Markov Decision Processes and Probabilistic Planning

Branislav Bošanský

PAH 2015/2016

Markov Decision Processes

- main formal model
- $\langle S, A, D, T, R \rangle$
  - states – finite set of states of the world
  - actions – finite set of actions the agent can perform
  - horizon – finite/infinite set of time steps $(1,2, \dots)$
  - transition function
    - $T: S \times A \times S \to [0,1]; \sum_{s' \in S} T(s, a, s') = 1$
  - reward function
    - $R: S \times A \times S \to \mathbb{R}$
    - typically bounded

MDPs – policy

- history-dependent policy
  - $\pi: H \times A \rightarrow [0,1]; \sum_{a \in A} \pi(h, a) = 1$
- for simple cases we do not need history and randomization
  - Markovian assumption
  - finite-horizon MDPs
  - infinite-horizon MDPs with reward discount factor $0 \leq \gamma < 1$
  - stochastic shortest path
  - (… and some others)
- from now on, policy is an assignment of an action in each state and time

.

MDPs – policy (2)

- $\pi : S \rightarrow A$
- **stationary policy**
  - when the policy is same every time state s is visited
  - otherwise – **nonstationary policy**
- **positional policy**
  - deterministic and stationary policy

- **Q: for which problems is the stationary policy sufficient?**

MDPs – value of a policy

- we can express an expected reward for every state and time-step when specific policy is followed
- $V_\pi^k(s) = \mathbb{E}\left[\sum_{t=0}^{k} \gamma^t \cdot R(s_t, a_t, s_{t+1}) \,|s_0 = s, a_t = \pi(s_t)\right]$
  - optimal policy : $\pi^{*,k}(s) = \underset{\pi}{\operatorname{argmax}}\, V_\pi^k(s)$

- for large (infinite) k we can approximate the value by dynamic programming
  - $V_\pi^0(s) = 0$
  - $V_\pi^k(s) = \sum_{s' \in S} T(s, a, s') \left[R(s, a, s') + \gamma V_\pi^{k-1}(s')\right] \qquad a = \pi(s)$
  -

MDPs – towards finding optimal policy

- we can exploit the concept of dynamic programming to find an optimal policy
- basic algorithm for solving MDPs based on Bellman's equation
- **value iteration**
  - $V^0(s) = 0 \quad \forall s \in S$
  - $V^k(s) = \max_{a \in A} \underbrace{\sum_{s' \in S} T(s,a,s') \left[R(s,a,s') + \gamma V^{k-1}(s')\right]}_{\text{Q-function } (Q(s,a))}$
  - Q-function $(Q(s,a))$
- for $k \to \infty$ values converges to optimum $V^k \to V^*$

MDPs – value iteration – convergence

- value iteration converges
  - for finite-horizon MDPs: $|D|$ steps
  - for infinite-horizon: asymptotically
    - we can measure residual r and stop if it is small enough ($\mathrm{r} \leq \varepsilon(1-\gamma)/\gamma$)
    - $r = \max_{s \in S} |V_{i+1}(s) - V_i(s)|$
    - convergence depends on γ

-

MDPs – extracting policy and policy iteration

- value iteration calculates only values
- the optimal policy can be extracted by using a greedy approach
  - $\pi^k(s) = \arg\max_{a \in A} \sum_{s' \in S} T^k(s,a,s') \left[R^k(s,a,s') + \gamma V^k(s')\right]$

- alternative algorithm – **policy iteration**
  - starts with an arbitrary policy
    - **policy evaluation:** recalculates value of states given the current policy π^k
    - **policy improvement:** calculates a new maximum expected utility policy π^{k+1}
  - until the strategy changes

MDPs – VI/PI – improvements

- value iteration is very simple
  - updates all states during each iteration
  - curse of dimensionality (huge state space)
  - **asynchronous VI**
    - select a single state to be updated in each iteration separately
    - each state must be updated infinitely often to guarantee convergence
    - lower memory requirements

- **Q: Can we use some heuristics to improve the convergence?**

MDPs – Heuristics

- initial values can be assigned better
  - we can use a heuristic function instead of 0
- **Q: Can you think of any heuristic function?**
  - e.g., remember FFReplan/Robust FF?
  - we can use a single run of a planner on the determinized version
- **Q: What if the values V are initialized incorrectly?**
-

MDPs – Prioritized VI

- initialize V and a priority queue q
- select state s from the top of q and perform a Bellman backup
- add all possible predecessors of s to q
- repeat until convergence
  - priorities: changes in utility, position in the graph, …

- but, values are still updated regardless on the current values
- consider a typical probabilistic planning problem
  - finite-horizon MDP with some goal states

.

MDPs – Real-Time Dynamic Programming

- updates the values only on the path from the starting state to the goal
- during one iteration updates one rollout/trial:
  - start with $s = s_0$
  - evaluate all actions using Bellman's Q-functions $Q(s, a)$
  - select action that maximizes current value: $\arg\max_{a \in A} Q(s, a)$
  - set $V(s) \leftarrow Q(s, a)$
  - get resulting state s'
  - if s' is not goal, then $s \leftarrow s'$ and go to step 2
- can be further improved with labeling (LRTDP) to identify solved states

