MODULE 3: PLANNING UNDER UNCERTAINTY WITH MARKOV DECISION PROCESS

Nicholas Ho, PhD

Institute of System Science, NUS

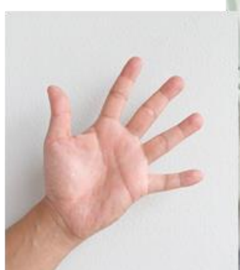
© 2020 National University of Singapore. All Rights Reserved

Interaction



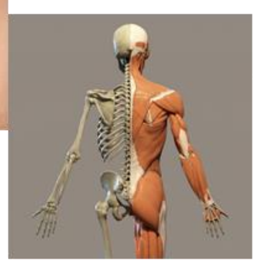












TASK

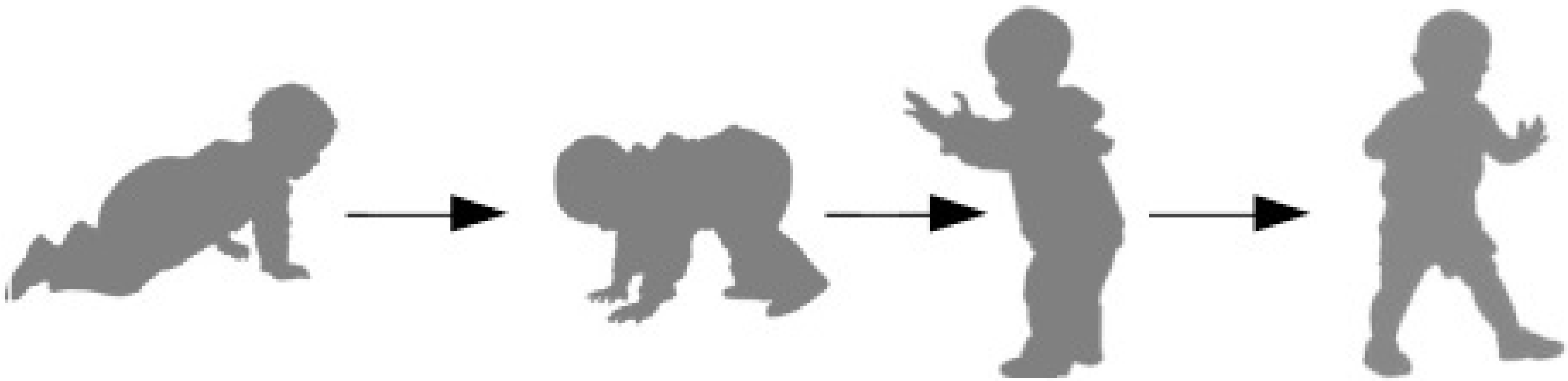
© 2020 National University of Singapore. All Rights Reserved

Interaction

TASK

© 2020 National University of Singapore. All Rights Reserved

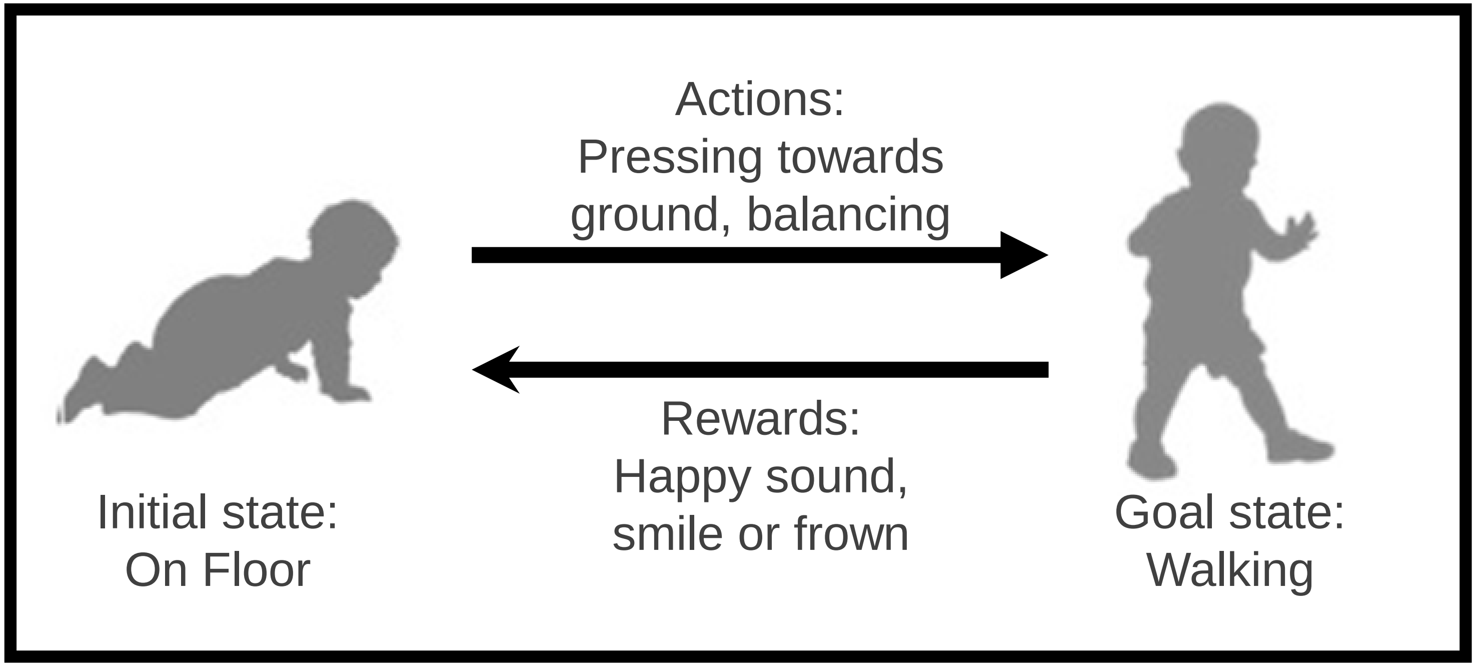
Understanding Reinforcement Learning (RL)



Baby Walking Analogy

© 2020 National University of Singapore. All Rights Reserved

Understanding Reinforcement Learning (RL)



Transition Representation

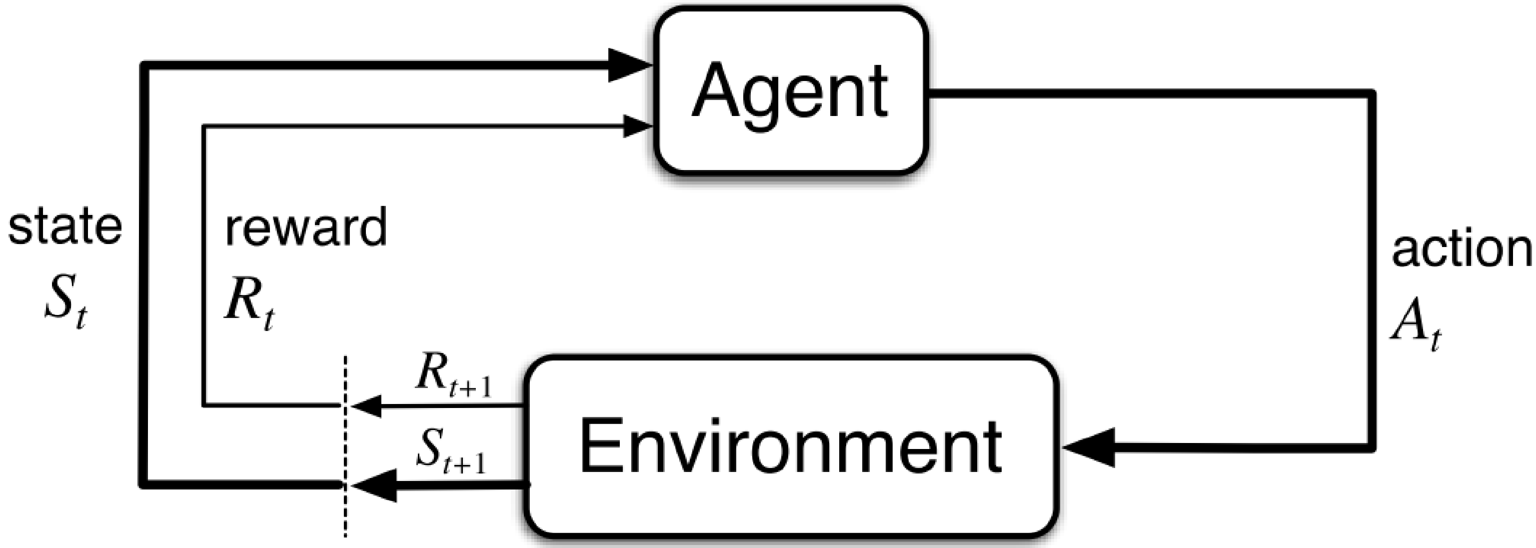
© 2020 National University of Singapore. All Rights Reserved

Explore vs Exploit

- **Explore:** Increase/reduce rewards suddenly (so as to try out more variety of scenarios); Long term and more time consuming
- **Exploit:** Increase rewards steadily (more concern with scenarios with more predictable higher rewards); Short term and more efficient
- Both important – Ideal to balance explore and exploit (e.g. ε-greedy method)

© 2020 National University of Singapore. All Rights Reserved

RECAP on RL



© 2020 National University of Singapore. All Rights Reserved

Many RL platforms

Examples:

- **OpenAI Gym**
  - Commonly used; supports TensorFlow, Theano, Keras
  - https://github.com/openai/gym
- OpenAI Universe
- DeepMind Lab
- ViZDoom
- Project Malmo

© 2020 National University of Singapore. All Rights Reserved

Markov Decision Process (MDP)

- MDP, the framework from which everything else is derived (e.g. POMDP, Q-Learning, Deep Q-Learning)
- In order to solve any RL problem, the problem should be defined or modeled as a MDP
- **Markov Property** – The future is independent of the past, given the present; **the system does not depend on any past historical data and the future only depends on the present data** (more practical unlike conditional probability)

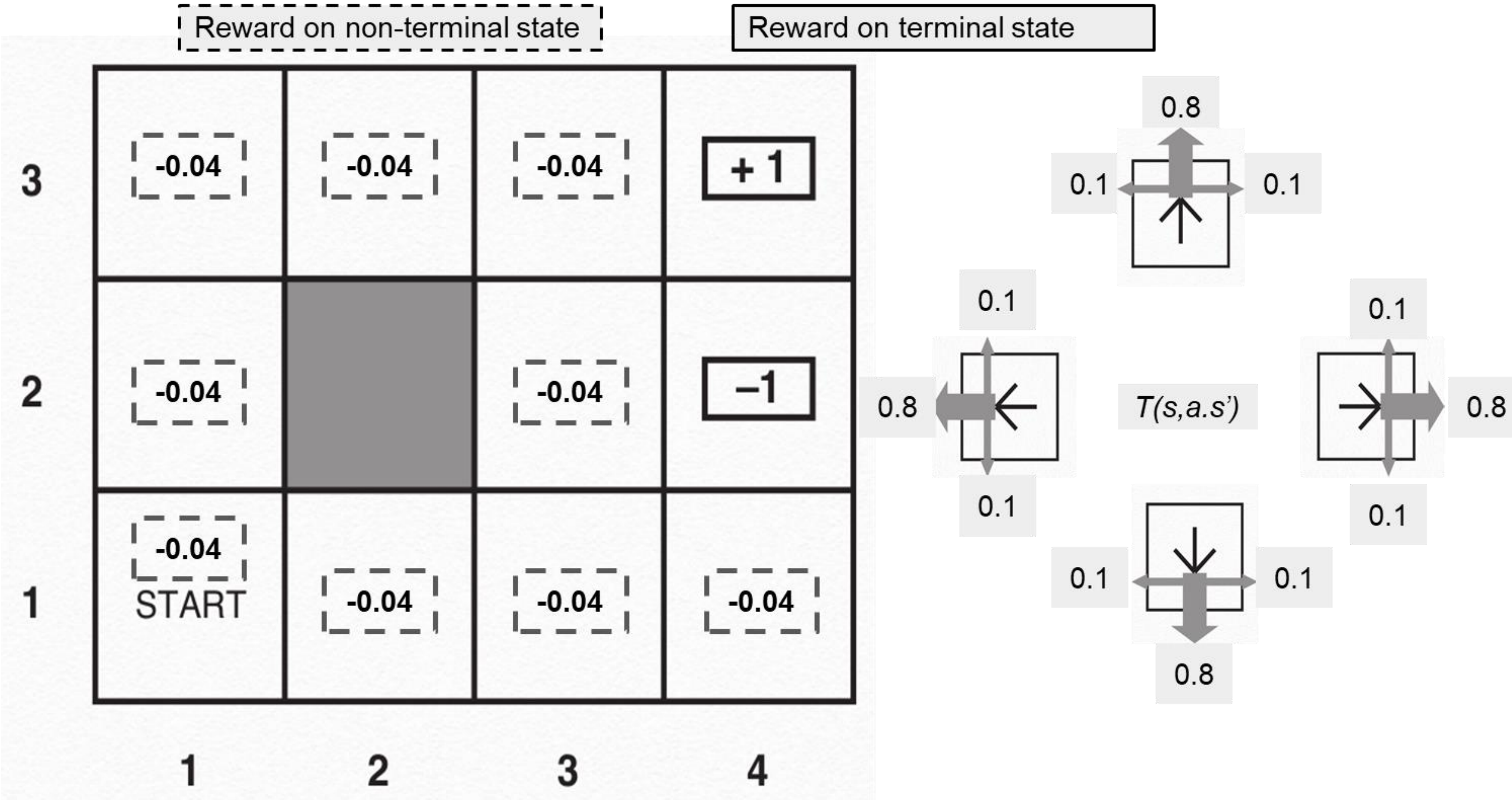
© 2020 National University of Singapore. All Rights Reserved

Markov Decision Process (MDP)

- **A Markov decision process (MDP) consists of four basic elements**
  - ➢ State space: *S* is a set of states
    - ○ E.g., a state *s* may represent the current robot vehicle position
  - ➢ Action space: *A* is a set of actions
    - ○ E.g., an action *a* may represent the speed, acceleration, or steering command for the robot vehicle
  - ➢ $T(s, a, s') = p(s' \mid s, a)$ is a probabilistic state transition function, which accounts for action uncertainty by prescribing a probability distribution for the new state *s'* at time *t+1* if the robot takes action *a* in state *s* at time *t*
  - ➢ $R(s, a)$ is a reward function, which gives the robot a real-valued reward if the robot takes action *a* in state *s*. It allows us to prescribe desirable robot behavior
    - ○ ***R(s)***, *R(s,a,s')* are also often used to define reward functions
- ***** MDPs consider action uncertainty, but still assume perfect sensing**

© 2020 National University of Singapore. All Rights Reserved

MDP Example



The **reward** function **R** and **transition** function **T** remain unchanged for this example unless stated.

