Lecture 3
IR, Spring '24
Andrea Bajcsy

01/24/24

Last time:

- [ ] dynamical systems
- [ ] need for safety
- [ ] uncertainty models

This time:

- [ ] optimal control
- [ ] dynamic programming

Optimal Control: The problem of optimal decision-making. In fact as we will see in the coming lectures, the <u>safety problem</u> can be posed as an optimal control problem. Right now: recap + how to solve.

minimize (decision)	objective	
s.t.	system dynamics	ctrl bounds
		obstacles
	other constraints	state constraints

An opt. control problem is an optimization problem, but with a temporal aspect to the decisions (i.e. optimization variable)

Examples.

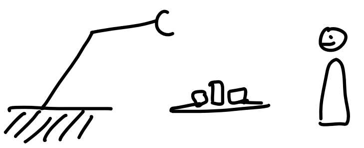
(1)

<u>objective</u>: goal reach, smoothness, fuel efficiency

<u>constraints</u>: collision avoid w/ vehicles, lane keeping

<u>system dynamics</u>: vehicle model

(2)

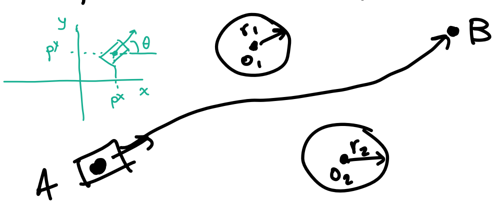


<u>objective</u>: food in mouth, reduce human effort/movement, don't want to spill, personalization (preference over food handling

<u>dynamics</u>: arm, morsels + end-effector interaction dynamics

<u>constraints</u>: safety (don't contact torso) (food + robot), smoothness, hold fork @ angle that is "comfortable"

$\boxed{\text{EXAMPLE}}$ move ground vehicle from A→B.



<u>objective</u>: reach B, fuel efficiency

<u>constraints</u>: obstacles, ctrl. bounds

<u>dynamics</u>: vehicle (x, y, θ)

running cost: <u>goal reach</u> — running cost: <u>effort</u> — <u>terminal cost</u>

$$\underset{u_{0:T-1}}{\text{minimize}} \sum_{t=0}^{T-1} \left\| \begin{matrix} p_t^x - B^x \\ p_t^y - B^y \end{matrix} \right\|^2 + \lambda \sum_{t=0}^{T-1} u_t^2 + \left\| \begin{matrix} p_T^x - B^x \\ p_T^y - B^y \end{matrix} \right\|^2$$

$$\text{s.t.} \quad \left.\begin{aligned} p_{t+1}^x &= p_t^x + \Delta T \cdot v \cos\theta \\ p_{t+1}^y &= p_t^y + \Delta T \cdot v \sin\theta \\ \theta_{t+1} &= \theta_{t+1} + \Delta t \cdot u_t \end{aligned}\right\} = x_{t+1} = f(x_t, u_t)$$

obstacle center

$$\forall o_i \in \Theta, \quad d(\underbrace{[p_t^x, p_t^y]}_{\text{position of vehicle}}, o_i) > r_i$$

set of obstacles — obstacle radius

e.g. $(p_t^x - o^x)^2 + (p_t^y - o^y)^2 > r^2$

$$p_0^x = A^x, \quad p_0^y = A^y, \quad \theta_0 = \theta^{init}$$
$$|u_t| \leq \omega^{max}, \quad t \in \{0, \dots, T\}$$

DISCRETE-TIME:

total cost starting from x_0, applying $u_{0:T-1}$

running — terminal

$$\min_{u_{0:T-1} \in \mathcal{U}_0^{T-1}} J(x_0, u_{0:T-1}) := \sum_{t=0}^{T-1} L(x_t, u_t) + \ell(x_T)$$

"set of all ctrl. sequences $u_0, u_1, u_2, \dots u_{T-1}$"

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

$\underline{u} \leq u_t \leq \overline{u}$, $\forall t \in \{0, \dots, T-1\}$ ← control bounds

CONTINUOUS-TIME:

$$\min_{u(\cdot) \in \mathcal{U}_0^T} J(x(0), u(\cdot)) := \int_{\tau=0}^{T} L(x(\tau), u(\tau)) d\tau + \ell(x(T))$$

"set of all ctrl. signals: $u(\cdot) : [t_0, T] \to \mathcal{U} \subseteq \mathbb{R}^m$"

s.t. $\dot{x} = f(x(t), u(t))$ $\forall t \in [0, T]$

$\underline{u} \leq u(t) \leq \overline{u}$

$$\boxed{\text{BIG Q: HOW TO ACTUALLY SOVE IT ?}}$$

Given the popularity of optimal control problems, there are a variety of tools / methods / algorithms for solving these:

(1) calculus of variations { "convert to unconstrained optimization problem via Lagrange multipliers"

(2) model predictive control { "similar @ high level to (1) but usually in discrete-time and continually resolve the optimization problem "receding horiz.""

(3) dynamic programming ← ⊛ going to focus on this b/c lay the foundation for safety analysis

(4) reinforcement learning { similar foundation to (3) but SOTA alg.s, approx. representation, simulation, and data to scale

Dynamic Programming — INTRO & DISCRETE-TIME O.C. (optimal control) Problem

Introduced by Richard Bellman in 1958 @ RAND corporation.

