Partially observable Markov decision processes

Jiří Kléma

Department of Computer Science,
Czech Technical University in Prague

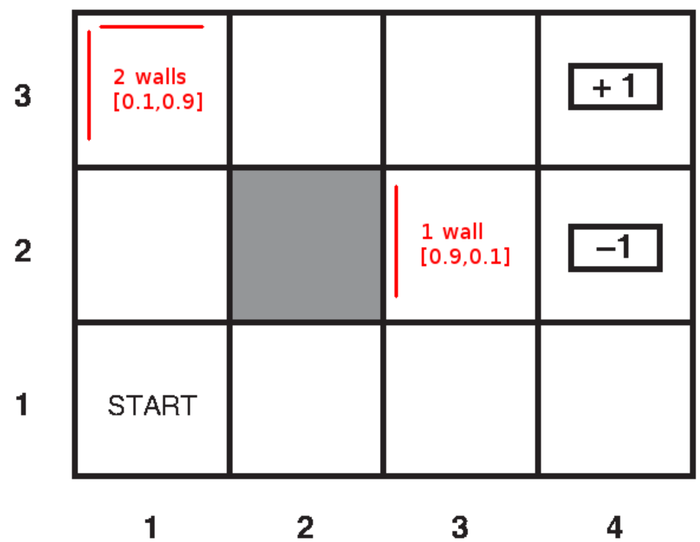
https://cw.fel.cvut.cz/wiki/courses/b4b36zui/prednasky

Agenda

- Previous lecture: Markov decision processes (MDPs)
  - stochastic process with a limited memory,
  - world/environment well defined by its transition and reward functions,
  - goal to find the optimal policy,
  - dynamic programming most frequently used,
- partially observable Markov decision processes (POMDPs)
  - the world is partially observable only, states are not available,
  - define a new stochastic process that generalizes MDP,
  - policy changes, complexity grows, theoretical and real solutions.
- reinforcement learning,
- generalization towards large/infinite state spaces.

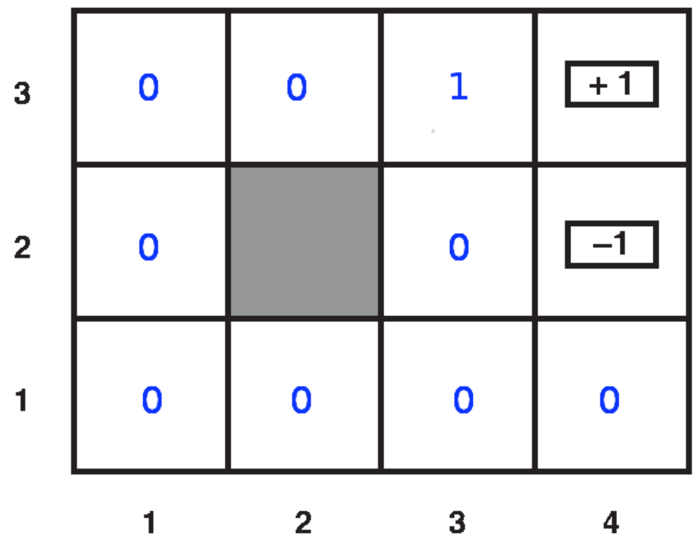
Changes in our running example – a robot in a grid world

- The grid, task and stochastic actions remain unchanged,
- the major change
  - the state in which the robor currently is remains hidden,
  - the robot only obtains a stochastic wall signal
    - ∗ the number of surrounding walls in the current state,
    - ∗ the wall sensor is wrong in 10% situations.