© 2020 National University of Singapore. All Rights Reserved

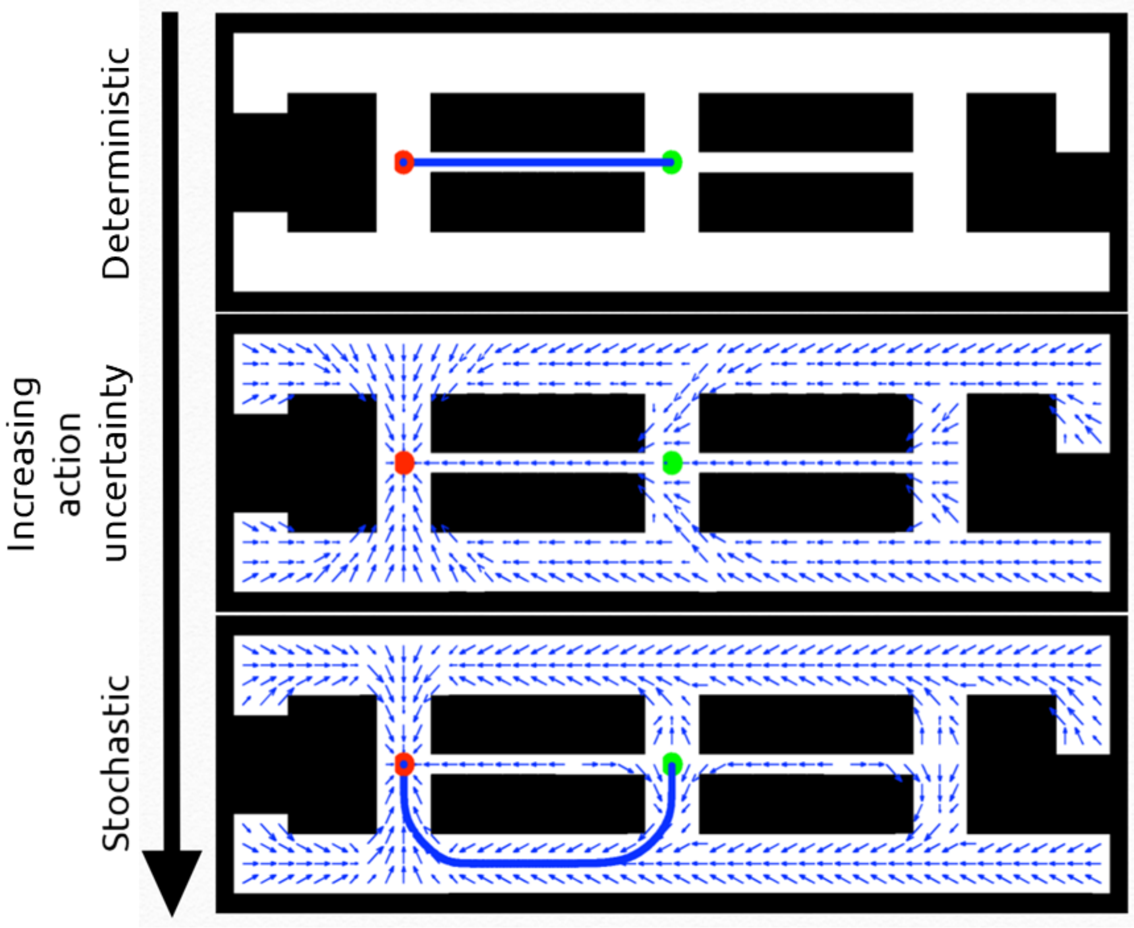
Deterministic vs Stochastic Actions

An *open-loop* plan

$$a_t = \pi(t)$$

A *closed-loop* plan, a.k.a., a *policy*

$$a_t = \pi(s_t)$$

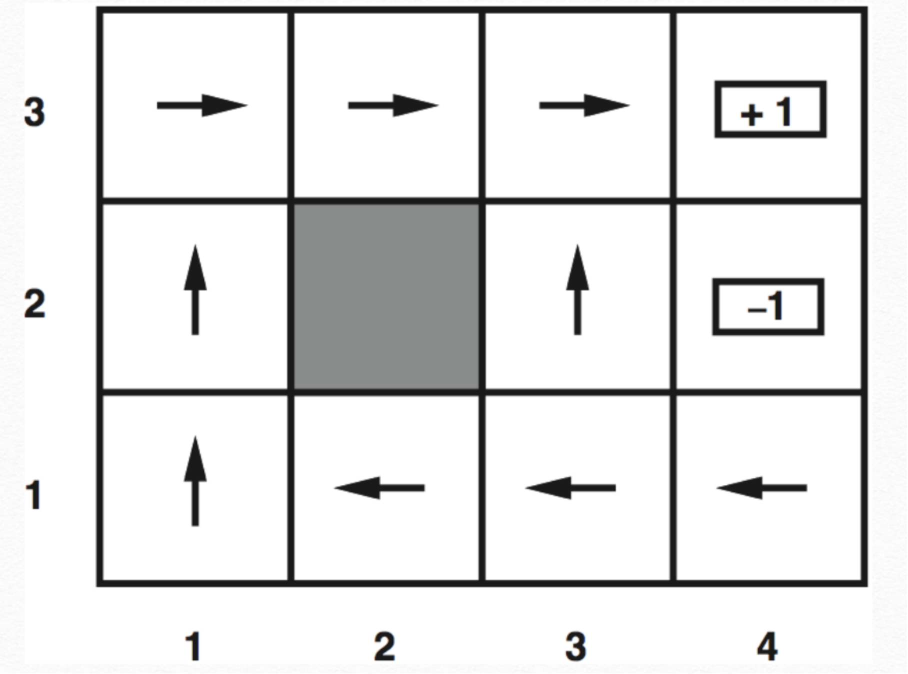


© 2020 National University of Singapore. All Rights Reserved

Solving MDPs

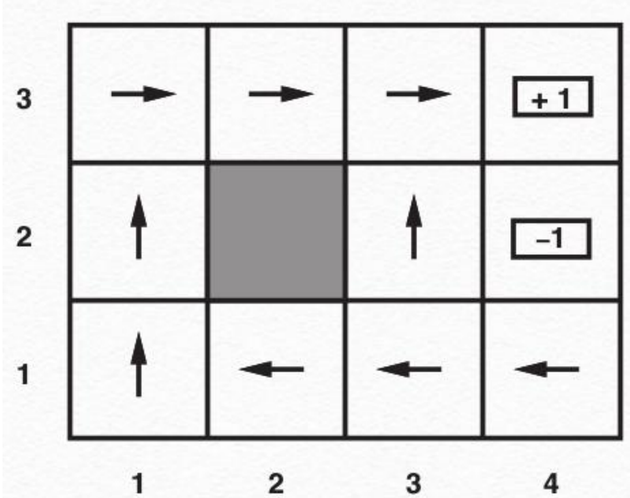
In MDPs, the aim is to find an optimal policy π^* .

Informally, $\pi^*(s)$ produces the "best" action for state *s*, such that π^* maximizes the *value/utility/happiness*.



Balancing Risk and Reward

How to decide on the Reward structure?



$R(s) = -0.04$

?

$R(s) < -1.6284$

?

$-0.4278 < R(s) < -0.0850$

?

$-0.0221 < R(s) < 0$

?

$R(s) > 0$

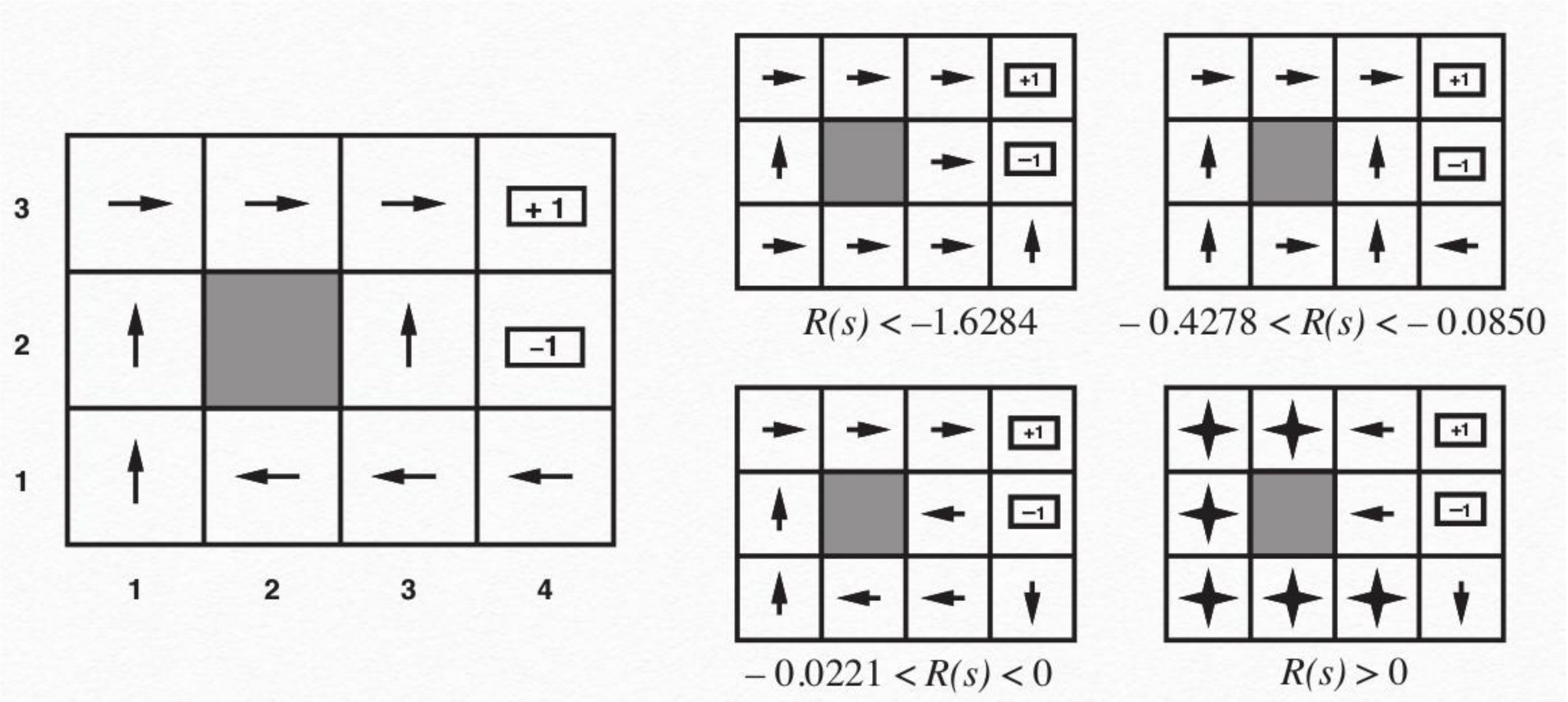
Illustrates the importance of constructing your reward structure

<u>FAQ:</u> How to choose?

© 2020 National University of Singapore. All Rights Reserved

Balancing Risk and Reward

How to decide on the Reward structure?



$R(s) = -0.04$

Illustrates the importance of constructing your reward structure

FAQ: How to choose?

© 2020 National University of Singapore. All Rights Reserved

Value of a sequence of states

Finite-horizon MDP planning: the value of a sequence of *N* visited states is simply the additive rewards

$$V := E[\sum_{t=0}^{N-1} R(s_t)]$$

Limited visited states

Infinite-horizon MDP planning: the value of a sequence of visited states is the discounted total rewards

$$V := E[\sum_{t=0}^{\infty} \gamma^t R(s_t)]$$

Where $\gamma \in [0, 1]$ is a discount factor

Why?

Unlimited visited states
(until it converges or hit terminal states)

© 2020 National University of Singapore. All Rights Reserved

Value of a single state

The <u>value of a state under a policy</u> π is defined to be the expected total discounted rewards when starting in *s* and following π thereafter

$$V^{\pi}(s) := E[\sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi, s_0 = s]$$

Formally, solving MDP is to solve for the optimal policy

$$\pi^* := \arg\max_{\pi} V^{\pi}(s)$$

Reiterate until optimal policy is obtained (corresponds to max Value!)

© 2020 National University of Singapore. All Rights Reserved

Policy vs Value

- **Policy function (π)**
  - Policy = the way in which the agent decides what **action** to perform to transit from the current state
  - The function that maps this transition is called the policy function
- **Value function (V^{π})**
  - Helps the agent determine whether it needs to stay in that particular state or transit to another state
  - Usually the total reward collected by the agent, starting from the initial state

© 2020 National University of Singapore. All Rights Reserved

Bellman Equation

The optimal value function and the optimal policy are stationary and do not depend on the time explicitly.

The optimal value function for an infinite-horizon MDP must satisfy

$$V^*(s) := R(s) + \gamma \max_{a \in A(s)} \sum_{s'} T(s, a, s') V^*(s')$$

The optimal action for each state

$$\boxed{\pi^*(s) := \arg \max_{a \in A(s)} \sum_{s'} T(s, a, s') V^*(s')}$$

The *Bellman equation* is used to find the optimal value function by <u>decomposing the value function into 2 parts: the immediate reward plus the discounted future values</u>

© 2020 National University of Singapore. All Rights Reserved

Bellman Equation

The optimal value function and the optimal policy are stationary and do not depend on the time explicitly.

