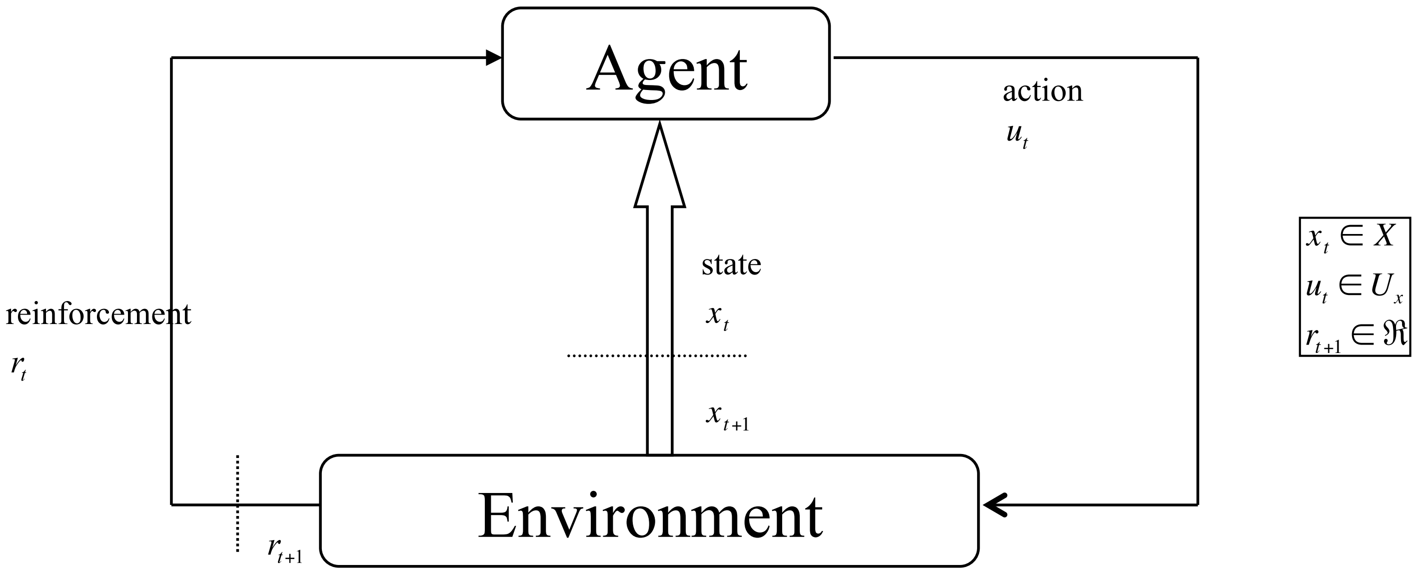
REINFORCEMENT LEARNING

Pedro Lima

pal@isr.ist.utl.pt
Instituto Superior Técnico (IST)
Instituto de Sistemas e Robótica (ISR)
Av.Rovisco Pais, 1
1049-001 Lisboa
PORTUGAL

May.2002
revised December 2009

Reinforcement Learning (RL)



