Lecture 4: Partial observability

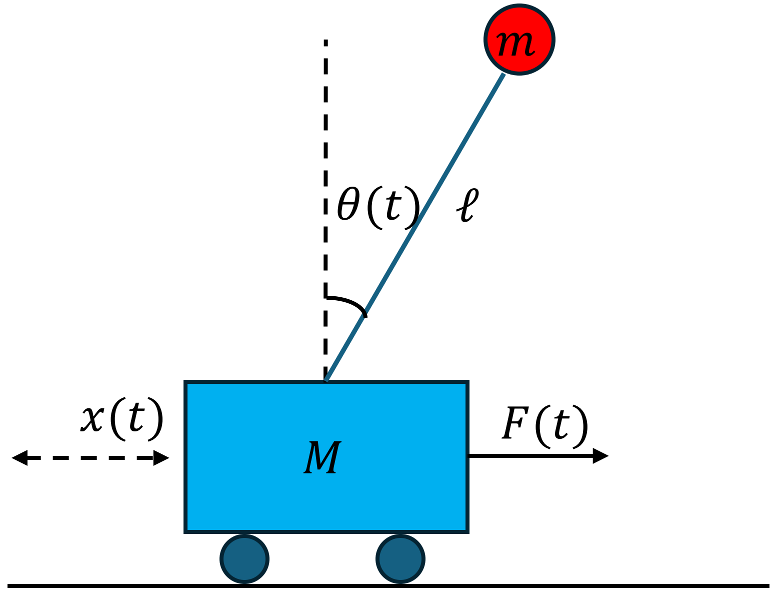
Optimal policies for Lagrangian turbulence – Dr. Robin Heinonen

Aqtivate workshop on data-driven and model-based tools for complex flows and complex fluids

June 3-7

The inverted pendulum revisited

- Recall: system state is $\theta(t), \omega(t) = \dot{\theta}(t)$
- In practice, we must **measure** these quantities
- Measurements *m* will come with **uncertainties** $\Delta\theta, \Delta\omega$
- Therefore, we **do not observe the state**
- But if we know (or can model) measurement likelihood $\Pr(m|\theta, \omega)$, gain **partial information about state** through measurements
- Uncertainity is everywhere. In practice, **partial observability is the rule rather than the exception**



The partial observable Markov decision process

- Consider an MDP with actions a, states s, rewards r.
- Suppose agent **does not access state**
- Instead, at each time step, make observation o drawn from *likelihood* $\Pr(o|s,a)$ depending on current s and previous a
- The agent will maintain a **probability distribution** over state space $b(s)$ called the *belief* or *posterior*
- $b(s)$ contains all the agent’s knowledge about the state

Partially observable Markov

Bayesian inference

$$\Pr(A|B) = \frac{\Pr(B|A)\Pr(A)}{\Pr(B)}$$

- How to convert observation, action sequence $o_1, a_1, o_2, a_2, \dots, o_t, a_t$ into probability distribution?
- Main tool: **Bayes' theorem.** If $\Pr(o|s,a)$ is known (or modeled), $b(s)$ can be updated using incoming observations:

$$\underset{\text{new belief}}{b(s|o,a)} \propto \underbrace{\Pr(o|s,a)}_{\text{likelihood of observation } o} \underbrace{\sum_{s'} \Pr(s|s',a)b(s')}_{\text{prob. of being in state } s} \longleftarrow \text{old belief}$$

- Requires o_t and o_{t-1} to be **independent**
- Need to choose **prior** $b_0(s)$ (problem dependent)

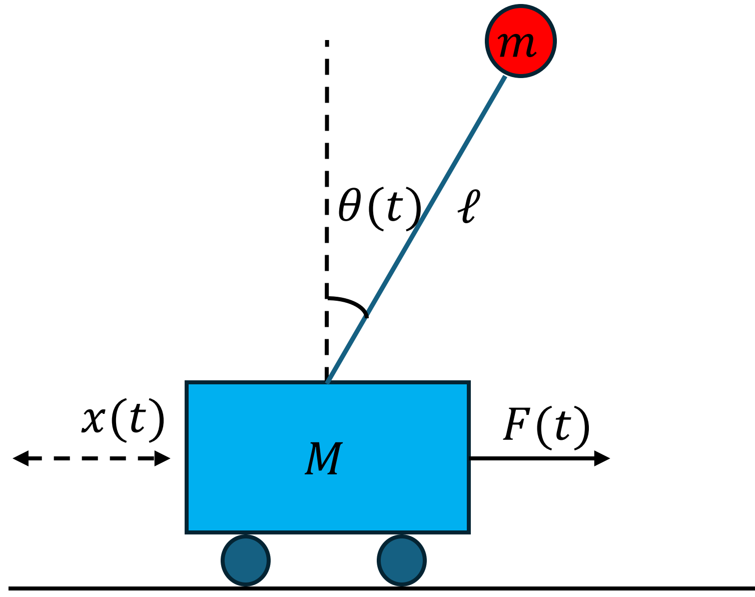
Example 1: inverted pendulum

- Measure θ, ω every timestep. Ground truth is θ^*, ω^*
- Reasonable assumption: independent Gaussians so

