

Some Complexity Results for Markov Decision Processes

Daniel S. Bernstein
Neil Immerman
Shlomo Zilberstein
University of Massachusetts

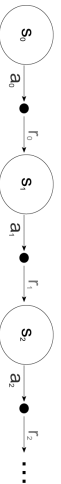
Robert Givan
Purdue University

April 24, 2001

Markov Decision Processes (MDPs)

- A simple, general framework for modeling sequential decision making in a stochastic environment
- Early work by Bellman and Howard in the 1950s
- Currently a popular model in OR and AI
- Used to model problems in robotics, finance, agriculture, etc.

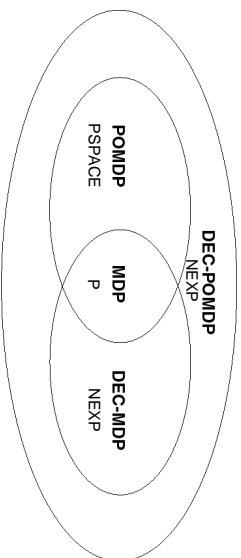
Markov Decision Processes (MDPs)



- The decision-making agent starts in some state and chooses an action according to its policy
- This determines an immediate reward and causes a stochastic transition to the next state
- Aim is to find the policy that leads to the highest total reward over T time steps (finite horizon)
- Because of the Markov property, the policy does not have to remember previous states

The formal decision problem - MDP

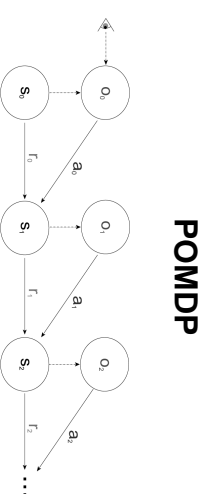
- Given $\langle S, A, P, R, T, K \rangle$
 - S is a finite state set (with start state s_0)
 - A is a finite action set
 - $P(s' | s, a)$ is a table of transition probabilities
 - $R(s, a, s')$ is a reward function
- Policy $\pi(s, t) = a$
- Is there a policy that yields total reward at least K over finite horizon T
- We will require that $T < |S|$



- Two natural dimensions of generalization:
 - Partial observability
 - Decentralization

POMDP

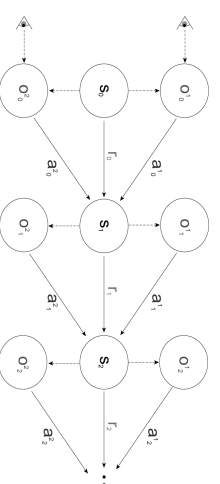
- Given $\langle S, A, P, R, \Omega, O, T, K \rangle$
 - Ω is a finite observation set
 - $O(o | s, a, s')$ is a table of observation probabilities
- Policy $\pi(O_1, \dots, O_t) = a$
- Does there exist a policy that yields total reward at least K over horizon T ?



POMDP

- The agent's observation is a function of the current state
- Now you may need to remember previous observations in order to act optimally

DEC-POMDP₂

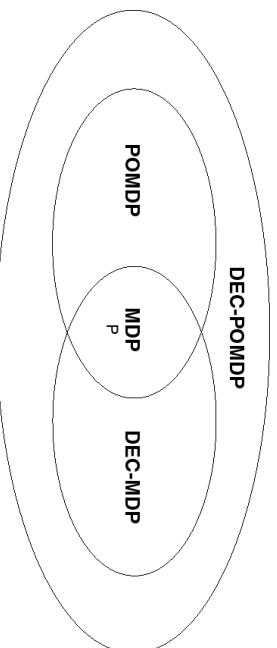


- Now two cooperating agents with different observation functions

DEC-POMDP₂

- Given $\langle S, A, P, R, \Omega_1, \Omega_2, O, T, K \rangle$
 - Ω_1 and Ω_2 are finite observation sets
 - $O(o^1, o^2 | s, a^1, a^2, s')$ is a table of observation probabilities
- Local policies $\pi^i(o^1_{t_1}, \dots, o^1_{t_i}) = a^1, \pi^2(o^2_{t_1}, \dots, o^2_{t_i}) = a^2$
- Joint policy $\langle \pi^1, \pi^2 \rangle$
- Does there exist a joint policy that yields total reward at least K over horizon T ?