Goal: choose the action sequence that maximizes

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

T may go to infinity, as long as $\gamma \neq 1$

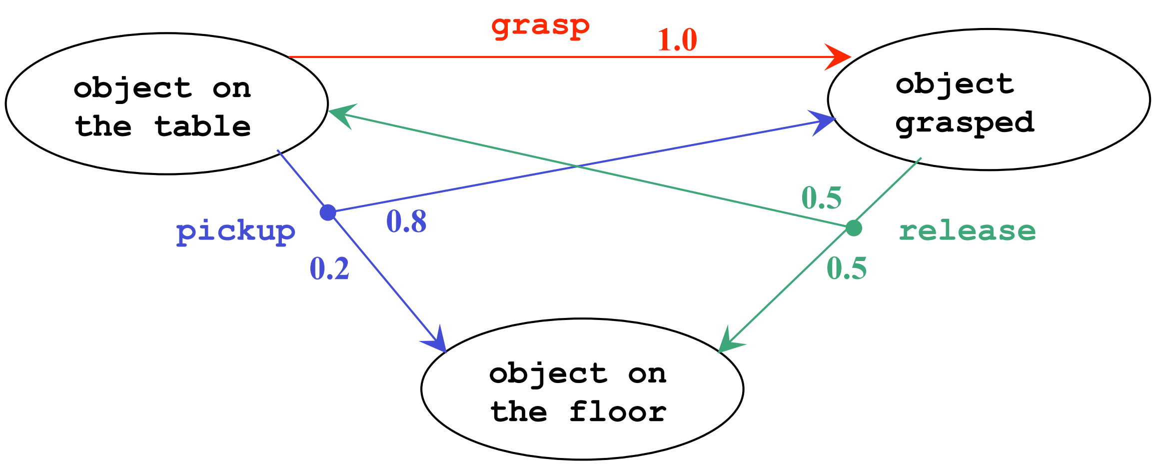
Rewards and state transitions after an action is executed are stochastic.

Markov Decision Process (MDP)

A Markov Chain (which by definition satisfies the Markov Property) with transition probabilities dependent on actions is known as Markov Decision Process (MDP)

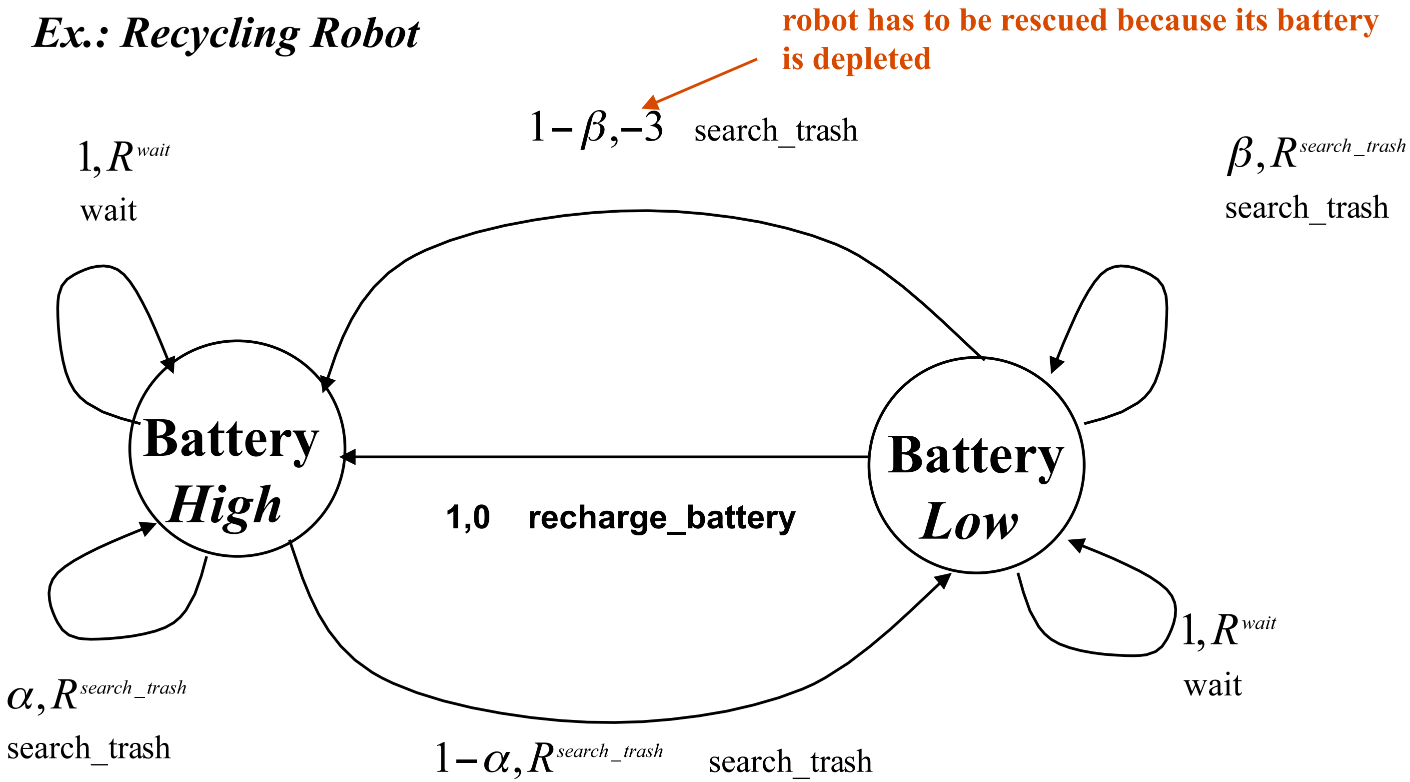
Example: conditional joint probability of state and reward:

$$\Pr\left\{x_{t+1} = x', r_{t+1} = r \,\middle|\, x_t, u_t, r_t, x_{t-1}, u_{t-1}, \ldots, r_1, x_0, u_0\right\} = \Pr\left\{x_{t+1} = x', r_{t+1} = r \,\middle|\, x_t, u_t\right\}$$



Markov Decision Process (MDP)

Ex.: Recycling Robot



transition probability ← α, R^{search_trash} → expected reward

search_trash → action taken

$$R^{search\_trash} > R^{wait}$$

Number of cans collected while performing the corresponding tasks

Value Functions

Policy $\equiv \quad \pi(x,u):(x \in X, u \in U_x) \rightarrow \pi(x,u)$ Probability of carrying out action *u* in state *x*

State value for policy π:

we leave the conditioning explicit here

α^k in the DP formulation

costs in the DP formulation

$$V_\infty^\pi(x) = \mathrm{E}_\pi\{R_t | x_t = x\} = \mathrm{E}_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | x_t = x\right\}$$

Expected value of starting in state *s* and following policy π thereafter.

NOTE: value of final state, if any, is always zero.

(state, action) value for policy π:

$$Q_\infty^\pi(x,u) = \mathrm{E}_\pi\{R_t | x_t = x, u_t = u\} = \mathrm{E}_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | x_t = x, u_t = u\right\}$$

Expected value of starting in state *x, carrying out action u*, and following policy π thereafter.

Markov Decision Process (MDP)

Given:

- States x
- Actions u
- Transition probabilities $p(x'|u,x)$
- Reward / expected payoff function $r(x,u)$

Wanted:

- Policy $\pi(x)$ that maximizes the future expected reward

Value Iteration

1. for all x do

$$\hat{V}(x) \leftarrow r_{\min}$$

2. endfor

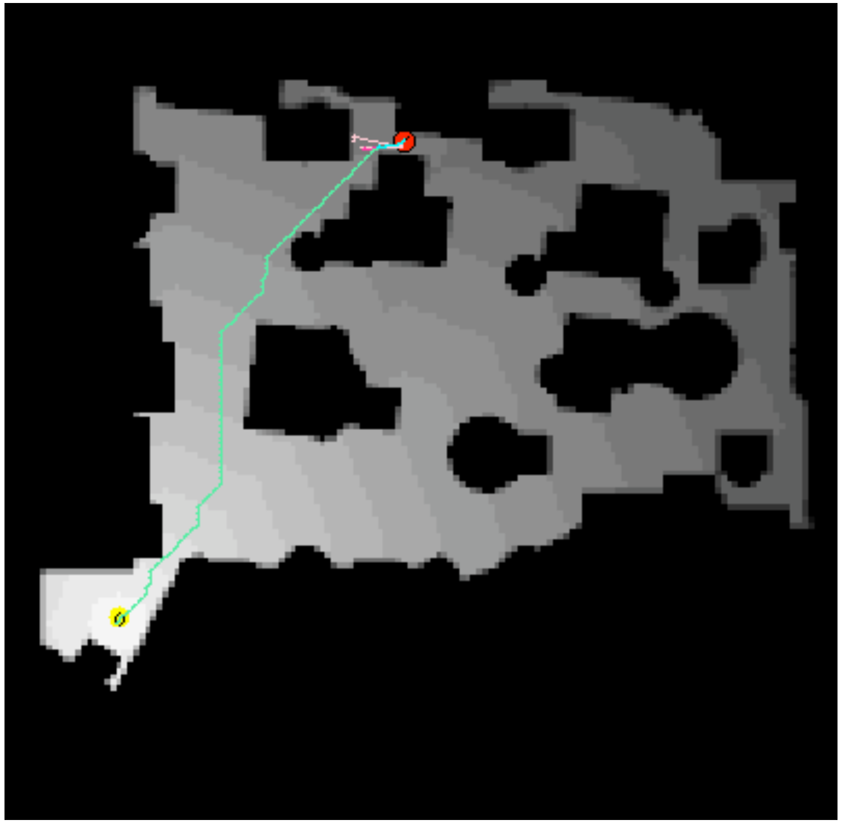
3. repeat until convergence
   1. for all x do