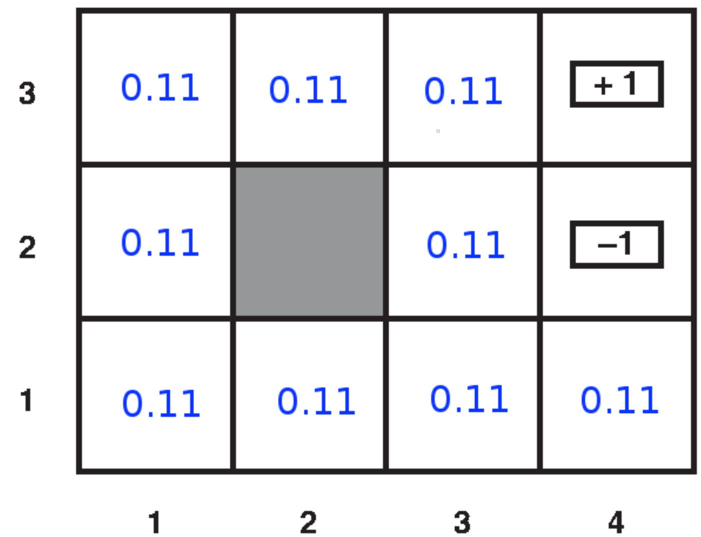


Changes in our running example – a robot in a grid world

- The major consequences of the change
  - policy cannot map between states and actions,
  - the state information maintained in the form of a probability distribution
    - the robot only has a state belief (vector),
  - policy is a mapping between state beliefs and actions.



Belief 1: a clear MDP like case, go east.



Belief 2: no state info, action selection difficult.

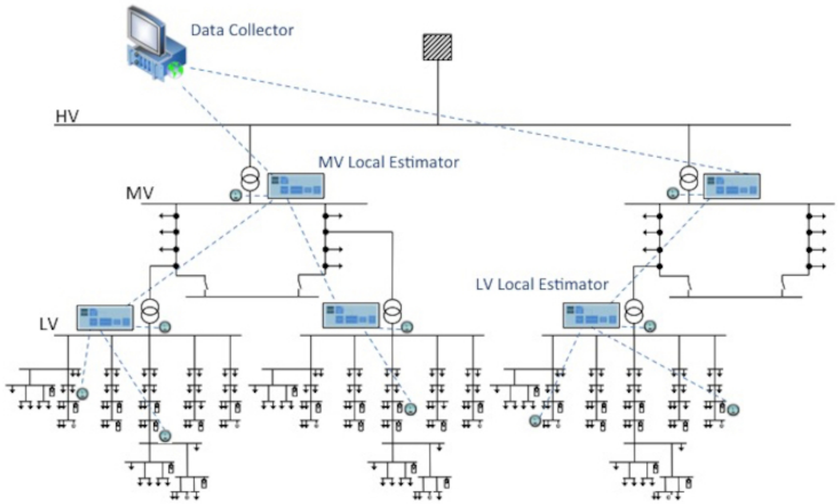
Partial observability

- MDPs work with the assumption of complete observability
  - assumption that the actual state s is always known is often non realistic,
  - examples
    - physical processes such as a nuclear reactor, complex machines,
    - we do know the physical laws that underlie the process,
    - we know the structure and characteristics of the machine and its parts,
    - however, do not know the initial state and subsequent states, can only measure temperature,
    - or have signals from various (unreliable) sensors.

POMDP – motivation

- industrial: machine maintenance

- business: power distribution systems



Partial observability

- partially observable Markov decision process (POMDP)
  - MDP generalization, states guessed from observations coupled with them,
  - POMDP $= \{S, A, P, R, O, \Omega\}$,
    - O is a set of observations,
    - Ω is a sensoric model that defines conditional observation probs

$$\Omega^a_{s'o} = Pr\{o_{t+1} = o \mid s_{t+1} = s', a_t = a\}$$

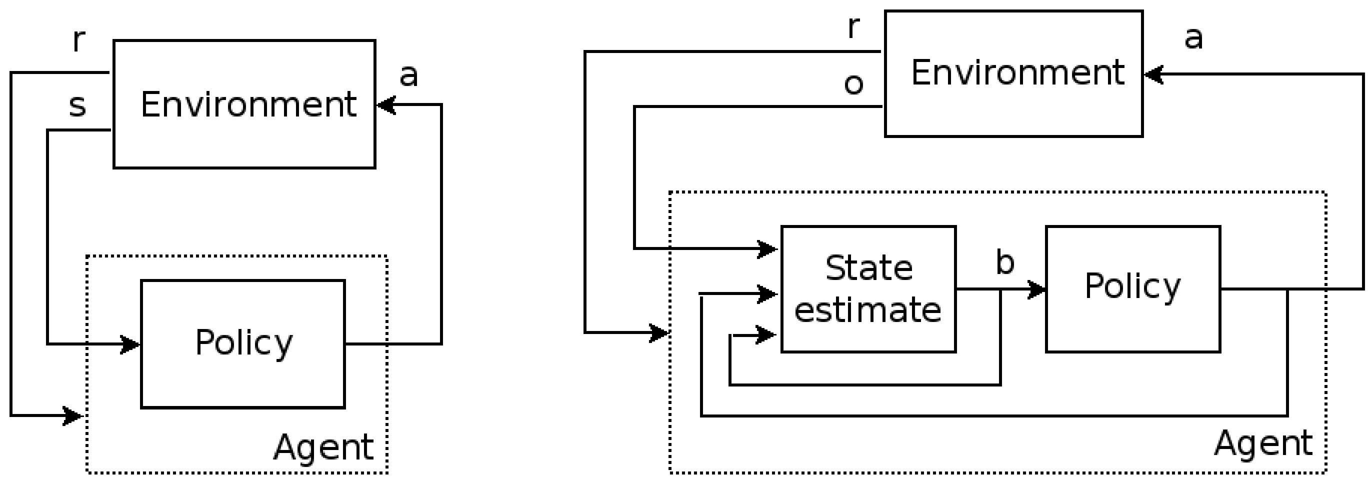
  - instead of s agent internally keeps prob distribution b (**belief**) across states
    - we perform a in unknown s (knowing $b(s)$ only) and observe o,
    - then we update our belief

