Sutton, R.S. and Barto, A.G. (1998). Reinforcement Learning: An Introduction, MIT Press, Chapter 3:

The Reinforcement Learning Problem

Jaakko Väyrynen
jaakko.j.vayrynen@tkk.fi

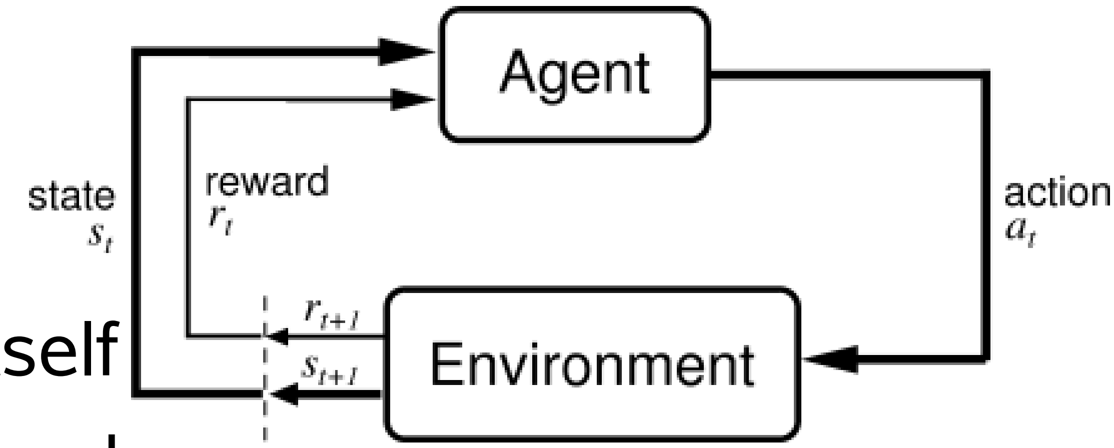
T-61.6020 Reinforcement Learning – Theory and Applications
January 31, 2006

Contents

Contents

- **Agent-environment interface framework**
- Goals, rewards, returns
- Markov property
- Markov Decision Process (MDP)
- Value functions
- Optimal value functions
- Summary

