Recap

- **Stochastic games**
  - unknown parts of the system can be modelled adversarially
  - zero-sum turn-based (or concurrent) stochastic games
    - dynamic programming (value iteration) generalises

- **Uncertain MDPs**
  - MDPs plus epistemic uncertainty: set of transition functions
    - each \( P \in \mathcal{P} \) is a transition function \( P : S \times A \times S \to [0,1] \)
  - rectangularity (dependencies)
  - control policies + robust control
  - environment policies

\[
V^*(s) = \max_{\pi \in \Pi} \min_{P \in \mathcal{P}} V_{\pi,P}(s)
\]
Course contents

• Markov decision processes (MDPs) and stochastic games
  ‣ MDPs: key concepts and algorithms
  ‣ stochastic games: adding adversarial aspects

• Uncertain MDPs
  ‣ MDPs + epistemic uncertainty, robust control, robust dynamic programming, interval MDPs, uncertainty set representation, challenges, tools

• Sampling-based uncertain MDPs
  ‣ removing the transition independence assumption

• Bayes-adaptive MDPs
  ‣ maintaining a distribution over the possible models
Robust control
Resolving uncertainty

• Now we consider a more dynamic approach to resolving uncertainty
  ‣ (which we will need to extend dynamic programming to this setting)

• An environment policy (or nature policy, or adversary) \( \tau \in \mathcal{T} \)
  ‣ is a mapping \( \tau : (S \times A)^* \times (S \times A) \rightarrow \text{Dist}(S) \)
  ‣ such that \( \tau(s_0, a_0, \ldots, s_n, a_n) \in \mathcal{P}_s^\alpha \)
  ‣ note: this assumes \((s,a)\)-rectangularity!

• Policies \( \pi, \tau \) yield
  ‣ a probability space \( \mathcal{P}_{s,\tau}^\pi \)
  ‣ random variables \( \mathbb{E}_{s,\tau}^\pi(X) \)
  ‣ and value functions \( V_{s,\tau}^\pi \)
Dynamic vs. static uncertainty

- Quantifying over environment policies $\tau \in \mathcal{T}$ is more exhaustive
  - than quantifying over transition probabilities $P \in \mathcal{P}$
  - $\{Pr_{s,P}^{\pi,P} : P \in \mathcal{P}\} \subseteq \{Pr_{s,\tau}^{\pi,\tau} : \tau \in \mathcal{T}\}$

- Memoryless (stationary) environment policies $\tau_m \in \mathcal{T}_m$
  - are mappings $\tau_m : S \times A \to \text{Dist}(S)$ such that $\tau_m(s,a) \in \mathcal{P}_s$
  - in this case, the semantics now coincide:
  - $\{Pr_{s,P}^{\pi,P} : P \in \mathcal{P}\} = \{Pr_{s,\tau_m}^{\pi,\tau_m} : \tau_m \in \mathcal{T}_m\}$

- We call this dynamic uncertainty ($\tau \in \mathcal{T}$) vs. static uncertainty ($P \in \mathcal{P}$)
  - which to use is a modelling decision (e.g., on the timing of events)
  - but there are also implications for tractability
  - similar situation to rectangularity (uncertainty set independence)
Robust control (revisited)

- Robust control
  - but quantifying over policies (rather than uncertainty sets)

- Now we have
  - optimal worst-case value
    \[
    V^*(s) = V_{\Pi,\mathcal{F}}(s) = \max_{\pi \in \Pi} \min_{\tau \in \mathcal{F}} V^{\pi,\tau}(s)
    \]
    notation for optimal value for sets of control/environment policy sets $\Pi, \mathcal{F}$

  - optimal worst-case policy
    \[
    \pi^* = \arg\max_{\pi \in \Pi} \min_{\tau \in \mathcal{F}} V^{\pi,\tau}(s)
    \]

- Note that we may want to quantify over mismatching sets of policies, e.g.:
  \[
  V_{\Pi,\mathcal{F}_m}(s) = \max_{\pi \in \Pi} \min_{\tau_m \in \mathcal{F}_m} V^{\pi,\tau_m}(s) = \max_{\pi \in \Pi} \min_{P \in \mathcal{P}} V^{\pi,P}(s)
  \]
  e.g. for static uncertainty
uMDPs vs stochastic games
Robust dynamic programming