$$b'(s') = \eta \, \Omega^a_{s'o} \sum_{s \in S} P^a_{ss'} b(s)$$

    - η is a normalization constant such that $\sum_{s' \in S} b'(s') = 1$.

Partial observability

- consequences of partial observability
  - it makes no sense to concern policy $\pi : S \to A$, shift to $\pi : B \to A$,
  - commonly computationally intractable, approximate solutions only
    - for n states, b is an n-dimensional real vector,
    - PSPACE-hard, worse than NP.

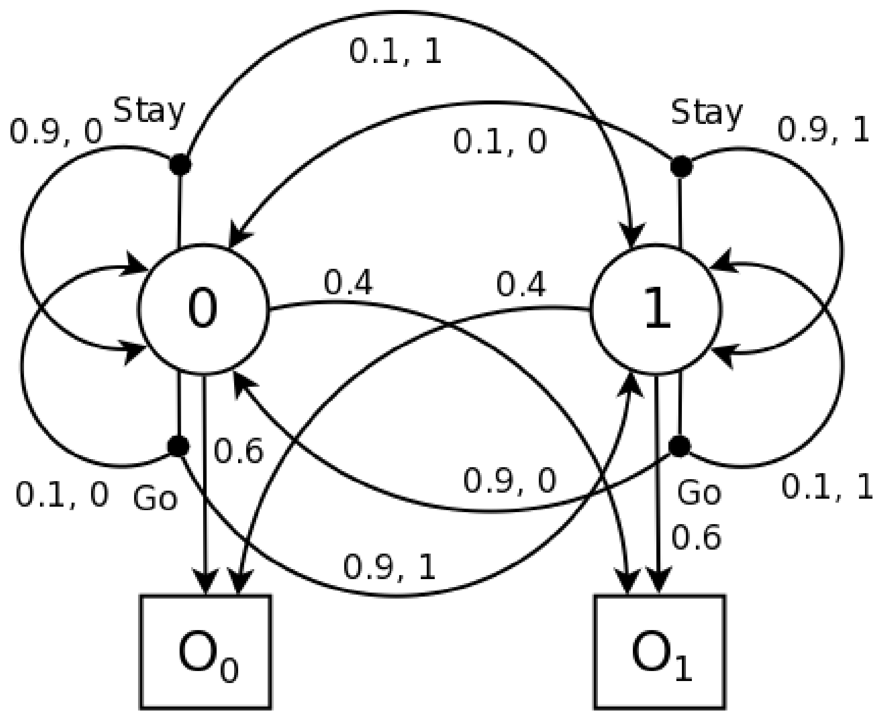


a) MDP b) POMDP

Partial observability – example

:: $S = \{0, 1\}$, $A = \{Stay, Go\}$, $O = \{o_0, o_1\}$,
$P(s_{t+1} = x | s_t = x, Stay) = .9$, $P(s_{t+1} = x | s_t = x, Go) = .1$,
$Pr(o_0|0) = .6$, $Pr(o_1|1) = .6$, $R(0) = 0$, $R(1) = 1$, $\gamma = 1$,

:: Goal: determine $V^*(b)$ (the main step for finding $a = \pi^*(b)$)



