Probabilistic Planning and Markov Decision Processes

Branislav Bošanský

PUI 2017/2018

Classical vs. Probabilistic Planning

- what have you learnt so far?
  - sequential decision making
  - deterministic effects of actions
  - static environment
  - perfect observation
  - perfect sensors

Classical vs. Probabilistic Planning

- the world is not perfect
  - actions take some time to execute
  - actions may fail or yield unexpected results
  - the environment may change due to other agents
  - the agent does not have knowledge about whole situation
  - other agents can have conflicting objectives
  - sensors are not precise

  - towards more realistic setting
  - planning with uncertainty

Classical vs. Probabilistic Planning

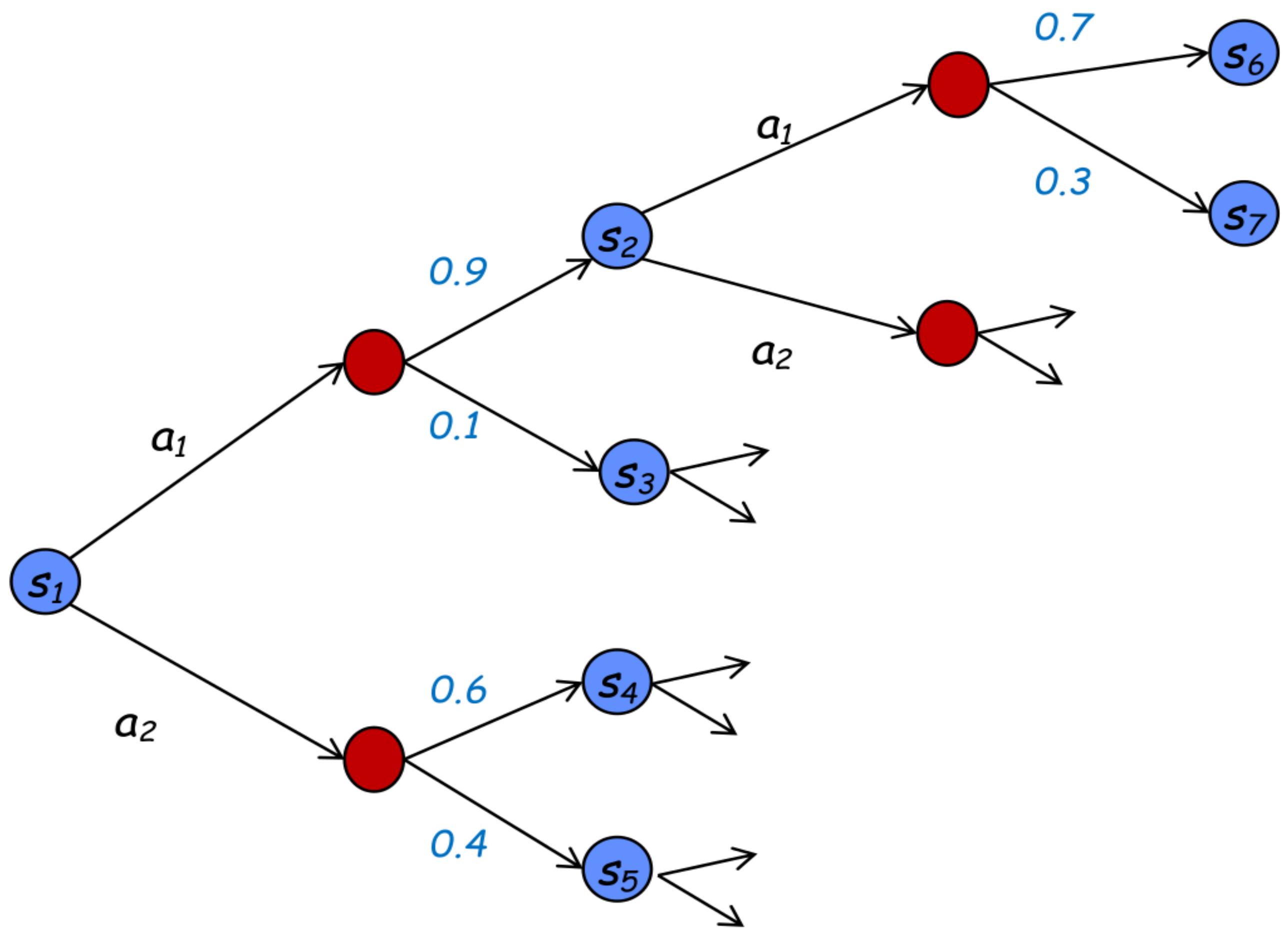
- Classical Planning: $\langle S, s_0, S_G, A, f, c \rangle$
  - states, initial state, goal state(s)
  - actions
  - transition function $f: S \times A \rightarrow S$
  - cost function

