PU*	*RF*
δ_1	*Sav*	*Stay*	*Sav*
δ_2	*Adv*	*Stay*	*Sav*

- With $E = 2$, δ_1 has 3 trajectories:
  - τ_1=(RU,Sav,PU,Stay,PU)
  - τ_2=(RU,Sav,RU,Sav,PU)
  - τ_3=(RU,Sav,RU,Sav,RU).

Possibilistic decision criteria



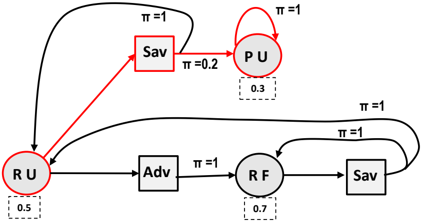
$u_{opt}(\delta_1, RU) = ?$

τ_1	0.5	0.2	0.3	1	0.3
τ_2	0.5	1	0.5	0.2	0.3
τ_3	0.5	1	0.5	1	0.5

Possibilistic qualitative utilities [Dubois et Prade, 1995; R. Sabbadin, 2001]

- Optimistic qualitative utility:
  $\delta_1 \succeq_{u_{opt}} \delta_2 \Leftrightarrow u_{opt}(\delta_1, s_0) \geq u_{opt}(\delta_2, s_0)$
  $u_{opt}(\delta, s_0) = \max_{\tau} \min\{\pi(\tau|s_0, \delta), \mu(\tau)\}$
- Pessimistic qualitative utility:
  $\delta_1 \succeq_{u_{pes}} \delta_2 \Leftrightarrow u_{pes}(\delta_1, s_0) \geq u_{pes}(\delta_2, s_0)$
  $u_{pes}(\delta, s_0) = \min_{\tau} \max\{1 - \pi(\tau|s_0, \delta), \mu(\tau)\}$

Possibilistic decision criteria



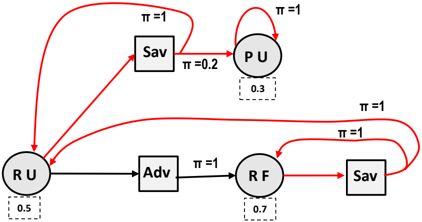
$u_{opt}(\delta_1, RU) = ?$

	μ	π	μ	π	μ	
τ_1	0.5	0.2	0.3	1	0.3	0.2
τ_2	0.5	1	0.5	0.2	0.3	
τ_3	0.5	1	0.5	1	0.5	

Possibilistic qualitative utilities [Dubois et Prade, 1995; R. Sabbadin, 2001]

- Optimistic qualitative utility:
  $\delta_1 \succeq_{u_{opt}} \delta_2 \Leftrightarrow u_{opt}(\delta_1, s_0) \geq u_{opt}(\delta_2, s_0)$
  $u_{opt}(\delta, s_0) = \max_{\tau} \min\{\pi(\tau|s_0, \delta), \mu(\tau)\}$
- Pessimistic qualitative utility:
  $\delta_1 \succeq_{u_{pes}} \delta_2 \Leftrightarrow u_{pes}(\delta_1, s_0) \geq u_{pes}(\delta_2, s_0)$
  $u_{pes}(\delta, s_0) = \min_{\tau} \max\{1 - \pi(\tau|s_0, \delta), \mu(\tau)\}$

Possibilistic decision criteria



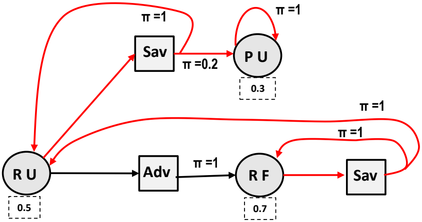
$u_{opt}(\delta_1, RU) = ?$

	μ	π	μ	π	μ	
τ_1	0.5	0.2	0.3	1	0.3	0.2
τ_2	0.5	1	0.5	0.2	0.3	0.2
τ_3	0.5	1	0.5	1	0.5	0.5

Possibilistic qualitative utilities [Dubois et Prade, 1995; R. Sabbadin, 2001]

- Optimistic qualitative utility:
  $\delta_1 \succeq_{u_{opt}} \delta_2 \Leftrightarrow u_{opt}(\delta_1, s_0) \geq u_{opt}(\delta_2, s_0)$
  $u_{opt}(\delta, s_0) = \max_{\tau} \min\{\pi(\tau|s_0, \delta), \mu(\tau)\}$
- Pessimistic qualitative utility:
  $\delta_1 \succeq_{u_{pes}} \delta_2 \Leftrightarrow u_{pes}(\delta_1, s_0) \geq u_{pes}(\delta_2, s_0)$
  $u_{pes}(\delta, s_0) = \min_{\tau} \max\{1 - \pi(\tau|s_0, \delta), \mu(\tau)\}$

Possibilistic decision criteria



$u_{opt}(\delta_1, RU) = 0.5$

	μ	π	μ	π	μ	
τ_1	0.5	0.2	0.3	1	0.3	0.2
τ_2	0.5	1	0.5	0.2	0.3	0.2
τ_3	0.5	1	0.5	1	0.5	0.5
						0.5

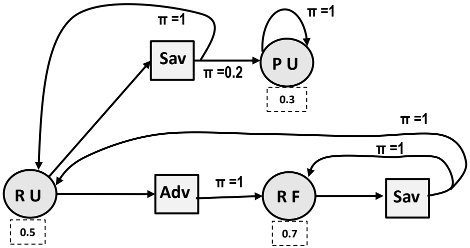
Possibilistic qualitative utilities [Dubois et Prade, 1995; R. Sabbadin, 2001]

- Optimistic qualitative utility:
  $\delta_1 \succeq_{u_{opt}} \delta_2 \Leftrightarrow u_{opt}(\delta_1, s_0) \geq u_{opt}(\delta_2, s_0)$
  $u_{opt}(\delta, s_0) = \max_{\tau} \min\{\pi(\tau|s_0, \delta), \mu(\tau)\} = 0.5$
- Pessimistic qualitative utility:
  $\delta_1 \succeq_{u_{pes}} \delta_2 \Leftrightarrow u_{pes}(\delta_1, s_0) \geq u_{pes}(\delta_2, s_0)$
  $u_{pes}(\delta, s_0) = \min_{\tau} \max\{1 - \pi(\tau|s_0, \delta), \mu(\tau)\} = 0.5$

