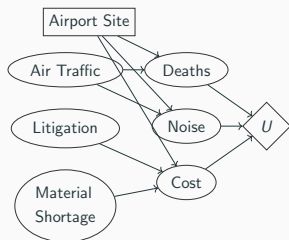


Lecture 29: Markov Decision Processes

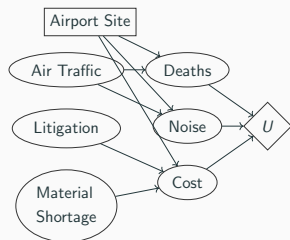
A Map of the Landscape

Benign	Chaotic
Fully Observable	Partially Observable
Deterministic	Stochastic
Episodic	Sequential
Static	Dynamic
Discrete	Continuous
Single Agent	Multi-Agent
Passive Reasoning	Agent can Act
Fixed Goal	Quantified Preferences

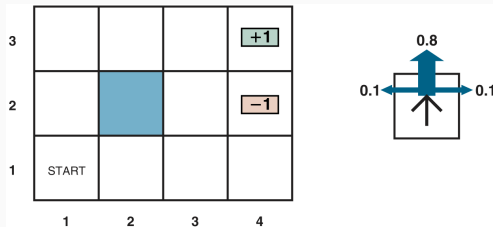


A Map of the Landscape

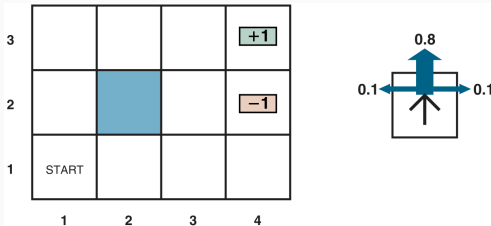
Benign	Chaotic
Fully Observable	Partially Observable
Deterministic	Stochastic
Episodic	Sequential
Static	Dynamic
Discrete	Continuous
Single Agent	Multi-Agent
Passive Reasoning	Agent can Act
Fixed Goal	Quantified Preferences



A Sequential Decision Problem



A Sequential Decision Problem

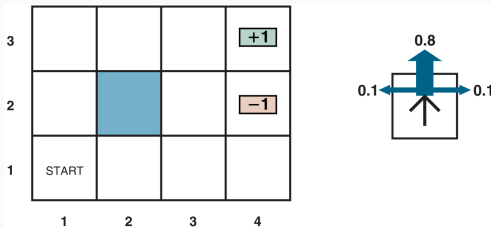


S_0 – initial state

$P(s'|s, a) = T(s, a, s')$ – transition model

$R(s)$ – reward function

A Sequential Decision Problem



S_0 – initial state

$P(s'|s, a) = T(s, a, s')$ – transition model

$R(s)$ – reward function

$$U_h(s_0, s_1, s_2, \dots)$$

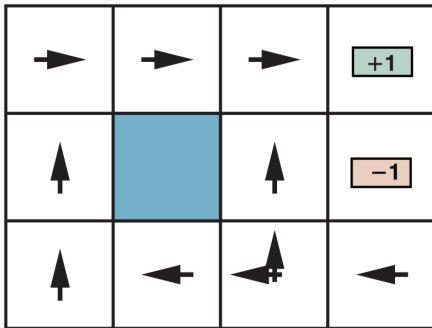
Markov Decision Process

$$MDP = \langle S_0, \mathcal{A}, T(s, a, s'), R(s) \rangle$$

$\pi(s)$ – **policy**

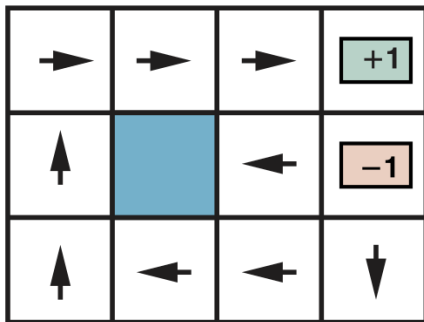
$\pi^*(s)$ – **optimal policy** (the one that yields highest expected utility)

