Lecture 4
IR, Spring '24
Andrea Bajcsy

01/29/2024

Last time:

- ☐ optimal control
- ☐ dynamic programming (discrete-time)

This time:

- ☐ dyn. prog. (cont. time)
- ☐ multi-agent games
- ☐ robust opt. ctrl.

Dynamic Programming — Continuous Time:

One of the advantages of D.P. is that it can be used to solve both discrete & continuous-time optimal control problems.

In continuous-time, principle of optimality enables:

$$V(x,t) := \min_{\substack{u(\cdot) \in \mathcal{U}_t^T \\ \text{"all control signals from } [t,T]\text{"}}} \left\{ \int_{\tau=t}^{T} L(x(\tau), u(\tau))\, d\tau + \ell(x(T)) \right\}$$

$$= \min_{u(\cdot) \in \mathcal{U}_t^T} \left\{ \int_{\tau=t}^{t+\delta} L(x(\tau), u(\tau))\, d\tau + \int_{s=t+\delta}^{T} L(x(s), u(s))\, ds + \ell(x(T)) \right\}$$

$$= \underbrace{\min_{u(\cdot) \in \mathcal{U}_t^{t+\delta}} \left\{ \int_{\tau=t}^{t+\delta} L(x(\tau), u(\tau))\, d\tau \right\}}_{\text{cost from } [t,\, t+\delta]} + \underbrace{\min_{u(\cdot) \in \mathcal{U}_{t+\delta}^{T}} \left\{ \int_{s=t+\delta}^{T} L(x(s), u(s))\, ds + \ell(x(T)) \right\}}_{\substack{:= V(x(t+\delta),\, t+\delta) \\ \text{By def. of optimal cost from } t+\delta \text{ onwards}}}$$

$$= \min_{u(\cdot) \in \mathcal{U}_t^{t+\delta}} \left\{ \int_{\tau=t}^{t+\delta} L(x(\tau), u(\tau))\, d\tau + V(x(t+\delta), t+\delta) \right\}$$

Lets focus on studying what happens when $\delta \to 0$. For now, assume $V(\cdot,\cdot)$ is everywhere differentiable. Lets go through an informal derivation based on finite-element analysis of how V changes when $\delta > 0$ but is very small.

$$V(x,t) = \min_{u(\cdot) \in \mathcal{U}_t^{t+\delta}} \left\{ \int_{\tau=t}^{t+\delta} L(x(\tau), u(\tau))\, d\tau + V(x(t+\delta), t+\delta) \right\}$$

$$V(x,t) \approx \min_{u(t) \in \mathcal{U}} \left\{ L(x(t), u(t)) \cdot \delta + \underline{V(x(t+\delta), t+\delta)} \right\}$$

Lets simplify by taking Taylor Series Expansion of $V(x(t+\delta), t+\delta)$ around current $(x(t), t)$.

$$V(x(t+\delta), t+\delta) \approx V(x(t), t) + \frac{\partial V}{\partial x}\cdot \underbrace{(x(t+\delta) - x(t))}_{\approx f(x(t), u(t))\cdot \delta} + \frac{\partial V}{\partial t}\cdot (t+\delta-t) + \text{h.o.t.}$$

Plugging expansion of $V(x(t+\delta), t+\delta)$ back in:

$$V(x,t) \approx \min_{u(t)\in \mathcal{U}} \Big\{ L(x(t), u(t))\cdot \delta + \boxed{V(x(t), t)} + \frac{\partial V}{\partial x}\cdot f(x(t), u(t))\,\delta + \boxed{\frac{\partial V}{\partial t}\cdot \underbrace{(t+\delta-t)}_{=\delta}} + \text{h.o.t.} \Big\}$$

$$\cancel{V(x,t)} = \cancel{V(x(t),t)} + \frac{\partial V}{\partial t}\cdot \delta + \min_{u(t)\in \mathcal{U}} \Big\{ L(x(t), u(t))\,\delta + \frac{\partial V}{\partial x}\cdot f(x(t), u(t))\,\delta \Big\}$$

$$0 = \frac{\partial V}{\partial t}\cdot \delta + \min_{u(t)\in \mathcal{U}} \Big\{ L(x(t), u(t))\,\delta + \frac{\partial V}{\partial x}\cdot f(x(t), u(t))\,\delta \Big\}$$

$$0 = \delta \left[\frac{\partial V}{\partial t} + \min_{u(t)\in \mathcal{U}} \Big\{ L(x(t), u(t)) + \frac{\partial V}{\partial x}\cdot f(x,u) \Big\} \right]$$