The optimal value function for an infinite-horizon MDP must satisfy

$$V^*(s) := R(s) + \gamma \max_{a \in A(s)} \sum_{s'} T(s, a, s') V^*(s')$$

The optimal action for each state

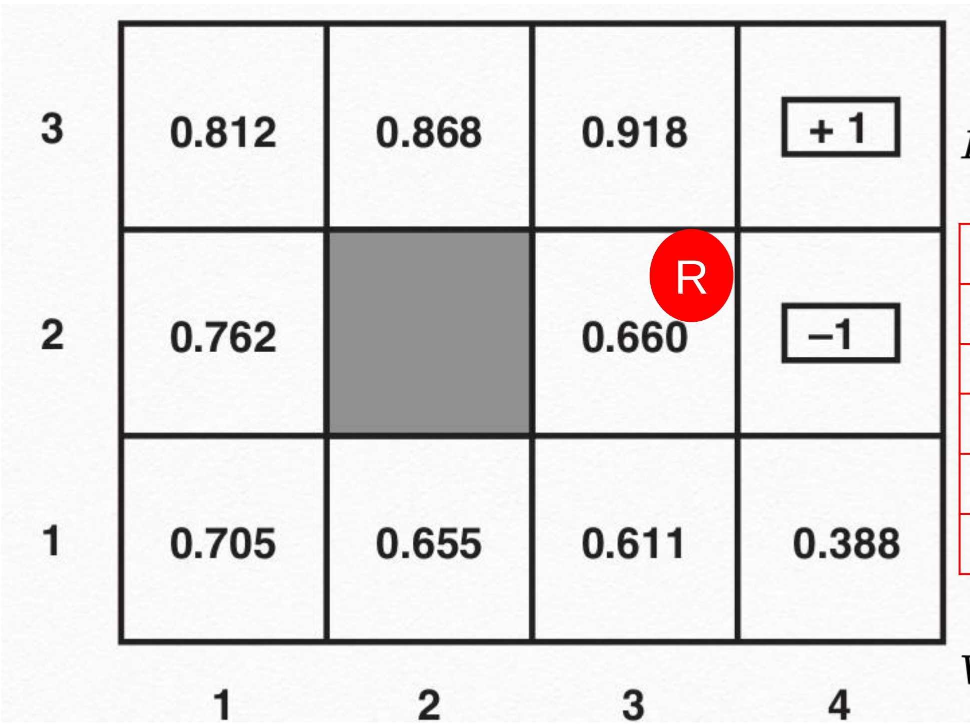
$$\boxed{\pi^*(s) := \arg \max_{a \in A(s)} \sum_{s'} T(s, a, s') V^*(s')}$$

The *Bellman equation* gives us the <u>ability to describe the value of a state s, with the value of the s' state</u>

© 2020 National University of Singapore. All Rights Reserved

Value of all states

$$V^*(s) := R(s) + \gamma \max_{a \in A(s)} \boxed{\sum_{s'} T(s,a,s') V^*(s')}$$



$R(s) = -0.04, \gamma = 1$

	Probability			Value
	0.8	0.1	0.1	
UP				
LEFT				
RIGHT				
DOWN				

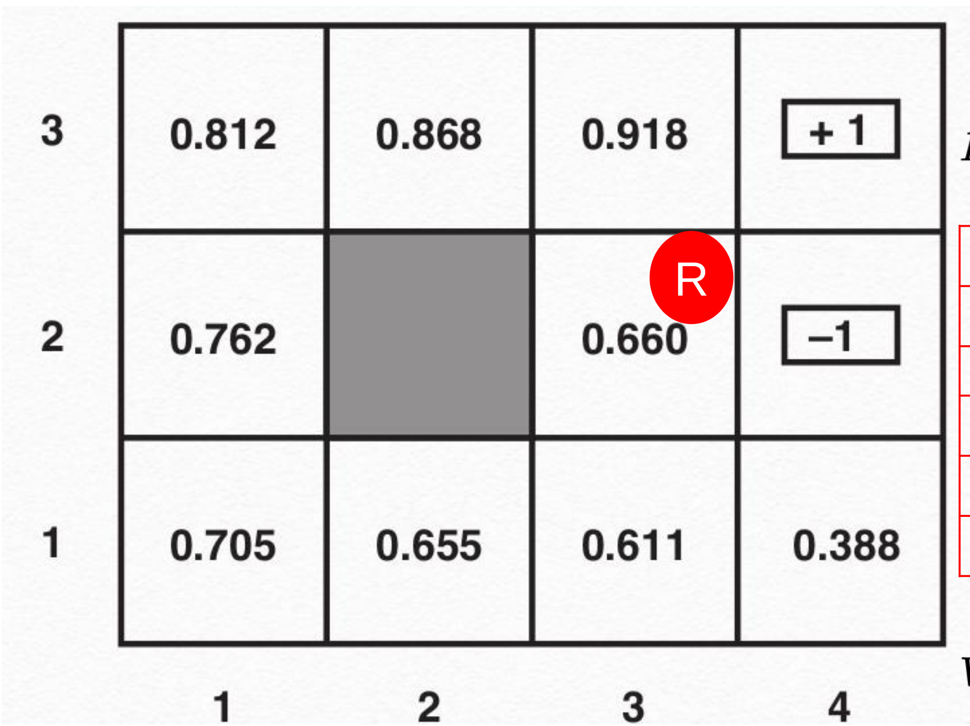
$V^*((3,2)) = -0.04 + 0.7004 = 0.6604$

UP, LEFT, RIGHT, DOWN are referring to the direction of the robot

© 2020 National University of Singapore. All Rights Reserved

Value of all states

$$V^*(s) := R(s) + \gamma \max_{a \in A(s)} \boxed{\sum_{s'} T(s,a,s') V^*(s')}$$



$R(s) = -0.04, \gamma = 1$

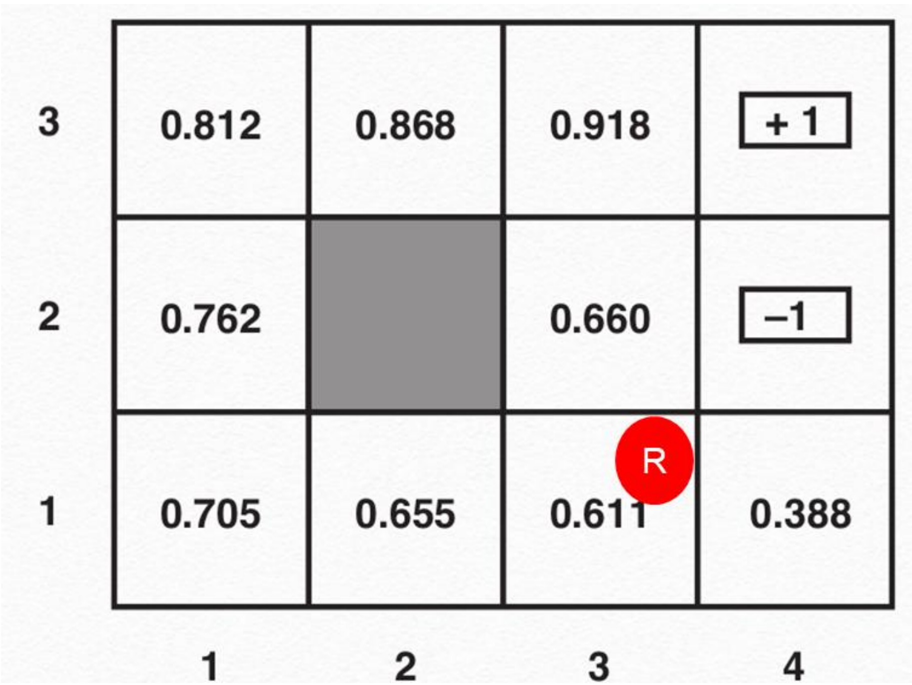
	Probability			Value
	0.8	**0.1**	**0.1**	
UP	0.918	0.66	-1	0.7004
LEFT	0.66	0.611	0.918	0.6809
RIGHT	-1	0.918	0.611	-0.6471
DOWN	0.611	-1	0.66	0.4548

$V^*((3,2)) = -0.04 + 0.7004 = 0.6604$

UP, LEFT, RIGHT, DOWN are referring to the direction of the robot

© 2020 National University of Singapore. All Rights Reserved

Value of all states (Example)



What is $\pi^*((3,1))$?

LEFT? RIGHT? UP? DOWN?
Why?

UP, LEFT, RIGHT, DOWN are referring to the direction of the robot

© 2020 National University of Singapore. All Rights Reserved

Value of all states (Example)

$$\pi^*(s) := \arg\max_{a \in A(s)} \sum_{s'} T(s,a,s') V^*(s')$$

	0.8	0.1	0.1	
UP	0.66	0.655	0.388	0.6323
DOWN	0.611	0.388	0.655	< UP
LEFT	0.655	0.611	0.66	0.6511
RIGHT	0.388	0.66	0.611	< LEFT

$$\pi^*((3,1)) = \text{LEFT}$$

UP, LEFT, RIGHT, DOWN are referring to the direction of the robot

© 2020 National University of Singapore. All Rights Reserved

Value Iteration Algorithm

$$V^*(s) := R(s) + \gamma \max_{a \in A(s)} \sum_{s'} T(s, a, s') V^*(s')$$

Initialize

$$V_0(s) = 0$$

Recurse

$$V_t(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} T(s, a, s') V_{t-1}(s')$$

Until $\|V_t - V_{t-1}\|_\infty \le \epsilon$, where $\|\cdot\|_\infty = \max_{s \in S} |\cdot|$ is the maximum norm.

Reiterate until it converges

© 2020 National University of Singapore. All Rights Reserved

Convergence

Theorem 1. For any two value vectors V_t and V_t', $\|V_{t+1} - V_{t+1}'\|_\infty \leq \gamma \|V_t - V_t'\|_\infty$

Proof. $\|V_{t+1} - V_{t+1}'\|_\infty$

$$
\begin{aligned}
&= \max_s |V_{t+1}(s) - V_{t+1}'(s)| && \text{(max norm)}\\
&= \max_s |\{R(s) + \gamma \max_{a \in A(s)} \sum_{s'} T(s,a,s')V_t(s')\} - \{R(s) + \gamma \max_{a \in A(s)} \sum_{s''} T(s,a,s'')V_t'(s'')\}| && \text{(value function)}\\
&= \gamma \max_s |\max_{a \in A(s)} \sum_{s'} T(s,a,s')V_t(s') - \max_{a \in A(s)} \sum_{s''} T(s,a,s'')V_t'(s'')| && \text{(rearranging)}\\
&\leq \gamma \max_s \max_{a \in A(s)} |\sum_{s'} T(s,a,s')V_t(s') - \sum_{s''} T(s,a,s'')V_t'(s'')|\\
&= \gamma \max_s \max_{a \in A(s)} \sum_{s'} T(s,a,s')|V_t(s') - V_t'(s')| && \text{(reindexing)}\\
&= \gamma \max_s |V_t(s') - V_t'(s')|\{\max_{a \in A(s)} \sum_{s'} T(s,a,s')\} && \text{(reordering)}\\
&= \gamma \max_s |V_t(s') - V_t'(s')| && \text{(identity)}\\
&= \gamma \|V_t - V_t'\|_\infty && \text{(max norm)}
\end{aligned}
$$

© 2020 National University of Singapore. All Rights Reserved

Convergence

Theorem2. If $\|V_{t+1} - V_t\|_\infty \leq \epsilon$ $\|V_t - V^*\|_\infty \leq \frac{\epsilon}{1-\gamma}$

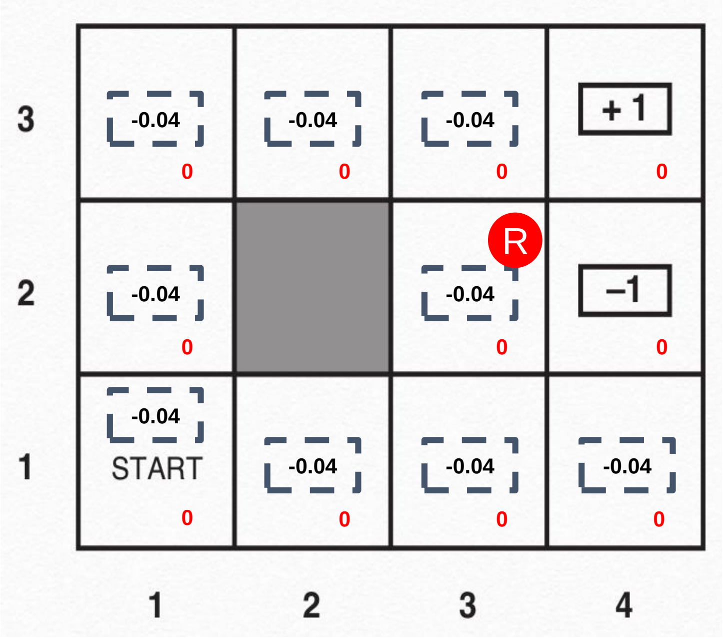
Proof.

$$\begin{aligned} \|V_t - V^*\|_\infty &= \|V_t - V_{t+1} + V_{t+1} - V^*\|_\infty \\ &\leq \|V_t - V_{t+1}\|_\infty + \|V_{t+1} - V^*\|_\infty \\ &\leq \epsilon + \gamma \|V_t - V^*\|_\infty \end{aligned}$$

Rearranging the inequality yields the final result.