$$p(\theta, \omega | \theta^*, \omega^*) \propto \exp\left(-\frac{(\theta - \theta^*)^2}{2\Delta\theta^2} - \frac{(\omega - \omega^*)^2}{2\Delta\omega^2}\right)$$

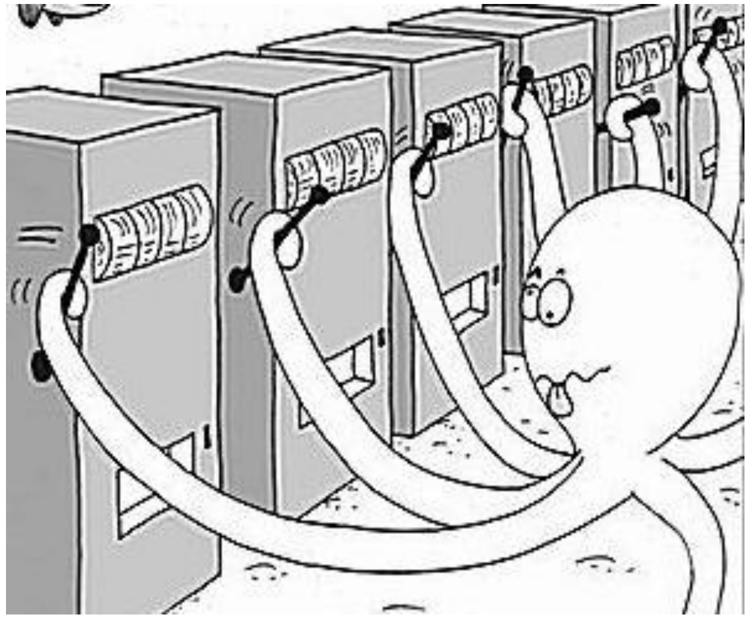
for some (known) $\Delta\theta$, $\Delta\omega$

- In order to solve problem, $\Delta\theta$, $\Delta\omega$ must be **known**
- Dynamics will quickly mix priors over θ, ω
- N.B. in linear regime, leads to "Kalman filtering"