Possibilistic value iteration algorithm

- Repeat modifications possibilistic value functions iteratively:
  - $u_{opt}(s) = \max_{a \in A_s} \max_{s' \in S} \min(\pi(s'|s,a), u_{opt}(s'))\}$
  - $u_{pes}(s) = \max_{a \in A_s} \min_{s' \in S} \max(1 - \pi(s'|s,a), u_{pes}(s'))\}$

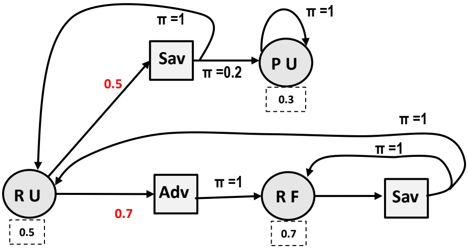
$\Rightarrow$ Choosing, for each state, an action that maximizes the utility $u_{opt}(s)$ or $u_{opt}(s)$ until:



Possibilistic value iteration algorithm

- Repeat modifications possibilistic value functions iteratively:
  - $u_{opt}(s) = \max_{a \in A_s} \max_{s' \in S} \min(\pi(s'|s,a), u_{opt}(s'))\}$
  - $u_{pes}(s) = \max_{a \in A_s} \min_{s' \in S} \max(1 - \pi(s'|s,a), u_{pes}(s'))\}$

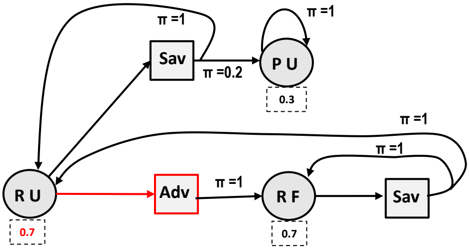
$\Rightarrow$ Choosing, for each state, an action that maximizes the utility $u_{opt}(s)$ or $u_{opt}(s)$ until:



Possibilistic value iteration algorithm

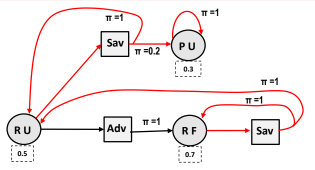
- Repeat modifications possibilistic value functions iteratively:
  - $u_{opt}(s) = \max_{a \in A_s} \max_{s' \in S} \min(\pi(s'|s,a), u_{opt}(s'))\}$
  - $u_{pes}(s) = \max_{a \in A_s} \min_{s' \in S} \max(1 - \pi(s'|s,a), u_{pes}(s'))\}$

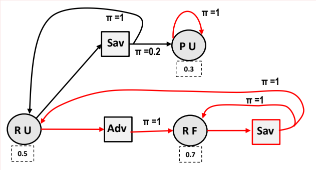
$\Rightarrow$ Choosing, for each state, an action that maximizes the utility $u_{opt}(s)$ or $u_{opt}(s)$ until:



Limitations of Possibilistic qualitative utilities

Drowning effect





$$\begin{array}{c} \tau_1 \\ \tau_2 \\ \tau_3 \end{array} \begin{pmatrix} 0.5 & 0.2 & 0.3 & 1 & 0.3 \\ 0.5 & 1 & 0.5 & 0.2 & 0.3 \\ 0.5 & 1 & 0.5 & 1 & 0.5 \end{pmatrix} \begin{array}{c} 0.2 \\ 0.2 \\ 0.5 \end{array}$$

$$u_{opt}(\delta_1) = u_{pes}(\delta_1) = 0.5$$

$$\begin{array}{c} \tau_4 \\ \tau_5 \end{array} \begin{pmatrix} 0.5 & 1 & 0.7 & 1 & 0.7 \\ 0.5 & 1 & 0.7 & 1 & 0.5 \end{pmatrix} \begin{array}{c} 0.5 \\ 0.5 \end{array}$$

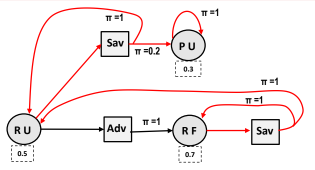
$$u_{opt}(\delta_2) = u_{pes}(\delta_2) = 0.5$$

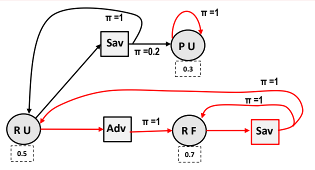
- $\tau_4, \tau_5 >> \tau_1$
- $\tau_4, \tau_5 >> \tau_2$
- $\tau_4 >> \tau_3$

Two policies are undistinguished although they give different consequences in some possible trajectories

Limitations of Possibilistic qualitative utilities

Drowning effect





$$\begin{array}{c} \tau_1 \\ \tau_2 \\ \tau_3 \end{array} \begin{pmatrix} 0.5 & 0.2 & 0.3 & 1 & 0.3 \\ 0.5 & 1 & 0.5 & 0.2 & 0.3 \\ 0.5 & 1 & 0.5 & 1 & 0.5 \end{pmatrix} \begin{array}{c} 0.2 \\ 0.2 \\ 0.5 \end{array}$$

$$u_{opt}(\delta_1) = u_{pes}(\delta_1) = 0.5$$

$$\begin{array}{c} \tau_4 \\ \tau_5 \end{array} \begin{pmatrix} 0.5 & 1 & 0.7 & 1 & 0.7 \\ 0.5 & 1 & 0.7 & 1 & 0.5 \end{pmatrix} \begin{array}{c} 0.5 \\ 0.5 \end{array}$$

$$u_{opt}(\delta_2) = u_{pes}(\delta_2) = 0.5$$

- $\tau_4, \tau_5 >> \tau_1$
- $\tau_4, \tau_5 >> \tau_2$
- $\tau_4 >> \tau_3$

Possibilistic utilities may fail to satisfy the Pareto efficiency

Objectives

Build efficient criteria

- Find a refinement $\succeq_x$ of $\succeq_{u_{opt}}$ s.t. :

$$\forall\, \delta_1, \delta_2 \in \Delta,\ \ \delta_1 \succ_{u_{opt}} \delta_2 \ \Rightarrow\ \delta_1 \succ_x \delta_2$$