© 2020 National University of Singapore. All Rights Reserved

Value Iteration Example (itr=0)



Initialize

$$V_0(s) = 0$$

Recurse

$$R(s) = -0.04, \gamma = 0.9$$

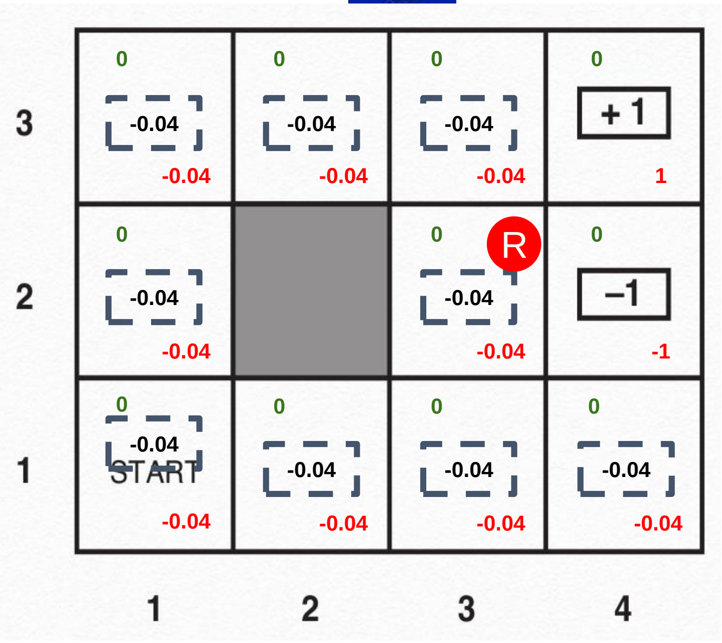
$$V_t(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} T(s, a, s') V_{t-1}(s')$$

Until $\|V_t - V_{t-1}\|_\infty \leq \epsilon$, where $\|\cdot\|_\infty = \max_{s \in S} |\cdot|$ is the maximum norm

© 2020 National University of Singapore. All Rights Reserved

Value Iteration Example (itr=1)

itr=1	Probability			Value	
	0.8	0.1	0.1		
UP	0	0	0	0	<-- MAX
LEFT	0	0	0	0	<-- MAX
RIGHT	0	0	0	0	<-- MAX
DOWN	0	0	0	0	<-- MAX
*V((3,2)) = -0.04 + 0.9 * 0 = -0.04*					



Recurse

$$R(s) = -0.04, \gamma = 0.9$$

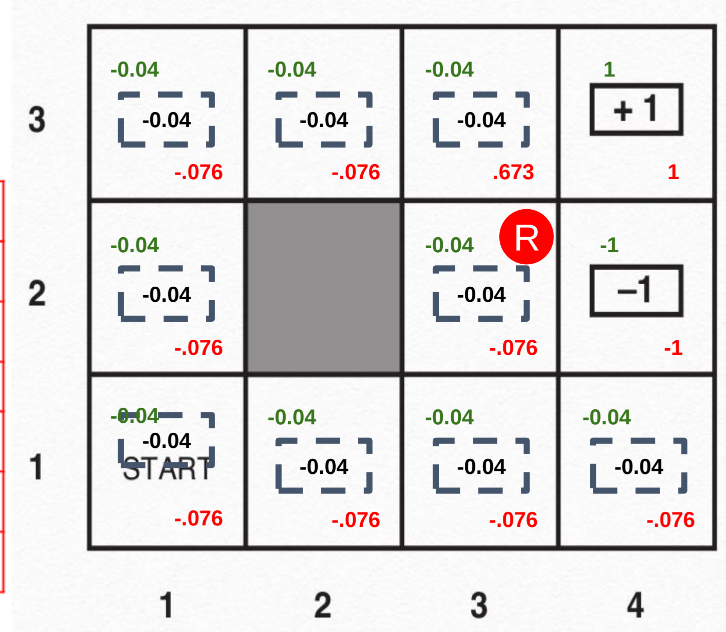
$$V_t(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} T(s, a, s') V_{t-1}(s')$$

Until $\|V_t - V_{t-1}\|_\infty \leq \epsilon$, where $\|\cdot\|_\infty = \max_{s \in S} |\cdot|$ is the maximum norm

© 2020 National University of Singapore. All Rights Reserved

Value Iteration Example (itr=2)

itr=2	Probability			Value	
	0.8	**0.1**	**0.1**		
UP	-0.04	-0.04	-1	-0.136	
LEFT	-0.04	-0.04	-0.04	-0.04	**<-- MAX**
RIGHT	-1	-0.04	-0.04	-0.808	
DOWN	-0.04	-1	-0.04	-0.136	
*V((3,2)) = -0.04 + 0.9 * (-0.04) = -0.076*					



Recurse

$$R(s) = -0.04, \gamma = 0.9$$

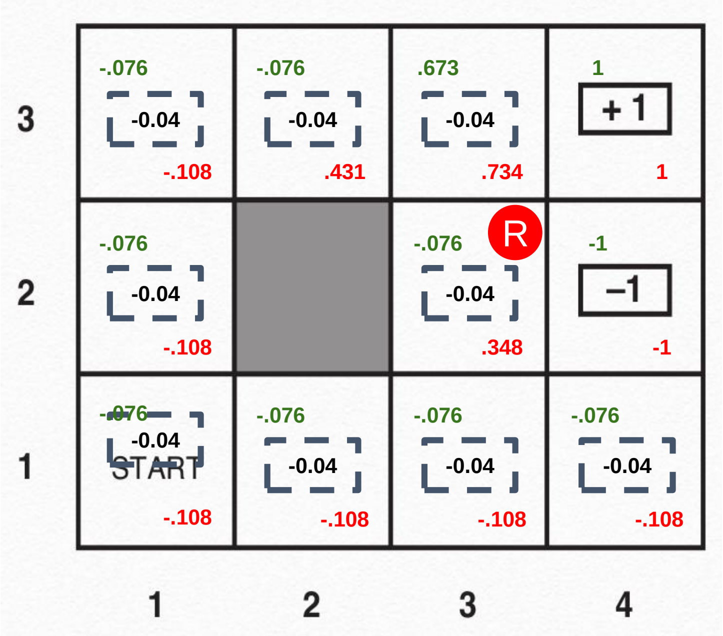
$$V_t(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} T(s, a, s') V_{t-1}(s')$$

Until $\|V_t - V_{t-1}\|_\infty \leq \epsilon$, where $\|\cdot\|_\infty = \max_{s \in S} |\cdot|$ is the maximum norm

© 2020 National University of Singapore. All Rights Reserved

Value Iteration Example (itr=3)

itr=3	Probability			Value	
	0.8	0.1	0.1		
UP	0.6728	-0.076	-1	0.43064	<-- MAX
LEFT	-0.076	-0.076	0.6728	-0.00112	
RIGHT	-1	0.6728	-0.076	-0.74032	
DOWN	-0.076	-1	-0.076	-0.1684	
*V((3,2)) = -0.04 + 0.9 * 0.43064 = 0.347576*					



Recurse

$$R(s) = -0.04, \gamma = 0.9$$

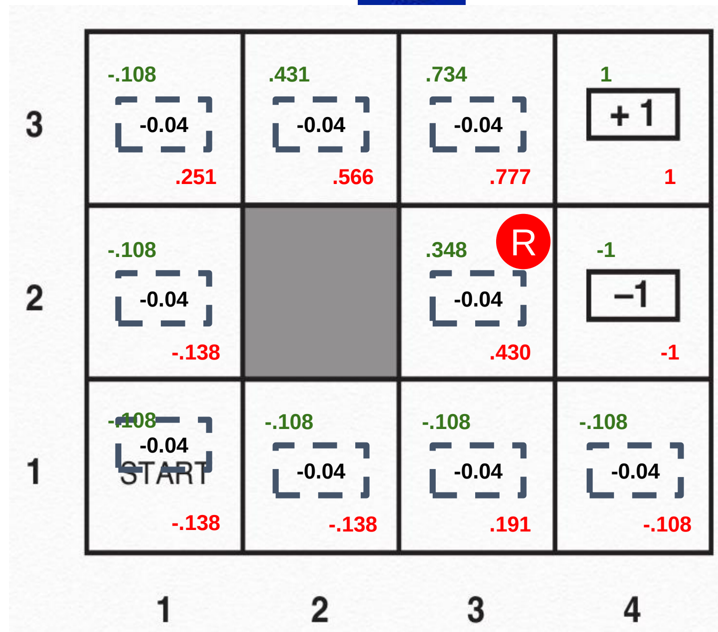
$$V_t(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} T(s, a, s') V_{t-1}(s')$$

Until $\|V_t - V_{t-1}\|_\infty \leq \epsilon$, where $\|\cdot\|_\infty = \max_{s \in S} |\cdot|$ is the maximum norm

© 2020 National University of Singapore. All Rights Reserved

Value Iteration Example (itr=4)

itr=4	Probability			Value	
	0.8	**0.1**	**0.1**		
UP	0.733712	0.347576	-1	0.5217272	**<-- MAX**
LEFT	0.347576	-0.1084	0.733712	0.340592	
RIGHT	-1	0.733712	-0.1084	-0.7374688	
DOWN	-0.1084	-1	0.347576	-0.1519624	
*V((3,2)) = -0.04 + 0.9 * 0.5217272 = 0.42955448*					



Recurse

$$R(s) = -0.04, \gamma = 0.9$$

$$V_t(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} T(s, a, s') V_{t-1}(s')$$

Until $\|V_t - V_{t-1}\|_\infty \leq \epsilon$, where $\|\cdot\|_\infty = \max_{s \in S} |\cdot|$ is the maximum norm

© 2020 National University of Singapore. All Rights Reserved

Value Iteration Example (itr=15)

0.509	0.650	0.795	1.000
0.398		0.486	-1.000
0.296	0.254	0.345	0.130

$R(s) = -0.04, \gamma = 0.9$

	1	2	3	4
3	0.812	0.868	0.918	$\boxed{+1}$
2	0.762		0.660	$\boxed{-1}$
1	0.705	0.655	0.611	0.388

$R(s) = -0.04, \gamma = 1$

High discount factor = more concern about long term rewards
Low discount factor = more concern about short term rewards

© 2020 National University of Singapore. All Rights Reserved

High vs Low Discount Factor

$$V := E[\sum_{t=0}^{\infty} \gamma^t R(s_t)]$$

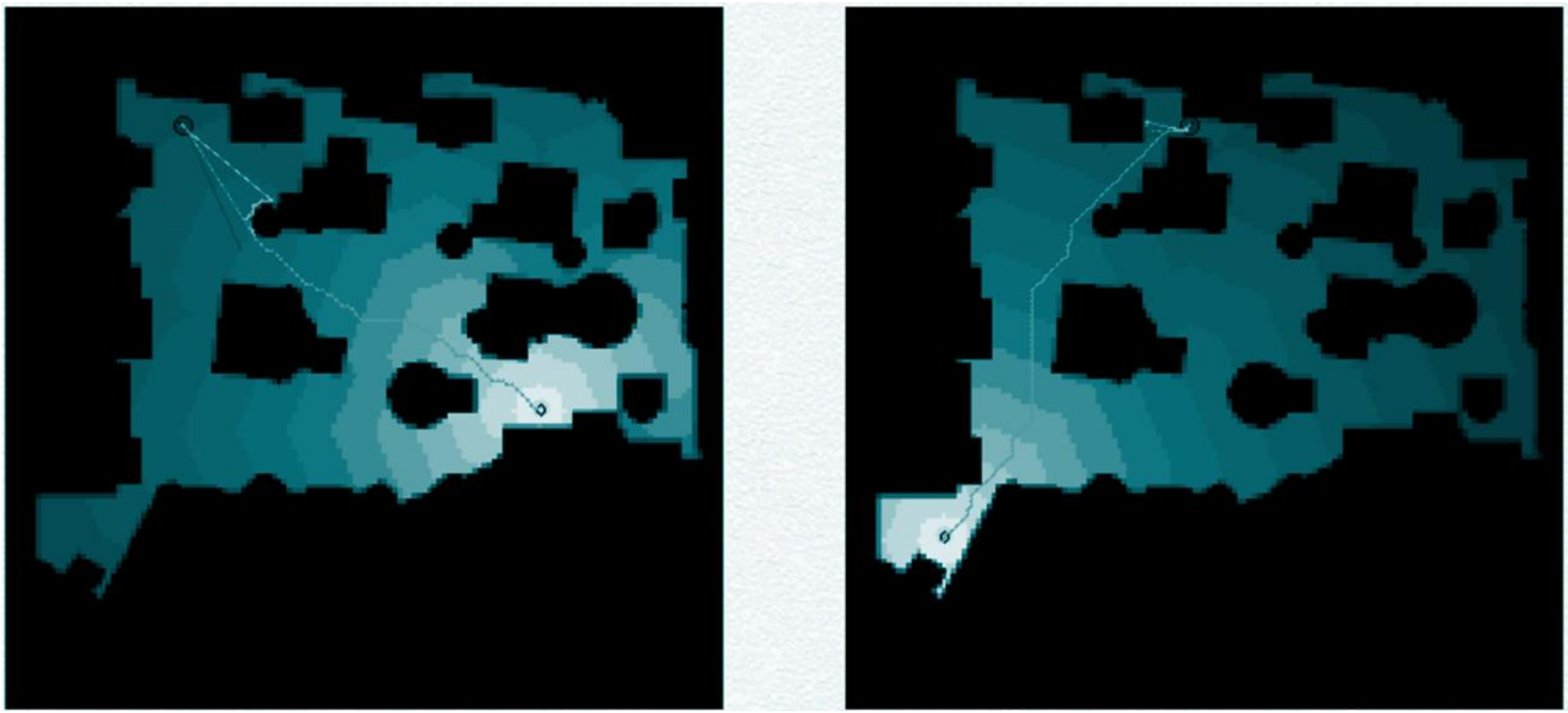
Discount Factor γ	**0.1**	**0.9**	**1**
γ^1 (when t=1)	0.1	0.9	1
γ^{10} (when t=10)	$1x10^{-10}$	0.35	1

High discount factor = more concern about long term rewards
Low discount factor = more concern about short term rewards

Discount factor introduced to ensure higher chance of convergence; rewards that the agent receive can be never-ending for certain scenarios

© 2020 National University of Singapore. All Rights Reserved

Value Iteration for Motion Planning



© 2020 National University of Singapore. All Rights Reserved

“Curse of dimensionality”

Value iteration computes value for *every state* and iterates *many times*

If we discretize a high-dimensional state space *S*, the resulting number of states is exponential in the dimension of *S*. The computational cost is unacceptable

Complexity:

$$O(N \cdot |S| \cdot |A| \cdot |S|)$$

$$V_t(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} T(s,a,s') V_{t-1}(s')$$

© 2020 National University of Singapore. All Rights Reserved

Extra: Policy Iteration Algorithm

Value iteration focuses on the value function. Often we are more interested in the policy