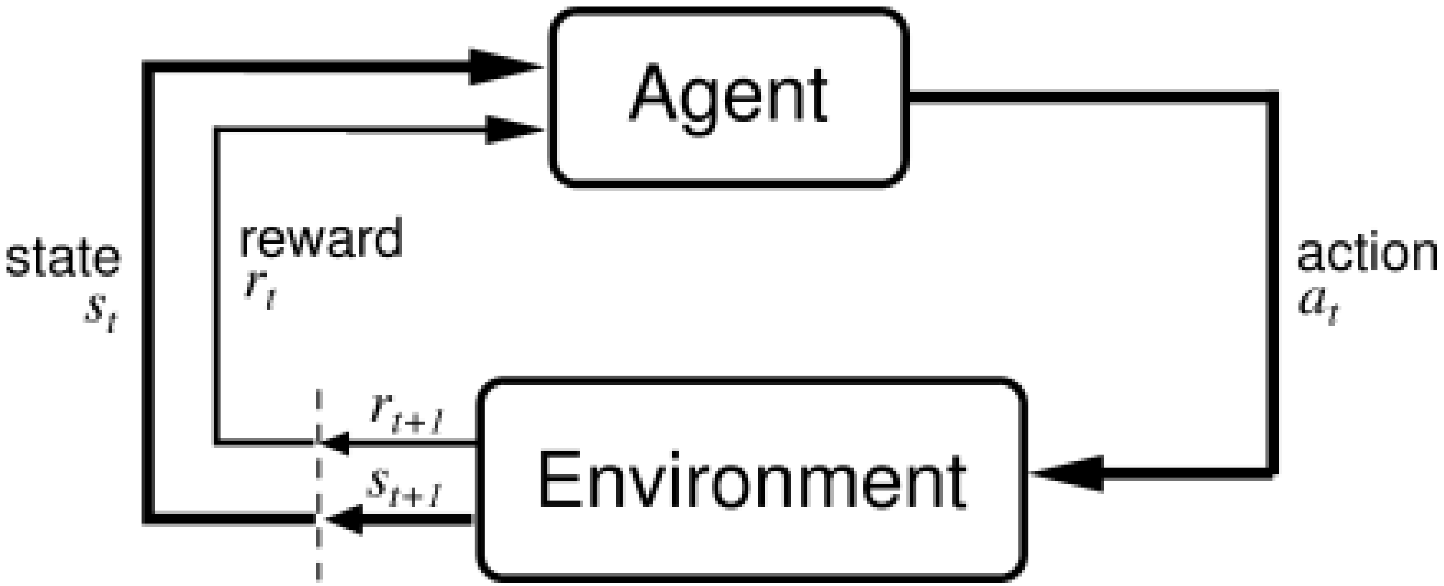
Agent-environment framework



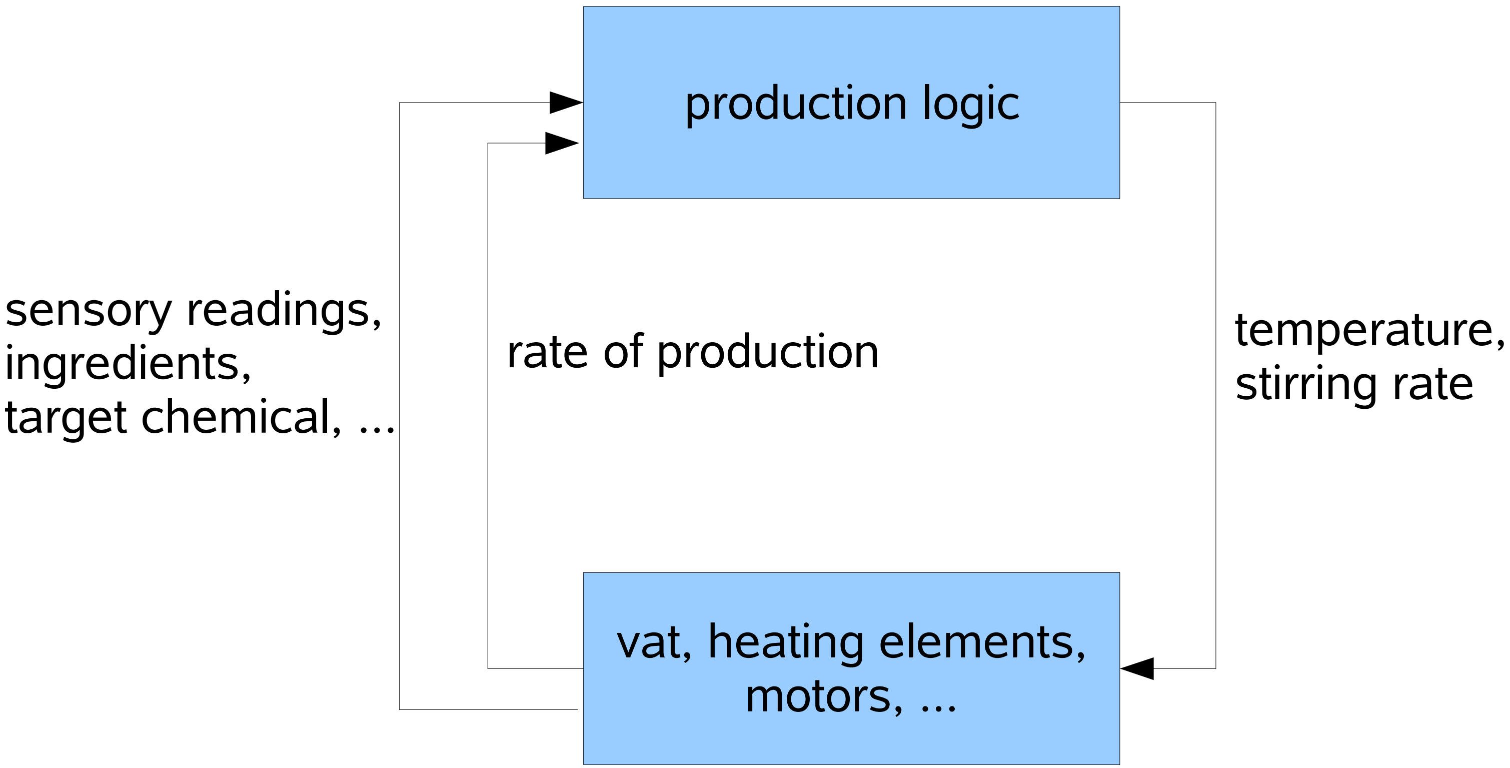
- Abstract framework
- Agent
  - complete control of itself
  - tries to maximize rewards
- Environment
  - defines the task with rewards
- Interaction: three signals
  - state, action, reward