Note: **DEC-POMDP₁ = POMDP**



DEC-MDP₂

- **DEC-POMDP₂** with requirement that the state is *uniquely* determined by the tuple of observations:
 $J : \Omega_1 \times \Omega_2 \rightarrow S$

Note: **DEC-MDP₁ = MDP**

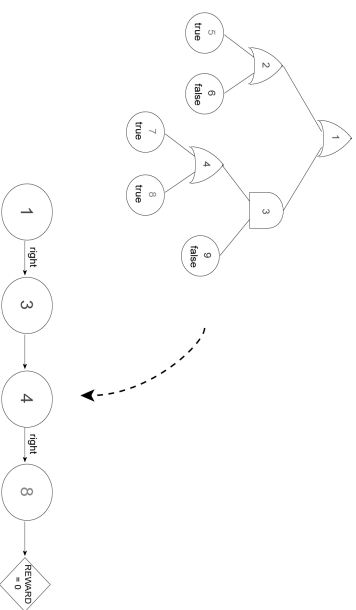
MDP $\in P$

- Can use dynamic programming to “back up” values from the end

$$V(s, t) = \max_{a \in A} \left(R(s, a) + \sum_{s' \in S} P(s' | s, a) V(s', t + 1) \right)$$

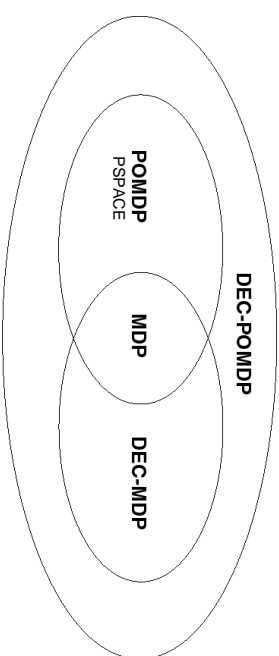
MDP is P-hard

- Papadimitriou and Tsitsiklis (1987)
- Reduction from Circuit Value Problem (CVP)



POMDP ∈ PSPACE

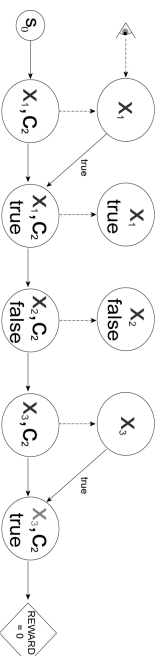
- Think of the possible outcomes of the process as a tree of depth T
- The leaves of the tree give the total reward
- To determine whether the optimal policy yields reward K , we traverse the tree, making decisions at nondeterministic nodes
- Need to be careful to make sure the same decision is made for the same observation sequence (can be done by grouping same observations at each internal tree node)

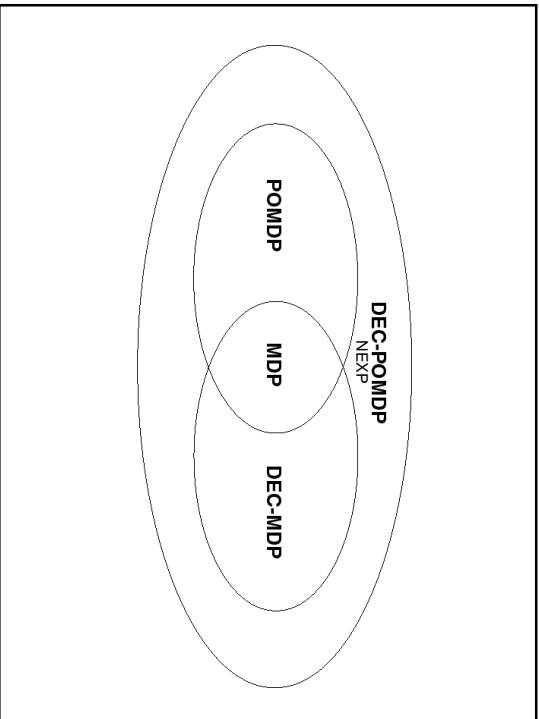


POMDP is PSPACE-hard

- Papadimitriou and Tsitsiklis (1987)
- Reduction from QBF

$$\exists x_1 \forall x_2 \exists x_3 (x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3)$$