- The $\succeq_x$ have to satisfy the principle of Pareto efficiency
- Compute optimal policies in possibilistic MDPs using these refinements

Lexicographic comparisons in non-sequential decision problems [Fargier and Sabbadin, 2005]

Refinement of max and min on vectors

- Lmax and lmin comparisons of vectors:

  $\vec{u} = (3, 2, 4), \vec{v} = (1, 2, 4)$

  1. Order the two vectors:
     - lmax: $\vec{u} = (3, 2, 4)$ $\vec{v} = (1, 2, 4)$

Lexicographic comparisons in non-sequential decision problems [Fargier and Sabbadin, 2005]

Refinement of max and min on vectors

- Lmax and lmin comparisons of vectors:

$\vec{u} = (3, 2, 4), \vec{v} = (1, 2, 4)$

1. Order the two vectors:
   - lmax: $\vec{u} = (4, 3, 2)$ $\vec{v} = (4, 2, 1)$

Lexicographic comparisons in non-sequential decision problems [Fargier and Sabbadin, 2005]

Refinement of max and min on vectors

- Lmax and lmin comparisons of vectors:
  1. Order the two vectors
  2. Compare the two vectors element by element:
     - lmax: $\vec{u} = (4, 3, 2)$ $\vec{v} = (4, 2, 1)$

Lexicographic comparisons in non-sequential decision problems [Fargier and Sabbadin, 2005]

Refinement of max and min on vectors

- Lmax and lmin comparisons of vectors:

  $\vec{u} = (3, 2, 4), \vec{v} = (1, 2, 4)$

  1. Order the two vectors
  2. Compare the two vectors element by element:
     - lmax: $\vec{u} = (4, 3, 2)$ $\vec{v} = (4, 2, 1)$

Lexicographic comparisons in non-sequential decision problems [Fargier and Sabbadin, 2005]

Refinement of max and min on vectors

- Lmax and lmin comparisons of vectors:

  $\vec{u} = (3, 2, 4), \vec{v} = (1, 2, 4)$

  1. Order the two vectors
  2. Compare the two vectors element by element:
     - lmax: $\vec{u} = (4, 3, 2) \succ_{lmax} \vec{v} = (4, 2, 1)$

Lexicographic comparisons in non-sequential decision problems [Fargier and Sabbadin, 2005]

Refinement of max and min on vectors

- Lmax and lmin comparisons of vectors:

  $\vec{u} = (3,2,4), \vec{v} = (1,2,4)$

  1. Order the two vectors
  2. Compare the two vectors element by element:
     - lmax: $\vec{u} = (4,3,2) \succ_{lmax} \vec{v} = (4,2,1)$
     - lmin: $\vec{u} = (2,3,4) \succ_{lmin} \vec{v} = (1,2,4)$

Lexicographic comparisons in non-sequential decision problems [Fargier and Sabbadin, 2005]

- Policy: a matrix of trajectories

$$\begin{matrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{matrix} \begin{pmatrix} 0.5 & 0.2 & 0.3 & 1 & 0.3 \\ 0.5 & 1 & 0.5 & 0.2 & 0.3 \\ 0.5 & 1 & 0.5 & 1 & 0.5 \end{pmatrix}$$

Main results

Lexicographic refinements of possibilistic utilities in MDPs

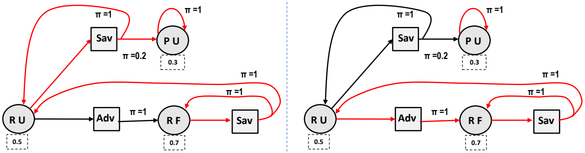
Lexicographic criteria on matrices of trajectories:

- lmax(lmin): refines u_{opt}
- lmin(lmax): refines u_{pes}

Proposition

- Lexicographic comparisons:
  - satisfy the principle of Pareto efficiency
  - satisfy the strict monotonicity property
  - satisfy transitivity

Lexicographic comparisons of matrices of trajectories



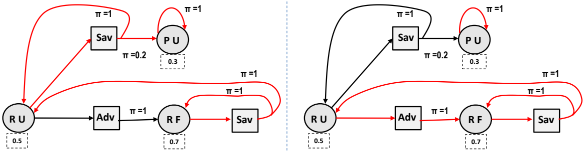
$$\begin{matrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{matrix} \begin{pmatrix} 0.5 & 0.2 & 0.3 & 1 & 0.3 \\ 0.5 & 1 & 0.5 & 0.2 & 0.3 \\ 0.5 & 1 & 0.5 & 1 & 0.5 \end{pmatrix}$$

$$\delta_1$$

$$\begin{matrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{matrix} \begin{pmatrix} 0.5 & 1 & 0.7 & 1 & 0.7 \\ 0.5 & 1 & 0.7 & 1 & 0.5 \\ e & e & e & e & e \end{pmatrix}$$

$$\delta_2$$

Lexicographic comparisons of matrices of trajectories

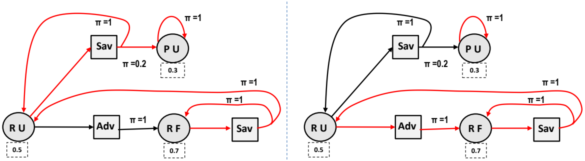


1. Order the elements of trajectories in increasing order

$$\delta_1: \begin{array}{c} \tau_1 \\ \tau_2 \\ \tau_3 \end{array} \begin{pmatrix} 0.2 & 0.3 & 0.3 & 0.5 & 1 \\ 0.2 & 0.3 & 0.5 & 0.5 & 1 \\ 0.5 & 0.5 & 0.5 & 1 & 1 \end{pmatrix}$$

$$\delta_2: \begin{array}{c} \tau_1 \\ \tau_2 \\ \tau_3 \end{array} \begin{pmatrix} 0.5 & 0.7 & 0.7 & 1 & 1 \\ 0.5 & 0.5 & 0.7 & 1 & 1 \\ e & e & e & e & e \end{pmatrix}$$