Reward function affects optimal policy



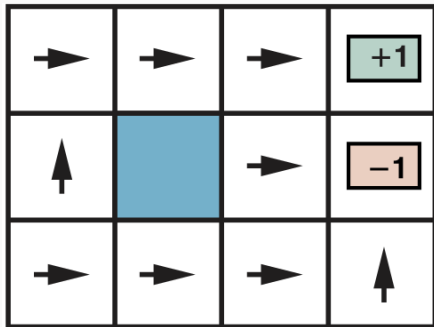
$$R(s) = -0.04$$

Reward function affects optimal policy



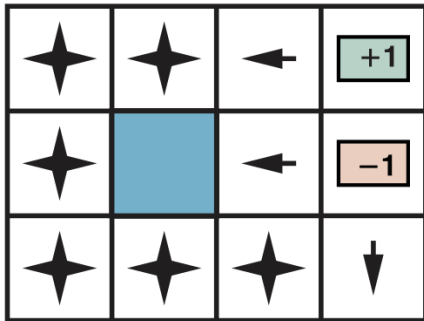
$$R(s) = -0.01$$

Reward function affects optimal policy



$$R(s) = -1.15$$

Reward function affects optimal policy



$$R(s) = 2.0$$

Finite vs Infinite Utilities

Infinite Horizon Utility Function:

$$U_h([s_1, s_2, \dots])$$

Finite Horizon Utility Function:

$$\forall k, U_h([s_1, s_2, \dots, s_N, \dots, s_{N+k}]) = U_h([s_1, s_2, \dots, s_N]) \quad \text{for fixed } N$$

$$U_h([s_0, s_1, s_2, \dots]) = R(s_1) + R(s_2) + \dots$$

Discounted Reward Formulation

$$\begin{aligned}U_h([s_0, s_1, s_2, \dots]) &= R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \gamma^3 R(s_3) + \dots \\&= \sum_{t=0}^{\infty} \gamma^t R(s_t) \\&\leq \sum_{t=0}^{\infty} \gamma^t R_{max} \\&= \frac{R_{max}}{1 - \gamma}\end{aligned}$$

Discounted Reward Formulation

$$\begin{aligned}U_h([s_0, s_1, s_2, \dots]) &= R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \gamma^3 R(s_3) + \dots \\&= \sum_{t=0}^{\infty} \gamma^t R(s_t) \\&\leq \sum_{t=0}^{\infty} \gamma^t R_{max} \\&= \frac{R_{max}}{1 - \gamma}\end{aligned}$$

Discounted Reward Formulation

$$\begin{aligned}U_h([s_0, s_1, s_2, \dots]) &= R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \gamma^3 R(s_3) + \dots \\&= \sum_{t=0}^{\infty} \gamma^t R(s_t) \\&\leq \sum_{t=0}^{\infty} \gamma^t R_{max} \\&= \frac{R_{max}}{1 - \gamma}\end{aligned}$$

Discounted Reward Formulation

$$\begin{aligned}U_h([s_0, s_1, s_2, \dots]) &= R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \gamma^3 R(s_3) + \dots \\&= \sum_{t=0}^{\infty} \gamma^t R(s_t) \\&\leq \sum_{t=0}^{\infty} \gamma^t R_{max} \\&= \frac{R_{max}}{1 - \gamma}\end{aligned}$$

How good is a given state?

$$U^\pi(s) = E \left[\sum_{t=0}^{\infty} \gamma^t R(S_t) \right]$$













$$\pi_s^* = \underset{\pi}{\operatorname{argmax}} U^\pi(s)$$

How good is a given state?




$$U^\pi(s) = E \left[\sum_{t=0}^{\infty} \gamma^t R(S_t) \right]$$

$$\pi_s^* = \underset{\pi}{\operatorname{argmax}} U^\pi(s)$$

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

(a) $\pi(s)$

0.8516	0.9078	0.9578	
0.8016		0.7003	
0.7453	0.6953	0.6514	0.4279

(b) $U^\pi(s)$

Finding the Optimal Policy: The Bellman Equation

$$\pi^*(s) = \underset{\pi}{\operatorname{argmax}} U^\pi(s)$$

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} P(s'|s, a) U^{\pi^*}(s')$$

Bellman Equation: $U^{\pi^*}(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s, a) U^{\pi^*}(s')$

Finding the Optimal Policy: The Bellman Equation

$$\pi^*(s) = \underset{\pi}{\operatorname{argmax}} U^\pi(s)$$

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} P(s'|s, a) U^{\pi^*}(s')$$

