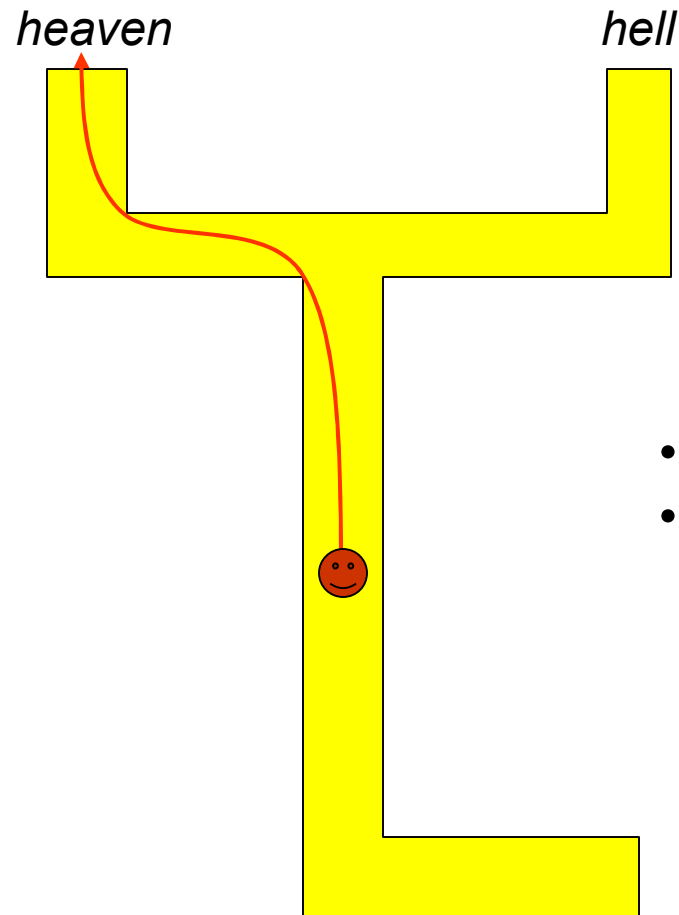


# Probabilistic Robotics:

## Probabilistic Planning and MDPs

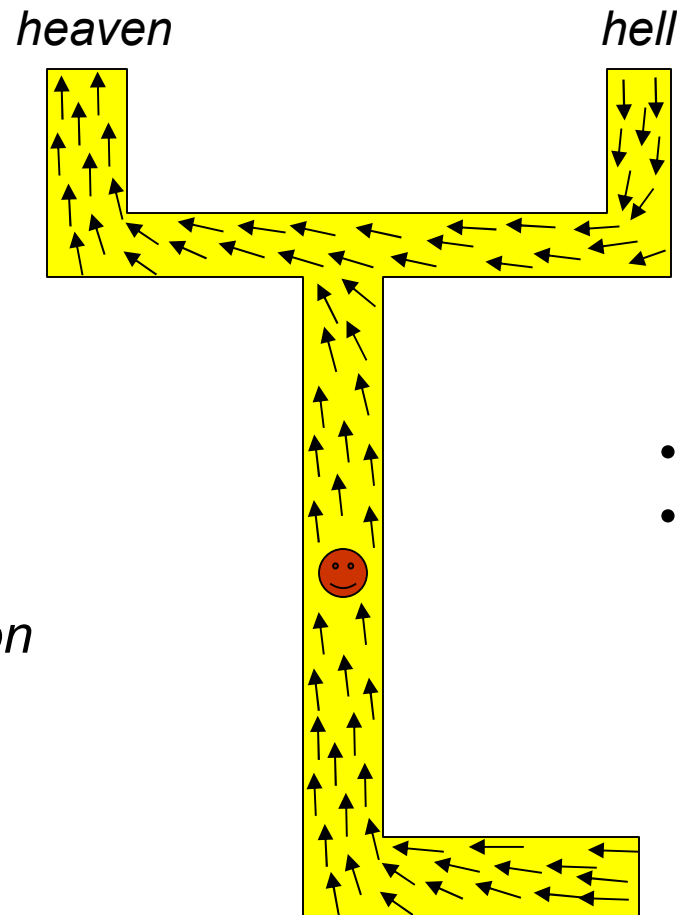
*Slide credits: Wolfram Burgard, Dieter Fox, Cyrill Stachniss, Giorgio Grisetti, Maren Bennewitz, Christian Plagemann, Dirk Haehnel, Mike Montemerlo, Nick Roy, Kai Arras, Patrick Pfaff and others*