Partial observability – example

- b space is 1D $\rightarrow V(b)$ is a real function of one variable,
- assumed that in near points of b space will be
  - very similar utility and identical policy,
- policy is equivalent to a **conditional plan** dependent on future observations
  - example, plan of length 2: $[Stay,$ if $O = o_0$ then Go else $Stay]$,
- let $\alpha_p(s)$ be the utility of plan p starting from state s
  - then the same plan executed from b has the utility

$$\sum_s b(s)\alpha_p(s) = b \cdot \alpha_p$$

  - α_p is a linear function of b (hyperplane for complex spaces),
  - optimal policy follows the plan with highest expected utility

$$V(b) = V^{\pi^*}(b) = \max_p b \cdot \alpha_p$$

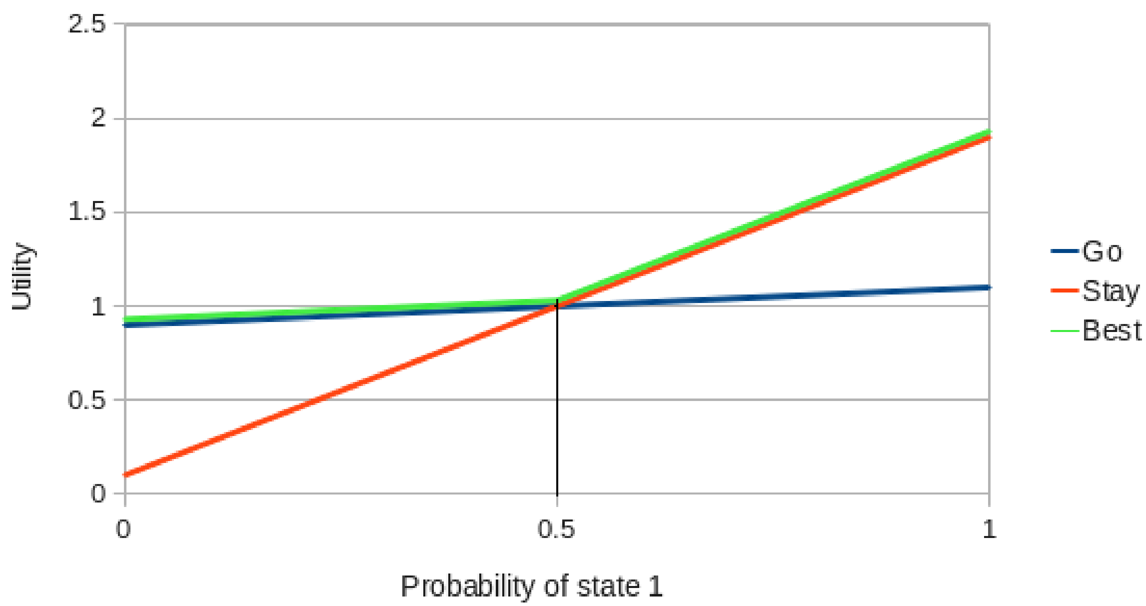
  - $V(b)$ is a partially linear function of b.

Partial observability – example

:: there are two plans of length 1, for them it holds

$$\begin{aligned}
\alpha_{[Stay]}(0) &= R(0) + \gamma(.9R(0) + .1R(1)) = 0.1 \\
\alpha_{[Stay]}(1) &= R(1) + \gamma(.9R(1) + .1R(0)) = 1.9 \\
\alpha_{[Go]}(0) &= R(0) + \gamma(.9R(1) + .1R(0)) = 0.9 \\
\alpha_{[Go]}(1) &= R(1) + \gamma(.9R(0) + .1R(1)) = 1.1
\end{aligned}$$

$$\alpha_{[Stay]}(b(1) = 0.3) = .7\alpha_{[Stay]}(0) + .3\alpha_{[Stay]}(1) = 0.64$$