Formal definition

- Discrete time steps
  - $t=0,1,2,...$
- State
  - $s_t \in S$
- Action
  - $a_t \in A(s_t)$
  - policy $\pi_t(s,a) = Pr\{a_t = a | s_t = s\}$
- Reward
  - $r_{t+1} \in \Re$

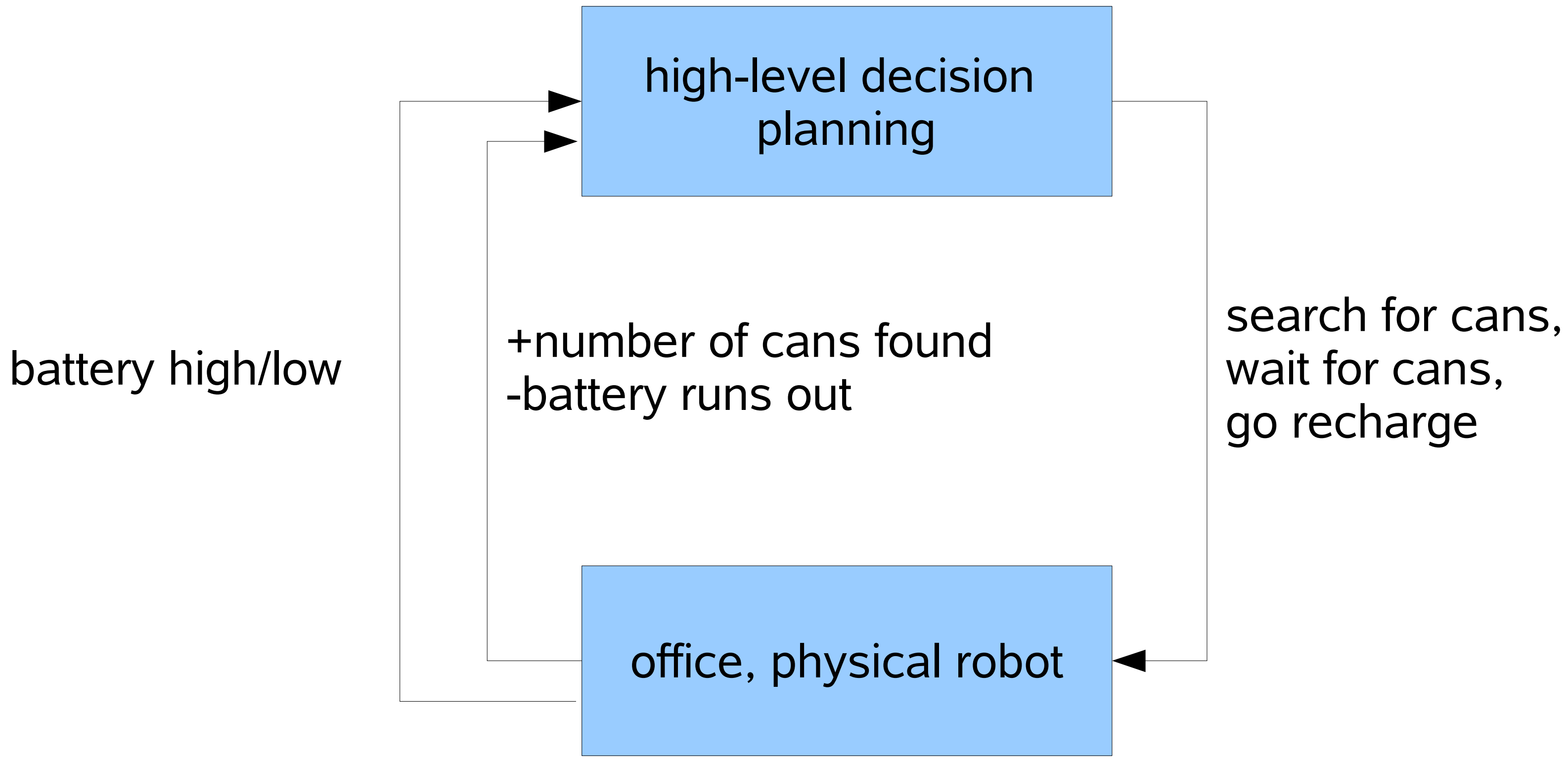


Example: Bioreactor



Task: produce target chemical

Example: Recycling robot



Task: choose best action on how to collect empty cans in an office environment

Contents

- Agent-environment interface framework
- **Goals, rewards, returns**
- Markov property
- Markov Decision Process (MDP)
- Value functions
- Optimal value functions
- Summary

Goal and reward

- Reward is a single real valued number
- Should determine the *goal of the task* (“what”)
- Should **not** assign sub-goals or prior knowledge of the problem (“how”)
- Informal goal of the agent: maximize reward in the long run
- Return R_t: function of future rewards
  - episode tasks
  - continuous tasks

Episode tasks and return

- Final time index T exists
  - games like checkers
  - escape from a maze
- Return R_t
- All states S^+
  - distribution of reset states