# Planning: Classical Situation



- *World deterministic*
- *State observable*

# Planning in a stochastic world

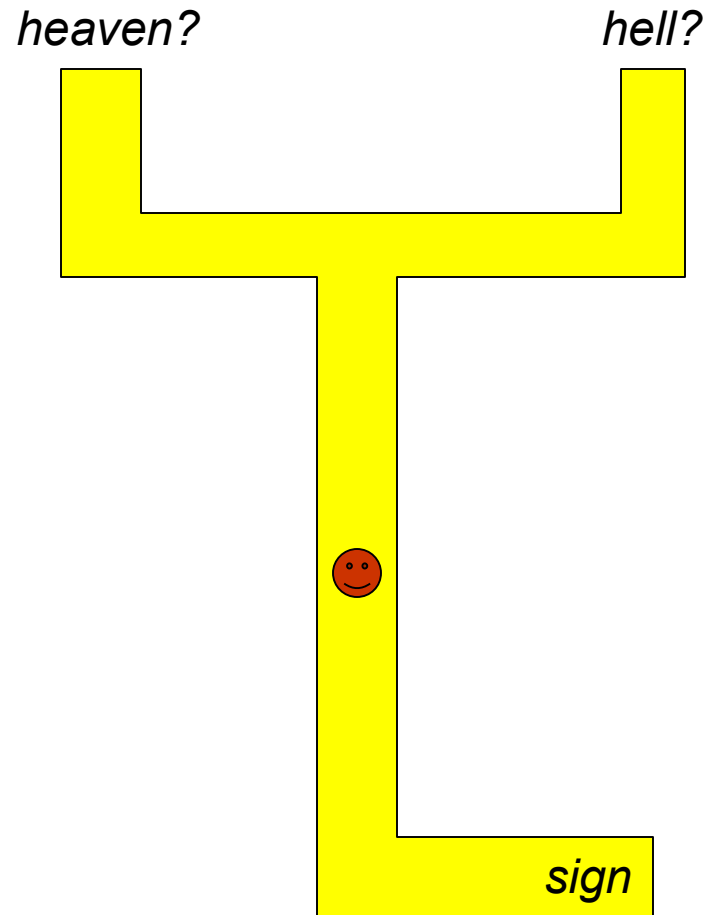


- *Policy*
- *Universal Plan*
- *Navigation function*

- *World stochastic*
- *State observable*

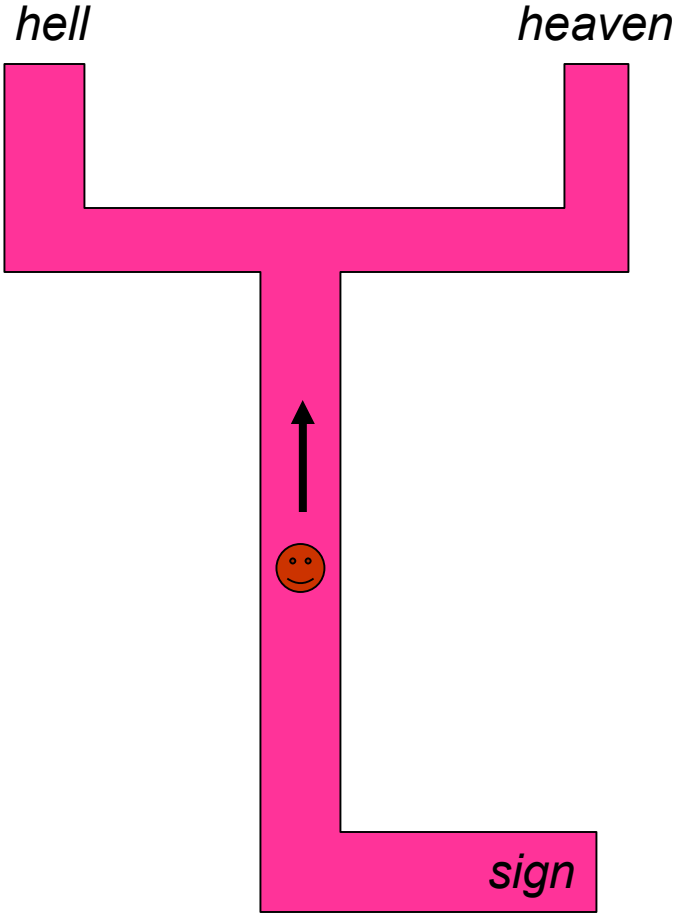
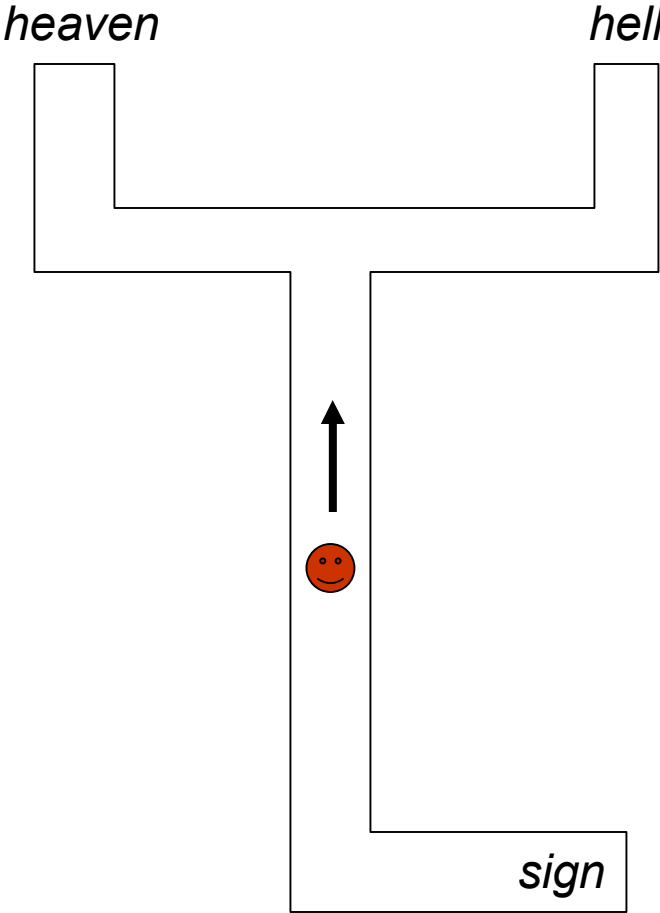
[Koditschek 87, Barto et al. 89]

# Stochastic, Partially Observable

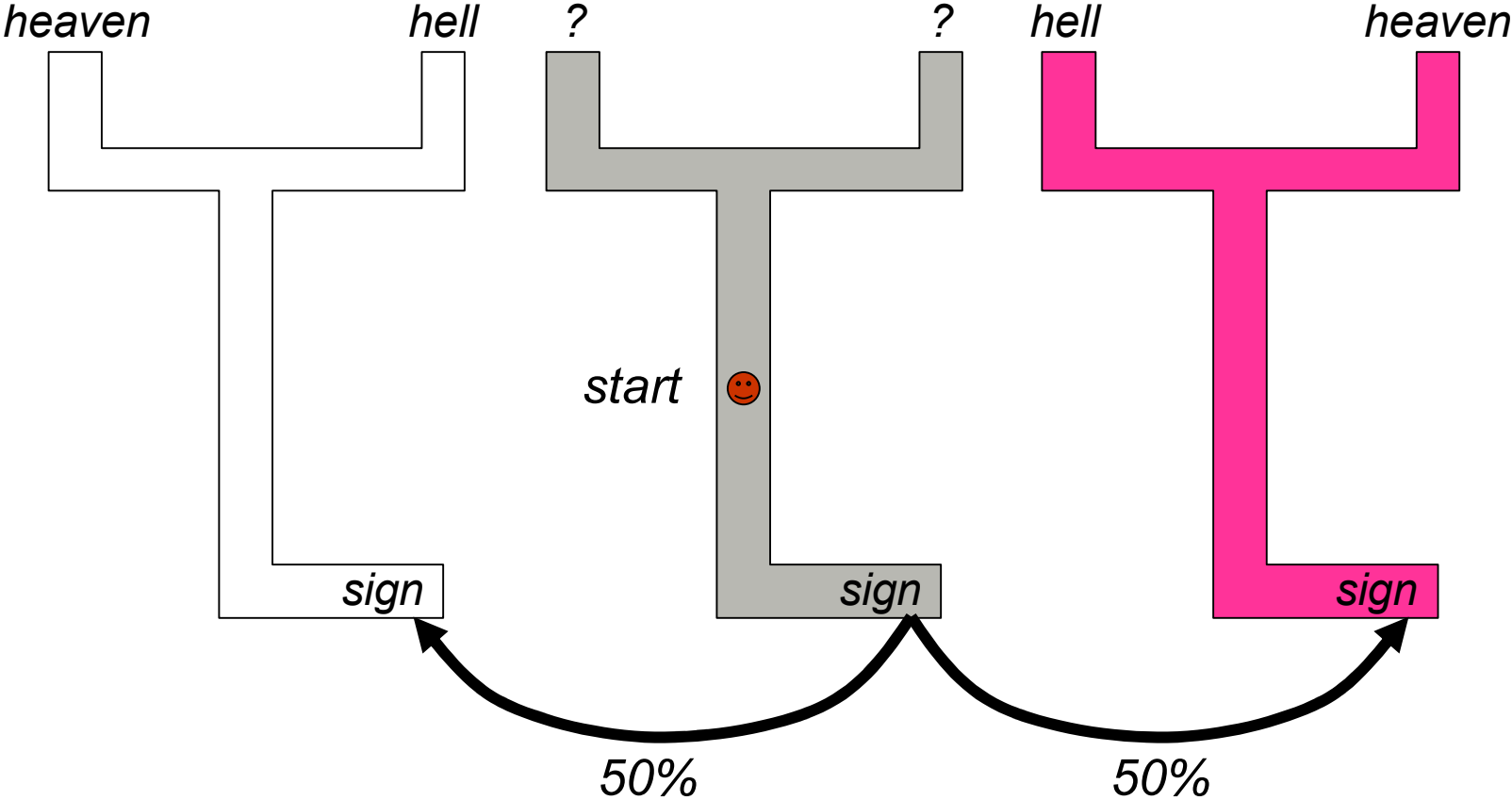


[Sondik 72] [Littman/Cassandra/Kaelbling 97]