Bellman Equation: $U^{\pi^*}(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s, a) U^{\pi^*}(s')$

Finding the Optimal Policy: The Bellman Equation

$$\pi^*(s) = \underset{\pi}{\operatorname{argmax}} U^\pi(s)$$

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} P(s'|s, a) U^{\pi^*}(s')$$

Bellman Equation: $U^{\pi^*}(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s, a) U^{\pi^*}(s')$

Bellman Equations

Bellman Equation:
$$U^{\pi^*}(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s, a) U^{\pi^*}(s')$$

Q: Problem with n states, how many bellman equations?

Bellman Equations

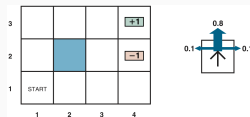
$$U^{\pi^*}(s_{\langle 1,1 \rangle}) = R(s_{\langle 1,1 \rangle}) + \gamma \max_a \sum_{s'} P(s'|s_{\langle 1,1 \rangle}, a) U^{\pi^*}(s')$$

$$U^{\pi^*}(s_{\langle 1,2 \rangle}) = R(s_{\langle 1,2 \rangle}) + \gamma \max_a \sum_{s'} P(s'|s_{\langle 1,2 \rangle}, a) U^{\pi^*}(s')$$

$$U^{\pi^*}(s_{\langle 1,3 \rangle}) = R(s_{\langle 1,3 \rangle}) + \gamma \max_a \sum_{s'} P(s'|s_{\langle 1,3 \rangle}, a) U^{\pi^*}(s')$$

$$U^{\pi^*}(s_{\langle 1,4 \rangle}) = R(s_{\langle 1,4 \rangle}) + \gamma \max_a \sum_{s'} P(s'|s_{\langle 1,4 \rangle}, a) U^{\pi^*}(s')$$

...



Bellman Equations

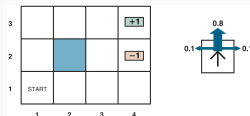
$$U^{\pi^*}(s_{\langle 1,1 \rangle}) = \underbrace{R(s_{\langle 1,1 \rangle})}_{\text{reward}} + \gamma \max_a \sum_{s'} \underbrace{P(s'|s_{\langle 1,1 \rangle}, a)}_{\text{transition}} \underbrace{U^{\pi^*}(s')}_{\text{recursive}}$$

$$U^{\pi^*}(s_{\langle 1,2 \rangle}) = R(s_{\langle 1,2 \rangle}) + \gamma \max_a \sum_{s'} P(s'|s_{\langle 1,2 \rangle}, a) U^{\pi^*}(s')$$

$$U^{\pi^*}(s_{\langle 1,3 \rangle}) = R(s_{\langle 1,3 \rangle}) + \gamma \max_a \sum_{s'} P(s'|s_{\langle 1,3 \rangle}, a) U^{\pi^*}(s')$$

$$U^{\pi^*}(s_{\langle 1,4 \rangle}) = R(s_{\langle 1,4 \rangle}) + \gamma \max_a \sum_{s'} P(s'|s_{\langle 1,4 \rangle}, a) U^{\pi^*}(s')$$

...



Value Iteration

Setup: Set Utility guesses to arbitrary values

$$U_0(s_{\langle 1,1 \rangle}) \leftarrow 0, U_0(s_{\langle 1,2 \rangle}) \leftarrow 0, \dots$$

Repeat: One-step update Utilities using bellman equation

$$U_{i+1}(s_{\langle 1,1 \rangle}) \leftarrow R(s_{\langle 1,1 \rangle}) + \gamma \max_a \sum_{s'} P(s'|s_{\langle 1,1 \rangle}, a) U_i(s')$$

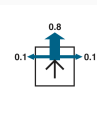
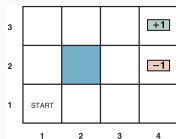
...

$$U_{i+1}(s_{\langle 3,4 \rangle}) \leftarrow R(s_{\langle 3,4 \rangle}) + \gamma \max_a \sum_{s'} P(s'|s_{\langle 3,4 \rangle}, a) U_i(s')$$