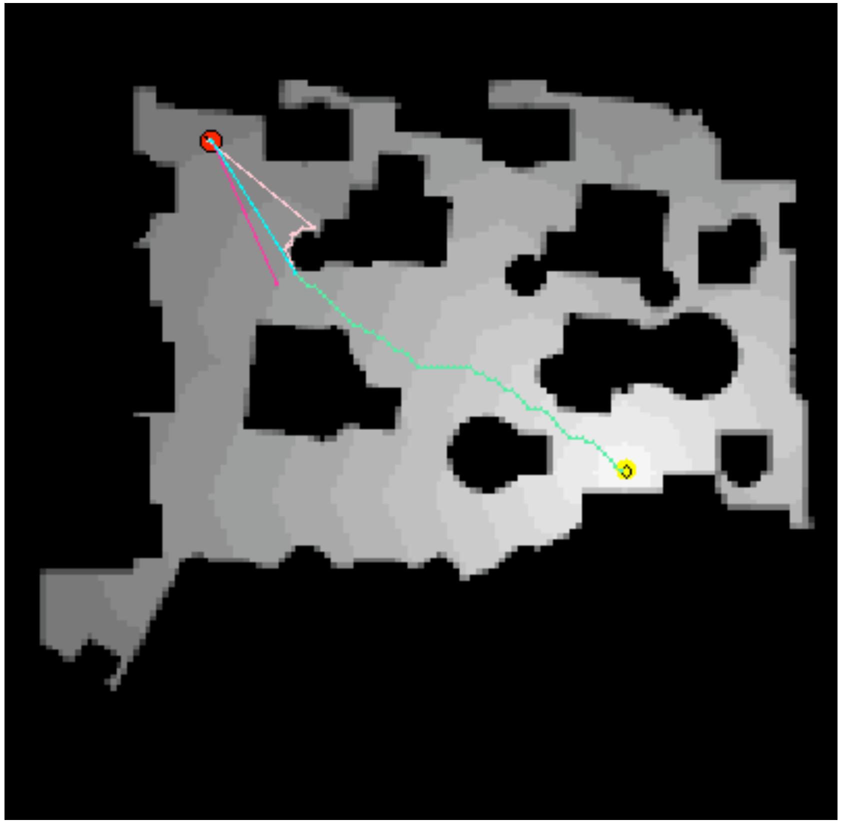
   $$\hat{V}(x) \leftarrow \max_u \left[r(x,u) + \gamma \sum_{x'} \hat{V}(x') p(x' \mid u,x) dx' \right]$$

   2. endfor
4. endrepeat

$$\pi(x) = \operatorname*{argmax}_u \left[r(x,u) + \gamma \sum_{x'} \hat{V}(x') p(x' \mid u,x) dx' \right]$$

Value Iteration for Motion Planning





Value Function and Policy Iteration

- Often the optimal policy has been reached long before the value function has converged.
- Policy iteration calculates a new policy based on the current value function and then calculates a new value function based on this policy.
- This process often converges faster to the optimal policy.