← must hold true for all $\delta > 0$, can divide both sides by δ to get:

Hamilton-Jacobi-Bellman (HJB) Equation

$$\frac{\partial V}{\partial t} + \underbrace{\min_{u(t)\in \mathcal{U}} \Big\{ L(x(t), u(t)) + \frac{\partial V}{\partial x}\cdot f(x,u) \Big\}}_{\text{called Hamiltonian}} = 0$$

$$V(x, T) = \ell(x)$$

- ⊛ continuous-time counterpart of Bellman Equation
- ⊛ terminal time PDE
- ⊛ optimal control:

$$u^*(x,t) = \arg\min_{u\in \mathcal{U}} \Big\{ L(x,u) + \frac{\partial V}{\partial x}\cdot f(x,u) \Big\}$$

! key equations that we will solve during safety analysis!

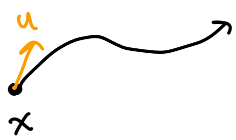
Let's add uncertainty back in! For now, we will focus on non-deterministic uncertainty, $d \in \mathcal{D}$, but I want to note that in the paper discussions + later in the course, we will see probabilistic uncertainty.

We will formulate this as a <u>robust</u> problem. More specifically, as a

$$\underbrace{\text{zero-sum}}_{\text{there is winner \& a loser}} \quad \underbrace{\text{dynamic}}_{\text{game evolves over time}} \quad \underbrace{\text{game}}_{\text{result of the interaction depends on 2+ players}}$$

Recall, <u>before</u>: $\longrightarrow$ <u>Now</u>:

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

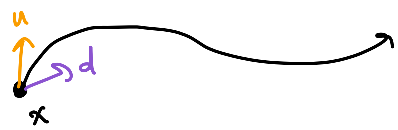


robot's input

other agent's input

"d" for disturbance

$\dot{x} = f(x, u, d)$



Maybe, one way we could pose the optimization problem:

$$V(x,t) = \min_{u(\cdot) \in \mathbb{U}_t^T} \max_{d(\cdot) \in \mathbb{D}_t^T} J(x, u(\cdot), d(\cdot), t)$$

$$\text{s.t. } \dot{x}(\tau) = f(x(\tau), u(\tau), d(\tau)), \quad \forall \tau \in [t, T]$$
$$u(\tau) \in \mathcal{U} \subseteq \mathbb{R}^m$$
$$d(\tau) \in \mathcal{D} \subseteq \mathbb{R}^d$$

INFORMATION PATTERNS:

When we have 2 players reacting to each other, their optimal strategy will depend on what information they each have access to.

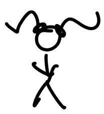
EXAMPLE Suppose there are 2 boxes, each with 2 slots. Each slot contains some prize money. Player A (you ☺) wants to maximize the prize money, while Player B (e.g. competition org.) wants to minimize Player A's prize money.