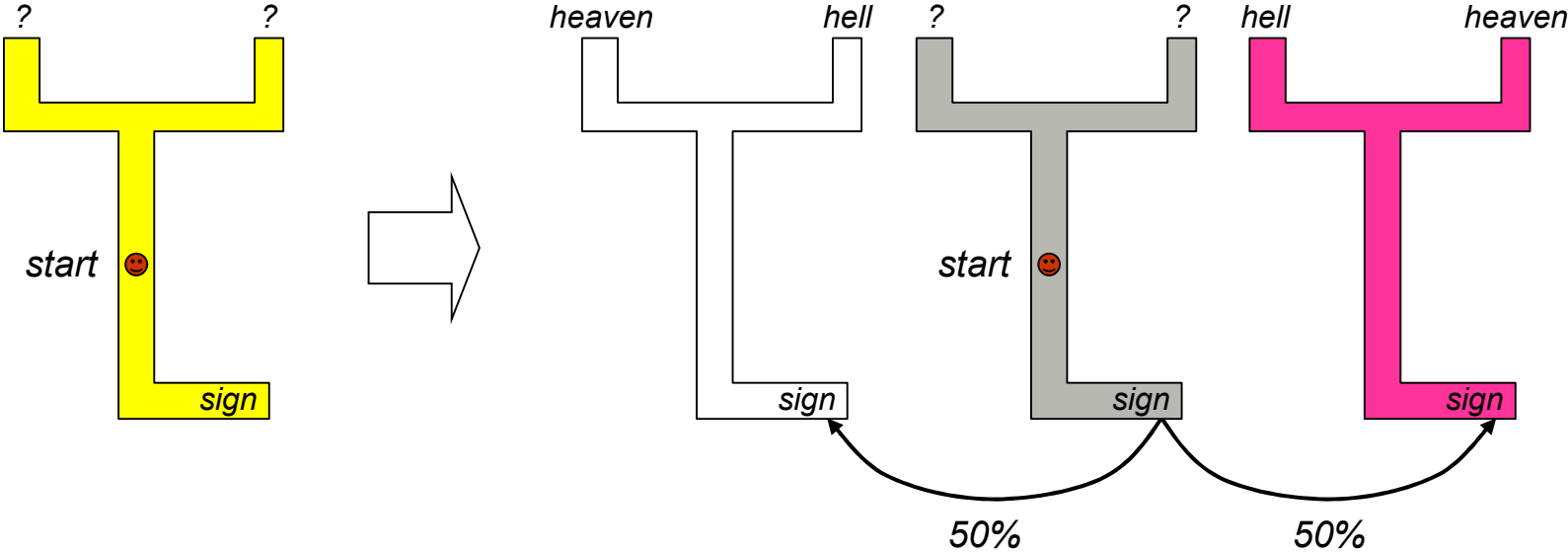
# Stochastic, Partially Observable



# Stochastic, Partially Observable



# Stochastic, Partially Observable



# A Quiz

<i># states</i>	<i>sensors</i>	<i>actions</i>	<i>size belief space?</i>
3	<i>perfect</i>	<i>deterministic</i>	3: $s_1, s_2, s_3$
3	<i>perfect</i>	<i>stochastic</i>	3: $s_1, s_2, s_3$
3	<i>abstract states</i>	<i>deterministic</i>	$2^3-1$ : $s_1, s_2, s_3, s_{12}, s_{13}, s_{23}, s_{123}$
3	<i>stochastic</i>	<i>deterministic</i>	2-dim continuous: $p(S=s_1), p(S=s_2)$
3	<i>none</i>	<i>stochastic</i>	2-dim continuous: $p(S=s_1), p(S=s_2)$
1-dim continuous	<i>stochastic</i>	<i>deterministic</i>	$\infty$ -dim continuous
1-dim continuous	<i>stochastic</i>	<i>stochastic</i>	$\infty$ -dim continuous
$\infty$ -dim continuous	<i>stochastic</i>	<i>stochastic</i>	aargh!



# MDP Planning

- Solution for Planning problem
  - Noisy controls
  - Perfect perception
  - Generates “universal plan” (=policy)

# Markov Models

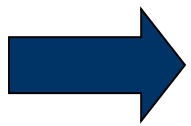
- Randomly changing systems
  - Stochastic variables
    - States  $X_t$  and probabilistic transitions  $P(X_{t+1}|X_t)$  among them
  - Markov property:
    - Future states depend only on the current state
  - Markov Chains:
    - Discrete-time Markov chains (DTMCs)
    - Continuous-time Markov chains (CTMCs) – Markov Processes
  - Hidden Markov Model (HMMs)
    - Markov model with hidden states and observation
  - Markov Decision Processes (MDPs)

# Markov Models

- Controllability and Observability
  - Observable state and non controllable actions:
    - Markov Chains
  - Not Observable state and not controllable actions:
    - Hidden Markov Models (HMMs)
  - Observable state and controllable actions:
    - Markov Decision Process (MDPs)
  - Not Observable state and controllable actions:
    - Partially Observable MDPs (POMDPs)

# What is the problem?

- Consider a non-deterministic robot/environment.
- Actions have desired outcome with a probability less than 1.
- What is the best action for a robot under this constraint?
  
- Example: a mobile robot does not exactly perform the desired action.



*Uncertainty about performing actions!*

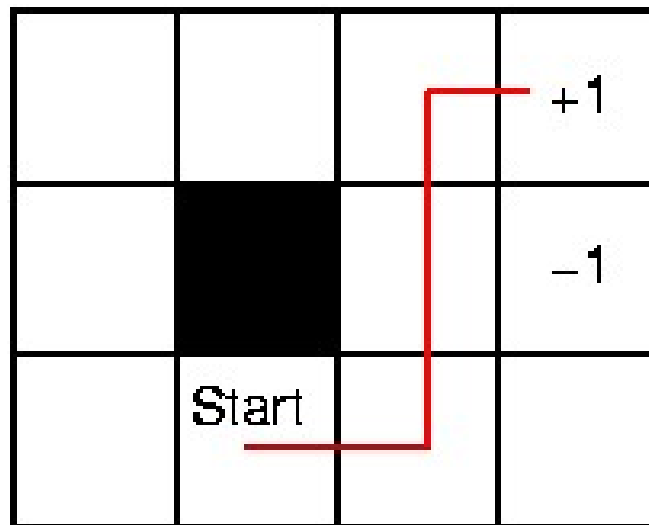
# Example (1)

			+1
			-1
	Start		

- *Bumping to wall "reflects" robot.*
- *"Reward" for free cells  $-0.04$  (travel cost).*
- *What is the best way to reach the cell labeled with  $+1$  without moving to  $-1$  ?*

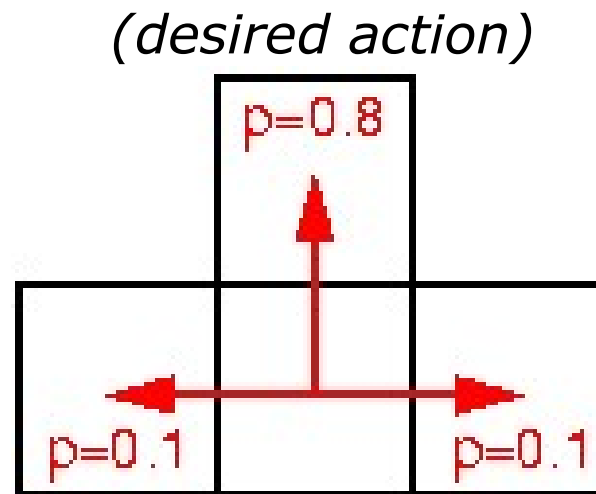
## Example (2)

- *Deterministic Transition Model:  
move on the shortest path!*



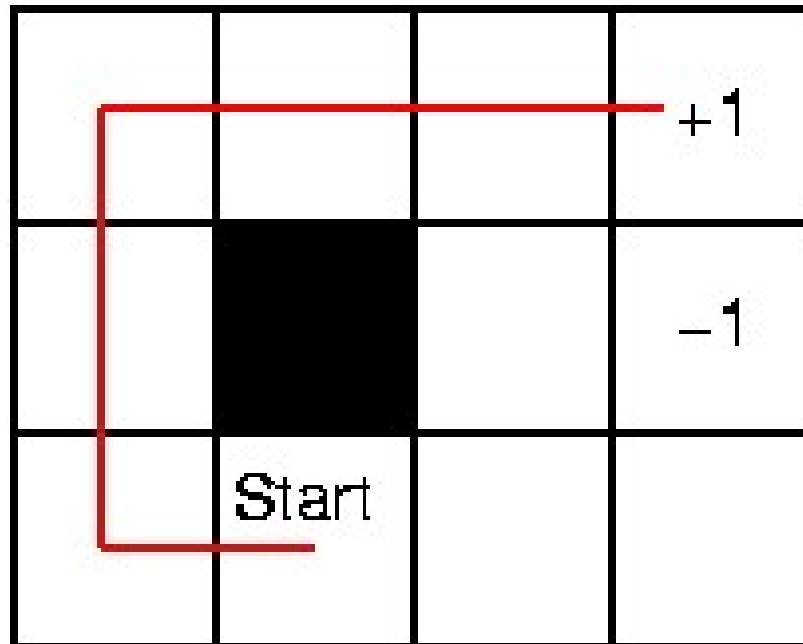
## Example (3)

- *But now consider the non-deterministic transition model (N / E / S / W):*



- *What is now the best way?*

## Example (4)



- *Use a longer path with lower probability to move to the cell labeled with  $-1$ .*
- *This path has the **highest overall utility!***



# Utility and Policy

- Compute for every state a **utility**:  
"What is the usage (utility) of this state for the overall task?"
- A **Policy** is a complete mapping from states to actions ("In which state should I perform which action?").

