Markov Decision Processes and Probabilistic Planning

PAH 2013/2014

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 $(1,2,\ldots)$
  - transition function
    - $T: S \times A \times S \times D \rightarrow [0,1]$
  - reward function
    - $R: S \times A \times S \times D \rightarrow \mathbb{R}$

MDPs – policy

- history-dependent policy
  - $\pi: H \times A \times D \to [0,1]$
- 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)

- Markov policy
  - $\pi: S \times D \to A$
- when the policy is same in every time-step – **stationary policy**
  - $\pi(s,t) = \pi(s,t')\ \forall t,t' \in D; t \neq t'$
- otherwise – **nonstationary 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
- $V_\pi^k(s) = \mathbb{E}\left[\sum_{t=0}^{k} \gamma^t \cdot R^t(s_t, a_t, s_{t+1}) \,|s_0 = s, a_t = \pi(s_t)\right]$

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

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

MDPs – extracting policy

- 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
  - updates using the same equations

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 ($\leq \varepsilon$)
    - $r = \max_{s \in S} |V_{i+1}(s) - V_i(s)|$
    - convergence depends on $\gamma, \ldots$

MDPs – value iteration – 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 admissible heuristic function?**
  - e.g., remember FFReplan/Robust FF?
  - we can use a single run of a planner on the determinized version
- 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

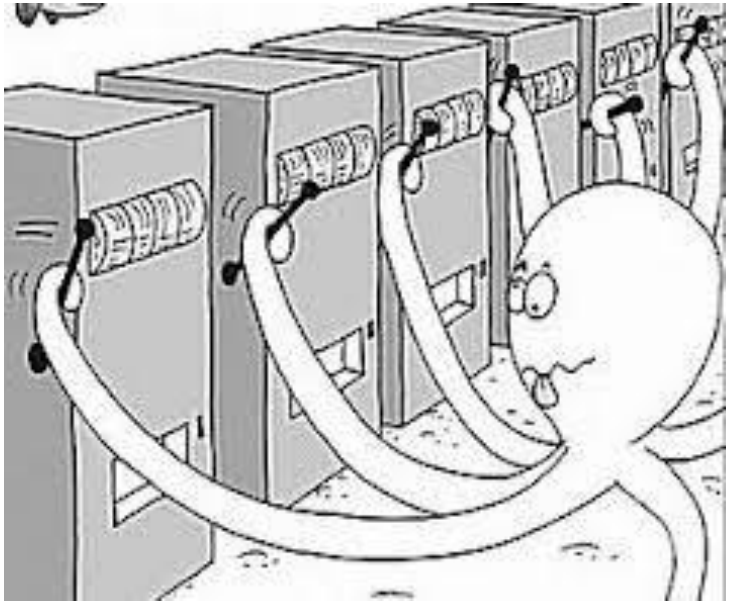
MDPs – Find and Revise

- 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 …

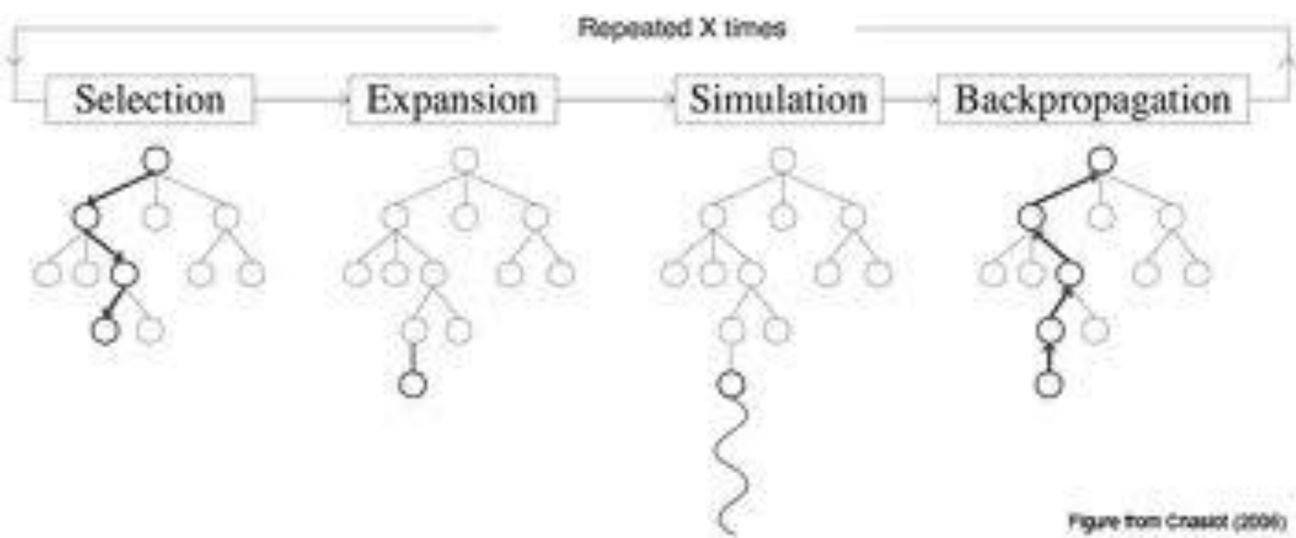
MDPs – Using Monte-Carlo Methods

- Monte-Carlo sampling is a well known method for searching through large state space
- exploiting MC in sequential decision making has first been successfully designed in 2006 (Kocsis, Szepesvari)
- foundations in mathematical theory
  - multi-armed bandit problem
  - exploration/exploitation
  - Upper Confidence Bounds (UCB)



MDPs – Monte-Carlo Tree Search – UCT

- using bandits in sequential decision making: MCTS



- UCB – selection function (UCB applied on trees – UCT)

MDPs – Monte-Carlo Tree Search – UCT

- UCB – selection function (UCB applied on trees – UCT)
  - for each action a_i applicable in s UCB selects the one that maximizes

$$c\sqrt{\frac{\log n}{n_i}} + \sum_{s' \in S} T(s, a_i, s')[R(s, a_i, s') + \gamma V(s')]$$

  - n – times the state is visited
  - $V(s)$ – average reward from the previous iterations
  - c - exploration constant (linear to expected utility)
- exploration factor ensures to evaluate actions that are evaluated rarely

MDPs – UCT in probabilistic planning

- winner of IPPC 2011 – PROST
- uses a number of improvements
- vanilla UCT is not that fast
- MCTS/UCT requires large number of iterations to converge
- large state-space does not allow this
  - depth-limited rollouts
- reducing branching factor
  - some actions are dominated, we can remove them

MDPs – UCT (2)

- UCT can also benefit from heuristics
  - values after expansion can be set better
    - PROST uses Q-value initialization on most-probable determinization
  - also random rollouts can be driven with some heuristic
- different update mechanism
  - Rapid Action Value Estimation (RAVE)
- many, many others …

MDPs – Beyond UCT

- UCT is far from optimal algorithm
  - there exist simple examples where vanilla UCT performs extremely bad
- number of reasons
  - learning the best action is different from learning the best (contingency) plan
  - situation that occur in states does not exactly correspond to multi-armed bandit (mathematically)
- there are modifications that improve these drawbacks
  - BRUE (Feldman & Domshlak, 2013)