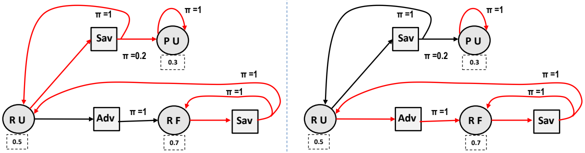
Lexicographic comparisons of matrices of trajectories



1. Order the elements of trajectories in increasing order
2. Order the trajectories in decreasing order

$$\delta_1: \begin{array}{c} \tau_3 \\ \tau_2 \\ \tau_1 \end{array} \begin{pmatrix} 0.5 & 0.5 & 0.5 & 1 & 1 \\ 0.2 & 0.3 & 0.5 & 0.5 & 1 \\ 0.2 & 0.3 & 0.3 & 0.5 & 1 \end{pmatrix} \qquad \delta_2: \begin{array}{c} \tau_1 \\ \tau_2 \\ \tau_3 \end{array} \begin{pmatrix} 0.5 & 0.7 & 0.7 & 1 & 1 \\ 0.5 & 0.5 & 0.7 & 1 & 1 \\ e & e & e & e & e \end{pmatrix}$$

Lexicographic comparisons of matrices of trajectories



1. Order the elements of trajectories in increasing order
2. Order the trajectories in decreasing order

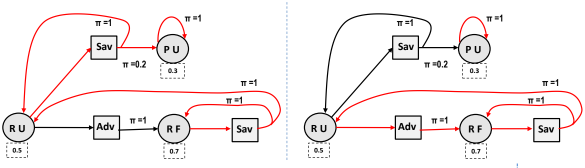
δ_1

τ_3	0.5 (uopt)	0.5	0.5	1	1
τ_2	0.2	0.3	0.5	0.5	1
τ_1	0.2	0.3	0.3	0.5	1

δ_2

τ_1	0.5 (uopt)	0.7	0.7	1	1
τ_2	0.5	0.5	0.7	1	1
τ_3	e	e	e	e	e

Lexicographic comparisons of matrices of trajectories



1. Order the elements of trajectories in increasing order
2. Order the trajectories in decreasing order
3. Compare the sets item by item

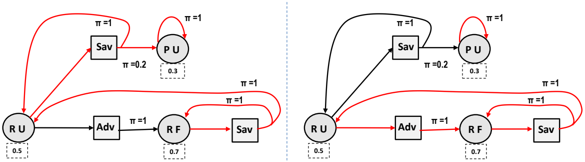
δ_1

τ_3	uopt 0.5	0.5	0.5	1	1
τ_2	0.2	0.3	0.5	0.5	1
τ_1	0.2	0.3	0.3	0.5	1

δ_2

τ_1	uopt 0.5	0.7	0.7	1	1
τ_2	0.5	0.5	0.7	1	1
τ_3	e	e	e	e	e

Lexicographic comparisons of matrices of trajectories



1. Order the elements of trajectories in increasing order
2. Order the trajectories in decreasing order
3. Compare the sets item by item

$\delta_2 \succ_{lmax(lmin)} \delta_1$

uopt

$$\delta_1 = \begin{array}{c} \tau_3 \\ \tau_2 \\ \tau_1 \end{array} \begin{pmatrix} 0.5 & 0.5 & 0.5 & 1 & 1 \\ 0.2 & 0.3 & 0.5 & 0.5 & 1 \\ 0.2 & 0.3 & 0.3 & 0.5 & 1 \end{pmatrix}$$

uopt

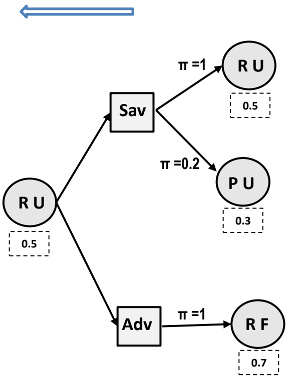
$$\delta_2 = \begin{array}{c} \tau_1 \\ \tau_2 \\ \tau_3 \end{array} \begin{pmatrix} 0.5 & 0.7 & 0.7 & 1 & 1 \\ 0.5 & 0.5 & 0.7 & 1 & 1 \\ e & e & e & e & e \end{pmatrix}$$

Finite-horizon lexicographic-value iteration algorithm

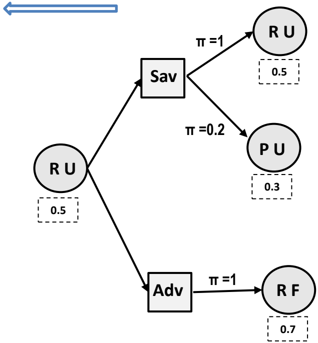
Algorithm 1: Lmax(lmin)-value iteration

Data: A possibilistic MDP and an horizon E
δ^*, the policy built by the algorithm, is a global variable
`// δ a global variable starts as an empty set`
Result: Computes and returns δ^* for MDP
begin
- $t \leftarrow 0$;
- **foreach** $s \in S$ **do** $U^t(s) \leftarrow ((\mu(s)))$;
- **foreach** $s \in S, a \in A_s$ **do** $TU_{s,a} \leftarrow T_{s,a} \times ((\mu(s')), s' \in succ(s,a))$;
- **repeat**
  - $t \leftarrow t + 1$;
  - **foreach** $s \in S$ **do**
    - $Q^* \leftarrow ((0))$;
    - **foreach** $a \in A$ **do**
      - $Future \leftarrow (U^{t-1}(s'), s' \in succ(s,a))$; `// Gather the matrices provided by the successors of s;`
      - $Q(s,a) \leftarrow (TU_{s,a} \times Future)^{lmaxlmin}$;
      - **if** $Q^* \preceq_{lmaxlmin} Q(s,a)$ **then** $Q^* \leftarrow Q(s,a)$; $\delta^t(s) \leftarrow a$;
    - $U^t(s) \leftarrow Q^*(s, \delta^t(s))$
- **until** $t == E$;
- $\delta^*(s) \leftarrow argmax_a Q(s,a)$
- **return** δ^*;