  - non-terminal states S
  - terminal states

$$R_t = \sum_{k=0}^{T-1} r_{t+k+1}$$

Continuous tasks and return

- There is no final time index
  - continual process-control task
  - robot with a long life span
- Return R_t

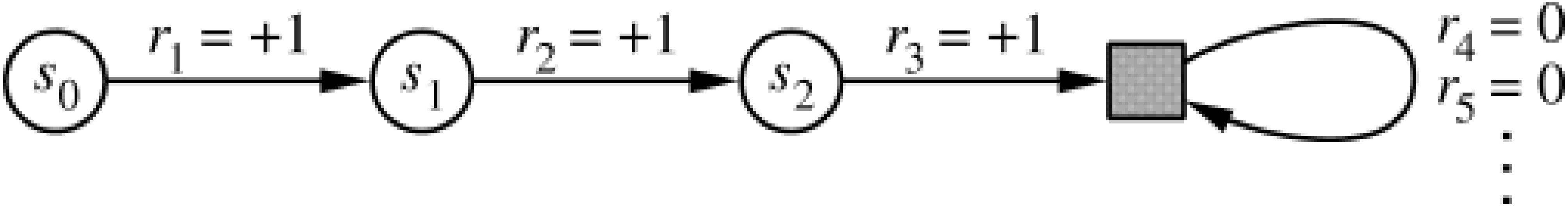
$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

- Discount rate $0 \leq \gamma \leq 1$

Unified notation

- Absorbing state
  - marks the end of an episode
  - transitions only to itself
  - zero reward
- Episodes start from time index 0
- Episode indexes are left out

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1}$$



Contents

- Agent-environment interface framework
- Goals, rewards, returns
- **Markov property**
- Markov Decision Process (MDP)
- Value functions
- Optimal value functions
- Summary

Markov property

- State signal represents everything the agent knows about its environment
- Markov state
  - preserve relevant information
  - only current state affects the next state
  - $Pr\{s_{t+1}=s', r_{t+1}=r | s_t, a_t\}$
- State in RL approximates Markov state