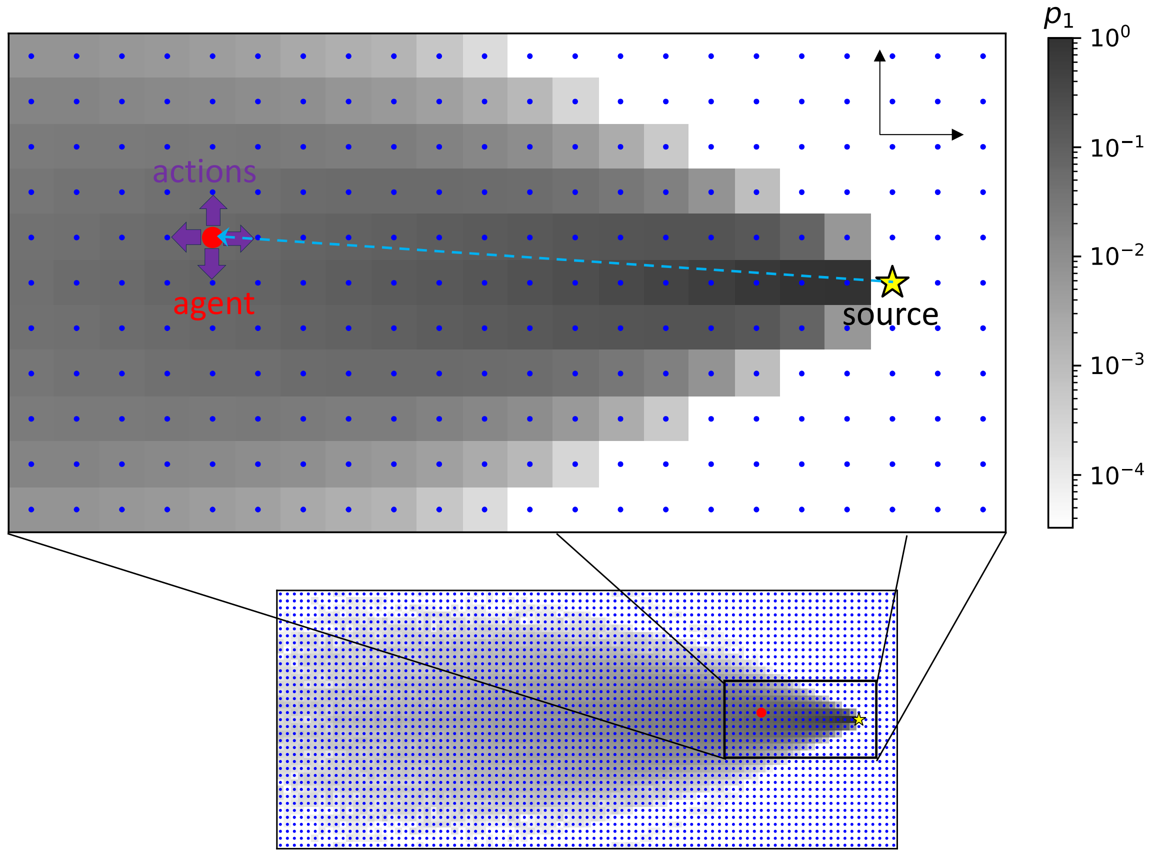
Example 2: multi-armed Bernoulli bandit

- $k \geq 1$ slot machines, each with unknown probability of unit payout p_i
- At each timestep, pull one lever, observe reward $r \in \{0,1\}$
- Want to maximize discounted payout $R = r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots$. This requires balancing **exploration** (discovering the p_i) and **exploitation** (pulling levers with highest p_i)
- Exploration/exploitation tradeoff is fundamental to partial observable problems
- What are states? Transitions? Observation likelihoods?
- Rare case where optimal policy can be specified exactly in closed form (Gittins 1974)



Example 3: olfactory search

- Set up spatial gridworld, source at fixed unknown position $\mathbf{x}_s$. State is agent position relative to source $\mathbf{x} = \mathbf{x}_a - \mathbf{x}_s$
- Observations $o_t \in \{0,1\}$. Concentration below/above threshold
- Likelihood of detection p_1 is function of $\mathbf{x}$. p_1 obtained either from model or empirically
- Agent moves between adjacent gridpoints
- $r = -1$ until source found (source absorbing state)



How to solve a POMDP?

- POMDPs can be viewed as **special cases of MDPs**. States are *probability distributions*, state transitions are governed by Bayes theorem
- Therefore, they obey a Bellman equation and can (in principle) be solved via dynamic programming.
- But solution is **very hard** because state space is **huge**
- Before we dive into POMDP solution, worth noting that there are often **powerful heuristics** available, vastly simpler to implement than a quasi-optimal policy. This will be subject of hands-on session (Friday)

Bellman equation for POMDPs

- For $r(s, a)$ deterministic (usual case), optimal value function obeys

$$V^*(b) = \max_a \left[\underbrace{\sum_s b(s) r(s,a)}_{\text{expected immediate reward}} + \gamma \underbrace{\sum_o \Pr(o|b,a)\, V^*\big(\overset{\text{belief after Bayesian update}}{b(\cdot\,|o,a)}\big)}_{\text{expectation of all future rewards}} \right]$$

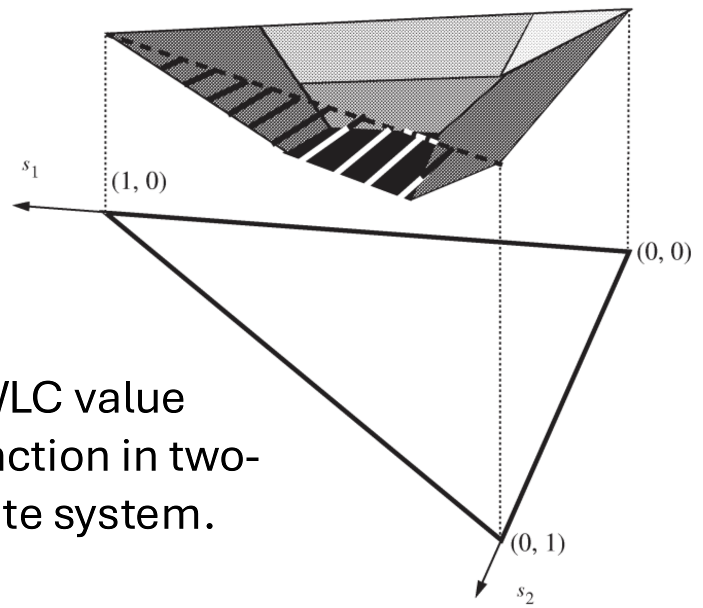
- Here, $\Pr(o|b,a) = \sum_{s,s'} \Pr(o|s,a) \Pr(s|s',a) b(s')$
- Need to solve and obtain V^* for all **distributions** over states $b(s)$. Uncountably many! **Approximation methods are necessary.**