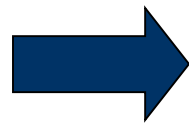
*policy* : States  $\mapsto$  Actions

# Markov Decision Problem (MDP)

- Compute the *optimal policy* in an accessible, stochastic environment with known transition model.

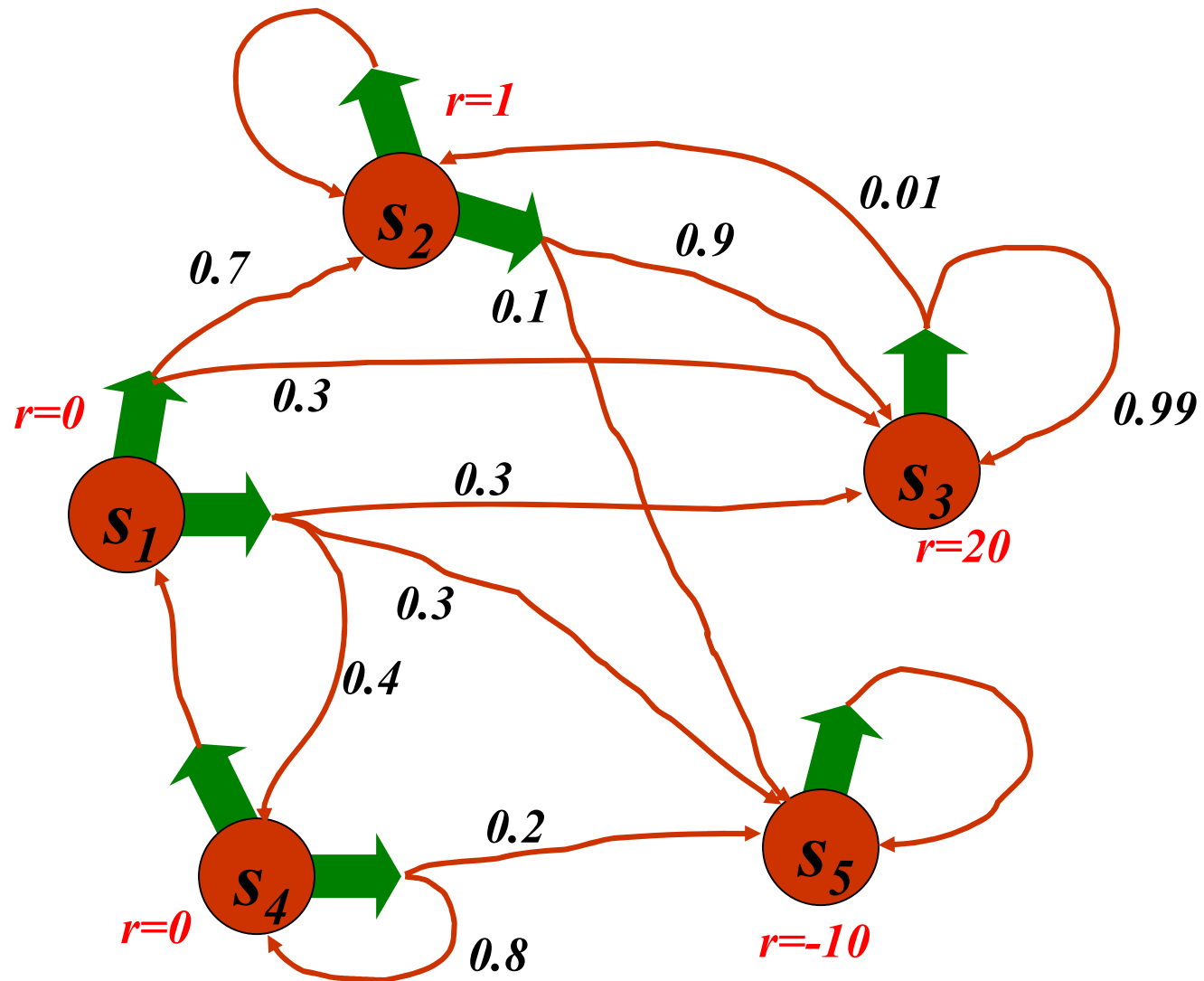
## Markov Property:

- The *transition probabilities* depend only on the current state and not on the history of predecessor states.



*Not every decision problem is a MDP.*

# Markov Decision Process (MDP)



# Markov Decision Process (MDP)

- **Given:**
  - States  $x$
  - Actions  $u$
  - Transition probabilities  $p(x' | u, x)$
  - Reward / payoff function  $r(x, u)$
- **Wanted:**
  - Policy  $\pi(x)$  that maximizes the future expected reward

# Rewards and Policies

- Policy (general case):

$$\pi: z_{1:t-1}, u_{1:t-1} \rightarrow u_t$$

- Policy (fully observable case):

$$\pi: x_t \rightarrow u_t$$

- Expected cumulative payoff:

$$R_T = E \left[ \sum_{\tau=1}^T \gamma^\tau r_{t+\tau} \right]$$

- $T=1$ : greedy policy
- $T>1$ : finite horizon case, typically no discount
- $T=\infty$ : infinite-horizon case, finite reward if discount  $< 1$

# Policies contd.

- Expected cumulative payoff of policy:

$$R_T^\pi(x_t) = E \left[ \sum_{\tau=1}^T \gamma^\tau r_{t+\tau} \mid \mathbf{u}_{t+\tau} = \pi(z_{1:t+\tau-1}, \mathbf{u}_{1:t+\tau-1}) \right]$$

- Optimal policy:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} R_T^\pi(x_t)$$

- 1-step optimal policy:

$$\pi_1(x) = \operatorname{argmax}_u r(x, u)$$

- Value function of 1-step optimal policy:

$$V_1(x) = \gamma \max_u r(x, u)$$

## 2-step Policies

- Optimal policy:

$$\pi_2(x) = \operatorname{argmax}_u \left[ r(x, u) + \int V_1(x') p(x' | u, x) dx' \right]$$

- Value function:

$$V_2(x) = \gamma \max_u \left[ r(x, u) + \int V_1(x') p(x' | u, x) dx' \right]$$

# T-step Policies

- Optimal policy:

$$\pi_T(x) = \operatorname{argmax}_u \left[ r(x, u) + \int V_{T-1}(x') p(x' | u, x) dx' \right]$$

- Value function:

$$V_T(x) = \gamma \max_u \left[ r(x, u) + \int V_{T-1}(x') p(x' | u, x) dx' \right]$$



# Infinite Horizon

- Optimal policy:

$$V_{\infty}(x) = \gamma \max_u \left[ r(x, u) + \int V_{\infty}(x') p(x' | u, x) dx' \right]$$

- Bellman equation
- Fix point is optimal policy
- Necessary and sufficient condition