Lexicographic-value iteration algorithm

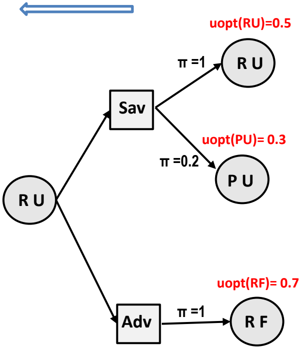


The best act (Sav or Adv) w.r.t uopt ?

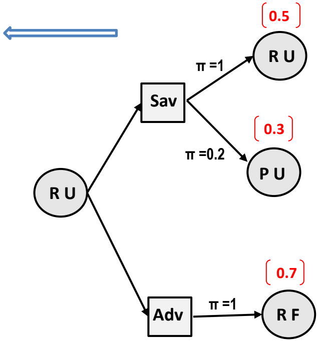


The best act (Sav or Adv) w.r.t lmax(lmin) ?

Lexicographic-value iteration algorithm

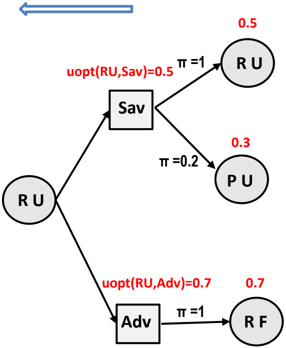


The best act (Sav or Adv) w.r.t uopt ?

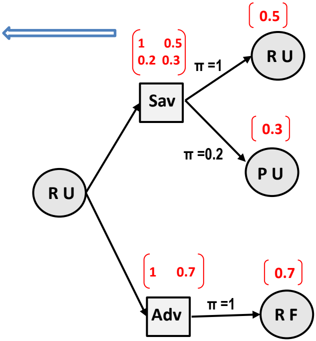


The best act (Sav or Adv) w.r.t lmax(lmin) ?

Lexicographic-value iteration algorithm

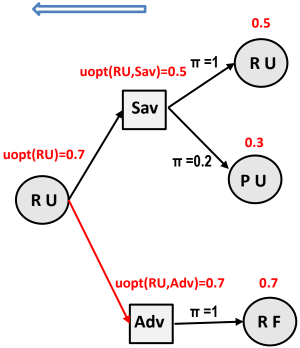


The best act (Sav or Adv) w.r.t uopt ?

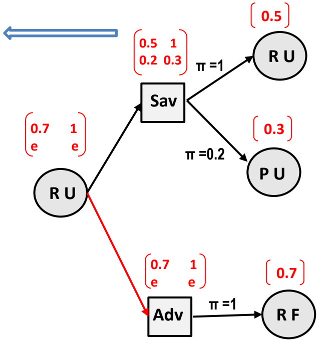


The best act (Sav or Adv) w.r.t lmax(lmin) ?

Lexicographic-value iteration algorithm



The best act is Adv

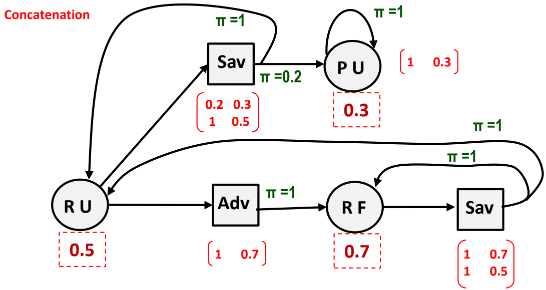


The best act is Adv

Lexicographic-value iteration algorithms

Finite-horizon case

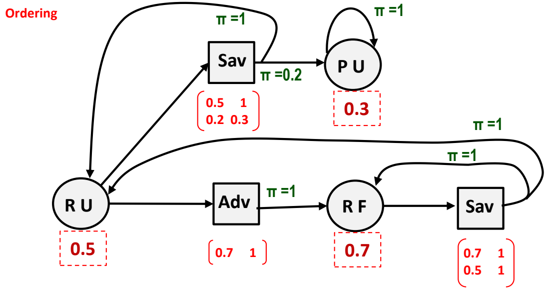
- Updates utilities of each state represented with a finite matrix of trajectories
- Number of iteration is fixed in advance



Lexicographic-value iteration algorithms

Finite-horizon case

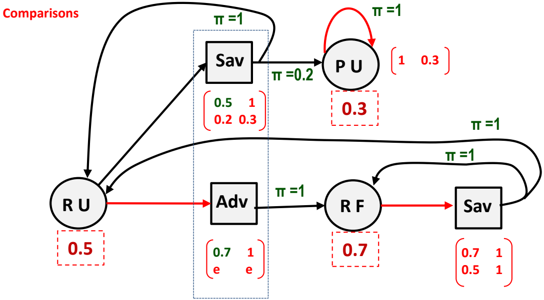
- Updates utilities of each state represented with a finite matrix of trajectories
- Number of iteration is fixed in advance



Lexicographic-value iteration algorithms

Finite-horizon case

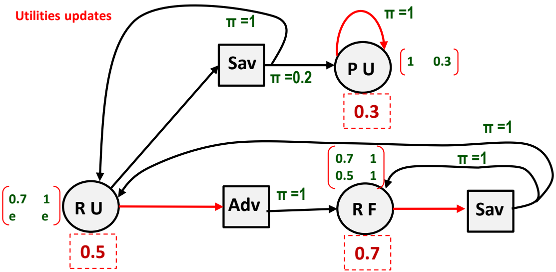
- Updates utilities of each state represented with a finite matrix of trajectories
- Number of iteration is fixed in advance