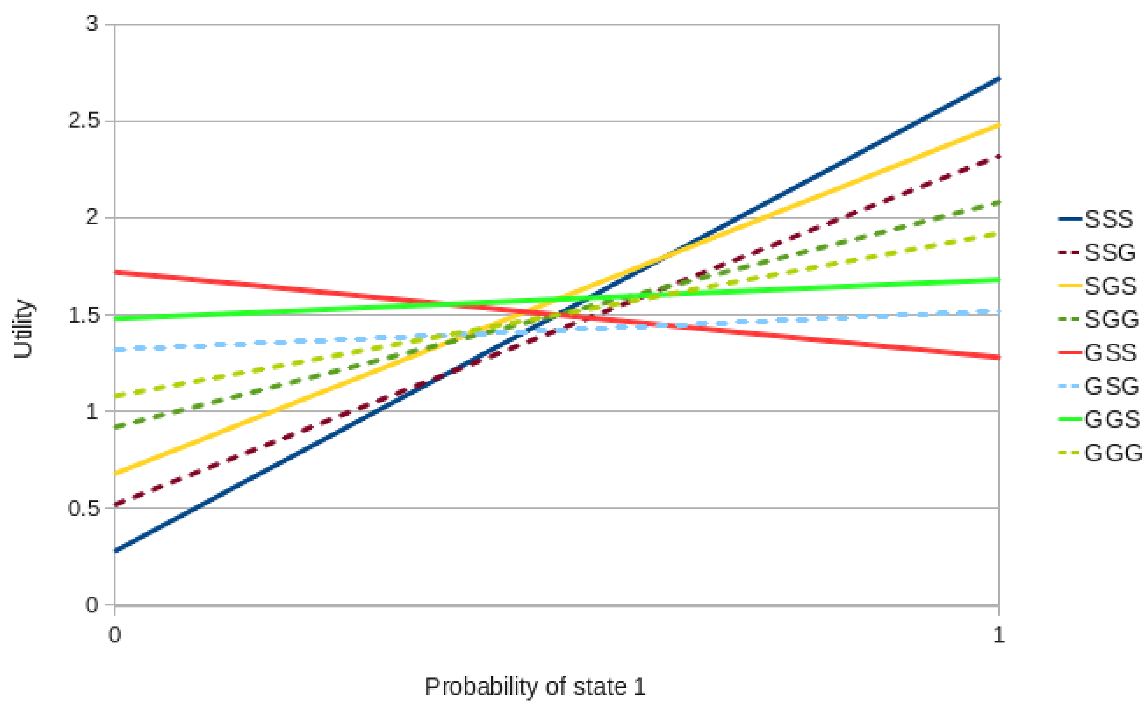


Partial observability – example

:: there are 8 plans of length 2 (4 dotted plans dominated by other plans)

$[Stay,$ if $O = o_0$ then Go else $Stay]$ encoded as $[SGS]$

$$\begin{aligned}
\alpha_{[SSS]}(0) &= R(0) + \gamma(.9\alpha_{[S]}(0) + .1\alpha_{[S]}(1)) &= 0.28 \\
\alpha_{[SSS]}(1) &= R(1) + \gamma(.9\alpha_{[S]}(1) + .1\alpha_{[S]}(0)) &= 2.72 \\
\alpha_{[SGS]}(0) &= R(0) + \gamma(.9(.6\alpha_{[G]}(0) + .4\alpha_{[S]}(0)) + .1(0.4\alpha_{[G]}(1) + .6\alpha_{[S]}(1)) &= 0.68 \\
\alpha_{[SGS]}(1) &= R(1) + \gamma(.9(.4\alpha_{[G]}(1) + .6\alpha_{[S]}(1)) + .1(0.6\alpha_{[G]}(0) + .4\alpha_{[S]}(0)) &= 2.48
\end{aligned}$$



Partial observability – plans of length d

- generalized formula to evaluate plans of length d

$$\alpha_p(s) = R(s) + \gamma\Big(\sum_{s'} P^a_{ss'} \sum_o \Omega^a_{s'o} \alpha_{p.o}(s')\Big)$$

  - recursive formula,
  - plan p with length d,
  - $p.o$ is its subplan with length $d-1$ without observation o,
  - pruning of dominated plans helps,
  - still, worst-case time complexity $\mathcal{O}(|A|^{|O|^{d-1}})$.

What happens if the world/environment model is unknown?

- We assume that the world is a Markov decision process
  - however, neither its transition model nor reward function is known,
- the goal is to learn the optimal policy again
  - agent needs to **interact** with the environment
    - it cannot purely use the model and compute the optimal policy,
  - agent **explores** (many) different actions in (many) different states,
  - analogous to learning and solving the underlying MDP
    - must be done simultaneously and thus iteratively,
- solved in terms of **reinforcement learning**.