Until: $U_{i+1}(s) \approx U_i(s)$ for all s

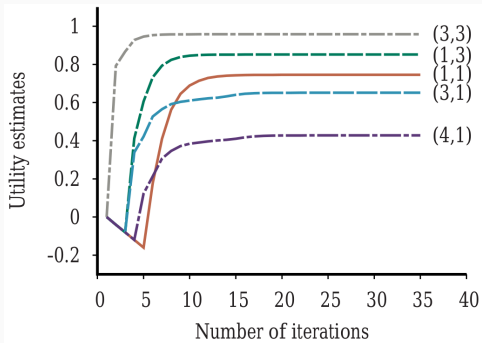
Example Update

$$U_{i+1}(s_{\langle 1,1 \rangle}) \leftarrow -0.04 + \gamma \max[\begin{aligned} &0.8U_i(s_{\langle 1,2 \rangle}) + 0.1U_i(s_{\langle 2,1 \rangle}) + 0.1U_i(s_{\langle 1,1 \rangle}), && (up) \\ &0.9U_i(s_{\langle 1,1 \rangle}) + 0.1U_i(s_{\langle 1,2 \rangle}), && (left) \\ &0.9U_i(s_{\langle 1,1 \rangle}) + 0.1U_i(s_{\langle 2,1 \rangle}), && (down) \\ &0.8U_i(s_{\langle 2,1 \rangle}) + 0.1U_i(s_{\langle 1,2 \rangle}) + 0.1U_i(s_{\langle 1,1 \rangle}) && (right) \end{aligned}]$$



Value Iteration in Action

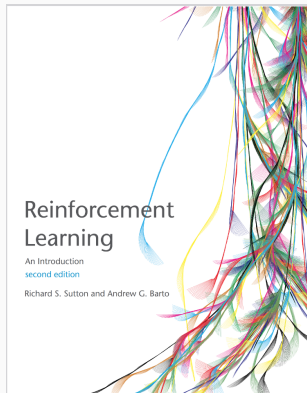
0.8516	0.9078	0.9578	+1
0.8016		0.7003	-1
0.7453	0.6953	0.6514	0.4279



Wait! Thats it?

Courses

- Robot and Reinforcement Learning



A Map of the Landscape

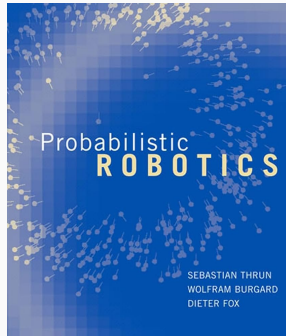
Benign	Chaotic
Fully Observable	Partially Observable
Deterministic	Stochastic
Episodic	Sequential
Static	Dynamic
Discrete	Continuous
Single Agent	Multi-Agent
Passive Reasoning	Agent can Act
Fixed Goal	Quantified Preferences

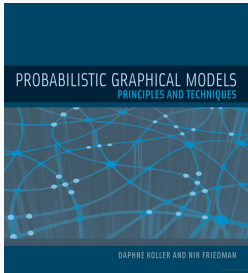
A Map of the Landscape

Benign	Chaotic
Fully Observable	Partially Observable
Deterministic	Stochastic
Episodic	Sequential
Static	Dynamic
Discrete	Continuous
Single Agent	Multi-Agent
Passive Reasoning	Agent can Act
Fixed Goal	Quantified Preferences

Courses

- Introduction to Mobile Robotics (MOB)
- Advanced Robotics





Courses:

- Probabilistic Modelling and Reasoning (PMR)
- Methods for Causal Inference (MCI)

I liked Constraint Solving / AC3 Stuff

The screenshot shows the Google OR-Tools website. The top navigation bar includes the Google OR-Tools logo, 'OR-Tools', and 'OR API'. Below this is a blue header with 'OR-Tools' and a menu with 'Installation', 'Guides', 'Reference', 'Examples', and 'Support'. A left sidebar contains a 'Filter' box and a list of categories: 'Get Started' (with a sub-item 'About OR-Tools'), 'MathOpt', 'CP-SAT', 'Network Flows', 'Linear Optimization', 'Integer Optimization', 'Assignment', 'Packing', 'Routing', and 'Scheduling'. The main content area is titled 'About OR-Tools' and includes a breadcrumb trail 'Home > Products > OR-Tools > Guides', a 'Was this helpful?' feedback prompt, and a 'Send feedback' button. The text explains that OR-Tools is open source software for combinatorial optimization and lists three example problem types: Vehicle routing, Scheduling, and Bin packing. It also mentions that OR-Tools uses state-of-the-art algorithms to narrow down the search set and includes solvers for Constraint Programming.

