Last Time

☐ Intro to Embodied AI Safety

This Time

☐ sequential decision-making review

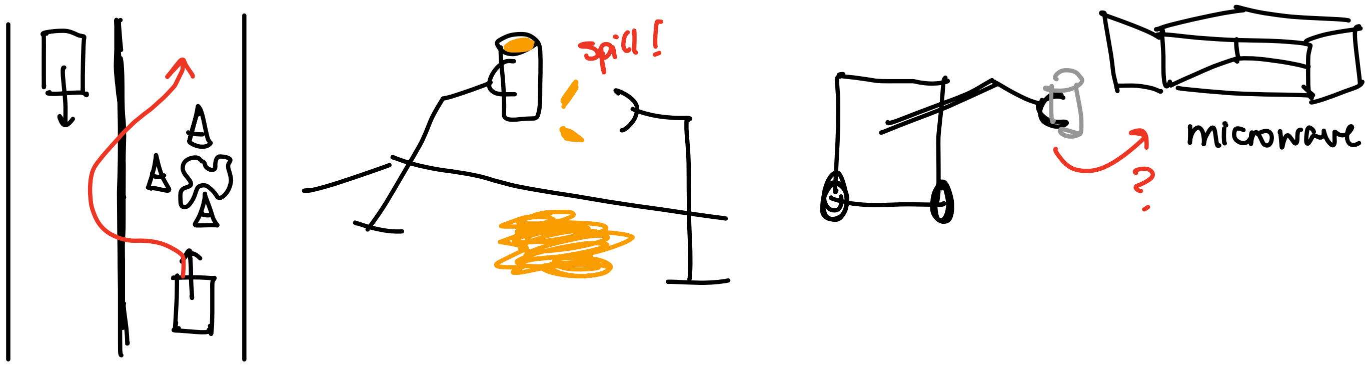
Lecture 1

EAIS SP'26

Andrea Bajcsy

Sequential Decision-Making – b/c the first half of the class is all about safety w.r.t decisions.

e.g. I want an embodied AI (EAI) system to make "good" decisions and prevent "bad" outcomes.



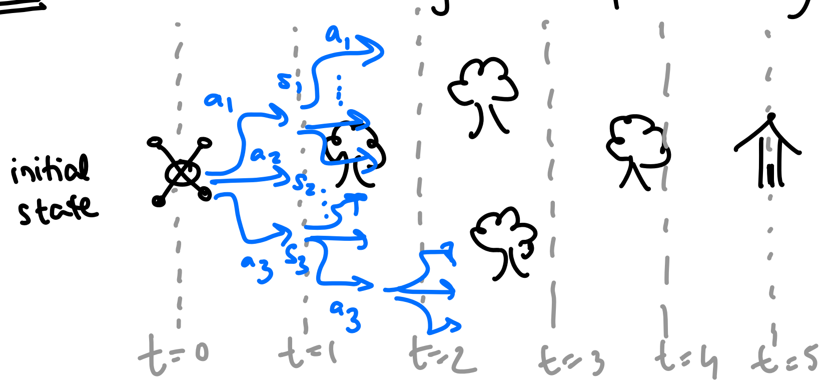
Our goal is to:

- mathematically model this decision-making process
- quantify the "goodness" of decisions
- compute these good decisions

This is where sequential decision-making (i.e. control theory) will provide us with a framework for answering these!

Q What makes sequential decision-making hard?