Active adaptive dynamic programming

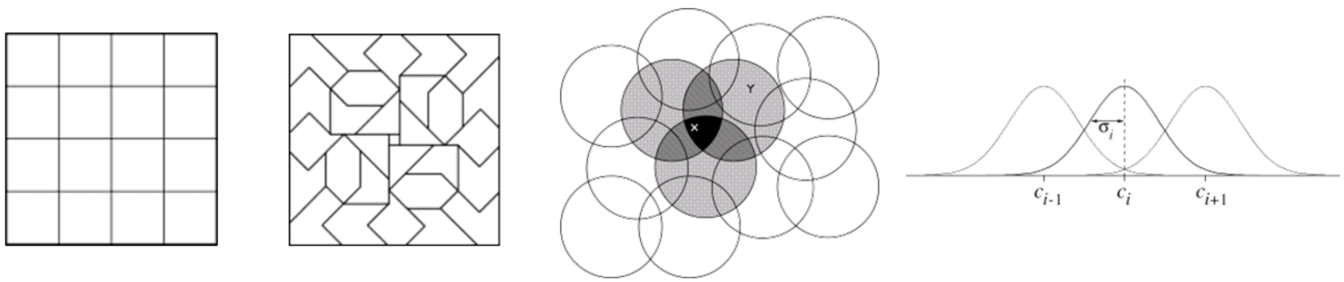
- **Active adaptive dynamic programming** is one of the RL approaches
  - the agent interacts with the environment
    - it executes actions and obtains percepts (= states and rewards),
    - single interaction = read a percept and execute an action,
  - after each interaction (step)
    - it updates its model = P and R,
    - it solves the underlying MDP for the current P and R = learns V,
    - executes an action based on the known state values,
  - a new important issue is how to select the action
    - criteria: optimal policy reached?, learning time, reward during learning,
    - a bad option: greedy action selection,
      - the agent exploits its current model but does not sufficiently explore,
      - the true model does not have to be learned,
    - a better option: ϵ-greedy action selection
      - only in ϵ situations random action, otherwise greedy,

The algorithm of active adaptive dynamic programming

```
function ACTIVE_ADP_AGENT(percept) returns an action
 inputs: percept, the current state s' and reward signal r'
 static:  V, a table of utilities, initially empty
  mdp, an MDP with model P, rewards R, discount γ
  Nsa, a state-action frequency table, initially zero
  Nsas', a state-action-state frequency table, initially zero
  s, a the previous state and action, initially null
 if s' is new then do V[s']= r'; R[s']= r'
 if s is not null, then do
  increment Nsa[s,a] and Nsas'[s,a,s']
  for each t such that Nsas'[s,a,t] is nonzero do
   P[s,a,t] = Nsas'[s,a,t] / Nsa[s,a]
 V = VALUE_ITERATION(mdp)
 if TERMINAL?[s'] then s,a=null else s,a = s',GET_ϵ_GREEDY(s',V)
 return a
```

Generalization and approximation

- up to now trivial demonstrations with limited state and action sets,
- for large state spaces, necessary approximation of value functions
  - learning from examples can be employed,
  - generalization assumes continuous and "reasonable" value functions,
  - states characterized by a parameter/attribute vector ϕ_s,



  - linear approximation with parameters $\theta(t)$: $V_t(s) = \theta_t^T \phi_s = \sum_i \theta_t(i)\phi_s(i)$,
  - non linear optimization by a neural network,
  - error function to be minimized: $MSE(\theta_t) = \sum_s P_s \left[V_t^\pi(s) - V_t(s)\right]^2$, P_s distribution of state weights, $V_t^\pi(s)$ real value, $V_t(s)$ its approximation,
  - regression, gradient optimization, back propagation etc.

Summary

- MDPs allow to search stochastic state spaces
  - computational complexity is increased due to stochasticity,
- problem solving = policy finding
  - policy assigns each state the optimal action, can be stochastic too,
  - basic approaches are policy iteration and value iteration,
  - other choices can be modified iteration approaches, possibly asynchronous,
- techniques similar to MDP
  - POMDP for partially observable environments,
  - RL for environments with unknown models,
- applications
  - agent technology in general, robot control and path planning in robotics,
  - network optimization in telecommunication, game playing.

Recommended reading, lecture resources

:: Reading

- Russell, Norvig: **AI: A Modern Approach**, Making Complex Decisions
  - chapter 17,
  - online on Google books: http://books.google.com/books?id=8jZBksh-bUMC,
- Sutton, Barto: **Reinforcement Learning: An Introduction**
  - MIT Press, Cambridge, 1998,
  - http://www.cs.ualberta.ca/~sutton/book/the-book.html.

Demo

- RL simulator
  - find the optimal path in a maze
  - implemented in Java
  - http://www.cs.cmu.edu/~awm/rlsim/