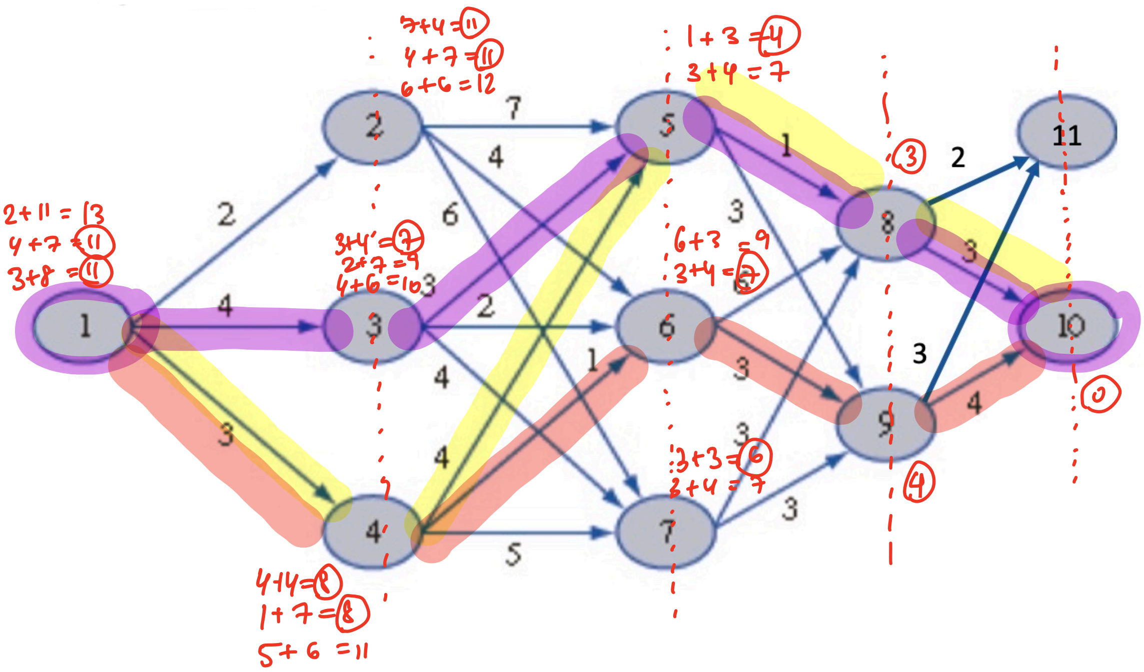
Bellman explains the name

The 1950s were not good years for mathematical research. We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word, research... His face would suffuse, he would turn red, and he would get violent if people used the term, research, in his presence. You can imagine how he felt, then, about the term, mathematical... I thought dynamic programming was a good name. It was something not even a Congressman could object to.

key idea:

principle of optimality

Find the shortest path from node 1 to node 10 in the network shown below.



<u>key properties of D.P.</u>:

(1) D.P. gives you an optimal path from <u>all nodes</u> to node 10. So you get intermediate solutions "for free"

(2) B/C exploring all intermediate trajectories, you globally optimal solution.

(3) D.P. gives significant computational savings over forward simulation.

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

Let's write this down mathematically:

cost from before.

We want to <u>solve</u>: $V_t(x_t) := \min_{u_{t:T-1}} J_t(x_t, u_{t:T-1})$ $\quad \sum L(x_t, u_t) + \ell(x_T)$

stores the best cost value, "cost-to-go" from x_t at time t

$\Rightarrow$ "value function"

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

depend on (x_t, u_t)

$$= \min_{u_{t:T-1}} \{ L(x_t, u_t) + J_{t+1}(x_{t+1}, u_{t+1:T-1}) \}$$

"cost from next state onwards"

$$= \min_{u_{t:T-1}} \{ L(x_t, u_t) + V_{t+1}(x_{t+1}) \}$$

principle of optimality? just restricting ourselves to optimal trajectories starting from next state x_{t+1}!

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

no need to optimize over sequence, only current action.

Bellman Equation $V_t(x_t) = \min_{u_t \in \mathcal{U}} \{ L(x_t, u_t) + V_{t+1}(x_{t+1}) \}$ $V_T(x_T) = \ell(x_T)$

(*) Beauty in the decomposition of decision-making into smaller subproblems solved recursively, pointwise optimizations over control

(*) $V(\cdot)$ is typically hard to solve in closed form for most dynamical systems & costs $\Rightarrow$ exception: LINEAR QUADRATIC REGULATOR! (LQR)

Q How do you get optimal control from this?

By defition of the value function, the optimal control is given by the value function minimizer:

$$u_t^*(x_t) := \arg\min_{u_t \in \mathcal{U}} \{ L(x_t, u_t) + V_{t+1}(x_{t+1}) \}$$

(*) will be useful later during human prediction!

$$Q(x_t, u_t) := L(x_t, u_t) + V_{t+1}(x_{t+1}) \quad // \text{ state-action value func.}$$

$$u_t^*(x_t) = \arg\min_{u_t} Q(x_t, u_t)$$

↳ feedback controller! B/c it depends on the state @ each t!

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:

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

"all control signals from $[t, T]$"

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

With the principle of optimality, we can create 2 subproblems from time $\tau = [t, t+\delta]$ and the other for finding ctrl signal from $\tau = [t+\delta, T]$

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

TO BE CONTINUED ...