A naive solution grows exponentially in time horizon

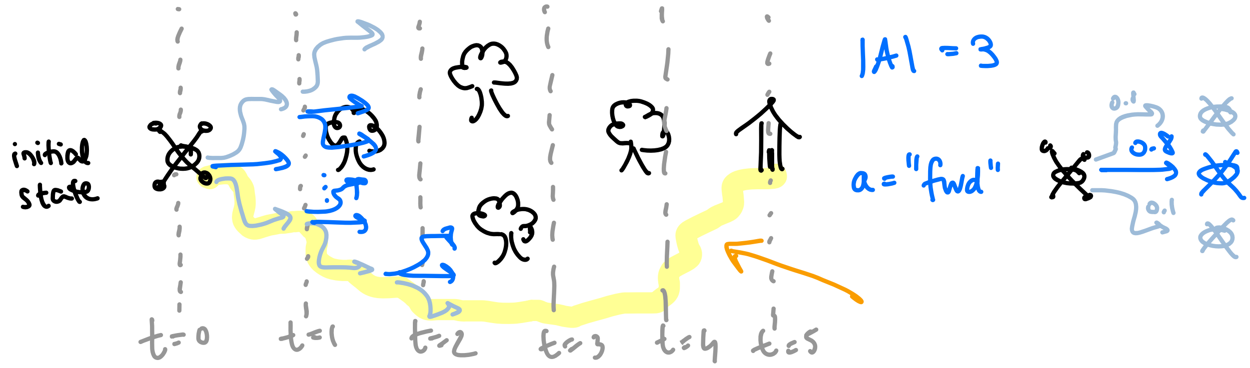


$|A| = 3$ { up, straight, down }

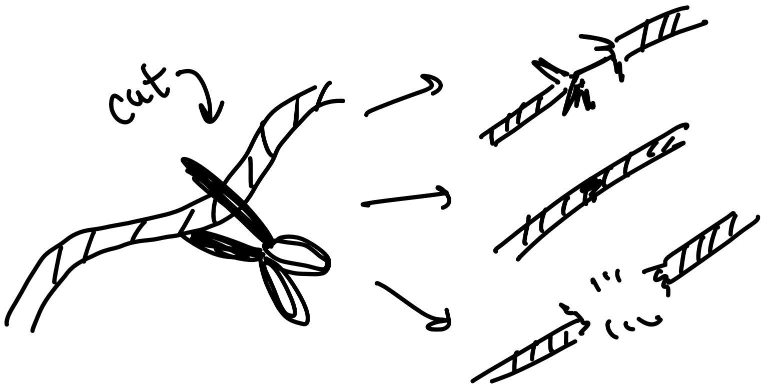
starting just from one state, you have $|A|^T$ possible seq. of decisions!

search: $\mathcal{O}((|S| \times |A|)^T)$

A2 outcomes of taking actions can be stochastic (or unknown)



A3 Some aspects of the world are hard to <u>represent</u> & <u>predict</u>

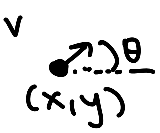


One framework that can help us describe these phenomenona are:

<u>state space repreprepresentations.</u>

<u>state</u>: $x(t) \in \mathbb{R}^n$ ← continuous time (often just x) // s_t in the AI lit.
$x_t \in \mathbb{R}^n$ ← discrete time

describes the minimal necessary characteristics of a system



ex. position in x-y plane of 2D vehicle, orientation, speed

control ↖ AI ↗
<u>control / action</u>: $u(t) \in \mathbb{R}^m$ // a_t in the AI lit.
$u_t \in \mathbb{R}^m$

inputs that we choose @ each instance in time.

ex. joint torques, acceleration

output / observation: $y(t) \in \mathbb{R}^L$ // o_t in the AI lit.
$y_t \in \mathbb{R}^L$

outputs that are actually measurable (typically through sensor)

ex. position through GPS, img. obs RGB camera,

! initially, we assume $y \equiv x$ "perfect observability" BUT it's **not** the case in general & it's a major safety challenge!

control

AI / RL

dynamics / transition: how the system evolves over time

	continuous-time $(t \in \mathbb{R})$	discrete time $(t \in \mathbb{Z})$
deterministic	$\underbrace{\dot{x}(t)}_{\text{change in state}} = f(\underbrace{x(t)}_{\text{state now}}, \underbrace{u(t)}_{\text{action now}})$	$\underbrace{x_{t+1}}_{\text{next state}} = f(\underbrace{x_t}_{\text{state}}, \underbrace{u_t}_{\text{action}})$
stochastic	$dX_t = f(X_t, u_t)dt + \underbrace{dW_t}_{\text{Weiner Process (infintessimal Gaussian Noise)}}$	$x_{t+1} \sim P(x_{t+1} \mid x_t, u_t)$

diffusion models!

Optimal Decision-Making

minimize Objective → AI Alignment
decisions

s.t. system dynamics

other constraints

Uncertainty quant.