DEC-POMDP_m ∈ NEXP

- Guess a joint policy
- The joint policy together with the DEC-POMDP yield an exponentially bigger Markov process with states being observation sequences
- The value of the joint policy can be determined using dynamic programming on this bigger Markov process

DEC-POMDP₂ is NEXP-hard

- Bernstein, Givan, Immerman, and Zilberstein (2000)
- Reduction from **TILING**
- Given n, L, H, V , is there a *consistent tiling*?

The diagram illustrates a tiling problem. It shows three sets of tiles: $L = \{ \text{white square}, \text{black square} \}$, $H = \{ \text{white square}, \text{black square}, \text{white square} \}$, and $V = \{ \text{white square}, \text{black square}, \text{white square}, \text{black square} \}$. Below these is a 4x4 grid labeled "a consistent tiling" with columns numbered 1-4 and rows numbered 1-4. The grid is filled with tiles from the sets L, H, and V such that they fit together perfectly.

DEC-POMDP₂ is NEXP-hard

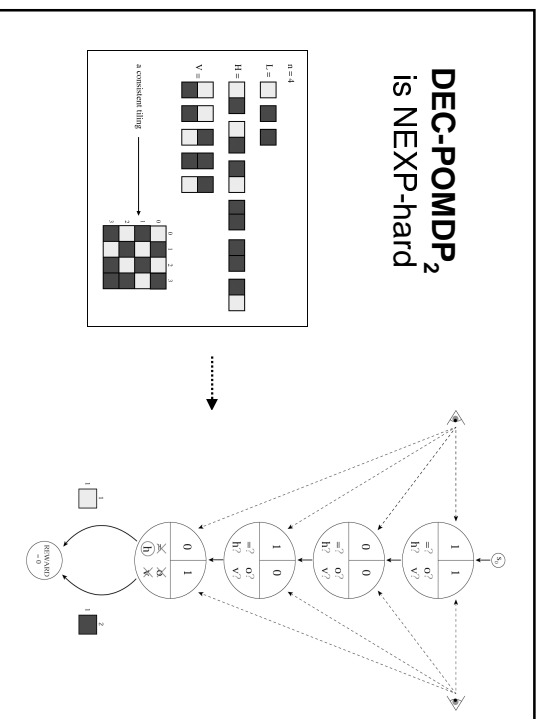
- Constructing an equivalent DEC-POMDP:

The diagram shows a DEC-POMDP construction. At the top, a circle labeled "Process randomly chooses two tile positions" has arrows pointing to "Agent 1 observes tile position 1" and "Agent 2 observes tile position 2". Below this, a central circle shows two tile positions, 1 and 2. "Agent 1 chooses a tile type 't'" and "Agent 2 chooses a tile type 't'" are shown with arrows pointing to the central circle. Below the central circle, a circle labeled "REWARD = 0" is shown with an arrow pointing to it from the central circle. A feedback loop arrow points from the reward back to the central circle. Below the diagram, text reads: "Process checks that the choices come from one consistent tiling".