• Let’s again focus on optimising MaxProb (the situation is similar for SSP)
  ‣ and recall: we need to assume \((s,a)\)-rectangularity

• Memoryless policies suffice, for both the controller and the environment

\[
V_{\Pi,\mathcal{F}}(s_0) = V_{\Pi_m,\mathcal{F}_m}(s_0) = V_{\Pi_m,\mathcal{F}}(s_0) = V_{\Pi,\mathcal{F}_m}(s_0)
\]

• Perfect duality:

\[
V_{\Pi,\mathcal{F}}(s_0) = \max_{\pi \in \Pi} \min_{\tau \in \mathcal{F}} V^{\pi,\tau}(s_0) = \min_{\tau \in \mathcal{F}} \max_{\pi \in \Pi} V^{\pi,\tau}(s_0)
\]

• The optimal value function satisfies the Bellman equation:

\[
V^*(s) = V_{\Pi,\mathcal{F}}(s) = \begin{cases} 
1 & \text{if } s \in \text{goal} \\
\max_{a \in A(s)} \inf_{P_a \in \mathcal{P}_a} \sum_{s' \in S} P_s^a(s') \cdot V_{\Pi,\mathcal{F}}(s') & \text{otherwise}
\end{cases}
\]
Robust value iteration

- Optimal values for uMDPs can be obtained using robust value iteration (robust VI)
  - from the limit of the vector sequence defined below
  - \( V^*(s) = \lim_{k \to \infty} x^k_s \) where:

\[
\begin{align*}
x^k_s &= \begin{cases} 
1 & \text{if } s \in S^1 \\
0 & \text{if } s \in S^0 \\
0 & \text{if } s \in S^? \text{ and } k = 0 \\
\max_{a \in A(s)} \inf_{P_s^a \in \mathcal{P}_s} \sum_{s' \in S} P_s^a(s') \cdot x^{k-1}_{s'} & \text{otherwise}
\end{cases}
\end{align*}
\]

- Again, this Bellman operator is (i) monotonic (ii) a contraction in the L\(_\infty\) norm
  - needs (s-a)-rectangularity, but no assumptions on convexity
  - (it suffices to take convex hull of each \( \mathcal{P}_s^a \))

We will re-use graph-based pre computation for MDPs
Uncertainty set representations

• The core step of robust VI comprises two nested optimisation problems:

\[
\max_{a \in A(s)} \inf_{P_s^a \in \mathcal{P}_s^a} \sum_{s' \in S} P_s^a(s') \cdot x_{s'}
\]

where \( x \) is some vector of values

• **Outer** problem (optimal control action)
• **Inner** problem (worst-case transition probabilities)

• **Computational cost**: robust VI potentially not much more expensive than VI for MDPs
  ‣ if the inner problem can solved efficiently
  ‣ note: uncertainty sets \( \mathcal{P}_s^a \) are usually infinite

• **Definition/representation** of uncertainty sets?
  ‣ trade off statistical accuracy vs. computation efficiency?

• First example: **intervals**, a simple uncertainty set representation
  ‣ which suit statistical estimates of confidence intervals for individual transition probabilities
Interval MDPs
Interval MDPs

- An interval MDP (IMDP) is of the form $\mathcal{M} = (S, s_0, A, \underline{P}, \overline{P})$ where:
  - states $S$, initial state $s_0$ and actions $A$ are as for MDPs
  - $\underline{P} : S \times A \times S \to [0,1]$ gives transition probability lower bounds
  - $\overline{P} : S \times A \times S \to [0,1]$ gives transition probability upper bounds
    - such that $\underline{P}(s, a, s') \leq \overline{P}(s, a, s')$ for all $s, a, s'$