- $\pi_0 \leftarrow$ An arbitrary initial policy
- Repeat until no change in π_t
  - **Policy evaluation**. Given a policy π_t, compute its corresponding value function

$$V_t(s) = E[\sum_{k=0}^{\infty} \gamma^k R(s_k) | \pi_t, s_0 = s]$$

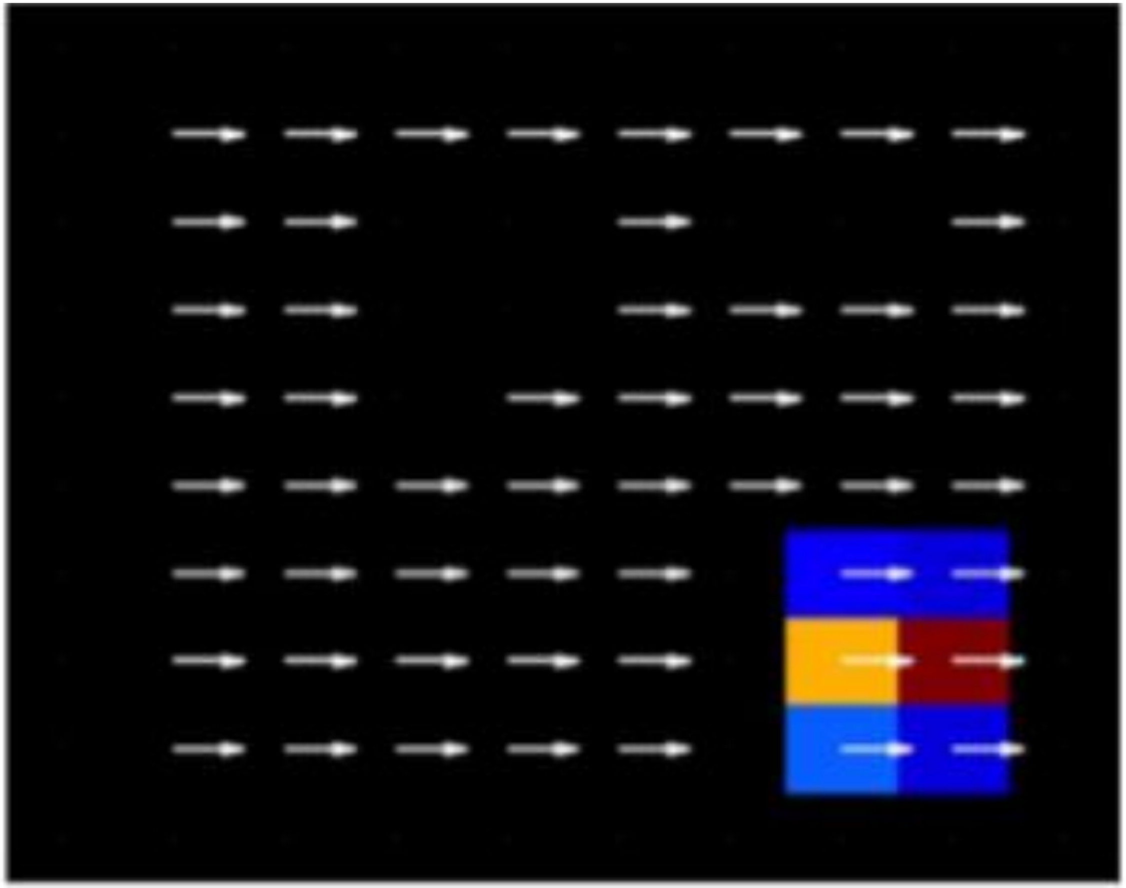
  - **Policy improvement**. Given $V_t(s')$, find a new policy

$$\pi_{t+1}(s) = \arg \max_{a \in A(s)} \sum_{s'} T(s, a, s') V_t(s')$$

- **Value Iteration = Policy determined based on max value**
- **Policy Iteration = Various policies explored until it is clear that there is a policy with max value, which is selected as optimal**

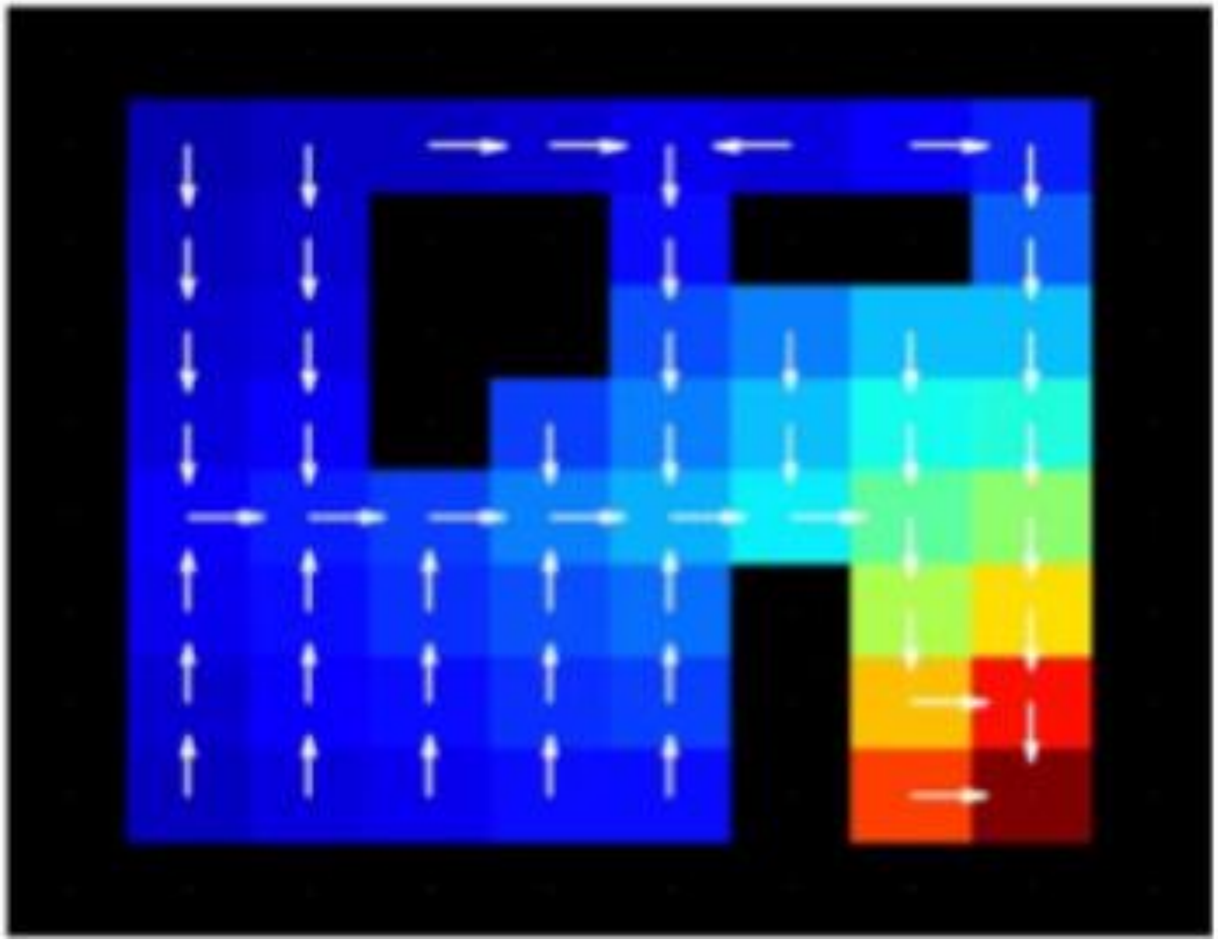
© 2020 National University of Singapore. All Rights Reserved

Extra: Policy Iteration Example (itr=1)



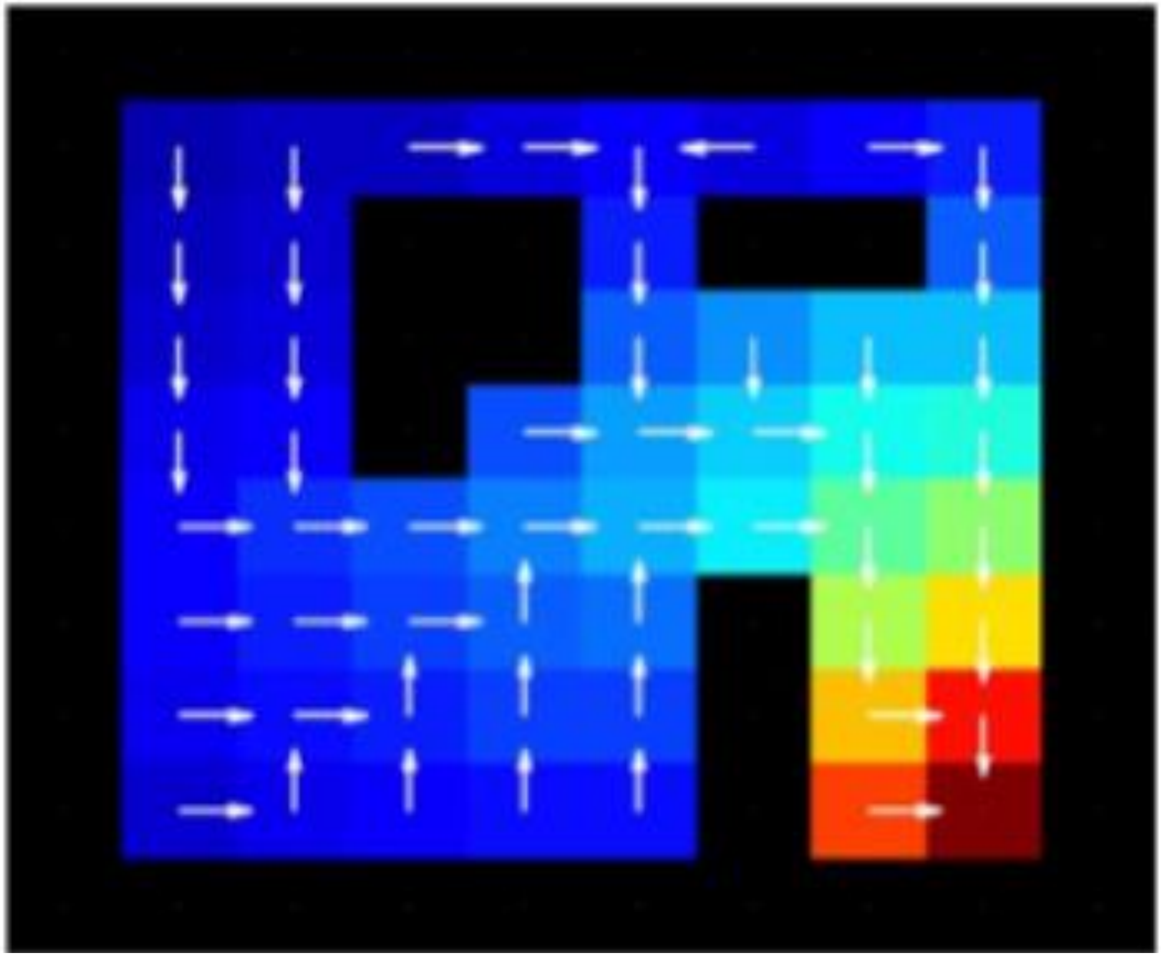
© 2020 National University of Singapore. All Rights Reserved

Extra: Policy Iteration Example (itr=2)



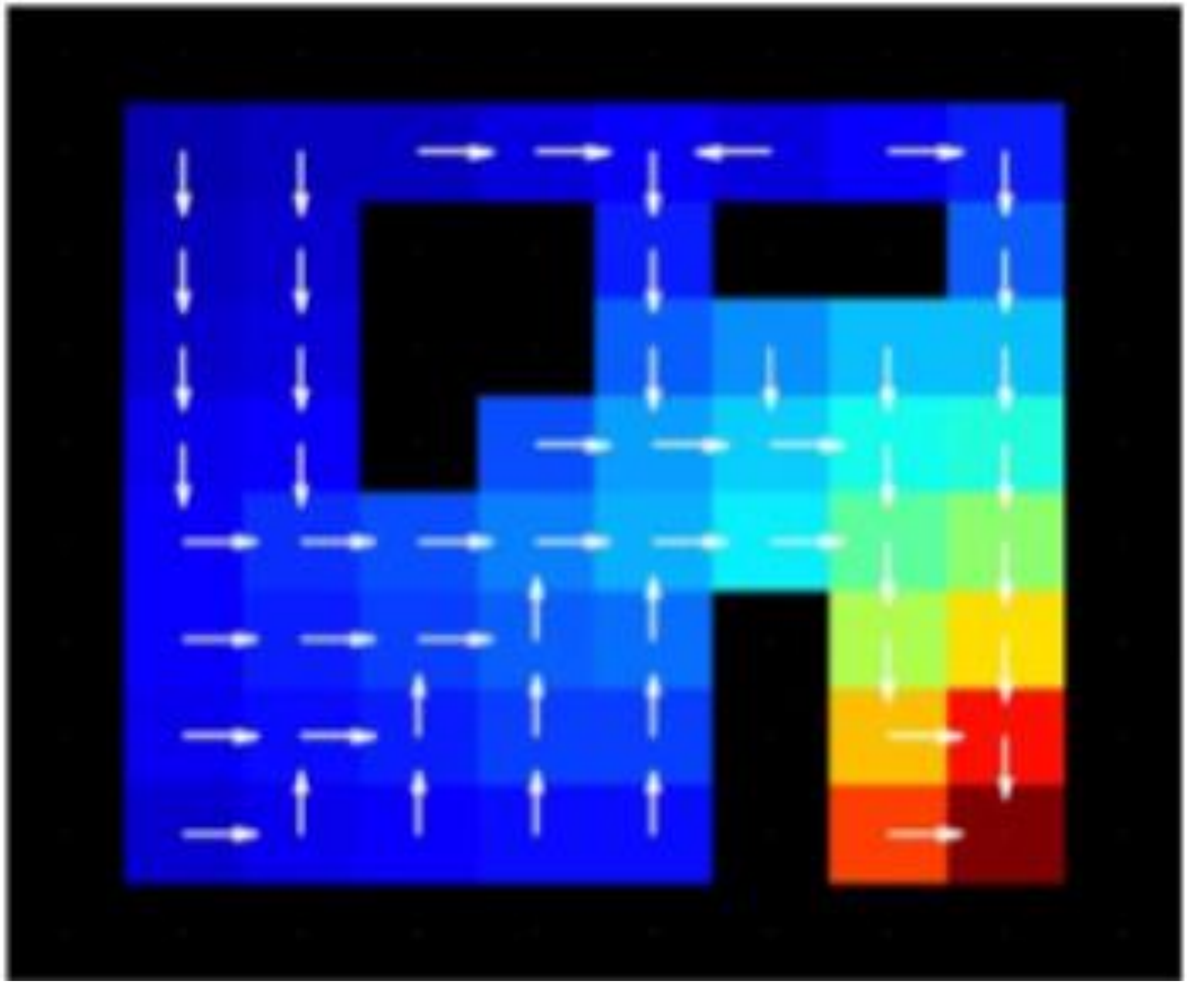
© 2020 National University of Singapore. All Rights Reserved