Contents

- Agent-environment interface framework
- Goals, rewards, returns
- Markov property
- **Markov Decision Process (MDP)**
- Value functions
- Optimal value functions
- Summary

Markov Decision Process (MDP)

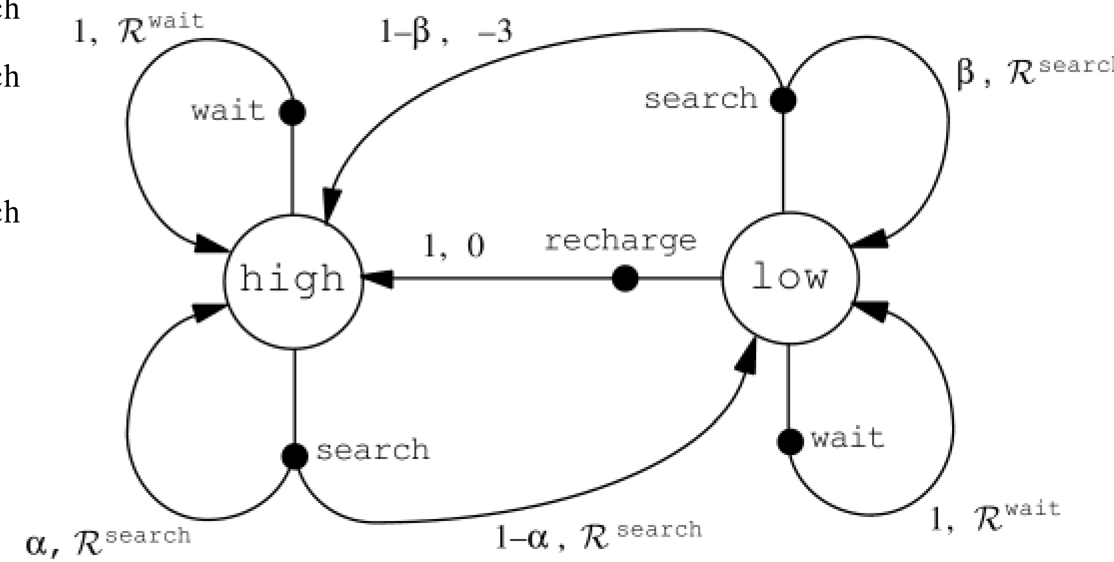
- Reinforcement learning task with the Markov property
- Finite MDP
  - state and action spaces are finite
- Transition probabilities
  - $P_{ss'}^{a} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$
- Expected reward R
  - $R_{ss'}^{a} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\}$

Example: Recycling robot

- Battery state: S={high, low}
- Actions A={search, wait, recharge}
  - A(high)={search, wait}
  - A(low)={search,wait,recharge}
- Transition probabilities for states P
- Expected rewards R

Example: Recycling robot

$s=s_t$	$s'=s_{t+1}$	$a=a_t$	P	R
high	high	search	α	R^{search}
high	low	search	1-α	R^{search}
low	high	search	1-β	-3
low	low	search	β	R^{search}
high	high	wait	1	R^{wait}
high	low	wait	0	R^{wait}
low	high	wait	0	R^{wait}
low	low	wait	1	R^{wait}
low	high	recharge	1	0
low	low	recharge	0	0



transition probabilities and expected rewards

transition graph

Contents

- Agent-environment interface framework
- Goals, rewards, returns
- Markov property
- Markov Decision Process (MDP)
- **Value functions**
- Optimal value functions
- Summary