Lexicographic-value iteration algorithms

Finite-horizon case

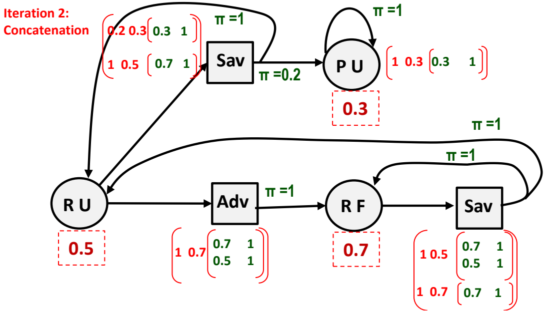
- Updates utilities of each state represented with a finite matrix of trajectories
- Number of iteration is fixed in advance



Lexicographic-value iteration algorithms

Finite-horizon case

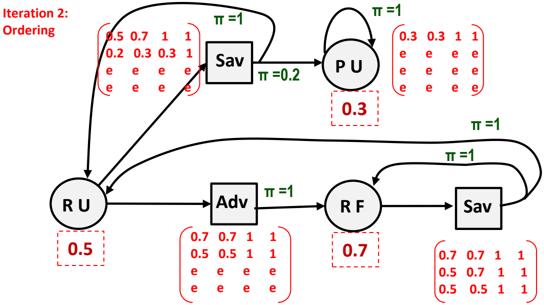
- Updates utilities of each state represented with a finite matrix of trajectories
- Number of iteration is fixed in advance



Lexicographic-value iteration algorithms

Finite-horizon case

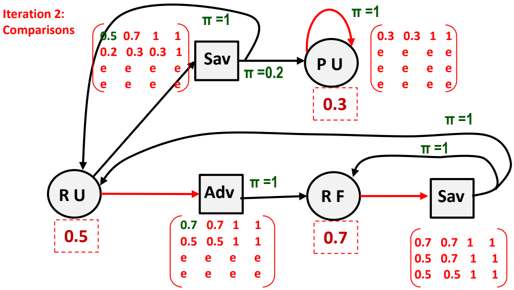
- Updates utilities of each state represented with a finite matrix of trajectories
- Number of iteration is fixed in advance



Lexicographic-value iteration algorithms

Finite-horizon case

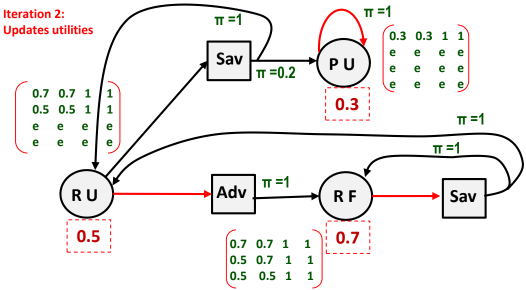
- Updates utilities of each state represented with a finite matrix of trajectories
- Number of iteration is fixed in advance



Lexicographic-value iteration algorithms

Finite-horizon case

- Updates utilities of each state represented with a finite matrix of trajectories
- Number of iteration is fixed in advance

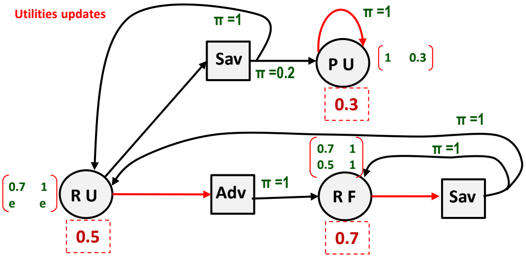


Lexicographic-value iteration algorithms

Unbounded lexicographic value iteration

- Complexity: $O(|S| \cdot |A| \cdot |E| \cdot b^E)$

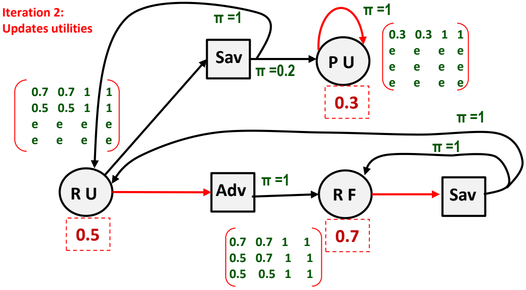
Example with Lmax(lmin)



Lexicographic-value iteration algorithms

Unbounded lexicographic value iteration

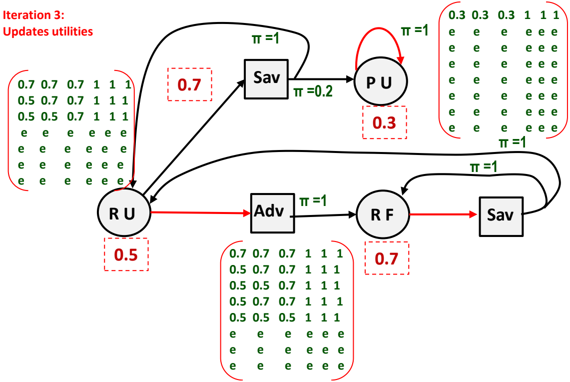
- Complexity: $O(|S| \cdot |A| \cdot |E| \cdot b^E)$



Lexicographic-value iteration algorithms

Unbounded lexicographic value iteration

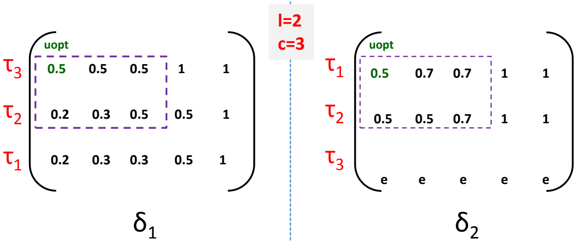
- Complexity: $O(|S| \cdot |A| \cdot |E| \cdot b^E)$