Box 1

s1	s2
2,000	10

Box 2

s1	s2
1	6,000



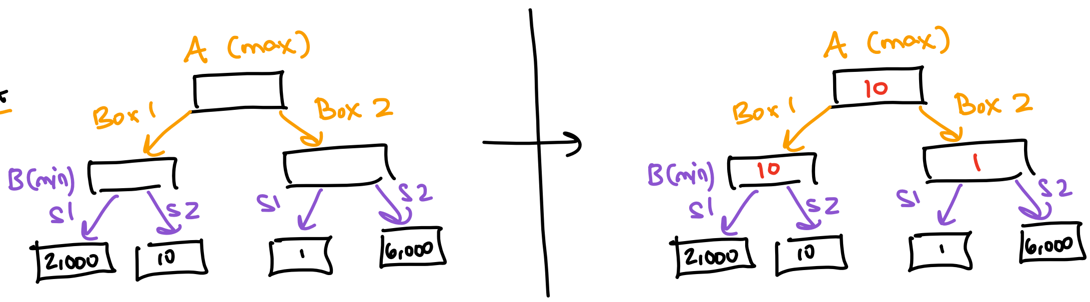
Player A

$u \in [\text{Box 1}, \text{Box 2}] = \mathcal{U}$

Player B

$d \in [\text{slot 1}, \text{slot 2}] = \mathcal{D}$

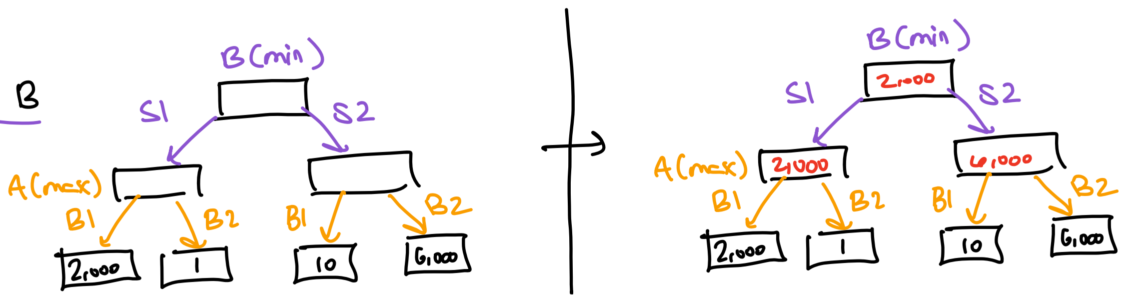
Suppose Player A goes first:



⇒ Best outcome for A is to pick Box 1, and get \$10.

$$\max_{u\in[B1,B2]}\left(\min_{d\in[s1,s2]} J(u,d)\right) = 10$$ ← Mathematically

Suppose Player B goes first:



Best strategy of Player B is to choose slot 1 and pay player A a reward of \$2,000.

$$\min_{d\in[s1,s2]}\left(\max_{u\in[B1,B2]} J(u,d)\right) = 2{,}000$$

This can be stated formally via the <u>minimax inequality</u>:

$$\max_{a}\left(\min_{b} J(a,b)\right) \leq \min_{b}\left(\max_{a} J(a,b)\right)$$

just talked about order of play

what about this?

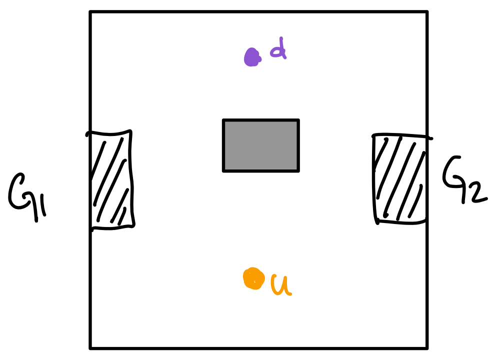
In dynamic games, the outcome depends on <u>WHEN</u> and <u>WITH WHAT INFO</u>. each player decides their inputs.

In fact, earlier I wrote one kind of information pattern:

<u>OPEN-LOOP INFORMATION PATTERN</u>

$$V(x,t) = \min_{u(\cdot)\in \mathbb{U}_t^T} \max_{d(\cdot)\in \mathbb{D}_t^T} J(x, u(\cdot), d(\cdot), t)$$

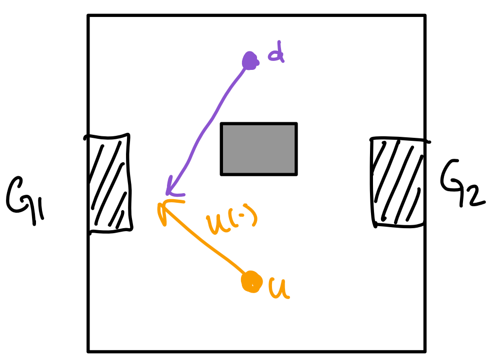
Let's consider example.