Value functions

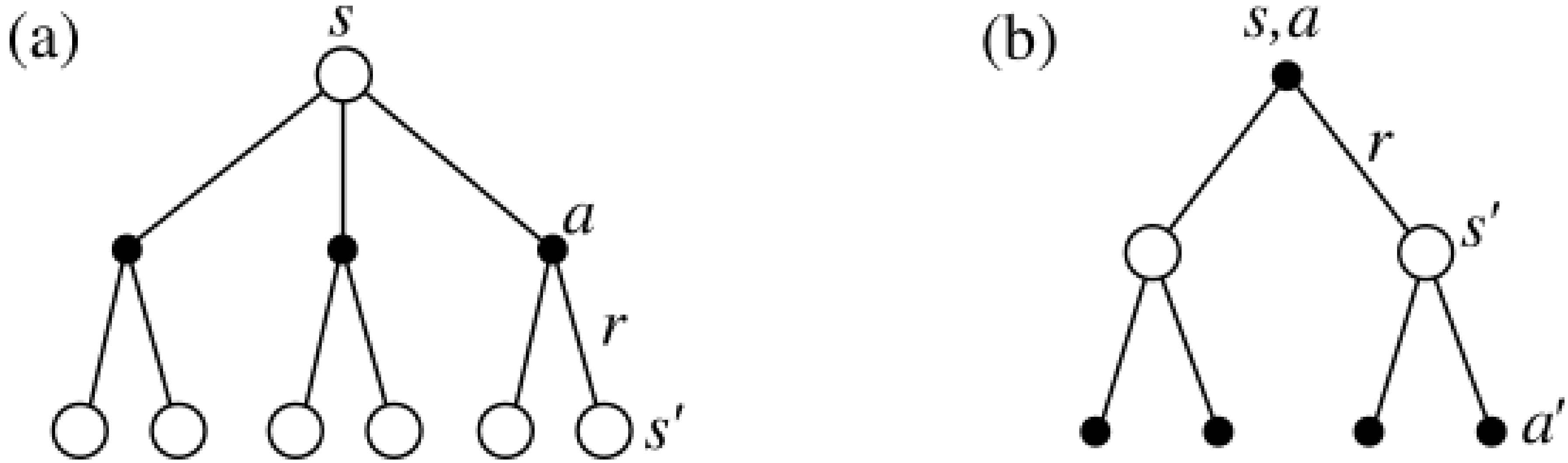
- Goodness of being in state (and taking action)
- Goodness is measured with expected return
- Evaluated with respect to a policy π
- State-value function for policy π
  - $V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\}$
- Action-value function for policy π
  - $Q^{\pi}(s, a) = E_{\pi}\{R_t | s_t = s, a_t = a\}$
    $= E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}$

Estimation of value functions

- Estimation from observations
  - Monte Carlo (MC) methods
    - average of actual returns for each state -> V
    - average of actual returns for actions in state -> Q
  - parameterized function approximations
    - compact representation