Policy Iteration

repeat

deterministic policy — *matrix of* p(*x'*|*u*,*x*) — *matrix of* r(*u*,*x*)

$$\hat{V} \leftarrow \text{Value\_Iteration}(\pi, \hat{V}, P, R)$$

$$\text{policy\_unchanged} \leftarrow \textit{True}$$

1. **for each state *x* do**

$$\text{if } \max_u \sum_{x'} p(x' | u, x)\hat{V}(x') > \sum_{x'} p(x' | \pi(x), x)\hat{V}(x')$$

$$\pi(x) \leftarrow \arg\max_u \sum_{x'} p(x' | u, x)\hat{V}(x')$$

$$\text{policy\_unchanged} \leftarrow \textit{False}$$

2. **end for**

until policy_unchanged

Reinforcement Learning

Previous (DP) methods to solve MDPs assume full knowledge of $p(x'|u,x)$

Dynamic Programming (DP)

- To determine V for |X| = N, a system of N non-linear equations must be solved.
- Well established mathematical method.
- A complete model of the environment is required (P *and* R known).
- Often faces the "*curse of dimensionality*" [Bellman, 1957]

Alternative approaches, if we do not know $p(x'|u,x)$

Monte Carlo

- Similar to DP, but P and R_s unknown.
- P and R determined from the average of several trial-and-error trials.
- Unappropriate for a step-by-step incremental approximation of V^*.

Temporal Differences

- Knowledge of *P e R* is not required
- Step-by-step incremental approximation of V.
- Mathematical analysis more complex.
- *Q-learning*

Value Functions cont'd

Bellman equation for *V* (discrete action and state spaces)

$$V_T^{\pi}(x) = \sum_{x'} p(x'|u,x)\left[r(x,u) + \gamma V_{T-1}^{*}(x')\right]$$

$$V_T^{*} = \max_{\pi} V_T^{\pi}(x) \quad \forall_x$$

$$V^{*}(x) = \max_{u} E\left[r(x,u) + \gamma V^{*}(x',u')\right]$$

$$Q_T^{\pi}(x,u) = \sum_{x'} p(x'|u,x)\left[r(x,u) + \gamma \max_{u'} Q_{T-1}^{\pi}(x',u')\right]$$

$$Q_T^{*} = \max_{\pi} Q_T^{\pi}(x,u) \quad \forall_{x,u}$$

$$V^{*}(x) = \max_{u} Q^{*}(x,u) \Rightarrow Q^{*}(x,u) = E\left[r(x,u) + \gamma \max_{u'} Q^{*}(x',u')\right]$$

Solutions are unique and equations are also met by the optimal functions

$$Q^{*}(x,u) = \mathrm{E}\left\{r_{t+1} + \gamma V^{*}(x_{t+1} = x') \middle| x_t = x, u_t = u\right\}$$

Q-Learning

- **Once V^* is known or learned, an apparently obvious solution for the RL problem would be:**

$$\pi^*(x) = \arg\max_{u(x)} E\left\{ r(x,u) + \gamma V^*(\delta(x,u)) \mid x_t = x, u_t = u \right\}, \quad \delta(x,u) \equiv \text{state transition function}$$

... but *r(x,u)* and *δ(x,u)* are unknown in the general case

- **However, if we know or learn Q^*, a different solution arises:**

$$Q^*(x,u) = \mathrm{E}\left\{ r_{t+1} + \gamma V^*(x_{t+1}) \middle| x_t = x, u_t = u \right\}$$

$$\pi^*(x) = \arg\max_{u(x)} Q^*(x,u)$$

In a stochastic environment, with unknown *P and R*, the agent's own experience when interacting with its environment can be used to learn Q^* and π^*.