- IMDP uncertainty sets
  - $\mathcal{P}_s^a = \{ P_s^a \in \text{Dist}(S) \mid \underline{P}(s, a, s') \leq P_s^a(s') \leq \overline{P}(s, a, s') \text{ for all } s' \}$
    - probabilities are independent (except for the need to sum to 1)
  - $\mathcal{P} = \times_{(s,a) \in S \times A} \mathcal{P}_s^a$
    - i.e., IMDPs are (s-a)-rectangular
IMDP uncertainty sets

- Interval uncertainty sets $\mathcal{P}_{\alpha}^{s}$ are convex subsets of $[0,1]^{|S|}$

We can delimit the intervals

- i.e., trim the interval bounds such that at least one possible distribution takes each extremal value
- e.g., $\underline{P}(s') := \max [\underline{P}(s'), 1 - \sum_{s \neq s'} \overline{P}(s)]$
  - e.g. $[0.1,0.4], [0.5,0.8] \rightarrow [0.2,0.4], [0.6,0.8]$
An assumption on IMDPs

- **Assumption**: IMDPs have a fixed underlying transition graph
  - i.e., for each $s,a,s'$ either: 
    1. $P(s,a,s') > 0$; or 
    2. $P(s,a,s') = \overline{P}(s,a,s') = 0$

- Otherwise behaviour can be qualitatively different for small changes in $P(s,a,s')$
  - For $\varepsilon > 0$, the probability to reach goal is always 1
  - For $\varepsilon = 0$, the probability to reach goal can be 0
  - (contrast to, e.g., a finite-horizon property $\text{MaxProb}^k(\text{goal})$)
Robust value iteration for IMDPs

- The inner problem for each iteration, and each \((s, a)\) is:
  \[
  \inf_{P_s^a \in \mathcal{P}_s} \sum_{s' \in S} P_s^a(s') \cdot x_{s'}
  \]

- Can be solved via a linear programming problem:
  - let \(p_s\) be \(|S|\) variables for the chosen probabilities \(P_s^a(s')\)
    - minimise \(\sum_{s'} p_s \cdot x_{s'}\) such that:
      \[
      P_s^a(s') \leq p_s \leq \overline{P}_s^a(s') \text{ for all } s' \text{ and } \sum_{s'} p_s = 1
      \]

- We can also solve this more directly by sorting
  - sort the values \(x_{s'}\) into ascending order
  - for increasing values \(x_{s_i}\) assign the maximum possible value to \(p_{s_i}\)
  - which is bounded by \(1 - (\text{the sum of actual/ min values for other } p_{s_j})\)
Running example: IMDPs and robust VI

• Example: MaxProb($goal_1$)
Running example: IMDPs and robust VI

• Example: MaxProb($\text{goal}_1$)
Running example: IMDPs and robust VI

• Example: MaxProb(\textit{goal}_1)

• Fix $x_4=1$ and $x_2=x_3=0$, just solve for $x_0, x_1$

• Iteration $k=0$: $x_0=x_1=0$

• Iteration $k=1$:

$$x_0 := \max(\min(0\cdot0.4 + 0\cdot0.6), \min(0\cdot p_1 + 0\cdot p_3 + 1\cdot p_4))$$

subject to:

- $0.09 \leq p_1 \leq 0.11$
- $0.49 \leq p_3 \leq 0.51$
- $0.39 \leq p_4 \leq 0.41$
- $p_1+p_3+p_4=1$

$$= \max(0, 0.39)$$

$$= 0.39$$  \hspace{1cm} p_4 = 0.39, \ldots$$

$$x_1 := \max(\min(0\cdot1), \min(0\cdot p_2 + 1\cdot p_4))$$

subject to:

- $0.46 \leq p_2 \leq 0.54$
- $0.46 \leq p_4 \leq 0.54$
- $p_2+p_4=1$

$$= \max(0, 0.46)$$

$$= 0.46$$  \hspace{1cm} p_4 = 0.46, \ldots$$
• Example: MaxProb(\textit{goal}_1)

\[
\begin{align*}
\text{x}_0 &= 0.39 \\
\text{x}_1 &= 0.46 \\
\text{x}_2 &= 0 \\
\text{x}_3 &= 0 \\
\text{x}_4 &= 1
\end{align*}
\]