Lexicographic-value iteration algorithms

Bounded lexicographic value iteration

- Restriction size of matrices to (l,c) lines and columns:
  - $l > 1$ and $c > 1$
  - $\geq_{lmaxlmin,1,1}$ corresponds to $\succeq_{opt}$
  - $\geq_{lmaxlmin,+\infty,+\infty}$ corresponds to $\geq_{lmaxlmin}$
- Complexity: $O(|E| \cdot |S| \cdot |A| \cdot (l \cdot c) \cdot b \; log((l \cdot c) \cdot b))$



Lexicographic-value iteration algorithms

Bounded lexicographic value iteration

- Restriction size of matrices to (l,c) lines and columns:
  - $l > 1$ and $c > 1$
  - $\geq_{lmaxlmin,1,1}$ corresponds to $\succeq_{opt}$
  - $\geq_{lmaxlmin,+\infty,+\infty}$ corresponds to $\geq_{lmaxlmin}$
- Complexity: $O(|E| \cdot |S| \cdot |A| \cdot (l \cdot c) \cdot b\ log((l \cdot c) \cdot b))$

l=2
c=3

$$\delta_1: \quad \begin{matrix} & \text{uopt} & & \\ \tau_1 & 0.5 & 0.5 & 0.5 \\ \tau_2 & 0.2 & 0.3 & 0.5 \end{matrix}$$

$$\delta_2: \quad \begin{matrix} & \text{uopt} & & \\ \tau_1 & 0.5 & 0.7 & 0.7 \\ \tau_2 & 0.5 & 0.5 & 0.7 \end{matrix}$$

Finite-horizon lexicographic-value iteration algorithms

Algorithm 2: Bounded-Lmax(lmin)-value iteration

Data: A possibilistic MDP and an horizon E

δ^*, the policy built by the algorithm, is a global variable

`// δ a global variable starts as an empty set`

Result: Computes and returns δ^* for MDP

begin

- $t \leftarrow 0$;
- **foreach** $s \in S$ **do** $U^t(s) \leftarrow ((\mu(s)))$;
- **foreach** $s \in S, a \in A_s$ **do** $TU_{s,a} \leftarrow T_{s,a} \times ((\mu(s')), s' \in succ(s,a))$;
- **repeat**
  - $t \leftarrow t + 1$;
  - **foreach** $s \in S$ **do**
    - $Q^* \leftarrow ((0))$;
    - **foreach** $a \in A$ **do**
      - $Future \leftarrow (U^{t-1}(s'), s' \in succ(s,a))$; `// Gather the matrices provided by the successors of s;`
      - $Q(s,a) \leftarrow [(TU_{s,a} \times Future)^{lmaxlmin}]_{l,c}$;
      - **if** $Q^* \leq_{lmaxlmin} Q(s,a)$ **then** $Q^* \leftarrow Q(s,a)$; $\delta^t(s) \leftarrow a$;
    - $U^t(s) \leftarrow Q^*(s, \delta^t(s))$
- **until** $t == E$;
- $\delta^*(s) \leftarrow argmax_a Q(s,a)$
- **return** δ^*;

Infinite-horizon lexicographic-value iteration algorithms

Algorithm 3: Infinite-horizon-Lmax(lmin)-value iteration

Data: A possibilistic MDP and an horizon E
δ^*, the policy built by the algorithm, is a global variable
`// δ a global variable starts as an empty set`
Result: Computes and returns δ^* for MDP
begin
  $t \leftarrow 0$;
  foreach $s \in S$ **do** $U^t(s) \leftarrow ((\mu(s)))$;
  foreach $s \in S, a \in A_s$ **do** $TU_{s,a} \leftarrow T_{s,a} \times ((\mu(s')), s' \in succ(s,a))$;
  repeat
    $t \leftarrow t + 1$;
    foreach $s \in S$ **do**
      $Q^* \leftarrow ((0))$;
      foreach $a \in A$ **do**
        $Future \leftarrow (U^{t-1}(s'), s' \in succ(s,a))$; `// Gather the matrices provided by the successors of s;`
        $Q(s,a) \leftarrow [(TU_{s,a} \times Future)^{lmaxlmin}]_{l,c}$;
        if $Q^* \leq_{lmaxlmin} Q(s,a)$ **then** $Q^* \leftarrow Q(s,a)$; $\delta^t(s) \leftarrow a$;
      $U^t(s) \leftarrow Q^*(s, \delta^t(s))$
  until $(U^t)^{lmaxlmin}_{l,c} == (U^{t-1})^{lmaxlmin}_{l,c}$;
  $\delta^*(s) \leftarrow argmax_a Q(s,a)$
  return δ^*;

Pessimistic lexicographic criterion (lmin(lmax))

- Same results and algorithms
- Trajectories : $\tau(\mu_0, 1-pi_1, \mu_1, 1-\pi_2, \ldots, 1-\pi_{E-1}, \mu_E)$

$$\begin{matrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{matrix} \begin{pmatrix} 0.5 & 1-0.2 & 0.3 & 1-1 & 0.3 \\ 0.5 & 1-1 & 0.5 & 1-0.2 & 0.3 \\ 0.5 & 1-1 & 0.5 & 1-1 & 0.5 \end{pmatrix}$$

$$\delta_1$$

$$\begin{matrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{matrix} \begin{pmatrix} 0.5 & 1-1 & 0.7 & 1-1 & 0.7 \\ 0.5 & 1-1 & 0.7 & 1-1 & 0.5 \\ e & 1-e & e & 1-e & e \end{pmatrix}$$

$$\delta_2$$

Experimental protocol

Objective

- Evaluation of possibilistic Markov decision processes:
  - Unbounded value iteration algorithm ($UL - VI$)
  - Bounded value iteration algorithm ($BL - VI$) with different values of (l,c)
- Comparative study of evaluation algorithms (Solutions, CPU time)