Q-Learning - Algorithm

stochastic approximation: $Q(x_t,u_t) \leftarrow Q(x_t,u_t) + \alpha\left[r(x_{t+1},u_{t+1}) + \gamma \max_u Q(x_{t+1},u) - Q(x_t,u_t)\right]$

```
Initialize Q(x,u) random or arbitrarily
Repeat forever (for each episode or trial):
      Initialize x
      Repeat (for each step n of the episode):
            Choose action u of x
            Execute action u and observe r and x'
```

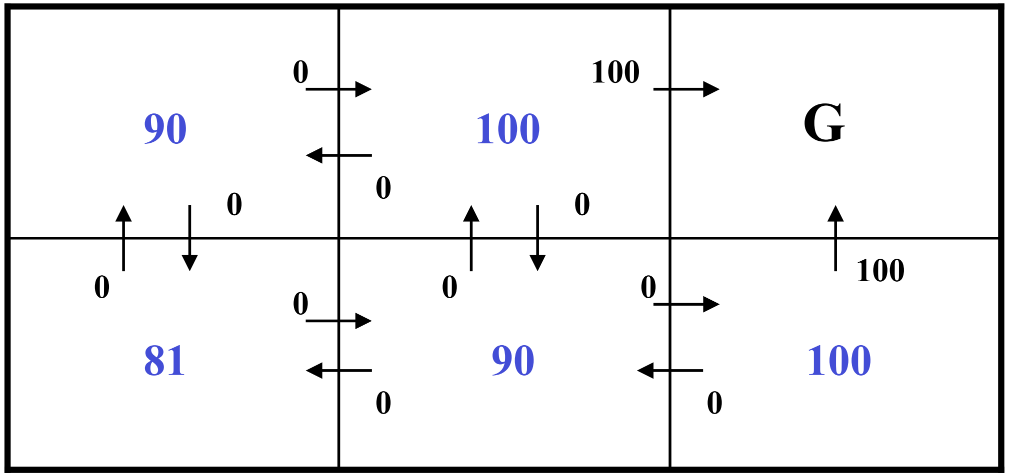
$$Q_{n+1}(x,u) \leftarrow Q_n(x,u) + \alpha_n\left[r(x,u) + \gamma \max_{u'} Q_n(x',u') - Q_n(x,u)\right]$$

$$x \leftarrow x';$$

```
      until x final.
```

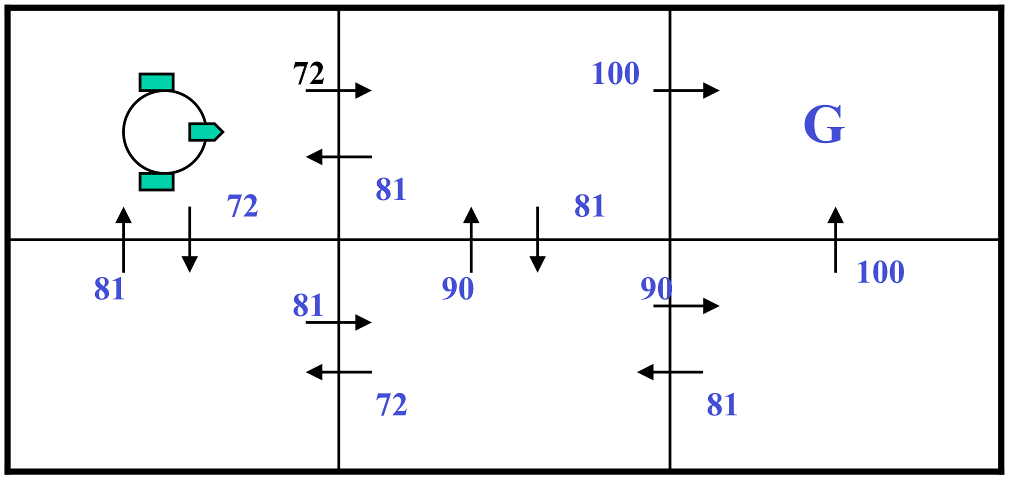
α constant allows adaptability to slow environment changes but it does not guarantee convergence – only possible with a temporal decay under given circumstances.