- Probabilistic Planning
  - probabilistic transition function $T: S \times A \times S \rightarrow [0,1]$

$$\sum_{s' \in S} T(s, a, s') = 1$$

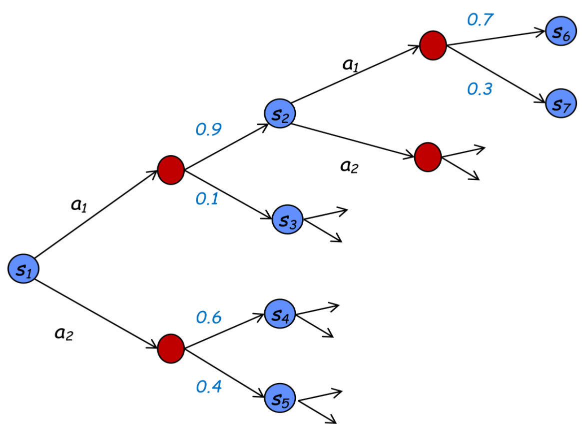
Q: why is this enough for modelling uncertainty in environment?

Probabilistic Planning - Visualization



Probabilistic Planning - Solution

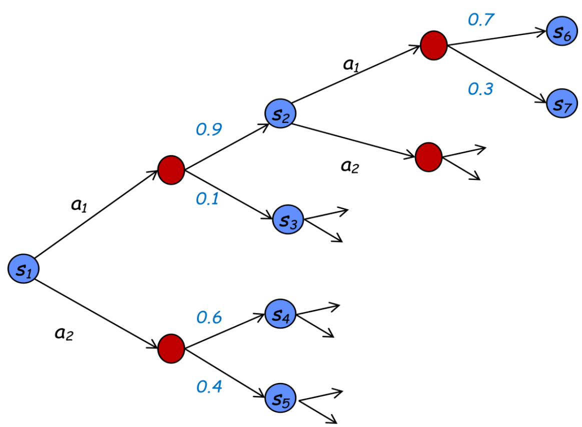
- what is the solution in classical planning?
  - sequence of (partially) ordered actions leading from initial state to the goal state



- this is not sufficient in the probabilistic case
  - what if the plan fails?
- we need **a contingency plan (policy)**
  - typically assumes *k* failures
  - if the number of failures is unbounded → policy

Probabilistic Planning - Solution

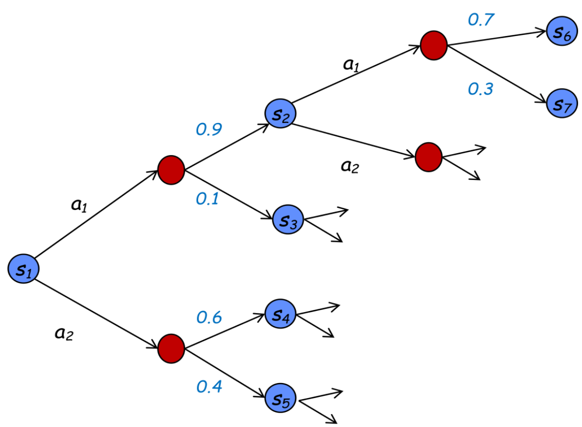
- in general we seek for a probabilistic history-dependent policy
  - $\pi: H \times A \rightarrow [0,1]$
  - where $h = s_1 a_1 s_2 a_2 \dots s_t$
  - note that the policy may prescribe randomization over actions



- now we have a representation for plans (policy)
  - we need a method for plan evaluation

Probabilistic Planning - Evaluation

- costs are assigned to triplets (s, a, s')
- typically termed rewards (i.e., positive sense)
- executing a policy yields a sequence of rewards
- policy value – linear additive utility
  - $u(R_1, R_2, \dots) = R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots$
  - $u\big(\pi(s_0)\big) = E[u(R_1, \dots)]$
- expected utility – what can happen?
  - optimal only for risk-neutral agent



Probabilistic Planning – Optimal Solution

- If the quality of every policy can be measured by its expected linear additive utility, there is a policy that is optimal at every time step.