PWLC representation

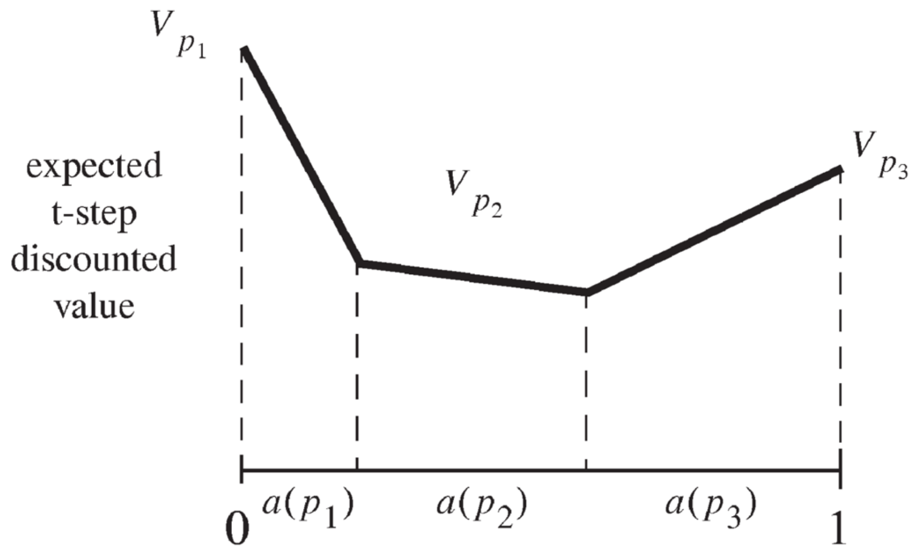
- Theorem: for finite-state system, V^* is **piecewise linear and convex** function (Sondik 1973)
- Can be parametrized as
$$V^*(b) = \max_{\alpha \in \mathcal{A}} \alpha \cdot b$$

where $\mathcal{A}$ is collection of "α-vectors" w/ same dimensionality as b, i.e. $|S|$

- Each α has associated action, defines a $|S|$-dimensional hyperplane
- Need algorithm to build $\mathcal{A}$



PWLC value function in two-state system.



Point-based value iteration

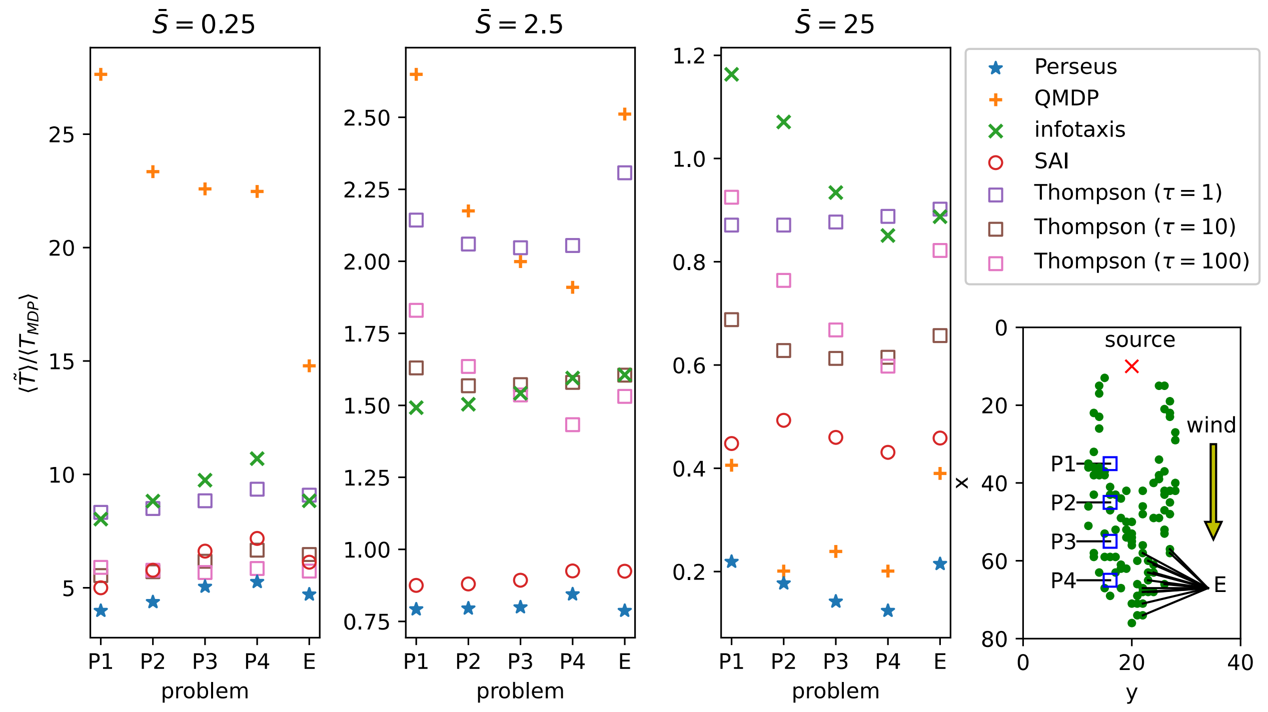
- Class of successful approx. algorithms for POMDPs. First introduced in Pineau *et al*. (2003)
- Idea: build up a representative set of beliefs $\mathcal{B}$, and perform value iteration on these beliefs only.
- Hope that in practice, any b encountered will be sufficiently close to an element of $\mathcal{B}$
- Value iteration accomplished via “backup operator”