- \text{S}_0 \xrightarrow{\text{east}} [0.4,0.4] \text{S}_1 \xrightarrow{\text{east}} [0.6,0.6] \text{S}_2
- \text{S}_0 \xrightarrow{\text{south}} [0.49,0.51] \text{S}_3
- \text{S}_1 \xrightarrow{\text{south}} [0.46,0.54] \text{S}_4

• Iteration k=2:

\[
x_0 := \max(\min(0.39 \cdot 0.4 + 0.46 \cdot 0.6), \min(0.46 \cdot p_1 + 0 \cdot p_3 + 1 \cdot p_4)) = \max(0.432, 0.436) = 0.436
\]

subject to:
- \(0.09 \leq p_1 \leq 0.11\)
- \(0.49 \leq p_3 \leq 0.51\)
- \(0.39 \leq p_4 \leq 0.41\)
- \(p_1 + p_3 + p_4 = 1\)

\[
\begin{align*}
p_3 &= 0.51 \\
p_1 &= \min(0.11, 1-(0.51+0.39)) = 0.1 \\
p_4 &= 1-(0.51+0.1) = 0.39
\end{align*}
\]

\[
x_1 := 0.46 \quad \text{(as before)}
\]
Running example: IMDPs and robust VI

- Example: MaxProb($\text{goal}_1$)

- Iteration $k=2$:

$$x_0 := \max(\min(0.39 \cdot 0.4 + 0.46 \cdot 0.6), \\
\min(0.46 \cdot p_1 + 0 \cdot p_3 + 1 \cdot p_4))$$

$$= \max(0.432, 0.436)$$

$$= 0.436$$

$$x_0 := \max(\min(0.39 \cdot 0.4 + 0.46 \cdot 0.6), \\
\min(0.46 \cdot p_1 + 0 \cdot p_3 + 1 \cdot p_4))$$

subject to:

$$0.09 \leq p_1 \leq 0.11$$

$$0.49 \leq p_3 \leq 0.51$$

$$0.39 \leq p_4 \leq 0.41$$

$$p_1 + p_3 + p_4 = 1$$

Finally: $x_0=0.46$, $x_1=0.46$
Interval MDPs - so far…

• Robust control is **computationally efficient** (robust value iteration)
  ‣ (s,a)-rectangular and inner problem is easy to solve
  ‣ another possibility not discussed here: convex optimisation [Puggelli et al.'13]

• For MaxProb (and SSP), optimal policies are memoryless (and deterministic)
  ‣ so computed policies are optimal worst case with respect to **static uncertainty**

  What about objectives that need memory? (e.g. finite horizon, or temporal logic)

• Intervals are a **simple, natural** way to model transition probability uncertainty

  How do we generate the intervals?

  Are there better models of uncertainty sets?
Policies with memory

• Quantifying over memoryless environment policies
  ‣ gives us worst-case behaviour over static uncertainty

\[ V^{\Pi, T_m}(s) = \max_{\pi \in \Pi} \min_{\tau_m \in T_m} V^{\pi, \tau_m}(s) = \max_{\pi \in \Pi} \min_{P \in \mathcal{P}} V^{\pi, P}(s) \]

• But for objectives that require non-memoryless control policies
  ‣ computation methods typically also assume non-memoryless environment policies

\[ V^{\Pi, T}(s) = \max_{\pi \in \Pi} \min_{\tau_m \in T} V^{\pi, \tau_m}(s) \]
  ‣ i.e., worst-case behaviour over dynamic uncertainty
  ‣ which is often (but not always) unrealistic (depends on time-scales)

• This however gives a conservative bound over static uncertainty

\[ V^{\Pi, T}(s) \leq \max_{\pi \in \Pi} \min_{P \in \mathcal{P}} V^{\pi, P}(s) \]
Memory (time dependencies)

- Objective: \( \text{MaxProb}^2(\text{goal}) \), i.e., get to \text{goal} in exactly 2 steps
  - so we need time-dependent strategies for the controller
  - computable via \( k \) steps of value iteration