Google OR-Tools OR-Tools OR API

OR-Tools

Installation Guides Reference Examples Support

Filter

Get Started ^

- About OR-Tools
- Get Started Guides

MathOpt v

CP-SAT v

Network Flows v

Linear Optimization v

Integer Optimization v

Assignment v

Packing v

Routing v

Scheduling v

Home > Products > OR-Tools > Guides

Was this helpful?

About OR-Tools

OR-Tools is open source software for *combinatorial optimization*, which seeks to find the best solution to a problem out of a very large set of possible solutions. Here are some examples of problems that OR-Tools solves:

- **Vehicle routing:** Find optimal routes for vehicle fleets that pick up and deliver packages given constraints (e.g., "this truck can't hold more than 20,000 pounds" or "all deliveries must be made within a two-hour window").
- **Scheduling:** Find the optimal schedule for a complex set of tasks, some of which need to be performed before others, on a fixed set of machines, or other resources.
- **Bin packing:** Pack as many objects of various sizes as possible into a fixed number of bins with maximum capacities.

In most cases, problems like these have a vast number of possible solutions—too many for a computer to search them all. To overcome this, OR-Tools uses state-of-the-art algorithms to narrow down the search set, in order to find an optimal (or close to optimal) solution.

OR-Tools includes solvers for:

[Constraint Programming](#)

A set of techniques for finding feasible solutions to a problem expressed as *constraints* (e.g., a room can't be used for two events simultaneously, or the distance to the crops must be less than the length of the hose, or no more than five TV shows can be recorded at once).

Send feedback

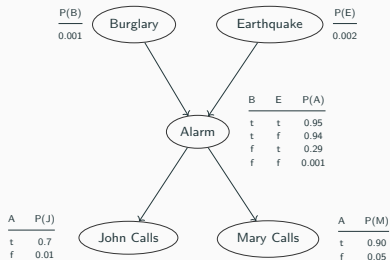


Competitions

The International Planning Competition 2023

- Classical Tracks
 - Daniel Fišer, Saarland University
 - Florian Pommerening, University of Basel
- Learning Tracks
 - Jendrik Seipp, Linköping University
 - Javier Segovia-Aguas, Universitat Pompeu Fabra
- Probabilistic Tracks
 - Ayal Tattler, University of Toronto
 - Scott Sanner, University of Toronto
- Numeric Tracks
 - Joan Espasa Arxer, University of St Andrews
 - Enrico Scala, University of Brescia
- HTN Tracks
 - Ron Alford, MITRE
 - Dominik Schreiber, Karlsruhe Institute of Technology
 - Gregor Behnke, University of Amsterdam

Where did these parameters come from?



Courses

- Machine Learning (MLG)
- Introductory Applied Machine Learning (IAML)
- Machine Learning Practical (MLP)
- Machine Learning Systems (MLS)
- Machine Learning Theory (MLT)
- Machine Learning and Pattern Recognition (MLPR)

I can make an AI agent...but should I?



Courses: Professional Issues, Modelling of Systems for Sustainability, Usable Security and Privacy

The Cynical Exam Prep Slide

- **Examiner Laziness:** Teachers rarely design exam Qs entirely from scratch. You can access the same resources they can (past exam papers are available online; tutorial solutions are out; required reading has exercises chapters...)
- **Not all Qs Created Equal:** The largest marks in compsci exams tend to be for algorithmic questions over bookwork Qs. Do you know how to implement the algorithms that have come up in the course?
- **Homo-Economicus:** Stuck on a 2-mark question? Move on and come back!
- **Worked Example Hinting:** Does a lecture follow up a particular method with a long worked-through example? Sounds like the lecturer really wants to make sure you understand this concept!
- **Pay Attention to “Weird Specifics”:** question explicitly instructs you to run through the algorithm “in a particular order”, “round to x decimal places”, “justify answer with respect to specific definition”? Might be a clue that this question could get really hard if you dont follow this advice!