MDPs – Real-Time Dynamic Programming

Algorithm 4.4: LRTDP

```
LRTDP(s_0, ε)
begin
    V_l ← h
    while s_0 is not labeled solved do
        LRTDP-TRIAL(s_0, ε)
    end
    return π*_{s_0}
end


LRTDP-TRIAL(s_0, ε)
begin
    visited ← empty stack
    s ← s_0
    while s is not labeled solved do
        push s onto visited
        if s ∈ G then
            break
        end
        a_best ← argmin_{a∈A} Q^{V_l}(s, a)
        V_l(s) ← Q^{V_l}(s, a_best)
        s ← execute action a_best in s
    end
    while visited ≠ empty stack do
        s ← pop the top of visited
        if ¬CHECK-SOLVED(s, ε) then
            break
        end
    end
end
```

MDPs – Find and Revise

- we can further combine selective updates with heuristic search
  - starts with admissible $V(s) \geq V^*(s)$ for all states
  - select next state s' that is:
    - reachable from s_0 using current greedy policy π_V, and
    - residual $r(s') > \varepsilon$
  - update s'
  - repeat until such states exist

- many further improvements and algorithms …

Variants of MDPs

TI-Dec-MDP

MMDP

EDI-Dec-MDP

EDI-CR

Dec-MDP

Dec-POMDP

M-POMDP

ND-POMDP

fDec-POMDP

OC-Dec-MDP

TD-POMDP

Dec-SIMDP

DPCL

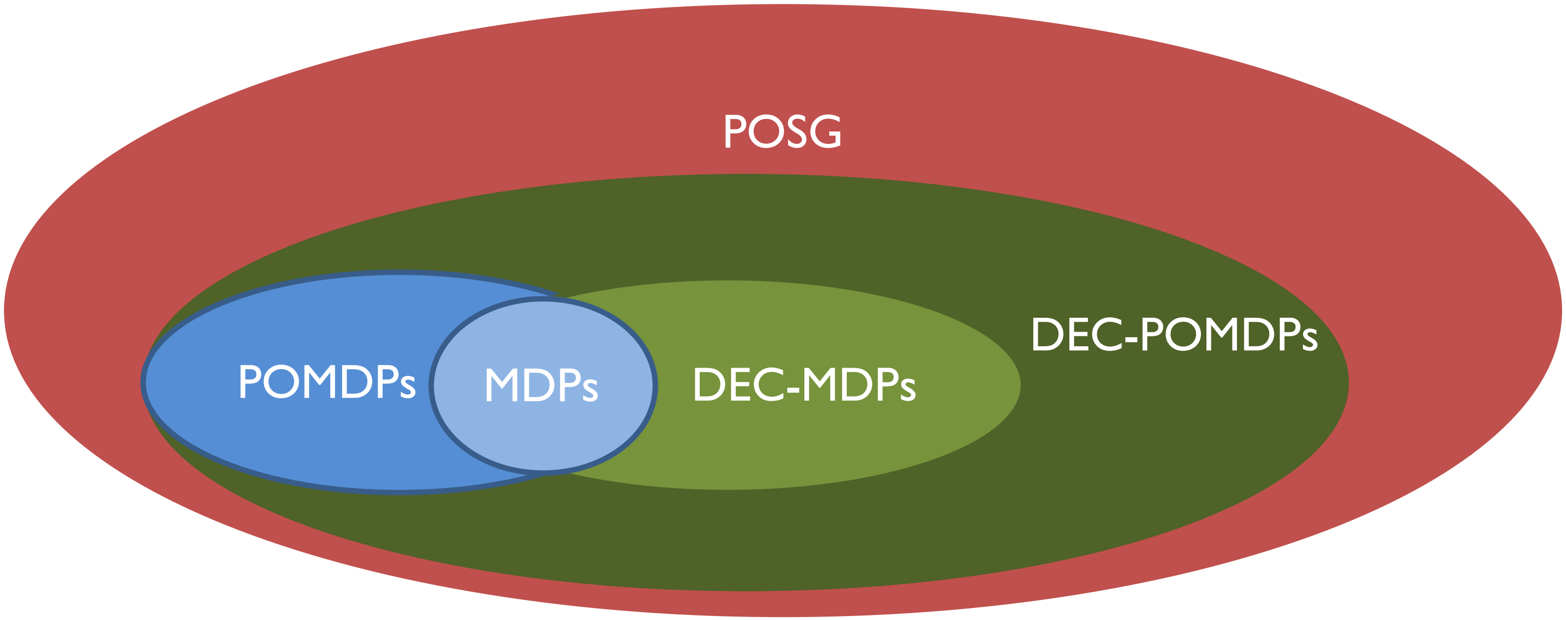
RC-Dec-MDP

IDMG

M-RMP

...

Beyond MDPs



MDP (finite horizon)	P-complete
POMDP (finite horizon)	PSPACE-complete
MDP (infinite horizon)	P-complete
POMDP (infinite horizon)	undecidable
DEC-MDP (finite horizon)	NEXP-complete
…	

Decentralized MDPs

- $\langle I, S, \{A_i\}, T, R \rangle$
  - agents – finite set of agents
  - actions – finite set of joint actions
    - $A = \prod_{i \in I} A_i$
  - other sets as in MDPs
  - common reward function
- if players have different reward function
  - Markov games (simultaneous move games)
- approaches to solve (TI-DEC-MDPs)
  - modification of dynamic programming
  - iterative best response
  -

Sources and Further Reading

- Planning with Markov Decision Processes: An AI Perspective
  - Mausam, Kolobov, 2003
- Policy Iteration for Decentralized Control of Markov Decision Processes
  - Bernstein et al. 2009
- Decentralized Control of Partially Observable Markov Decision Processes
  - Amato et al. 2013