# Value Iteration

- for all  $x$  do

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

- endfor

- repeat until convergence

- for all  $x$  do

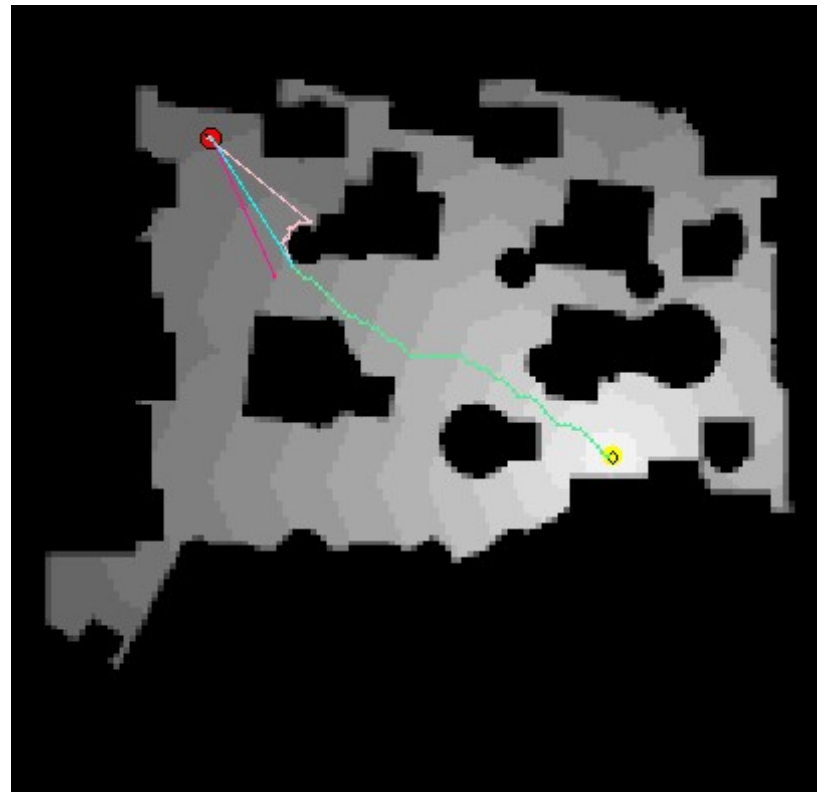
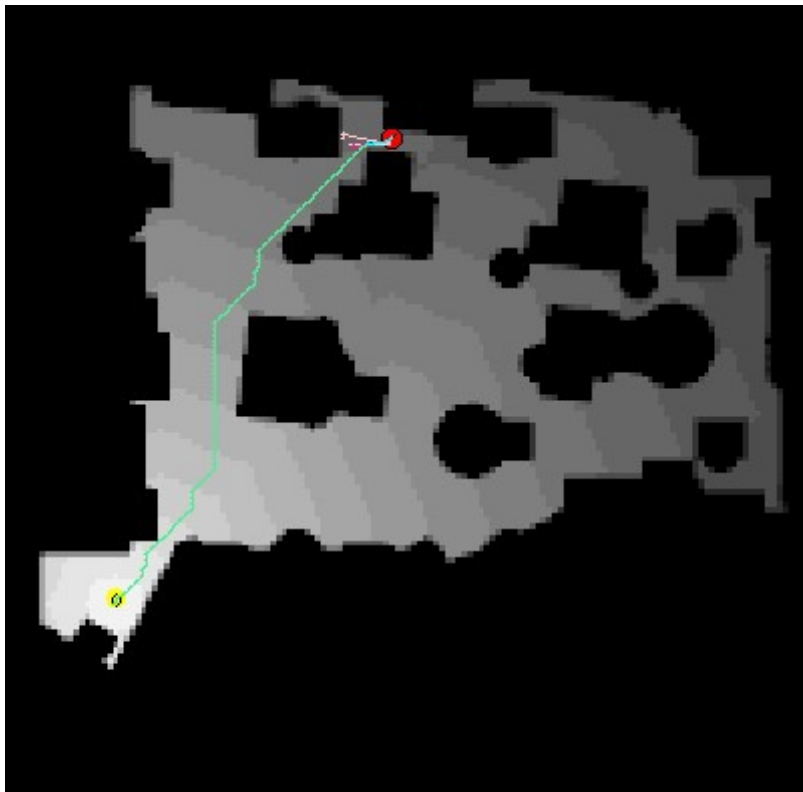
$$\hat{V}(x) \leftarrow \gamma \max_u \left[ r(x, u) + \int \hat{V}(x') p(x' | u, x) dx' \right]$$

- endfor

- endrepeat

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

# Value Iteration for Motion Planning



# The optimal Policy

$$\text{policy}^*(i) = \operatorname{argmax}_a \sum_j M_{ij}^a \cdot U(j)$$

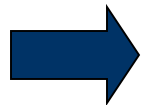
$M_{ij}^a$  = Probability of reaching state  $j$  from state  $i$  with action  $a$ .

$U(j)$  = Utility of state  $j$ .

- *If we know the utility we can easily compute the optimal policy.*
- *The problem is to compute the correct utilities for all states.*

# The Utility (1)

- *To compute the utility of a state we have to consider a **tree of states**.*
- *The utility of a state depends on the utility of **all successor states**.*



- *Not all utility functions can be used.*
- *The utility function must have the property of separability.*
- *E.g. additive utility functions:*

$$U([s_0, s_1, \dots, s_n]) = R(s_0) + U([s_1, \dots, s_n])$$

*(R = reward function)*

## The Utility (2)

- *The utility can be expressed similar to the policy function:*

$$U(i) = R(i) + \max_a \sum_j M_{ij}^a \cdot U(j)$$

- *The reward  $R(i)$  is the “utility” of the state itself (without considering the successors).*

# Dynamic Programming

- *This Utility function is the basis for "dynamic programming".*
- *Fast solution to compute  $n$ -step decision problems.*
- *Naive solution:  $O(|A|^n)$ .*
- *Dynamic Programming (Bellman, 1952):  $O(n|A||S|)$ .*
- *But what is the correct value of  $n$ ?*  
 $n \rightarrow \infty$
- *If the graph has loops: ???*

# Iterative Computation

## ***Idea:***

- *The Utility is computed iteratively:*

$$U_{t+1}(i) = R(i) + \max_a \sum_j M_{ij}^a \cdot U_t(j)$$

- *Optimal utility:  $U^* = \lim_{t \rightarrow \infty} U_t$*
- *Abort, if change in the utility is below a threshold.*



# The Value Iteration Algorithm

**function** VALUE-ITERATION( $M, R$ ) **returns** a utility function

**inputs:**  $M$ , a transition model

$R$ , a reward function on states

**local variables:**  $U$ , utility function, initially identical to  $R$

$U'$ , utility function, initially identical to  $R$

**repeat**

$U \leftarrow U'$

**for each** state  $i$  **do**

$U'[i] \leftarrow R[i] + \max_a \sum_j M_{ij}^a U[j]$

**end**

**until** CLOSE-ENOUGH( $U, U'$ )

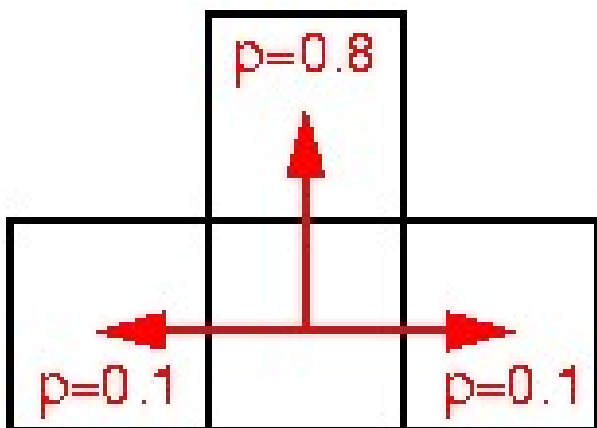
**return**  $U$

# Value Iteration Example

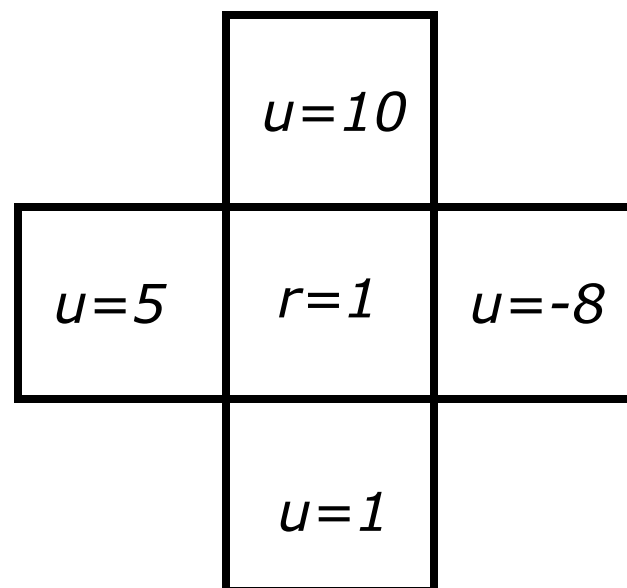
- Calculate utility of the center cell

$$U_{t+1}(i) = R(i) + \max_a \sum_j M_{ij}^a \cdot U_t(j)$$

(desired action=North)