Here, u wants to get to G1 or G2, without interception from d. But d wants to intercept u.

1) <u>OPEN-LOOP INFO. PATTERN</u>

$$V(x,t) = \min_{u(\cdot)\in \mathbb{U}_t^T} \max_{d(\cdot)\in \mathbb{D}_t^T} J(x, u(\cdot), d(\cdot), t)$$



$u(\cdot)$ will declare it's <u>entire</u> control signal $[t,T]$

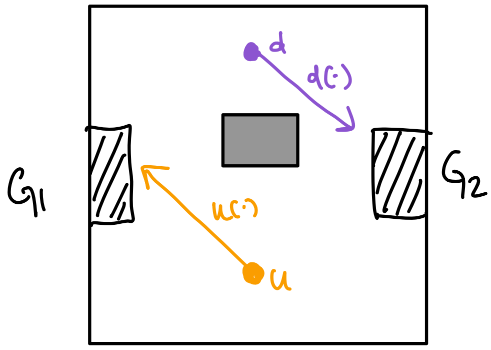
$d(\cdot)$ gets to see this entire signal before choosing its own.

"d can see "future" and u cannot change their mind"

$\Rightarrow$ OVERLY PESSIMISTIC

SWAP ORDER OF PLAY:

$$V(x,t) = \max_{d(\cdot)\in\mathbb{D}_t^T} \min_{u(\cdot)\in\mathbb{U}_t^T} J(x, u(\cdot), d(\cdot), t)$$



d(·) will commit /declare its entire control signal from [t,T]

u(·) gets to see this before choosing its own.

⇒ overly optimistic

2) CLOSED-LOOP (FEEDBACK) INFORMATION PATTERNS

These above formulations are not suitable for most practical systems.

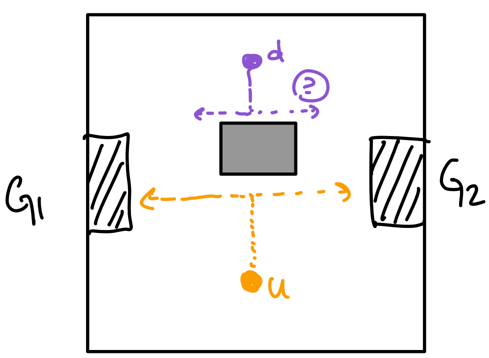
We would like the controller to adapt as the interaction evolves,

BUT respect the fact that @ time t, we only have information up to time t.

In differential games, such strategies are called

NON-ANTICIPATIVE STRATEGIES:

$$V(x,t) = \min_{u(\cdot)\in\Gamma_u} \max_{d(\cdot)\in\Gamma_d} J(x, u(\cdot), d(\cdot), t)$$



In this case, u(·) does not have to declare that they want to go left or right @ start of interaction.

Hence, d(·) cannot simply intercept u(·) before it reaches goal

Definition: Non-anticipative strategies:

$$\Gamma_d(t,T) = \{ \delta : \mathbb{U}_t^T \to \mathbb{D}_t^T \text{ s.t. if } \forall u_1(\cdot), u_2(\cdot) \in \mathbb{U}_t^T, \forall \tau \in [t,T]$$

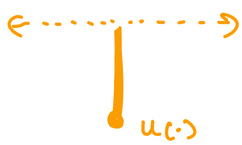
$$(u_1(s) = u_2(s) \quad \text{a.e.} \quad s \in [t,\tau]) \Rightarrow$$

"if the controls are the same almost everywhere.."

$$(\delta[u_1](s) = \delta[u_2](s) \quad \text{a.e.} \quad s \in [t,\tau]) \}$$

" then the disturbance must react the same way over that t-hor.

- δ is a map from u's control signals to d's ctrl. signals
- Intuition: the disturbance δ cannot pre-emptively start to adapt to a change in u UNTIL SUCH A CHANGE BEGINS!



} d cannot change behavior until u does