(Stated in various forms by Bellman, Denardo, and others)

- we seek for π^* s.t. $u(\pi^*) \geq u(\pi)$ for all other policies π

- Q: Can there be a case where the policy cannot be measured by expected linear additive utility?
  - yes (infinite state-space with non-discounted rewards, dead-ends, ...)

Markov Decision Processes

- Main formal model
- $\langle S, A, D, T, R \rangle$
  - states – a finite set of states of the world
  - actions – a finite set of actions the agent can perform
  - horizon – a 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

MDP – 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
  - Markov 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

MDP – 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

Probabilistic Planning – Algorithms

- this lecture
  - using classical planning to probabilistic planning
  - straightforward approach (FF-replan)
  - improved approach (Robust FF)
  - algorithms that directly use probability and uncertainty
    - formal definition MDP, strategy/policy iteration

- next lectures
  - MCTS, current approaches for solving MDPs
  - uncertainty in observations
    - formal definition and current approaches for solving POMDPs

Probabilistic Planning – First Approach

- 2004 – first international probabilistic planning competition
- several participants, mainly based on MDP solvers
- winner?
  - FF-Replan
  - possibly the simplest algorithm you can think of …

FF-Replan

- outline of the algorithm

1. determinize the input domain (remove all probabilistic information from the problem)
2. synthesize a plan
3. execute the plan
4. should an unexpected state occur, replan

FF-Replan - Determinization

- what information can be discarded?
- two main heuristics
  - keep only one from all probabilistic outcomes of an action in a state (e.g., using the outcome with the highest probability)
  - keep all outcomes
    - generate a separate action for each possible outcome

- very simple, not sound, not optimal, but still good enough for simple domains
  - (outperformed also all participants in IPPC-06)
  - Q: In which cases should you adopt such techniques?

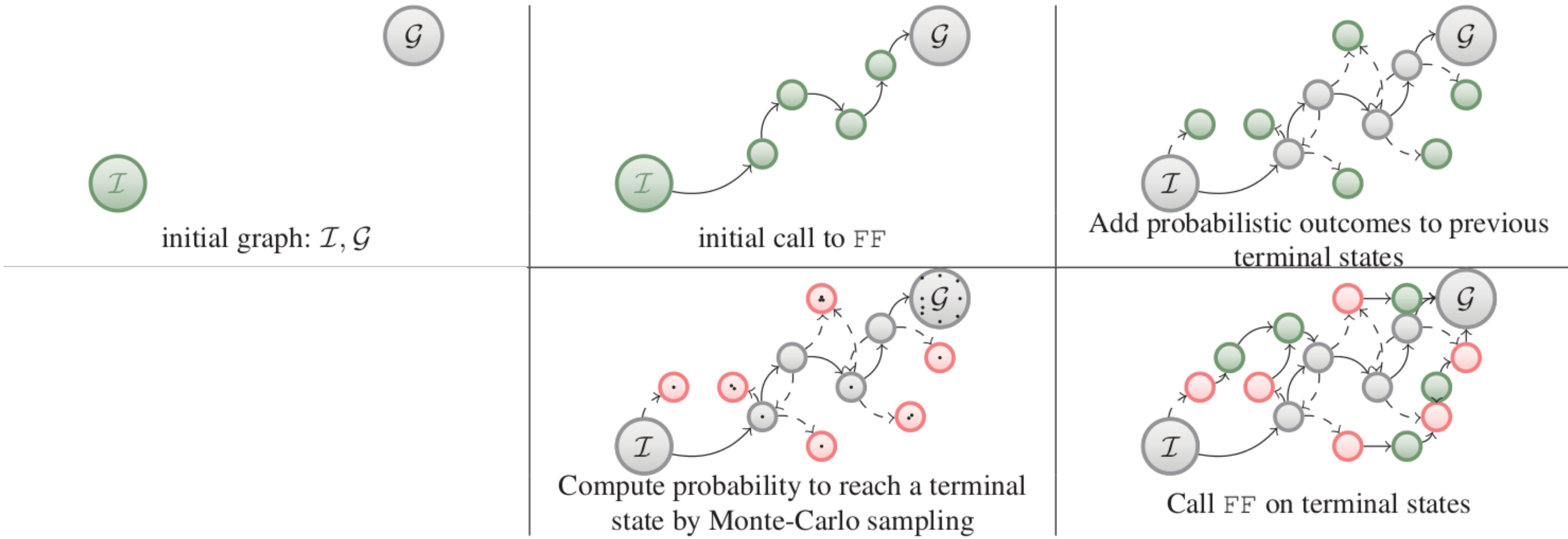
Probabilistic Planning (2)