Safe ctrl

{ state constraints
control bounds
⋮ }

An optimization problem, but it has a temporal aspect to it and we want sequence of decisions that are <u>optimal</u>.

More formally, in this class we will see:

Discrete-time:

$\mathcal{X} \times \mathcal{U}^T \to \mathbb{R}$

total cost of starting from x_0 & applying ctrl $u_{0:T-1}$

$$\min_{u_{0:T-1}} J(x_0, u_{0:T-1})$$

s.t. $x_{t+1} = f(x_t, u_t)$

upper limit of ctrl.

$\underline{u} \le u_t \le \bar{u}$ ← control bounds

lower limit

$\forall t \in \{0, \dots, T-1\}$

Continuous-time:

$$\min_{u(\cdot) \in \mathcal{U}_0^T} J(x(0), u(\cdot))$$

s.t. $\dot{x}(t) = f(x(t), u(t))$

$\underline{u} \le u(t) \le \bar{u}$

$\forall t \in [0, T]$

ex. of popular cost function (also called objective/reward)

$$J := \sum_{t=0}^{T-1} \underbrace{L(x_t, u_t)}_{\text{running cost}} + \underbrace{\ell(x_T)}_{\text{terminal cost}}$$

BIG Q: How to solve?

Each method listed below has its pros / cons -- BUT there are MANY solution strategies we can try! :)

① Calculus of Variations — converts constrained opt. → unconstrained via Lagrange multipliers

② Model Predictive Control (MPC) — like ① but usually in discrete-time and it replan in "receeding horizon"

③ Dynamic Programming — leverages the recursive structure in opt. control problems to compute policy

④ Reinforcement Learning — same foundations as ③ but uses sim, func. approx, data to scale.

⇒ feature: unknown dynamics

Dynamic Programming

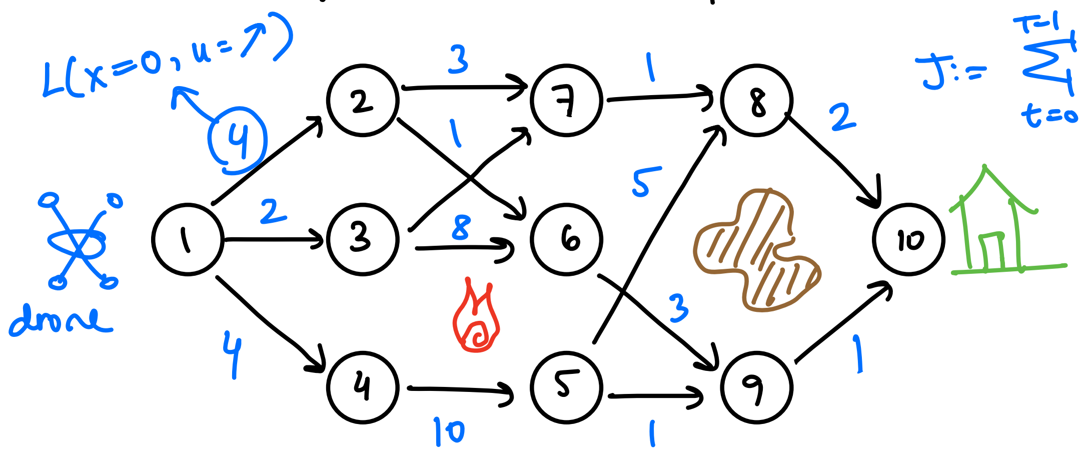
We will study the method of dynamic programming to solve optimal control problems. Dynamic programming relies on the principle of optimality developed by Richard Bellman around 1958 when he was working @ the RAND corporation. Since then dynamic prog. has been used in CS, operations research, controls, robotics, and many other domains.

We can intuit dynamic programming via an example.

ex. DRONE rescue during California wildfires.