$$\text{backup}: (b, \mathcal{A}) \mapsto \alpha$$

which produces new α from given b and current $\mathcal{A}$. Just a clever manipulation of Bellman equation

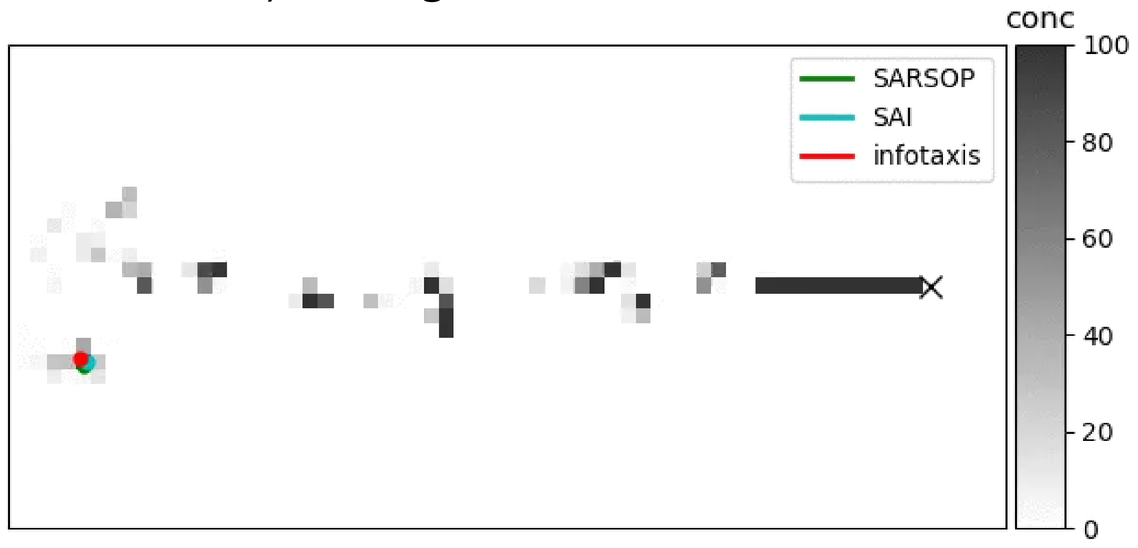
- Several different algorithms. Mostly differ by (a) method of generating $\mathcal{B}$ and (b) choosing order of $b \in \mathcal{B}$ to backup. See Shani *et al*. (2012) for review

POMDP solution of olfactory search



POMDP solution using Perseus algorithm (Spaan and Vlassis 2005) beats every tested heuristic for olfactory search, on both single points and an ensemble of points. From Heinonen *et al.* (2023)

Example of SARSOP algorithm (Kurniawati *et al*. 2012) beating heuristics in real flow data



Alternate method of solution

- Loisy and Eloy (2022) instead represented V^* by deep neural network
- Solved olfactory search POMDP using stochastic gradient descent, with loss function

$$L = E[(V^* - HV^*)^2]$$

(HV^* RHS of Bellman eq., expectation over belief samples)

- Belief samples obtained by using "experience replay," borrowing techniques from (model-free) **reinforcement learning**
- RL will be subject of final lecture