Extra: Policy Iteration Example (itr=3)



© 2020 National University of Singapore. All Rights Reserved

Extra: Policy Iteration Example (itr=4)



© 2020 National University of Singapore. All Rights Reserved

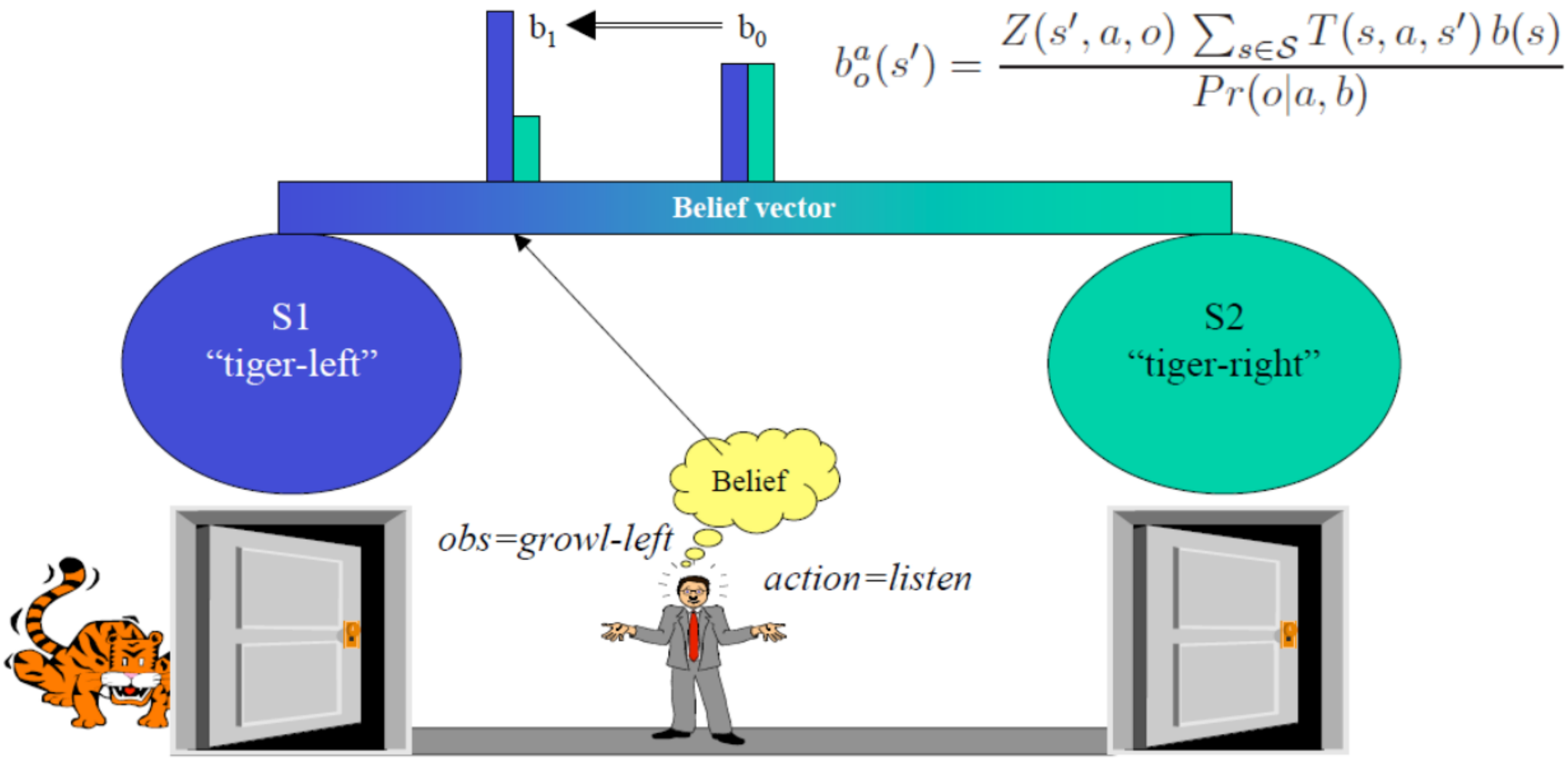
Partially Observable Markov Decision Process

- A (discrete) Partially Observable Markov decision process (POMDP) consists of seven basic elements
  - *S* is a set of **states**
  - *A* is a set of **actions**
  - *Z* is a set of **observations**
  - *T(s, a, s') = p(s' | s, a)* is a probabilistic state **transition** function.
  - *M(a, s', z)=p(z | a, s')* is a probabilistic **observation** function.
  - *R(s)* is a **reward** function
  - *b0* is the probability distribution for the initial state.
- *** MDPs consider action uncertainty, but still assume perfect sensing
- *** POMDPs considers uncertainty in both action and observation

© 2020 National University of Singapore. All Rights Reserved

Partially Observable Markov Decision Process

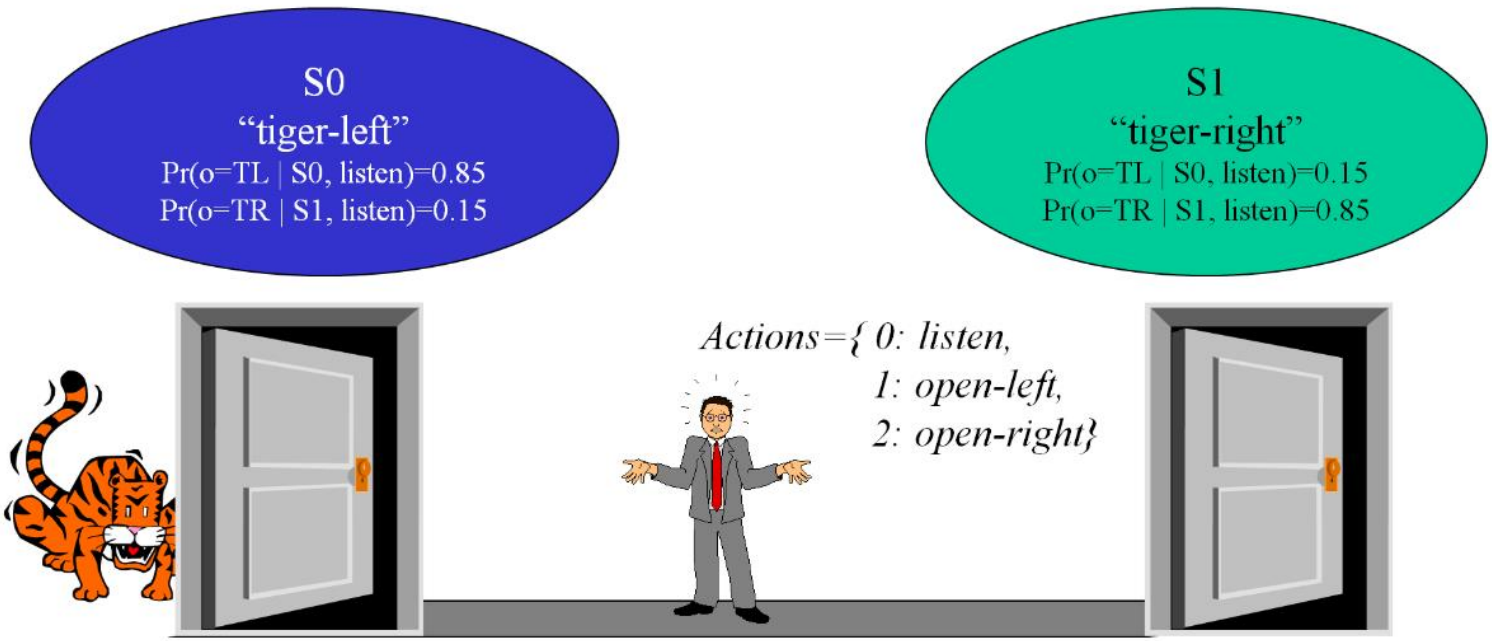
Tiger Example:



© 2020 National University of Singapore. All Rights Reserved

Partially Observable Markov Decision Process

Tiger Example (cont):



Reward Function

- *Penalty for wrong opening: -100*
- *Reward for correct opening: +10*
- *Cost for listening action: -1*

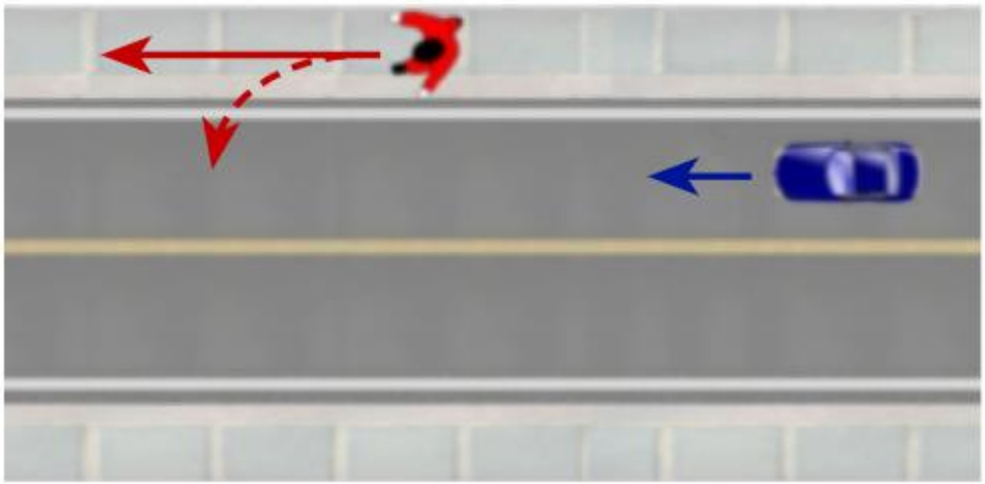
Observations

- *to hear the tiger on the left (TL)*
- *to hear the tiger on the right(TR)*

© 2020 National University of Singapore. All Rights Reserved

Partially Observable Markov Decision Process

Autonomous Vehicle Example:



The robot vehicle approaches a pedestrian on the sidewalk.

© 2020 National University of Singapore. All Rights Reserved

Partially Observable Markov Decision Process: Example 1

Source: https://www.youtube.com/watch?v=VprJZEOD5NE

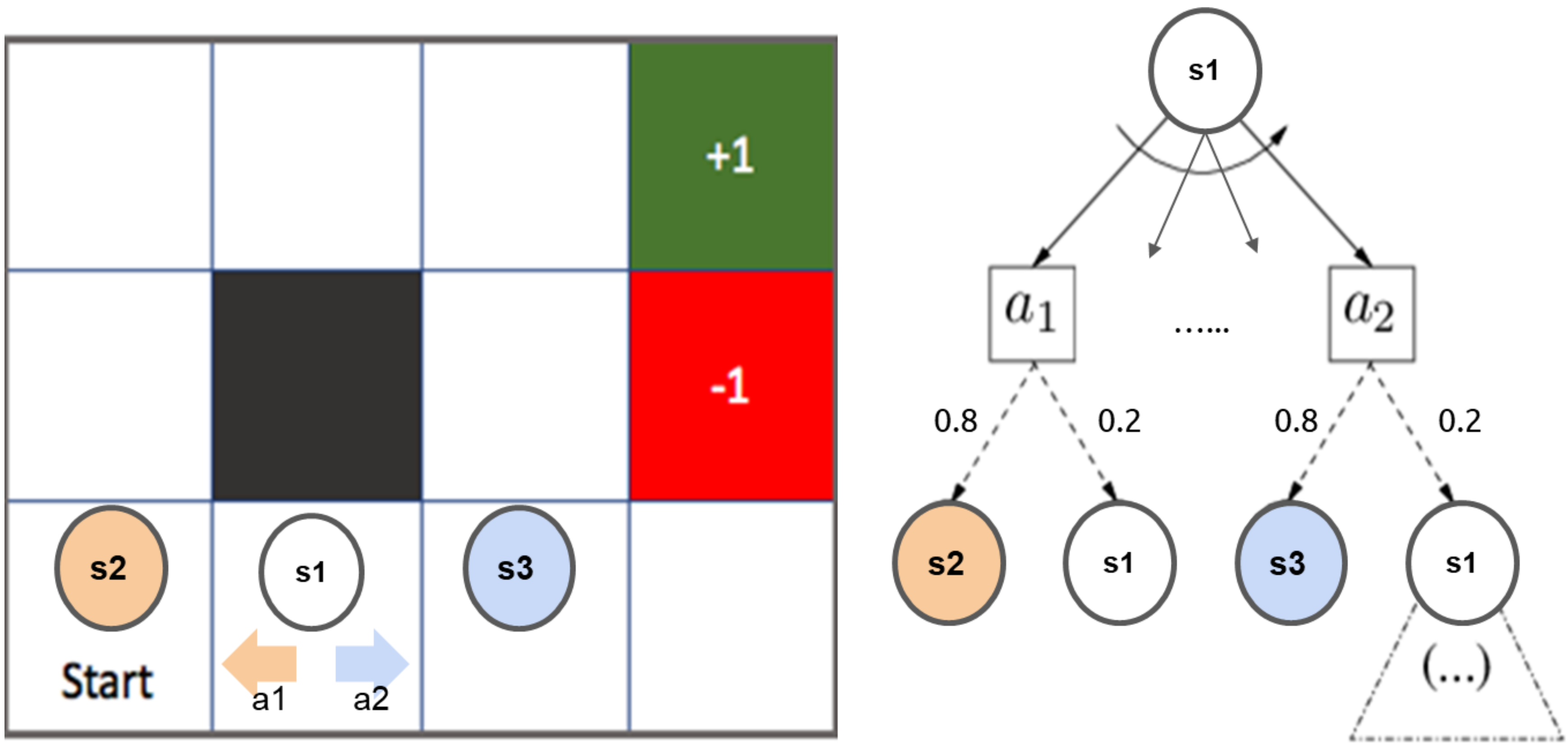
© 2020 National University of Singapore. All Rights Reserved