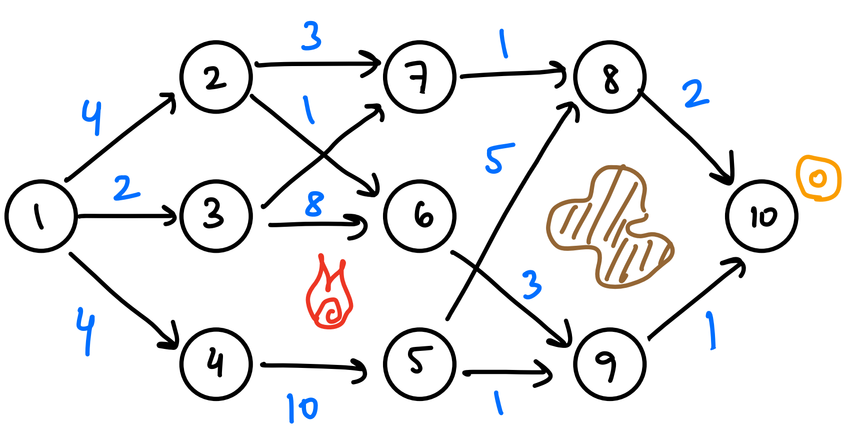
$\Rightarrow$ need to find shortest path from ① $\rightarrow$ ⑩

$$J := \sum_{t=0}^{T-1} L(x_t, u_t) + \ell(x_T)$$



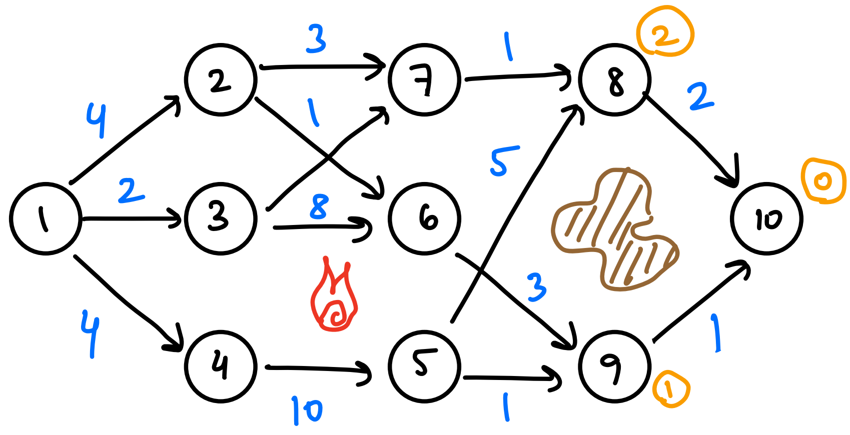
Q What's your strategy to solve this?

A Work backwards!

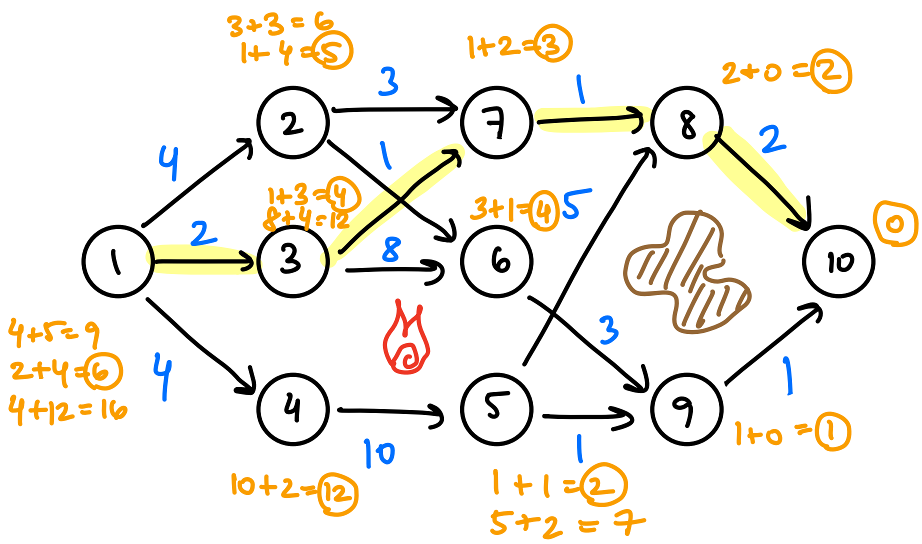


0 First, if I'm already @ ⑩, then cost = 0

i.e. $\ell(x_T = 10) = 0$



Now, you look @ one timestep backwards @ node 8 and 9 and evaluate what is the cost to go from 8 → 10 & 9 → 10; pick the min!