- Worst-case probabilities (time-dependent environment strategies)
  - “b,b” 0.2 (optimal)
  - “a,b”: 0
  - “a,a”: \( \min\{p_1(1 - p_2) : p_1, p_2 \in [0.4,0.6]\} = 0.4 \cdot (1 - 0.6) = 0.16 \) (too conservative)

- Worst-case probabilities (memoryless environment strategies)
  - “b,b”: 0.2
  - “a,b”: 0
  - “a,a”: \( \min\{p(1 - p) : p \in [0.4,0.6]\} = 0.4 \cdot (1 - 0.4) = 0.24 \) (better bound) (now optimal)
Memory (temporal logic objectives)

- Temporal logic (in particular LTL) allows more complex objectives, e.g.:
  - $P_{\text{max}} =? \left[ (G\neg \text{hazard}) \land (GF \text{ goal}_1) \right]$ - “maximise probability of avoiding hazard and also visiting goal 1 infinitely often”
  - $P_{\text{max}} =? \left[ \neg \text{zone}_3 \lor (\text{zone}_1 \land (F \text{ zone}_4)) \right]$ - “maximise probability of patrolling zone 1 (whilst avoiding zone 3) then zone 4”

- For MDPs, we generate optimal policies by:
  - converting the LTL formula to a deterministic automaton
  - building a product of the MDP and the automaton
  - optimising a simpler objective (e.g. MaxProb) on the product MDP

- The techniques extend to uMDPs/IMDPs [Wolff et al.’12]
  - but (like for MDPs), optimal policies need memory
Automata for LTL objectives

• For co-safe LTL (satisfaction occurs in finite time), we use finite automata

\[ \neg \text{zone}_3 \cup (\text{zone}_1 \land (F \text{zone}_4)) \]

(avoiding hazard and also visiting goal 1 infinitely often)

• For general LTL, we use e.g. Rabin automata

\[ (G\neg \text{hazard}) \land (GF \text{goal}_1) \]

(visit zone 1 (whilst avoiding zone 3) then zone 4)
Optimising for LTL on a product MDP

MDP $M$

Product MDP $M \otimes \mathcal{A}$

Optimal memoryless policy of $M \otimes \mathcal{A}$ corresponds to finite-memory optimal policy of MDP $M$
Generating IMDP intervals

• Some examples of IMDP generation

• Unmanned aerial vehicle
  • robust control in turbulence
  • continuous-space dynamical model with unknown noise
  • discrete abstraction + finite “scenarios” of sampled noise yields IMDP abstraction
  
  [Badings et al.’23]

• Deep reinforcement learning
  • worst-case analysis of abstractions of probabilistic policies for neural networks
  • intervals between IMDP abstract states constructed by sampling the policy
  
  [Bacci&Parker’20]

• Robust anytime MDP learning
  • sampled MDP trajectories
  • IMDPs constructed and solved periodically to yield robust predictions on current model
  • PAC or Bayesian interval learning
  
  [Suilen et al.’22]
Learning IMDP intervals

• One approach: sampling from the (fixed, but unknown) “true” MDP
  ‣ generate sample paths and keep separate counts of transition frequencies

• Gives confidence intervals around point estimates for transition probabilities $P^a_s(s_i)$
  ‣ using probably approximately correct (PAC) guarantees
  ‣ we fix an error rate $\gamma$ and compute an error $\delta$
  ‣ standard method of maximum a-posteriori probability (MAP) estimation to infer point estimates of probabilities

• For each state $s$, we have sample counts $N = \#(s, a)$ and $k_i = \#(s, a, s_i)$
  ‣ point estimate of the transition probability $P^a_s(s_i)$ is: $\tilde{P}^a_s(s_i) \approx k_i/N$
  ‣ confidence interval for the transition probability: $\tilde{P}^a_s(s_i) \pm \delta$ where $\delta = \sqrt{\log(2/\gamma)/2N}$
  ‣ then we have: $Pr(P^a_s(s_i) \in \tilde{P}^a_s(s_i) \pm \delta) \geq 1 - \gamma$ (via Hoeffding’s inequality)
Learning IMDP intervals

- If desired, we can lift the PAC guarantee from individual transitions to the uMDP