Partially Observable Markov Decision Process: Example 2

Source: https://www.youtube.com/watch?v=zxb6pLwydfg

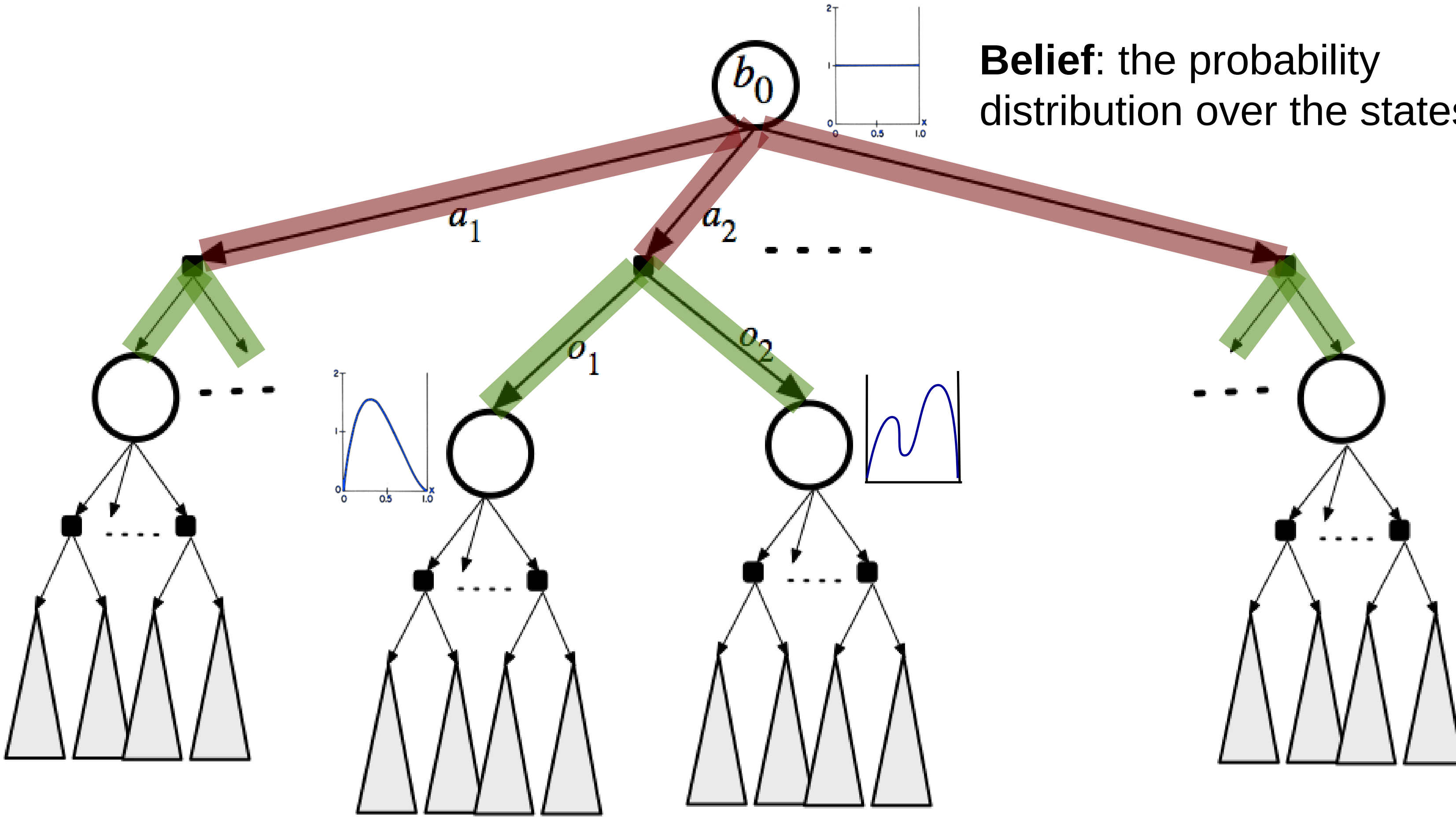
© 2020 National University of Singapore. All Rights Reserved

Recall: Markov Decision Process (MDP)



© 2020 National University of Singapore. All Rights Reserved

Belief Tree Search

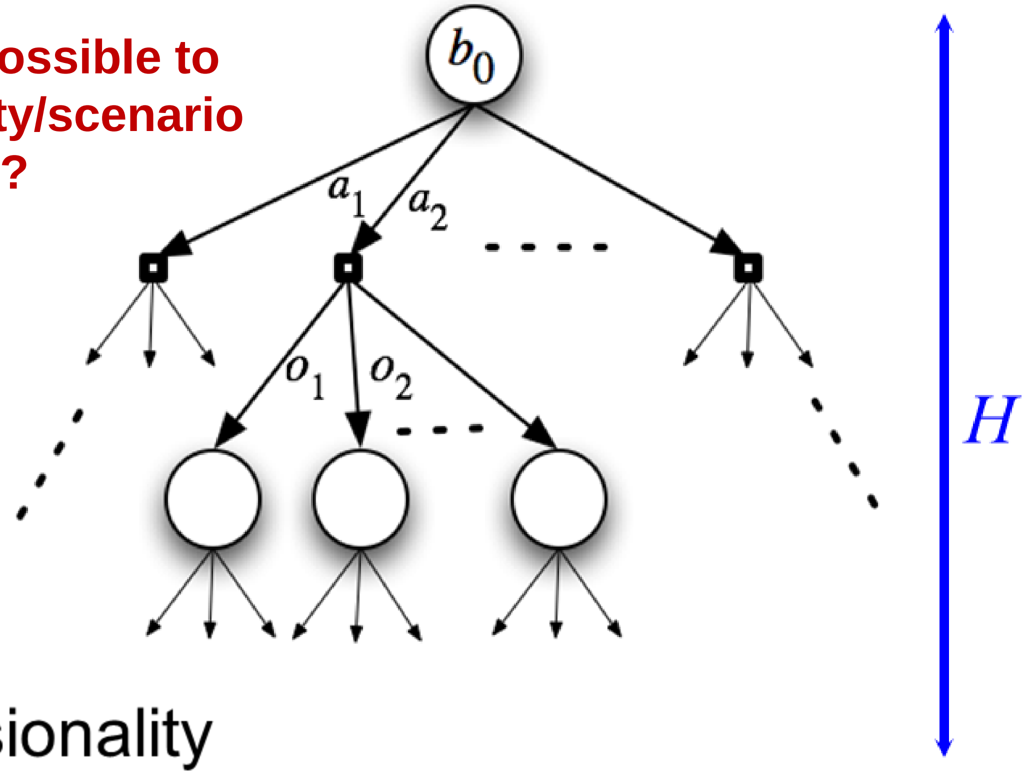


Belief: the probability distribution over the states

© 2020 National University of Singapore. All Rights Reserved

Planning under uncertainty is challenging

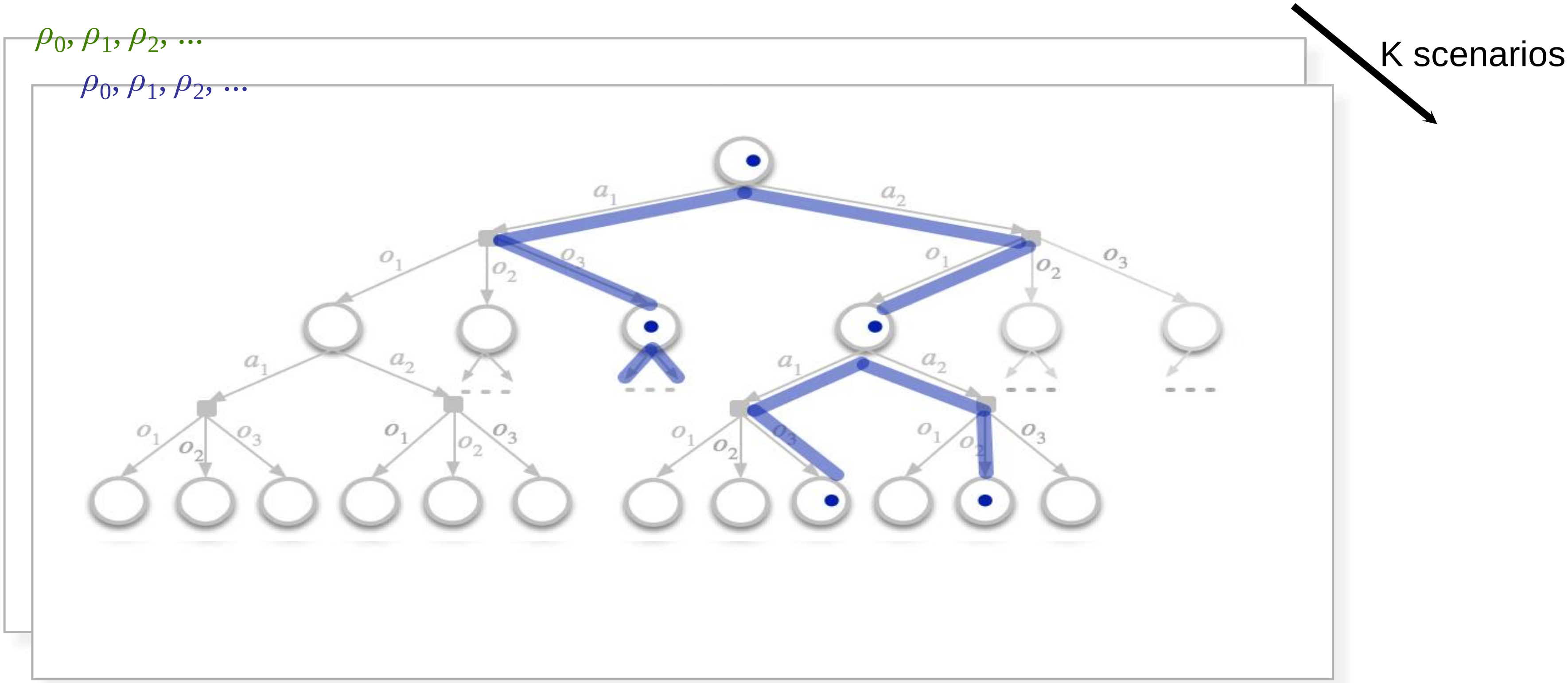
- **Not practical as it is impossible to consider every possibility/scenario**
- **So whats the solution???**



- $O(|A|^H|O|^H)$
- Curse of dimensionality

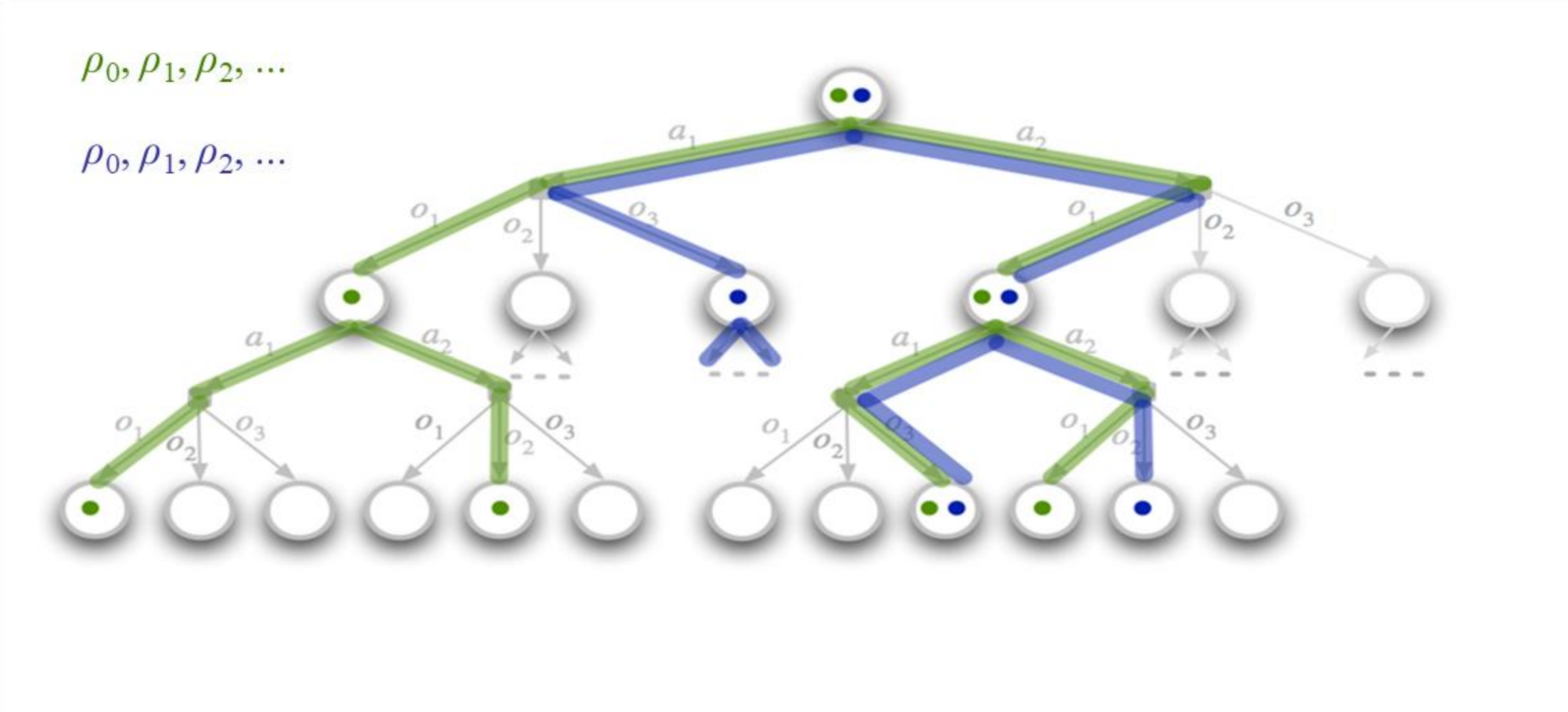
© 2020 National University of Singapore. All Rights Reserved