- winner of IPPC 2008
  - Robust-FF
    - (Incremental Plan Aggregation for Generating Policies in MDPs, Konigsbuch, Kuter, Infantes 2010)
  - generalizes FF-Replan

1. determinize the problem
2. use classical planner to find partial plans
3. aggregate these plans into the partial policy
4. continue until the probability of replanning is below given threshold

Robust-FF

- outline of the algorithm



Robust-FF

- pseudocode of the algorithm

Algorithm 1: $\texttt{RFF}(M, s_0, G, \rho, N)$

$\mathcal{D} \leftarrow$ a deterministic relaxation of M
$T \leftarrow \{s_0\}$; $\pi \leftarrow \emptyset$; $\omega(s_0, \pi, s_0) \leftarrow 1$
repeat
    $T' \leftarrow \emptyset$ `// new terminal states`
    $X \leftarrow \emptyset$ `// new expanded states`
    for $s \in T$ *such that* $\omega(s_0, \pi, s) > \rho$ **do**
        pick $G_{\texttt{FF}} \subseteq G \cup S_\pi$
        $p \leftarrow \texttt{FF}(\mathcal{D}, s, G_{\texttt{FF}})$
        if $p \neq failure$ **then**
            $s' \leftarrow s$; let $p = \langle \hat{a}_1, \ldots, \hat{a}_k \rangle$
            for $1 \leqslant i \leqslant k$ **do**
                $X \leftarrow X \cup \{s'\}$
                $\pi(s') \leftarrow a_i$
                $T' \leftarrow T' \cup succ(s', a_i) \setminus (S_\pi \cup G)$
                $s' \leftarrow succ_{\mathcal{D}}(s', \hat{a}_i)$
        else $X \leftarrow X \cup \{s\}$
    $T \leftarrow (T \setminus X) \cup T'$
    $\{\omega(s_0, \pi, s) \mid s \in T)\} \leftarrow \texttt{Fail\_Prob}(s_0, \pi, T, N)$
    $\Omega(s_0, \pi) = \sum_{s \in T} \omega(s_0, \pi, s)$
    `// Next line is optional`
    Optimize the shortest stochastic path in S_π by considering all states in T as if they were unsolvable
until $\Omega(s_0, \pi) \leqslant \rho$ **or** $T = \emptyset$
if $\pi \neq \emptyset$ **then return** π
else return $failure$

Robust-FF

- number of options
  - selecting determinization (most probable, all outcomes)
  - selecting goals (only problem goals, random goals, best goals)
    - random/best goals – include also expanded states into G_{FF}; either k random, or k "best ones"
  - calculating probability of reaching terminal states (dynamic programming, Monte Carlo simulations)

- soundness vs. completeness of the algorithm?
  - only with selected methods (RFF_{AO})

- not (approximately) optimal in general

FF-Hindsight

- Approximate the value of a state
  - sample a set of determinized problems originating from a state
  - then solve these problems and combine their values

- Optimal value function
  - $$V^*(s,T) = \max_{\pi} \boldsymbol{E}[R(s,F,\pi)]$$
  - from state s, horizon T, policy π, random variable F, reward function R

- HOP value approximation
  - $$V^*(s,T) = \boldsymbol{E}[\max_{\pi} R(s,F,\pi)]$$

MDP – 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
  - $V_\pi^0(s) = 0$
  - $V_\pi^k(s) = \sum_{s' \in S} T(s, a, s') \left[R(s, a, s') + \gamma V_\pi^{k-1}(s')\right] \qquad a = \pi(s)$

MDP – 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**
  - $V^0(s) = 0 \quad \forall s \in S$
  - $V^k(s) = \max_{a \in A} \underbrace{\sum_{s' \in S} T(s, a, s') \left[R(s, a, s') + \gamma V^{k-1}(s')\right]}_{\text{Q-function } (Q(s,a))}$
- for $k \to \infty$ values converge to optimum $V^k \to V^*$

MDP – convergence of value iteration

- 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

MDP – 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

MDP – 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?**

MDP – VI/PI 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?**

MDP – VI/PI with priority

- 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 – 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 …

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