- Distribute the chosen error rate $\gamma$ across all transitions:
  - $\gamma_P = \gamma / (\Sigma(s, a) \in S \times A \mid Succ_1(s, a) \mid )$
  - where $Succ_1(s, a) = \{ s \in S : 0 < P_s^a(s') < 1 \}$ is the set of successor states of each $(s, a)$ with more than one successor

- To construct the IMDP, we use:
  - $P^a_s(s_i) = \max(\varepsilon, \tilde{P}^a_s(s_i) - \delta_P)$
  - $\overline{P}^a_s(s_i) = \min(\tilde{P}^a_s(s_i) + \delta_P, 1)$

- Then we have: $Pr(P \in \mathcal{P}) \geq 1 - \gamma$

[Suilen et al.'22]
Likelihood uncertainty sets

• **Likelihood models** suit experimentally determined transition probabilities
  ‣ and are less conservative than interval representations

• Uncertainty sets are:
  ‣ are derived from empirical frequencies $F_s^a(s')$ of a transition to $s'$ after action $a$ in state $s$
  ‣ are described by likelihood regions: $\mathcal{P}_s^a = \{P_s^a \in \text{Dist}(S) \mid \sum_{s'} F_s^a(s') \log(P_s^a(s')) \geq \beta_s^a\}$
  ‣ where $\beta_s^a$ is the uncertainty level (can be estimated for a desired confidence level)
  ‣ $\beta_s^a < \beta_{s, \max}^a$ where $\beta_{s, \max}^a = \sum_{s'} F_s^a(s') \log(F_s^a(s'))$ is the optimal log-likelihood

• Inner optimisation problems
  ‣ can be solved (approximately) using a bisection algorithm
  ‣ to within an accuracy $\delta$ in time $O(\log(x_{\max}/\delta))$ where $x_{\max}$ is the maximum value in vector $x$
Uncertainty set models - Summary

- **Intervals & likelihood models**
  - both quite computationally tractable and statistically meaningful
  - interval models are more conservative (sometimes projected to as an estimate)

- **Finite scenarios** ("sampled"): \( \mathcal{P}_a^s = \{ P^a_{s,1}, \ldots, P^a_{s,k} \} \)
  - inner optimisation is simple (min over finite set)
  - but worst-case choice can be very conservative

- Many other possibilities, e.g.:
  - maximum a posteriori models, entropy models, ellipsoidal models, …
  - most have similar (approximate) optimisation approaches to likelihood models
  - see: [Nilim & Ghaoui’05] for details
Tool support: PRISM

- **PRISM**: probabilistic model checking tool
  - formal modelling and analysis (using temporal logic properties) of:
    - Markov chains, Markov decision processes,
    - interval Markov chains, interval Markov decision processes,
    - stochastic games (via PRISM-games), and much more…

- See: [www.prismmodelchecker.org](http://www.prismmodelchecker.org)
  - download, documentation, tutorials, papers, case studies, …

- Supporting files for ESSAI examples here:
  [www.prismmodelchecker.org/courses/essai23/](http://www.prismmodelchecker.org/courses/essai23/)
• ERC-funded project FUN2MODEL, based at Oxford
  ‣ lead by Marta Kwiatkowska
  ‣ model-based reasoning for learning and uncertainty

• Postdoc position available now
  ‣ http://www.fun2model.org/
  ‣ http://www.prismmodelchecker.org/news.php

Email:  david.parker@cs.ox.ac.uk
        marta.kwiatkowska@cs.ox.ac.uk
Summary (part 3)

• Uncertain MDPs
  ‣ environment policies - static vs dynamic uncertainty
  ‣ robust value iteration (robust dynamic programming)
  ‣ implementation with interval MDPs (IMDPs)
  ‣ non-memoryless policies (static uncertainty)
  ‣ generating / learning intervals
  ‣ uncertainty set representations
  ‣ tool support: PRISM

• Up next: Sampling-based uncertain MDPs
  ‣ removing the transition independence assumption (rectangularity)
References (part 3)

• IMDPs and uMDPs
References (part 3)

• Learning and using IMDPs