Transition Model

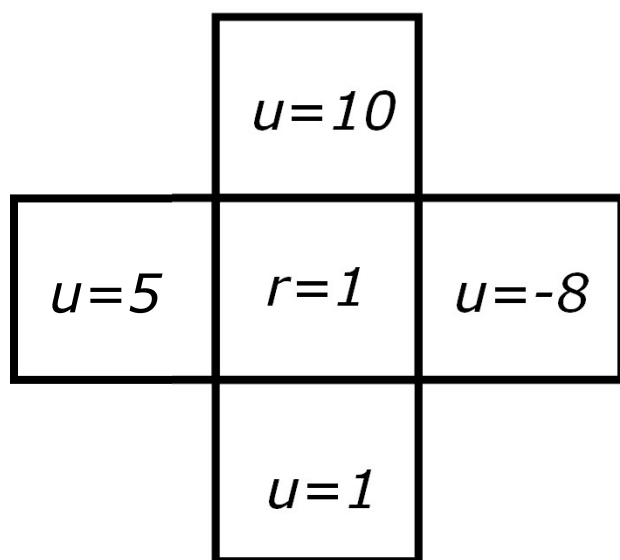


State Space

( $u$ =utility,  $r$ =reward)

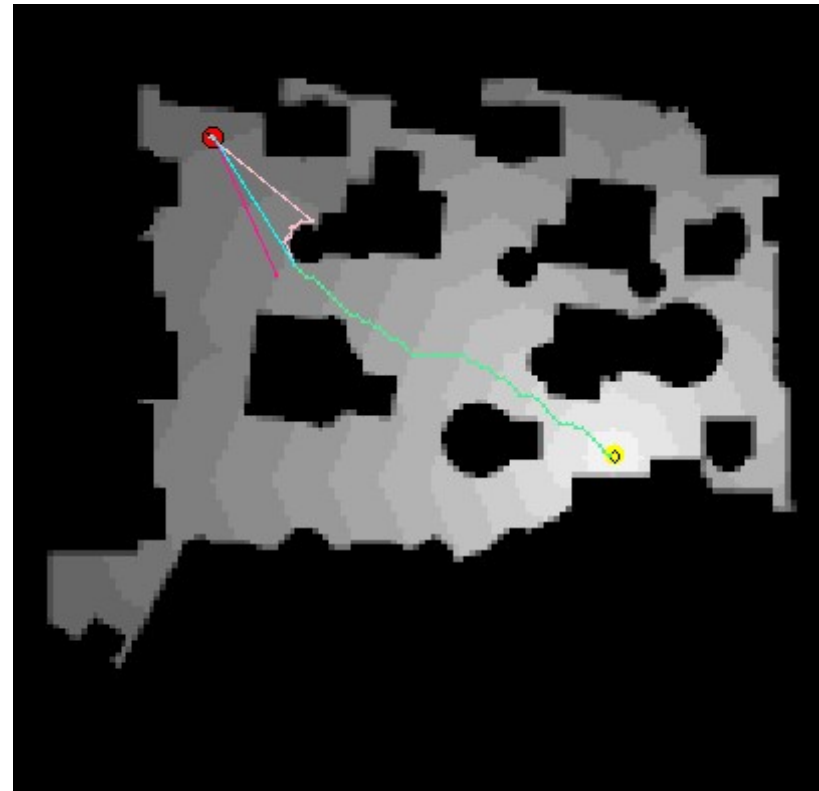
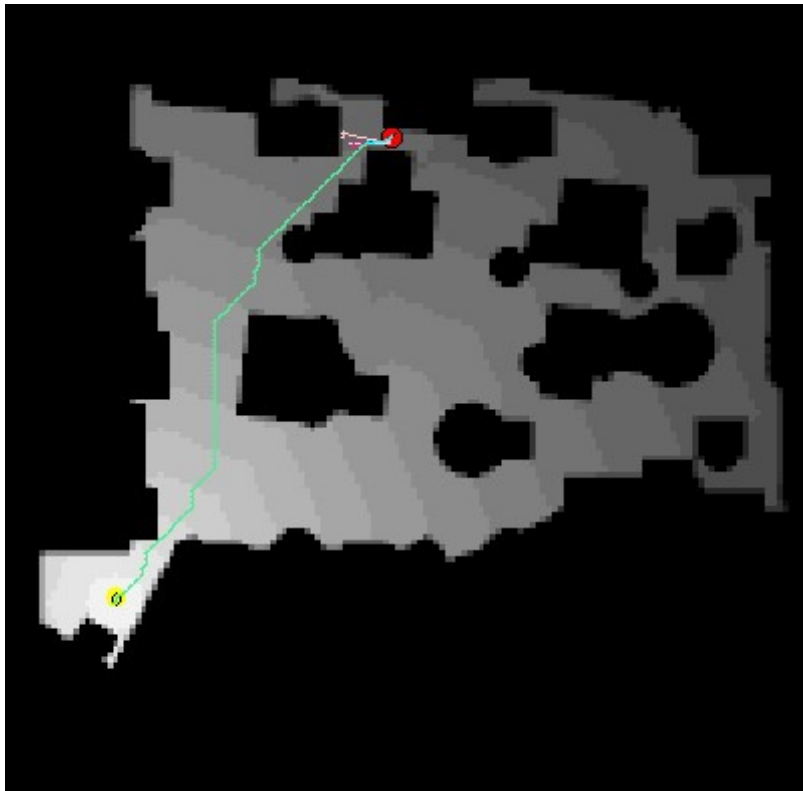
# Value Iteration Example