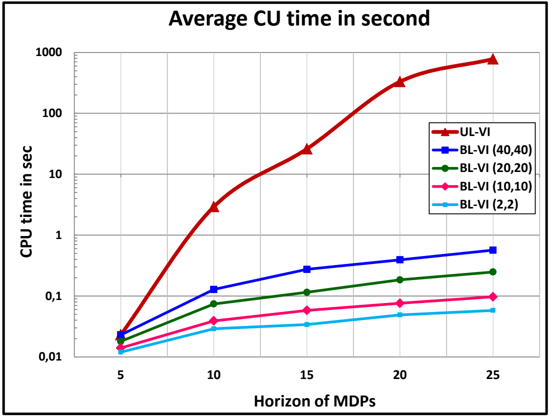
Data

- 100 Possibilisic MDPs randomly generated
- $|S| = 25$ and $|A_s| = 4$
- Values of utilities and conditional possibilities of decisions are chosen randomly

Experimental results

Execution CPU time

- $BL - VI$ is obviously faster than $UL - VI$



Experimental results

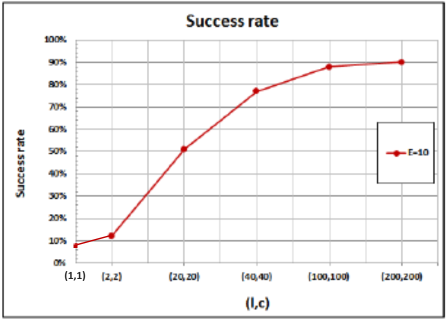
Pairwise success rate

$Success\frac{BL-VI}{UL-VI}$: Percentage of solutions provided w.r.t $BL-VI$ that are optimal w.r.t. $UL-VI$:

- the higher $Success\frac{BL-VI}{UL-VI}$: the more important effectiveness of cutting matrices
- the lower $Success\frac{BL-VI}{UL-VI}$: the more important drowning effect

Results

- $BL-VI$ provides good approximation (when (l, c) increase)



Experimental results

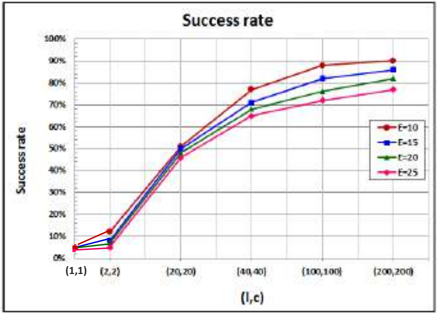
Pairwise success rate

$Success_{\frac{BL-VI}{UL-VI}}$: Percentage of solutions provided w.r.t $BL-VI$ that are optimal w.r.t. $UL-VI$:

- the lower $Success_{\frac{BL-VI}{UL-VI}}$: the more important drowning effect
- the higher $Success_{\frac{BL-VI}{UL-VI}}$: the more important effectiveness of cutting matrices

Results

- $BL-VI$ provides good approximation (when (l, c) increase)



Conclusion & Future work

Conclusion

- Extend lexicographic refinements to possibilistic Markov decision processes
- Escape the drowning effect of possibilistic utilities in policies
- Propose lexicographic value iteration algorithm

Future work

- Simulation based algorithms (Reinforcement learning)
- Solve lexico-possibilistic Influence diagrams

Thank you for your attention

Lexicographic comparisons and Expected utility: Example

Drowning effect

Two policies are undistinguished although they give different consequences in some possible trajectories

Possibilistic utilities may fail to satisfy the Pareto efficiency

$\forall \delta, \delta' \in \Delta$, if:

$$\left.\begin{array}{l} \forall D \in \text{Common}(\delta^1, \delta^2),\ \ \delta^1_D \succeq_O \delta^2_D \\ \exists D \in \text{Common}(\delta^1_D, \delta^2_D),\ \ \delta^1_D \succ_O \delta^2_D \end{array}\right\} \textit{then } \delta^1 \succ_O \delta^2$$

- $\text{Common}(\delta, \delta')$: set of decision nodes for both δ and δ'
- δ_D: is the restriction of δ to the subtree rooted in D.

Possibilistic utilities do not satisfy the property of strict monotonicity

$\forall \delta_1, \delta_2, \delta_3 \in \Delta$, a criterion O: $\quad \delta_1 \succeq_O \delta_2 \iff \delta_1 + \delta_3 \succeq_O \delta_2 + \delta_3$

Lexicographic comparisons

lexicographic comparisons on trajectories

- $\tau \succeq_{lmin} \tau'$ iff $(\mu_0, \pi_1, \ldots, \pi_E, \mu_E) \succeq_{lmin} (\mu'_0, \pi'_2, \ldots, \pi'_E, \mu'_E)$
- $\tau \succeq_{lmax} \tau'$ iff $(\mu_0, 1-\pi_1, \ldots, 1-\pi_E, \mu_E) \succeq_{lmax} (\mu'_0, 1-\pi'_1, \ldots 1-\pi'_E, \mu'_E)$

lexicographic comparisons on policies

- $\delta \succeq_{lmax(lmin)} \delta'$ 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* $\tau_{\mu(i^*)} \succ_{lmin} \tau'_{\mu(i^*)}$
- $\delta \succeq_{lmin(lmax)} \delta'$ 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* $\tau_{\sigma(i^*)} \succ_{lmax} \tau'_{\sigma(i^*)}$

Lexicographic comparisons and Expected utility

Transformation function

- Transformation of a possibilistic DT into a probabilistic one using: $\phi : L \rightarrow [0,1]$ s.t. $\phi(0_L) = 0$ and $\phi(1_L) = 1$
  - EU_{opt} refines u_{opt}
  - EU_{pes} refines u_{pes}

Result

- $C_{h,b}$: $\forall \alpha, \alpha' \in L, \alpha > \alpha' : \phi(\alpha)^{h+1} > b^h \phi(\alpha')$
- $\forall$ DT , $C_{h,b}$ is Sufficient condition to get:

$$\delta_1 >_{u_{opt}} \delta_2 \Rightarrow \delta_1 >_{EU_{opt}} \delta_2, \forall (\delta_1, \delta_2) \in \Delta$$

$$\delta_1 \succeq_{lmax(lmin)} \delta_2 \iff \delta_1 \succeq_{EU_{opt}} \delta_2, \forall (\delta_1, \delta_2) \in \Delta$$

.