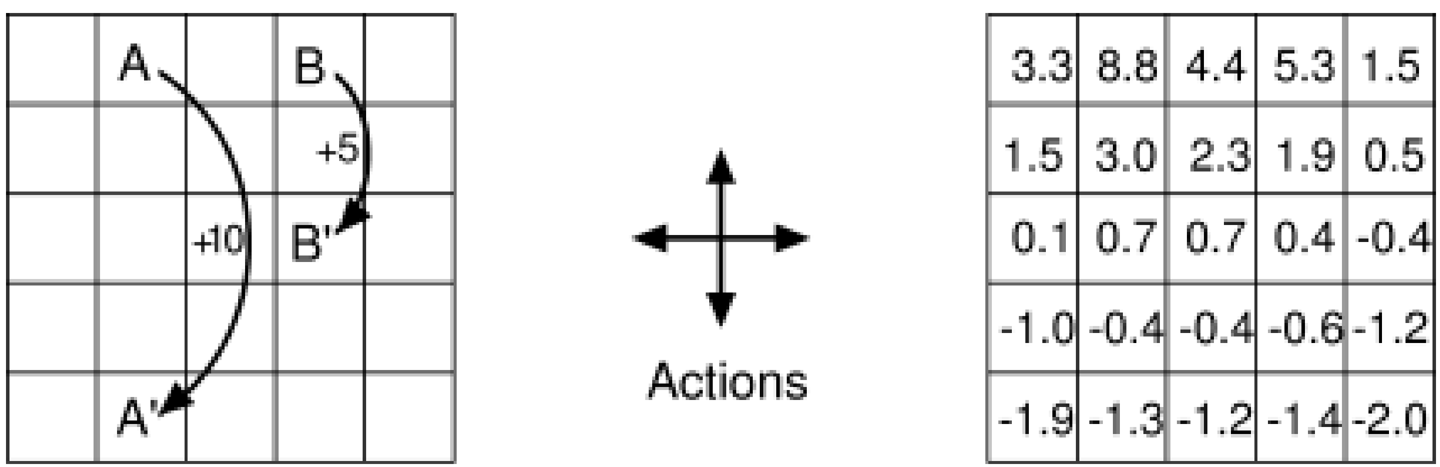
Bellman equation for value function

- Value functions hold the consistency condition
  - $V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P^{a}_{ss'} [R^{a}_{ss'} + \gamma V^{\pi}(s')]$
- Basis for computing, approximating and learning V^{π}
- Backup diagrams for (a) V^{π} and (b) Q^{π}



Example: Gridworld

- State: current cell
- Actions: move one cell {north, east, south, west}
- Rewards
  - -1 for trying to leave the grid, cell does not change
  - +10 for leaving cell A, relocation to cell A'
  - +5 for leaving cell B, relocation to cell B'



(a) (b)

Contents

- Agent-environment interface framework
- Goals, rewards, returns
- Markov property
- Markov Decision Process (MDP)
- Value functions
- **Optimal value functions**
- Summary

Value functions for finite MDPs

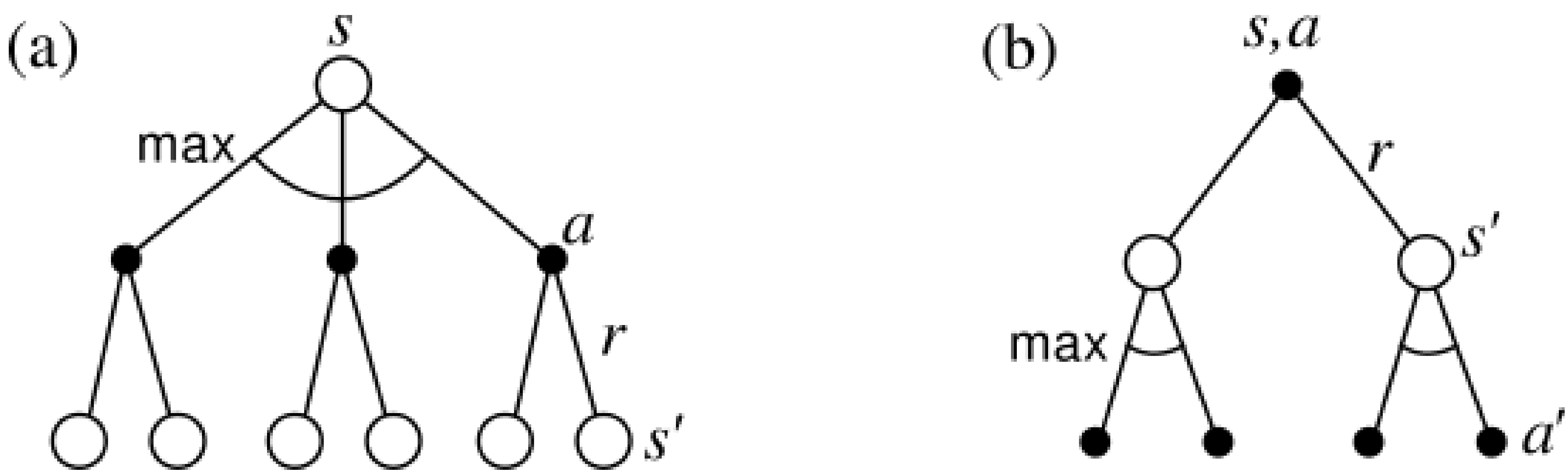
- Partial order over policies
  - there is at least one optimal policy π^*
- Optimal state-value function
  - $V^*(s) = max_\pi V^\pi(s) \quad \forall s \in S$
- Optimal action-value function
  - $Q^*(s, a) = max_\pi Q^\pi(s, a) \quad \forall s \in S, a \in A$