It's more interesting for node 5. There are 2 ways to go ↗ → but we only have to look one step ahead at the next nodes optimal cost to go!

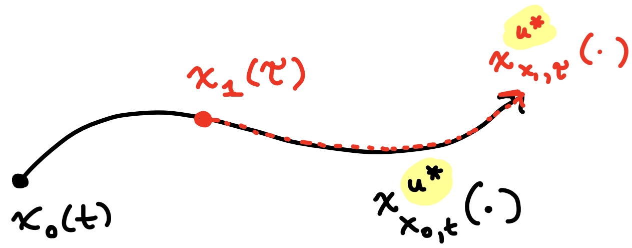
There are a few key properties of Dynamic programming

- DP gives you the optimal path from all nodes to node 10. You get intermediate sol^n "for free"
- Globally optimal solution
- DP gives computational gains over fwd. sim.

Let's understand underlying mathematical principle. DP relies on:

> Principle of optimality: "In an optimal sequence of decisions or choices, each subsequence must also be optimal. Thus, if we take any state along the opt. state trajectory, then the remaining subtrajectory is also optimal"

In the example earlier, if we take any intermediate node along the optimal route, we still take optimal route to destination.



Let's write this principle down mathematically:

We want to solve:

$$= \sum_{\tau=t}^{T-1} L(x_\tau, u_\tau) + \ell(x_T)$$

"value function" $\leftarrow$ $V_t(x_t) = \min_{u_{t:T-1}} \boxed{J_t}(x_t, u_{t:T-1})$

define V_t as storing the best-case "cost-to-go" from x @ time t to the end.

Let's expand out the RHS over time w/ our cost function:

depends on (x_t, u_t)

$$V_t(x_t) := \min_{u_{t:T-1}} \Big\{ L(x_t, u_t) + \underbrace{L(x_{t+1}, u_{t+1}) \cdots L_{T-1}(x_{T-1}, u_{T-1}) + \ell(x_T)}_{} \Big\}$$

$$= \min_{u_{t:T-1}} \Big\{ L(x_t, u_t) + J_{t+1}(x_{t+1}, u_{t+1:T-1}) \Big\}$$ cost from next state x_{t+1} onwards

$$= \min_{u_t} \Big\{ L(x_t, u_t) + \underbrace{\Big[\min_{u_{t+1:T-1}} J_{t+1}(x_{t+1}, u_{t+1:T-1}) \Big]}_{:= V_{t+1}} \Big\}$$ pull other min inside b/c u_t only infl. L & x_{t+1}

$$= \min_{u_t} \Big\{ L(x_t, u_t) + V_{t+1}(x_{t+1}) \Big\}$$ principle of optimality! just restrict ourselves to optimal trajectories starting from the next state

$$= \min_{u_t} \Big\{ L(x_t, u_t) + V_{t+1}(f(x_t, u_t)) \Big\}$$

no longer need to optimize over sequence, only current action!

w/ terminal condition $V_T(x) = \ell(x)$

ASIDE: if x_{t+1} stochastic then:

$$\min_{u_t} \mathbb{E}_{x_{t+1} \sim P(\cdot | x_t, u_t)} \left[L + V(x_{t+1}) \right]$$

Bellman Equation:

$$V_t(x_t) = \min_{u_t} \left[L(x_t, u_t) + V_{t+1}(x_{t+1}) \right],$$

$$V_T(x_T) = \ell(x_T)$$

ASIDE: this is for finite horiz. The discounted form (γ) you may have seen helps for ∞-horiz convergence: $\gamma \cdot V$

- The beauty is this lets us decompose decision-making problems into smaller subproblems and solve recursively, pointwise optim. over ctrl.
- $V(\cdot)$ is typically hard to solve in closed-form for most dynamical systems but for some you can!

$\boxed{\text{Exercise}}$ (offline): Linear Quadratic Regulator (LQR)

i.e. $x_{t+1} = Ax_t + Bu_t$ (lin. dyns.)

$L(x_t, u_t) = x^T Q x + u^T R u$ (quad. cost)