$$U_{t+1}(i) = R(i) + \max_a \sum_j M_{ij}^a \cdot U_t(j)$$

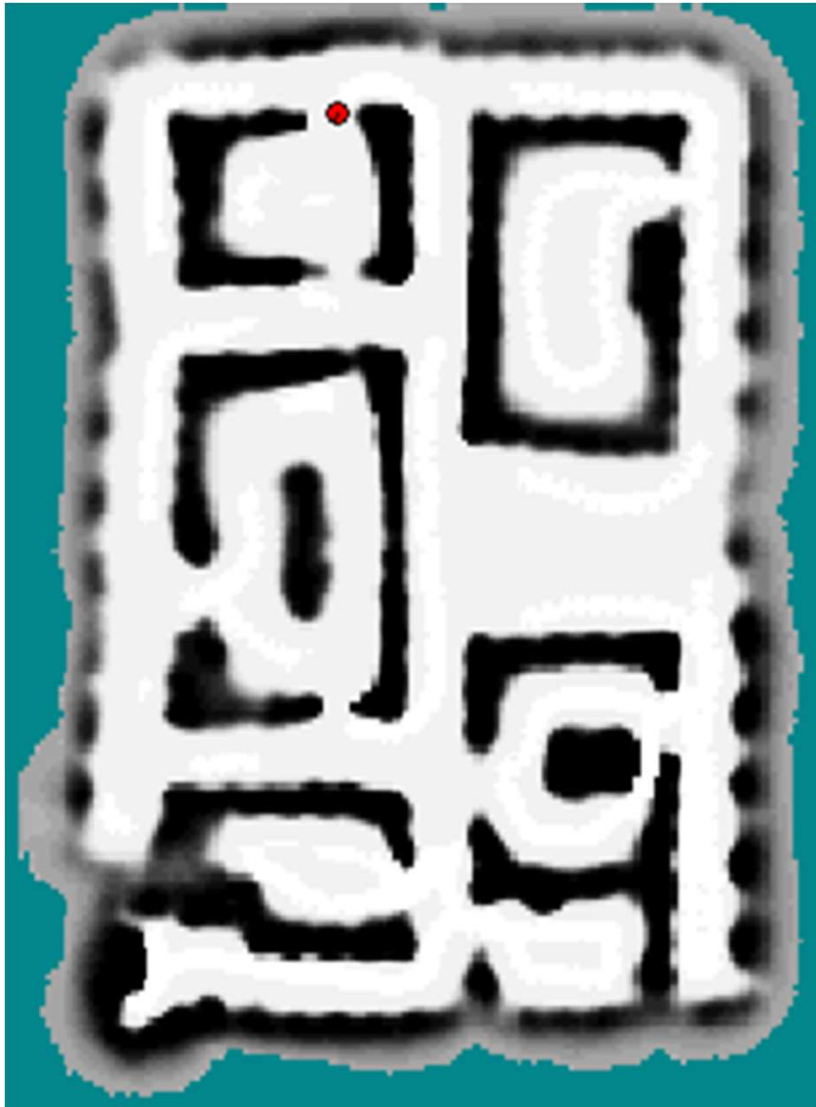


$$\begin{aligned} &= \text{reward} + \max_j \{ \\ &\quad 0.1 \cdot 1 + 0.8 \cdot 5 + 0.1 \cdot 10 \quad (\leftarrow), \\ &\quad 0.1 \cdot 5 + 0.8 \cdot 10 + 0.1 \cdot -8 \quad (\uparrow), \\ &\quad 0.1 \cdot 10 + 0.8 \cdot -8 + 0.1 \cdot 1 \quad (\rightarrow), \\ &\quad 0.1 \cdot -8 + 0.8 \cdot 1 + 0.1 \cdot 5 \quad (\downarrow) \} \\ &= 1 + \max\{5.1 (\leftarrow), 7.7 (\uparrow), \\ &\quad -5.3 (\rightarrow), 0.5 (\downarrow)\} \\ &= 1 + 7.7 \\ &= 8.7 \end{aligned}$$

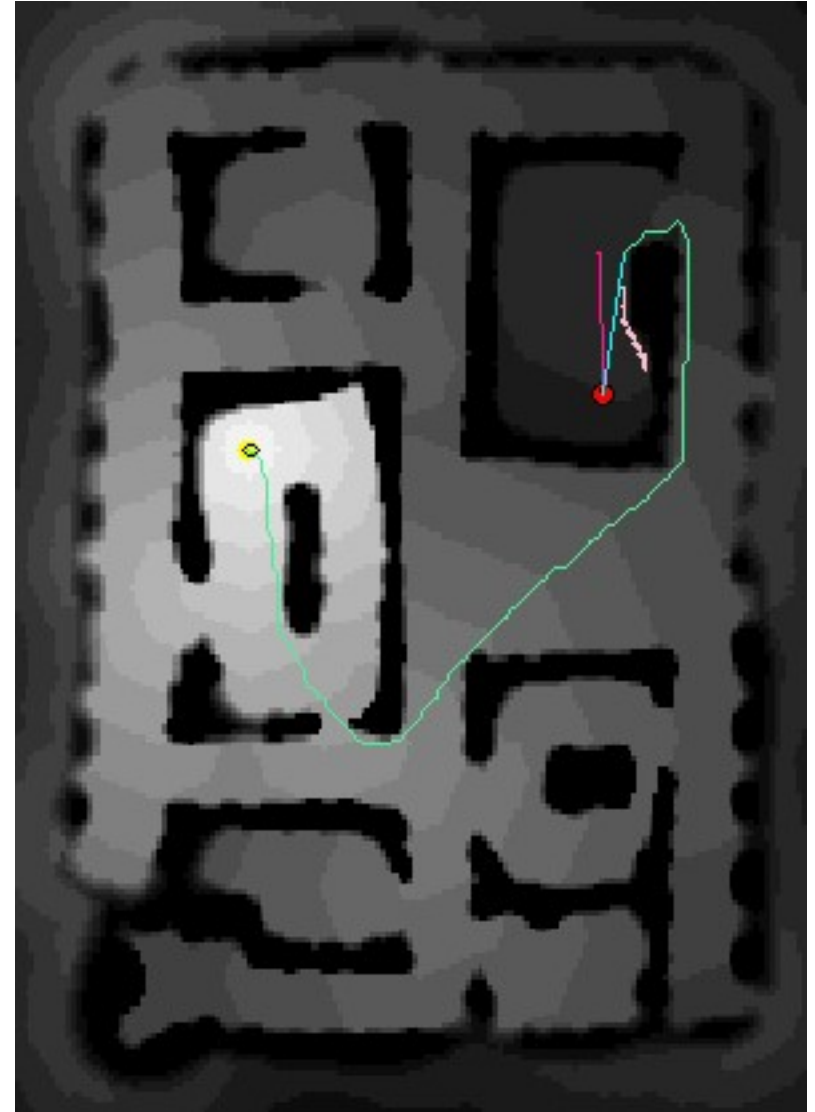
# Value Iteration: Example



# Another Example



*Map*

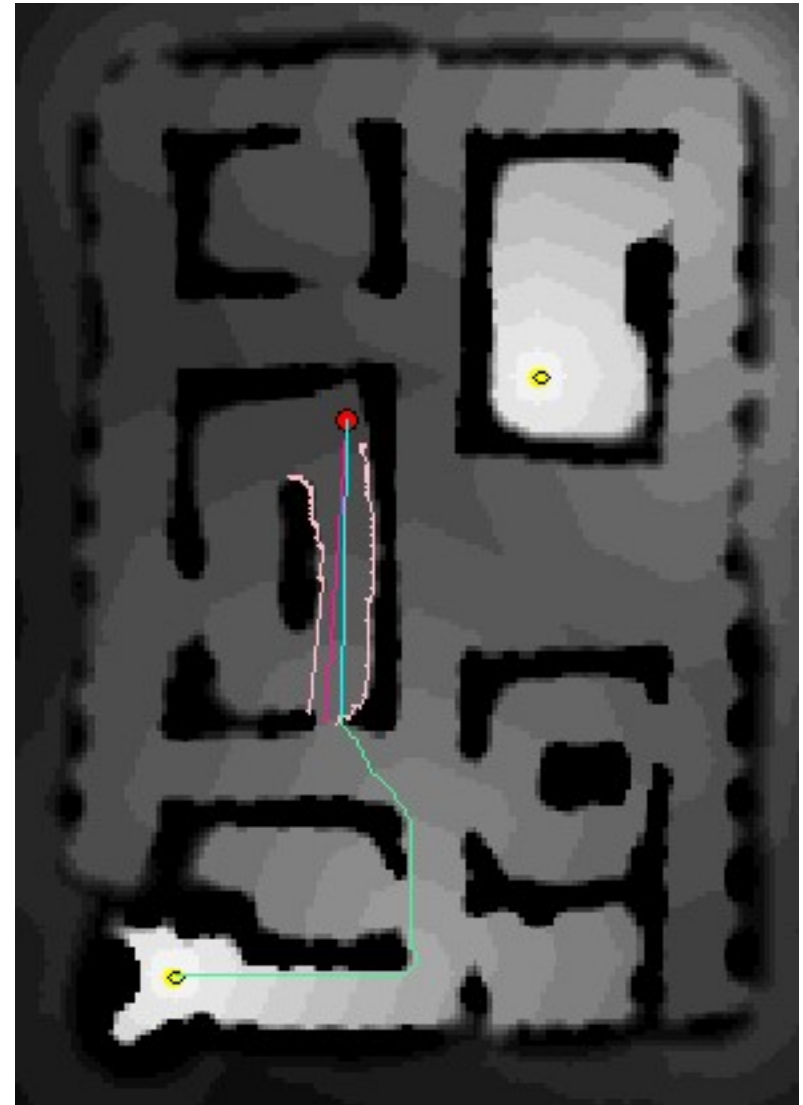


*Value Function and Plan*

# Another Example



*Map*



*Value Function and Plan*

# From Utilities to Policies

- Computes the *optimal utility* function.
- Optimal Policy can easily be computed using the optimal utility values:

$$\text{policy}^*(i) = \operatorname{argmax}_a \sum_j M_{ij}^a \cdot U^*(j)$$

- Value Iteration is an *optimal solution* to the Markov Decision Problem!