    $= E\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\}$

Bellman optimality equations

- With π* the value of state equals expected return with the optimal action
  - $$V^*(s) = \max_{a \in A(s)} Q^{\pi^*}(s,a)$$
    $$= \max_a E\{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\}$$
    $$= \max_{a \in A(s)} \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V^*(s')]$$
  - $$Q^*(s,a) = E\{r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') | s_t = s, a_t = a'\}$$
    $$= \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma \max_{a'} Q^*(s', a')]$$

Backup diagrams for optimal policy

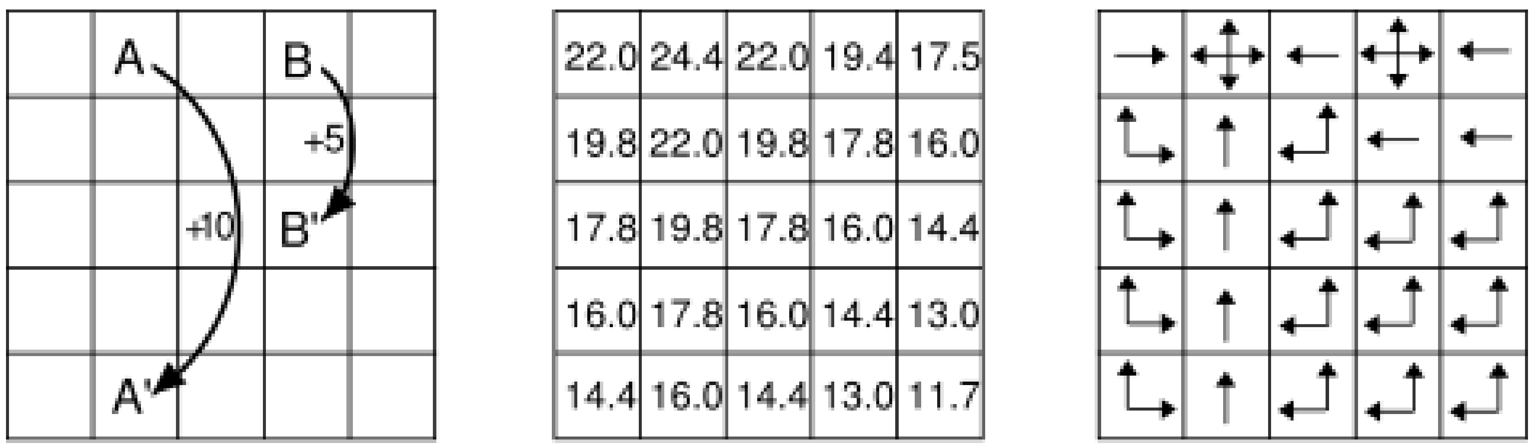


Solving Bellman optimality equation

- Unique solution exists with finite MDPs
- Optimal value function brings global information to local states
  - optimal policy can be calculated in a greedy manner
- System of equations
  - N states, N unknowns in N equations
  - known dynamics $R_{ss'}^{a}, P_{ss'}^{a}$
  - solution from the nonlinear equations
- Explicit solutions may not be practical

Example: Gridworld

- Optimal solution to gridworld example
  - unique optimal state-value function
  - multiple optimal policies



22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

a) gridworld b) V^* c) π^*

Approximations for solving MDPs

- Heuristics
- Dynamic programming
- Observed transitions in place of expected
- Parameterized function approximations
- Concentrate on making good decisions with frequent states

Summary

- Reinforcement learning problem
  - mathematical foundation
- Interaction of the agent and the environment
  - the interface defines a particular task
- Value functions, Bellman equations
  - optimal value functions, optimal policies
- Trade-off between tractability and applicability
  - approximation of the optimal value function