- \exists policy with expected reward 0 \leftrightarrow \exists consistent tiling

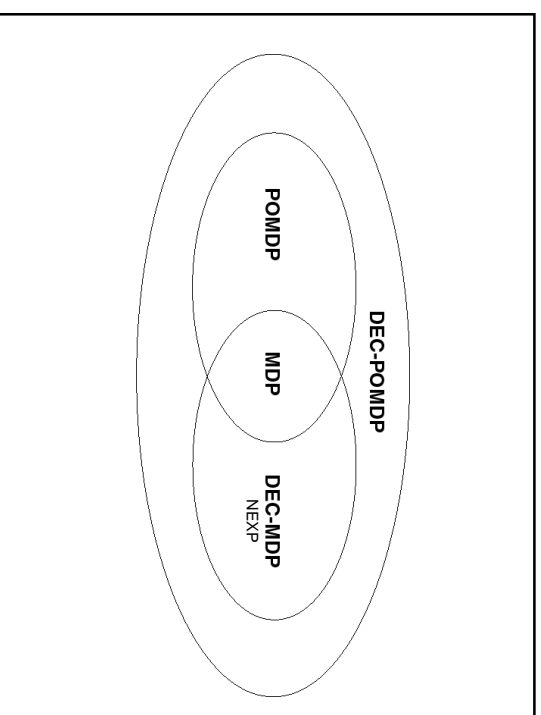
DEC-POMDP₂ is NEXP-hard

- Problem: process has a state for every possible pair of tile positions
- So the state space is exponentially bigger than the tiling instance (because the grid size n was given in binary)!
- Solution: notice that only certain info about the positions needs to be remembered to check whether constraints are violated
- Choose positions bit-by-bit and only remember important information in the state



DEC-POMDP₂ is NEXP-hard

- Each piece of info has a corresponding "small" automaton
- Equal tile positions: $((00) + (11))^*$
- Position 1 is origin: $((00) + (01))^*$
- Horizontally adjacent: $((00) + (11))^{(log n)} [01] [01] ((00) + (11))^*$
- Vertically adjacent: $[10] [01] ((00) + (11)) (00) + (11))^{(log n)}$
- Allows us to construct a DEC-POMDP that does the same thing as before but has only *polynomially* many states



DEC-MDP_m ∈ NEXP

- Because a DEC-MDP is a restricted DEC-POMDP

DEC-MDP₃ is NEXP-hard

- Reduce from DEC-POMDP₂ by adding a third agent that sees all but can do nothing

What about DEC-MDP₂?

- Can't we just reduce from DEC-POMDP₂ by cleverly encrypting the hidden state information?
- Surprisingly, the agents can use seemingly useless information to "cheat"

What about DEC-MDP₂?

- Example to demonstrate core problem: unsatisfiable sentence
 $(x \vee y) \wedge (\neg x \vee y) \wedge (x \vee \neg y) \wedge (\neg x \vee \neg y)$

- Encode clause with Latin square – let x see row and y see column

	$y = F$	$y = T$	$y = T$	$y = F$
$x = T$	$x \vee y$	$x \vee \neg y$	$\neg x \vee y$	$\neg x \vee \neg y$
$x = F$	$x \vee \neg y$	$x \vee y$	$\neg x \vee \neg y$	$\neg x \vee y$
$x = T$	$\neg x \vee \neg y$	$\neg x \vee y$	$x \vee \neg y$	$x \vee y$
$x = F$	$\neg x \vee y$	$\neg x \vee \neg y$	$x \vee y$	$x \vee \neg y$

- With the "useless" row/column information, they can now satisfy all clauses

What about DEC-MDP₂?

- Solution:
 - Process chooses tile positions
 - Agents choose tiles
 - Agents repeat back the tile positions, while relationship is calculated
 - Reward of zero obtained only if legal tile choice and positions repeated correctly
- Now the agents can observe the state of the automaton because they have already chosen tiles
- Argument for why this works is subtle

Conclusion

NEXP	DEC-MDP _m , DEC-POMDP _m
PSPACE	POMDP
P	MDP

Some other results

- Infinite-horizon:
 - MDP P-complete
 - POMDP undecidable (Madani et al., 1999)
- Compact representations:
 - MDP PSPACE-complete (Mundhenk, 2000)

Questions

- How similar is this to work in model checking?
- How does this work relate to interactive proof systems?
 - IP = PSPACE
 - 2IP = NEXP
 - Jump in complexity due to additional prover