Q-Learning – an Example



$r(x,u)$

$V^*(x)$

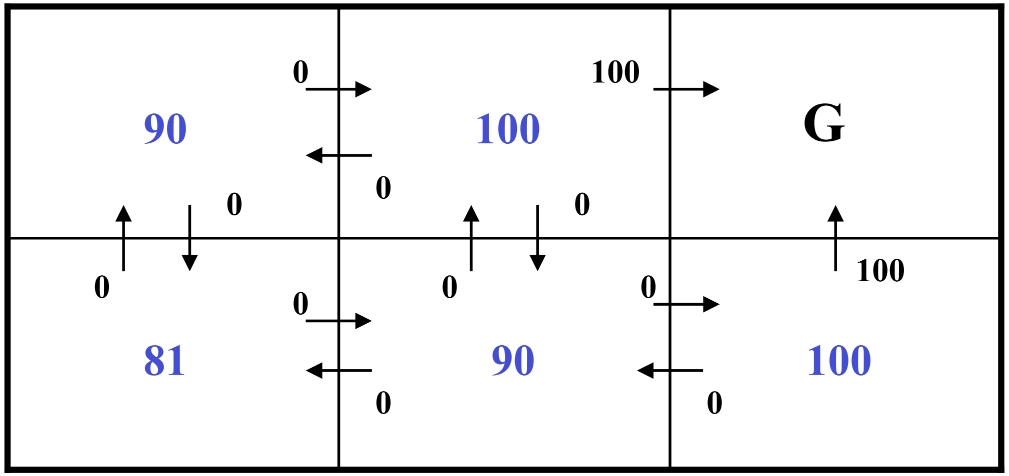


$Q_n^\pi(x,u)$

$\alpha = 1$

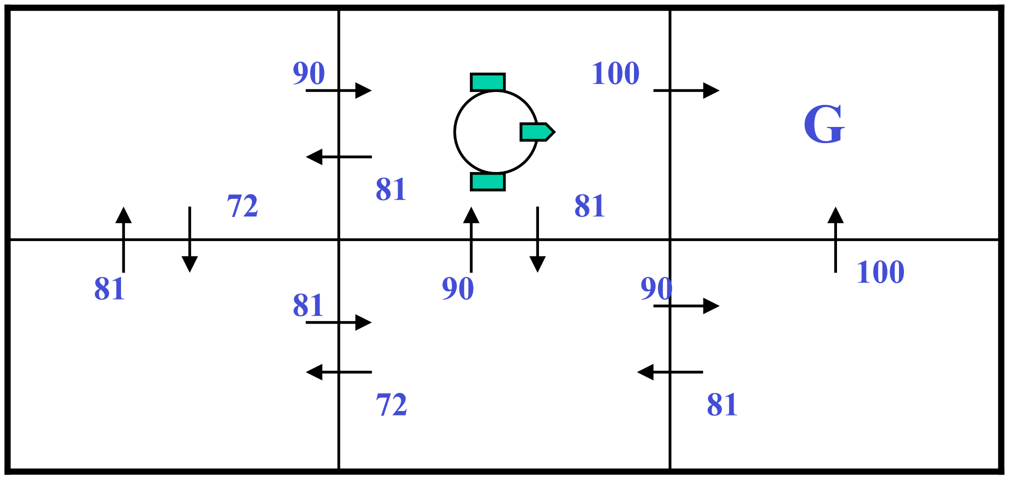
$\gamma = 0.9$

Q-Learning – an Example



$r(x,u)$

$V^*(x)$

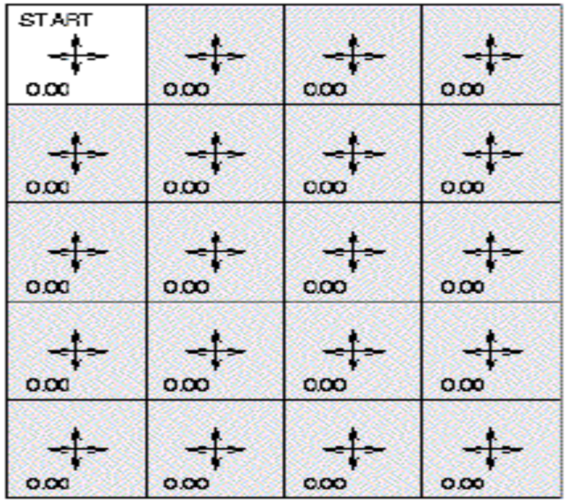


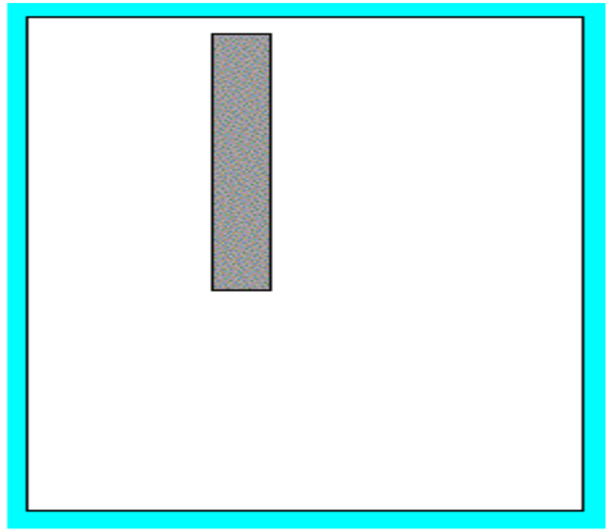
$Q_n^\pi(x,u)$

INSTITUTO SUPERIOR TÉCNICO

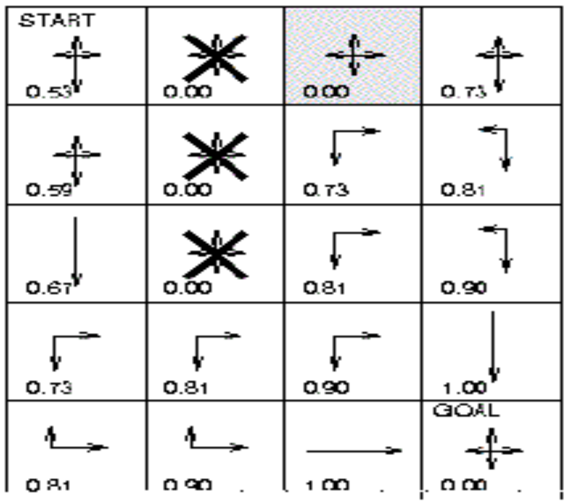
Q-Learning – another Example

Initial Situation





After some learning steps



Q-Learning – Algorithm Convergence

Should each pair (x,u) be visited an infinite number of times, with

$$0 \leq \alpha_n < 1$$

$$\sum_{i=1}^{\infty} \alpha_{n(i,x,u)} = \infty$$

$$\sum_{i=1}^{\infty} \alpha_{n(i,x,u)}^2 < \infty$$

then $\forall x,u \quad \Pr\left[\lim_{n \to \infty} \hat{Q}_n(x,u) = Q(x,u)\right] = 1$

Action Selection: Exploration *vs* Exploitation

Exploration: less promising actions, which may lead to good results, are tested.
Exploitation: takes advantage of tested actions which are more promising, i.e., which have a larger $Q(x,u)$.

- ε- *greedy:* at each step n, picks the best action so far with probability 1-ε , for small ε, but can also pick with probability ε , in an uniformly distributed random fashion, one of the other actions.
- *softmax:* at each step n, picks the action to be executed according to a Gibbs or Boltzmann distribution:

$$\pi_n(x,u) = \frac{e^{Q_n(x,u)/\tau}}{\sum_{u'(x)} e^{Q_n(x,u')/\tau}}$$

References and Further Reading

• K. Narendra, M. Thathatchar, *Learning Automata – An Introduction*, Prentice Hall, 1989

• D. P. Bertsekas, J.N. Tsitsiklis, *Neurodynamic Programming*, Athena Scientific, 1996

• T. Mitchell, *Machine Learning*, McGraw-Hill, Computer Science Series, 1997

• R. Sutton, A. Barto, *Reinforcement Learning – An Introduction*, The MIT Press, 1999