# Convergence “close-enough”

- *Different possibilities to detect convergence:*
  - *RMS error – root mean square error*
  - *Policy Loss*
  - *...*



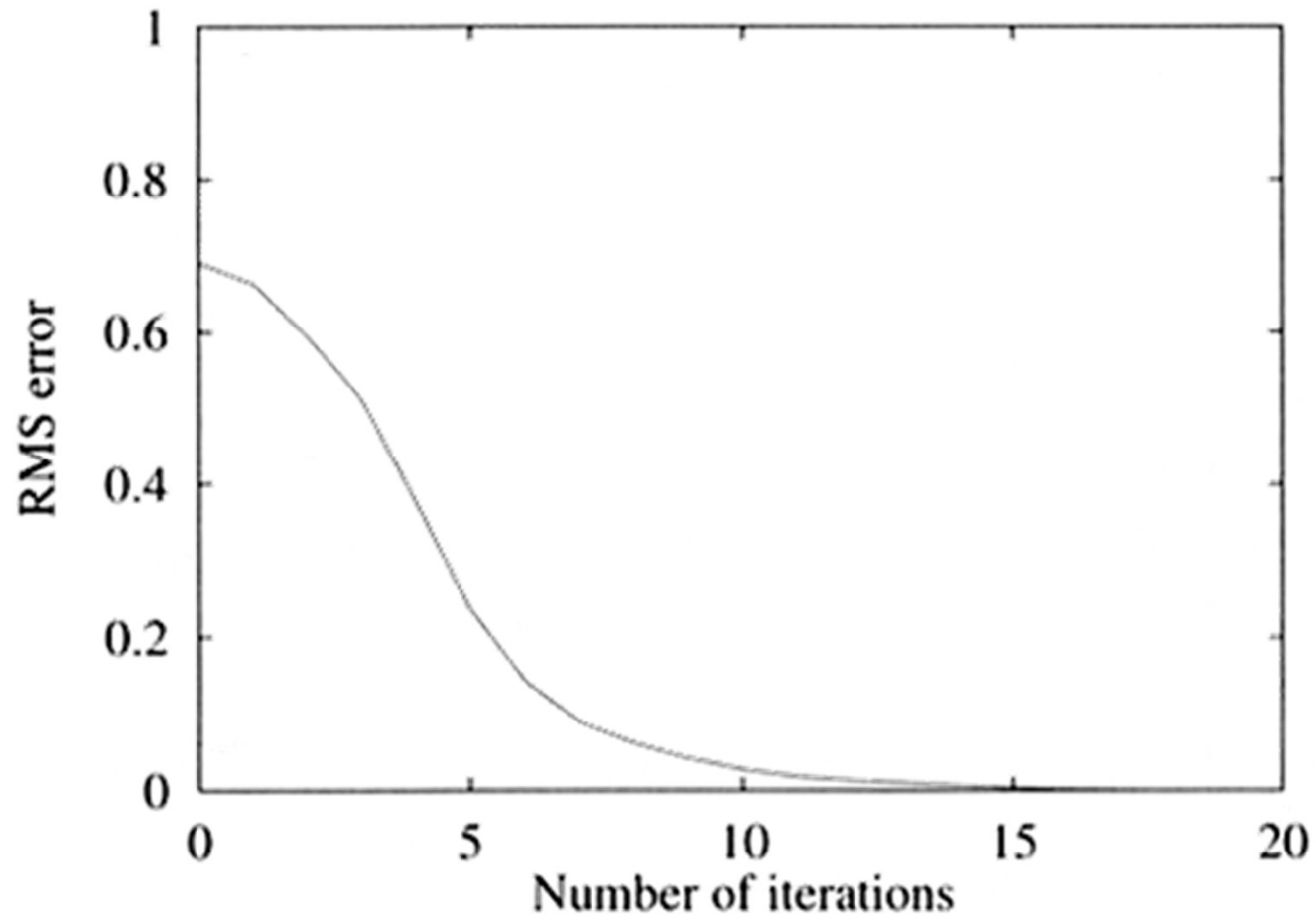
# Convergence-Criteria: RMS

$$RMS = \frac{1}{|S|} \cdot \sqrt{\sum_{i=1}^{|S|} (U(i) - U'(i))^2}$$

- *CLOSE-ENOUGH* ( $U, U'$ ) in the algorithm can be formulated by:

$$RMS(U, U') < \epsilon$$

# Example: RMS-Convergence



# Example: Value Iteration

			+1
			-1

*1. The given environment.*

# Example: Value Iteration

			+1
			-1

*1. The given environment.*

0.812	0.868	0.912	+1
0.762		0.660	-1
0.705	0.655	0.611	0.388

*2. Calculate Utilities.*

# Example: Value Iteration

			+1
			-1

*1. The given environment.*

0.812	0.868	0.912	+1
0.762		0.660	-1
0.705	0.655	0.611	0.388

*2. Calculate Utilities.*

→	→	→	+1
↑		↑	-1
↑	←	←	←

*3. Extract optimal policy.*

# Example: Value Iteration

			+1
			-1

1. *The given environment.*

0.812	0.868	0.912	+1
0.762		0.660	-1
0.705	0.655	0.611	0.388

2. *Calculate Utilities.*

→	→	→	+1
↑		↑	-1
↑	←	←	←

3. *Extract optimal policy.*

			+1
			-1

4. *Execute actions.*

# Example: Value Iteration

0.812	0.868	0.912	+1
0.762		0.660	-1
0.705	0.655	0.611	0.388

*The Utilities.*

→	→	→	+1
↑		↑	-1
↑	←	←	←

*The optimal policy.*

- *(3,2) has higher utility than (2,3). Why does the policy of (3,3) point to the left?*

# Example: Value Iteration

0.812	0.868	0.912	+1
0.762		0.660	-1
0.705	0.655	0.611	0.388

*The Utilities.*

→	→	→	+1
↑		↑	-1
↑	←	←	←

*The optimal policy.*

- *(3,2) has higher utility than (2,3). Why does the policy of (3,3) points to the left?*
- *Because the Policy is **not** the gradient!*

*It is:*

$$policy^*(i) = \operatorname{argmax}_a \sum_j M_{ij}^a \cdot U(j)$$



# Convergence of Policy and Utilities

- *In practice: policy converges faster than the utility values.*
- *After the relation between the utilities are correct, the policy often does not change anymore (because of the argmax).*
- *Is there an algorithm to compute the optimal policy faster?*

# Policy Iteration

- **Idea** for faster convergence of the policy:
  1. Start with one policy.
  2. Calculate utilities based on the current policy.
  3. Update policy based on policy formula.
  4. Repeat Step 2 and 3 until policy is stable.

# The Policy Iteration Algorithm

**function** POLICY-ITERATION( $M, R$ ) **returns** a policy

**inputs:**  $M$ , a transition model

$R$ , a reward function on states

**local variables:**  $U$ , a utility function, initially identical to  $R$

$P$ , a policy, initially optimal with respect to  $U$

**repeat**

$U \leftarrow$  VALUE-DETERMINATION( $P, U, M, R$ )

$unchanged? \leftarrow$  true

**for each** state  $i$  **do**

**if**  $\max_a \sum_j M_{ij}^a U[j] > \sum_j M_{ij}^{P[i]} U[j]$  **then**

$P[i] \leftarrow \arg \max_a \sum_j M_{ij}^a U[j]$

$unchanged? \leftarrow$  false

**end**

**until**  $unchanged?$

**return**  $P$

*Value Determination*

$$U(s_i) = R(s_i) + \sum_j P_{ij}^{\pi(s_i)} U(s_j)$$

$$U'(s_i) \leftarrow R[i] + \sum_j P_{ij}^{\pi(s_i)} U(s_j)$$