Dealing with Large Search Tree -- sampling scenarios



© 2020 National University of Singapore. All Rights Reserved

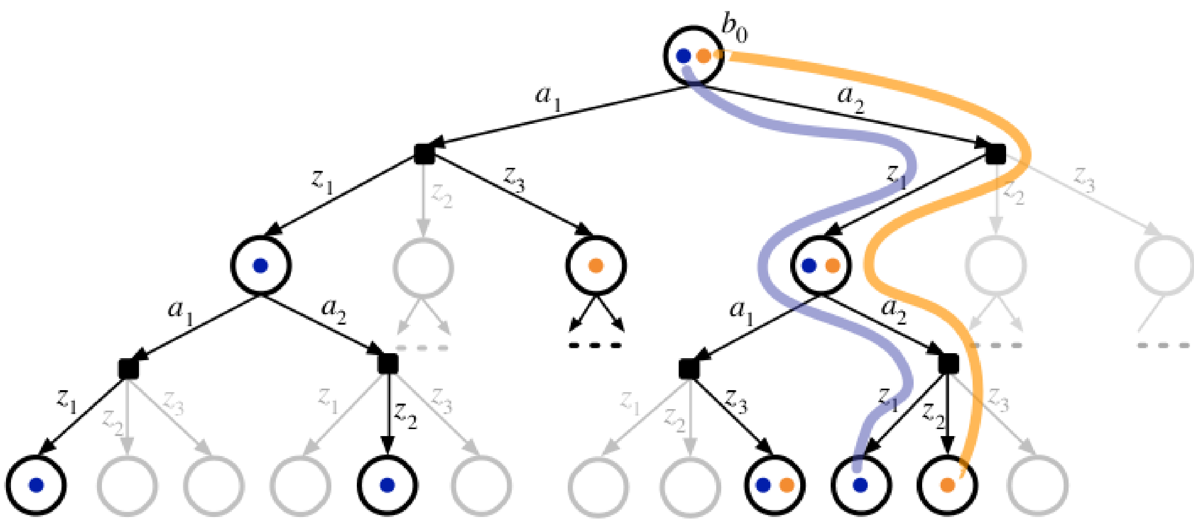
Dealing with Large Search Tree -- sampling scenarios



© 2020 National University of Singapore. All Rights Reserved

DESPOT[1]

Key idea: sample *"scenarios"* that capture uncertainty approximately and compute an optimal or near-optimal policy under sampled scenarios



DESPOT with 2 scenarios with height 2

Full belief tree >> DESPOT ≈ Deterministic planning

$O(|A|^H|O|^H)$ $O(|A|^H K)$ $O(|A|^H)$

[1] Somani, Adhiraj, et al. "DESPOT: Online POMDP planning with regularization." *Advances in neural information processing systems*. 2013.

© 2020 National University of Singapore. All Rights Reserved

POMDP for Autonomous Driving in NUS

Source: http://www.youtube.com/watch?v=v2-YLTtxYIU

© 2020 National University of Singapore. All Rights Reserved

POMDP for Autonomous Driving in NUS

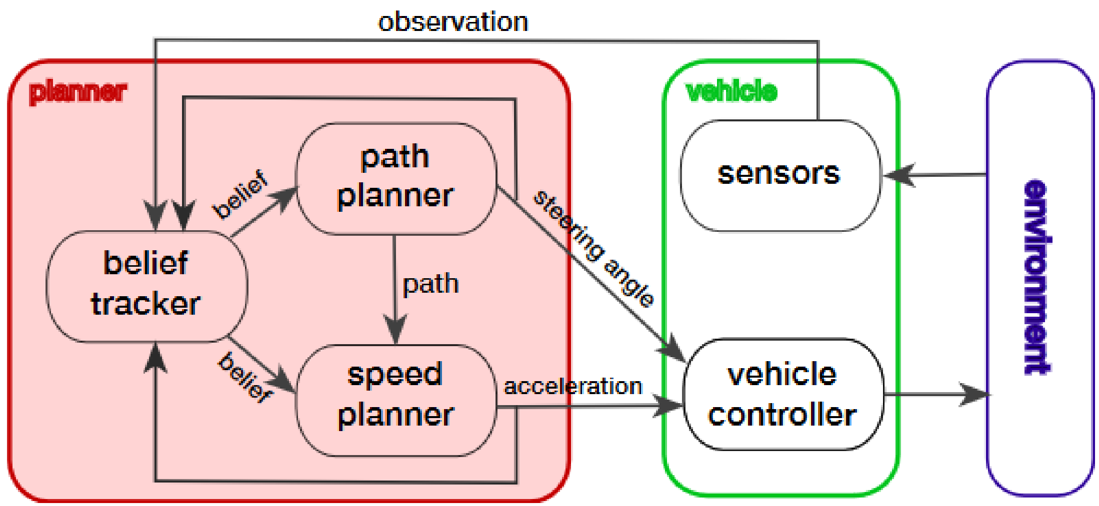


Fig. 3. Two-level POMDP-based planning for autonomous driving.

- The belief tracker maintains a belief over system states
- At each time step, it performs updates to incorporate new information from vehicle actions and sensor data into the belief
- The path planner then plans a minimum-cost path for the current belief and outputs the vehicle steering angle for the current step. Given the path and the current belief, the POMDP speed planner computes a conditional plan

https://www.comp.nus.edu.sg/~leews/publications/bai2015intention.pdf

© 2020 National University of Singapore. All Rights Reserved

POMDP for Autonomous Driving in NUS

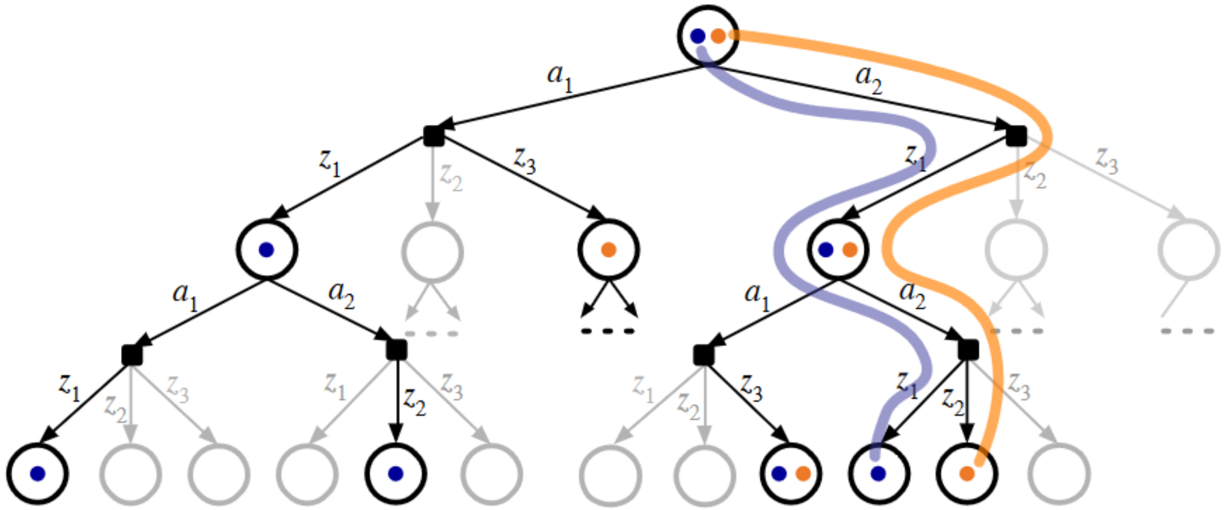
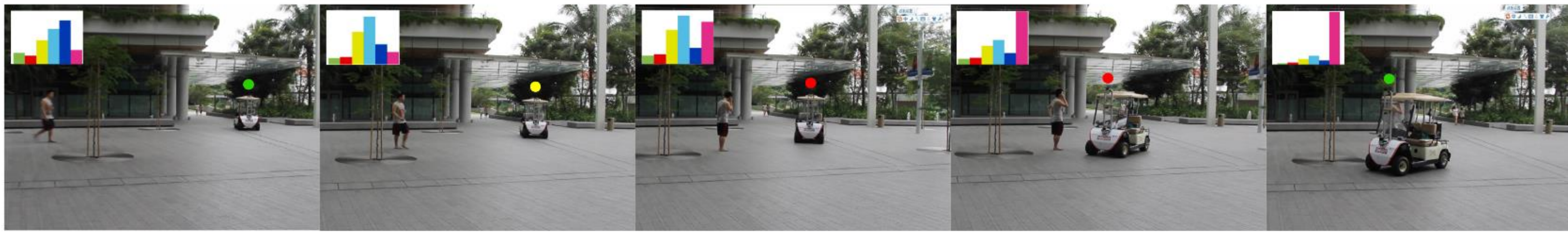


Fig. 4. A belief tree of height $H = 2$ (gray) and a corresponding DESPOT tree (black) obtained with 2 sampled scenarios, shown in blue and orange. The blue and orange curves indicate the execution paths of a same policy under the two scenarios.

https://www.comp.nus.edu.sg/~leews/publications/bai2015intention.pdf

© 2020 National University of Singapore. All Rights Reserved

POMDP for Autonomous Driving in NUS



(a) (b) (c) (d) (e)

Fig. 5. The vehicle encounters a pedestrian who stops to make a phone call. Histograms indicate beliefs over pedestrian intentions. See Fig. 6 for color codes. The colored dots indicate vehicle actions: green for ACCELERATE, yellow for MAINTAIN, and red for DECELERATE.

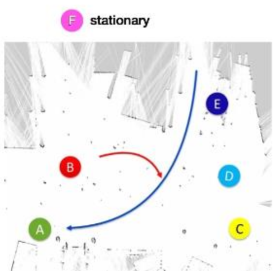


Fig. 6. A top-down view of the plaza area with a map built from LIDAR data. "A"–"F" indicate pedestrian intentions. The blue and the orange lines roughly correspond to the vehicle and the pedestrian paths, respectively, for the test run in Fig. 5.

TABLE I

COMPARISON OF POMDP PLANNING AND REACTIVE CONTROL.

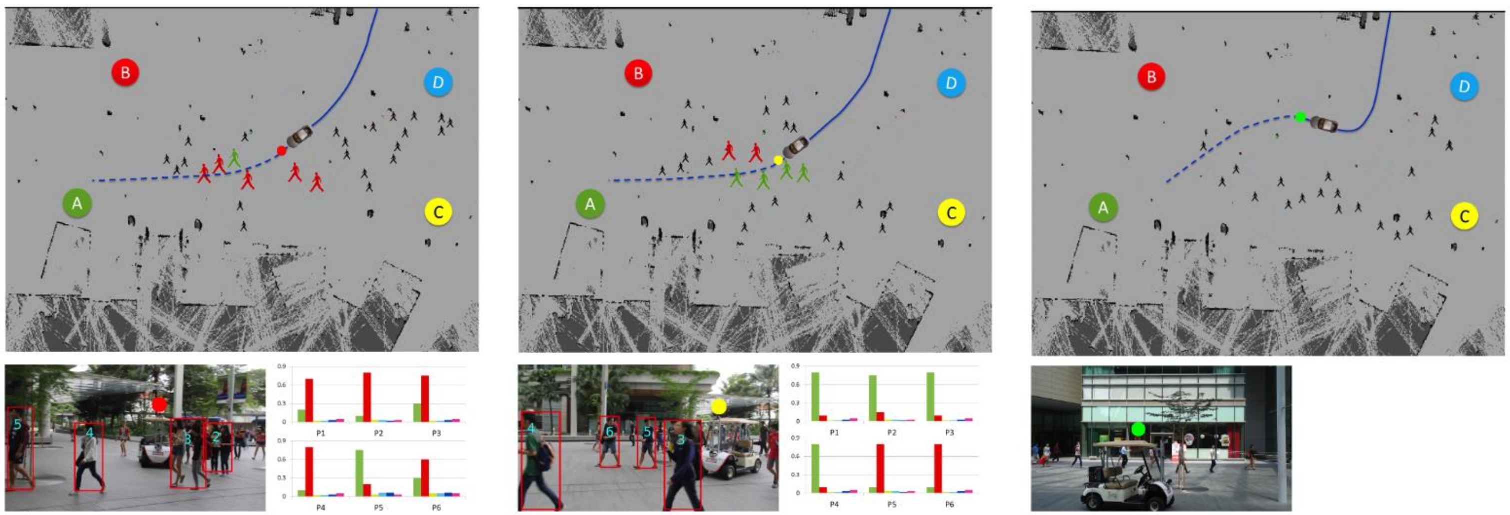
	Risk	Time (s)	Total Acceleration (m/s^2)
POMDP	0.0043 ± 0.0013	38.57 ± 0.16	6.31 ± 0.03
Reactive	0.0192 ± 0.0021	48.43 ± 0.27	7.85 ± 0.03

For this work, **Pedestrian intentions are modeled as goal locations** ("A"–"E" in Fig. 6), which correspond to entrances to office buildings, shops, restaurants, etc., as well as a bus stop; **"F" = Pedestrian remains stationary**

https://www.comp.nus.edu.sg/~leews/publications/bai2015intention.pdf

© 2020 National University of Singapore. All Rights Reserved

POMDP for Autonomous Driving in NUS



(a) (b) (c)

Fig. 7. The vehicle drives amongst a dense pedestrian crowd. The dashed blue line indicates the planned path. Each figurine indicates a pedestrian detected. The six pedestrians closest to the vehicle are tracked for the planning purpose, and the colors of their corresponding figurines indicate their most likely intentions. Each histogram on the lower right of a subfigure indicates the belief over the intentions of a tracked pedestrian. The last subfigure contains no histograms, as the pedestrians are all far away and not tracked.

https://www.comp.nus.edu.sg/~leews/publications/bai2015intention.pdf

© 2020 National University of Singapore. All Rights Reserved

Comparison between Markov Chain, MDP, HMM & POMDP

Markov Models		Do we have control over the state transitions?	
		NO	YES
Are the states completely observable?	YES	**Markov Chain**	**MDP** Markov Decision Process
	NO	**HMM** Hidden Markov Model	**POMDP** Partially Observable Markov Decision Process

© 2020 National University of Singapore. All Rights Reserved

THANK YOU

Email: nicholas.ho@nus.edu.sg