Mod4-RL2: Dynamic Programming & DT LQR

References:

- Chapter 2.1.7, 10.4/10d.4 Callier & Desoer [C&D]
- Lecture 10 (preview) and 20 — 22 Hespanha [JH]

DT LQR Problem

Objective: $J(u) = x_N^\top Q_f x_N + \sum_{k=0}^{N-1} \left(x(k)^\top Q x(k) + u(k)^\top R u(k) \right)$

state dynamics: $x_{t+1} = Ax_t + Bu_t$, initial data x_0

control input: $u = (u_0, \ldots, u_{N-1}) \in \mathbb{R}^{N \cdot m}$

LQR Problem: $\min_u \{ J(u) \mid x_{t+1} = Ax_t + Bu_t \quad \forall t = 0, \ldots, N \}$

Cost Matrices: $Q = Q^\top \succeq 0,\ Q_f = Q_f^\top \succeq 0,\ R = R^\top \succ 0$

Comparing to Similar Problems

- Least norm input that steers x_0 to the origin:
  - cost: $\|u\|^2 = u^\top u$ (i.e, $R = I$)
  - final state: $x_N = 0$
  - no state cost: $Q = 0$
  - final state cost: $Q_f \gg I$ (e.g., $Q_f = 10^8 \cdot I$)

- multi-objective optimization
  - cost matrices: $R = \rho \cdot I,\ Q = Q_f = C^\top C$

$$J(u) = \underbrace{\sum_{k=0}^{N} \|y(k)\|^2}_{\text{output cost: } J_o(u)} + \rho \underbrace{\sum_{k=0}^{N-1} \|u(k)\|^2}_{\text{input cost: } J_i(u)} , \quad y = Cx$$

Comparing to Similar Problems

- LQR as Least Squares:
  - $X := (x_0, \ldots, x_N), \quad u := (u_0, \ldots, u_{N-1})$
- Stacked up state equation:

$$\underbrace{\begin{bmatrix} x_0 \\ \vdots \\ x_N \end{bmatrix}}_{X} = \underbrace{\begin{bmatrix} 0 & \cdots & \cdots & \cdots \\ B & 0 & \cdots & \cdots \\ AB & B & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}}_{G} \underbrace{\begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix}}_{u} + \underbrace{\begin{bmatrix} I \\ A \\ \vdots \\ A^N \end{bmatrix}}_{H} x_0$$

- Least squares cost:

$$J(u) = \left\| \text{diag}(Q^{1/2}, \ldots, Q^{1/2}, Q_f^{1/2})(Gu + Hx_0) \right\|^2 + \left\| \text{diag}(R^{1/2}, \ldots, R^{1/2})u \right\|^2$$

Bellman's Principle

Consider the general DT dynamics

and suppose we want to minimizer the cost

Dynamic Programming

- gives an efficient, recursive method to solve LQR least-squares problem; cost is $O(N \cdot n^3)$
- DP is a useful and important idea on its own (it is applied in a number of domains including search, RL, MDPs, etc.)

Definition. For $t = 0, \ldots, N$, define the value function $V_t : \mathbb{R}^n \to \mathbb{R}$ by

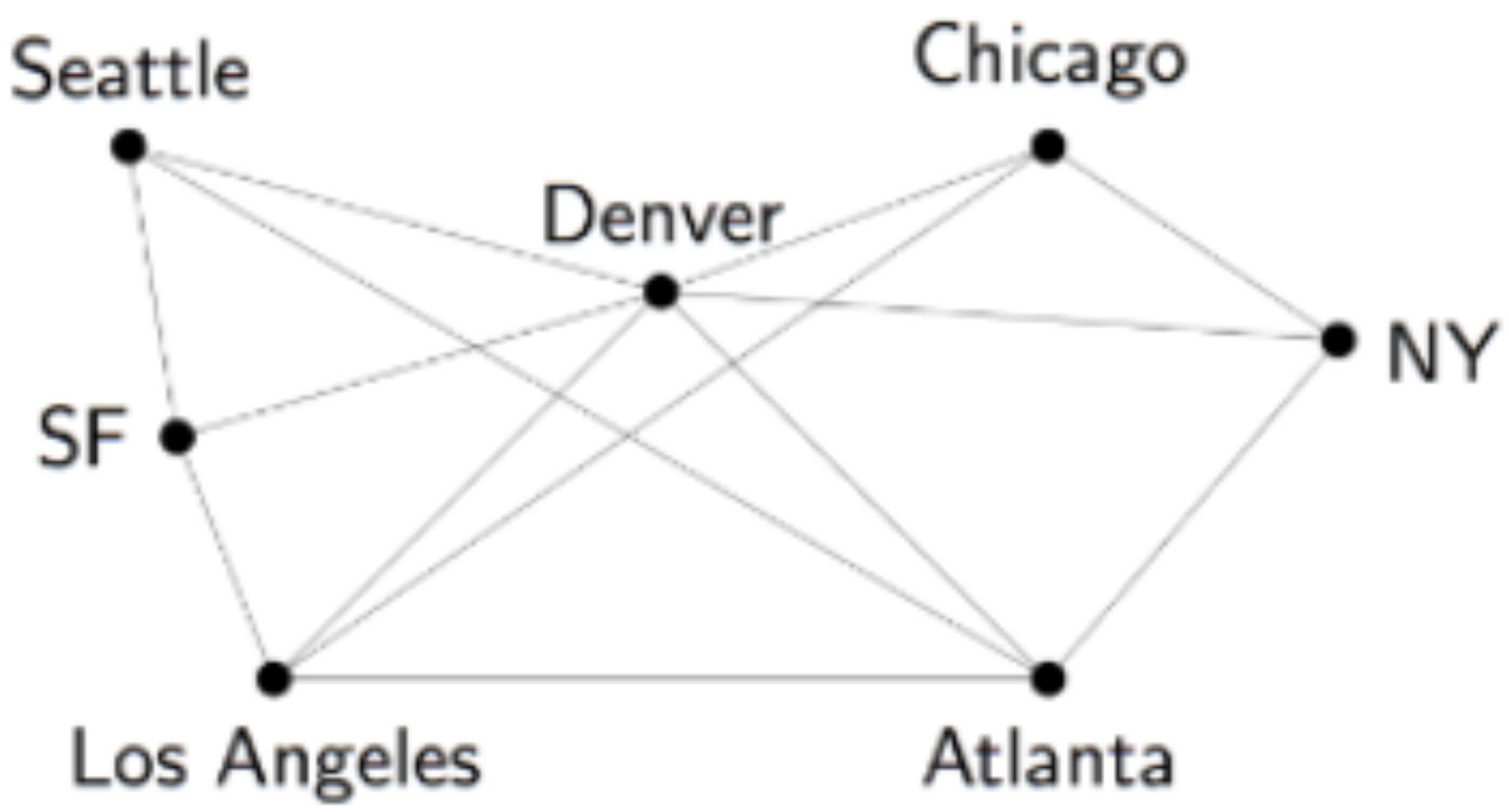
$$V_t(z) = \min_{u_t, \ldots, u_{N-1}} \sum_{k=t}^{N-1} (x_k^\top Q x_k + u_k^\top R u_k) + x_N^\top Q_f x_N \qquad \text{s.t. } x_t = z,\ x_{k+1} = Ax_k + Bu_k\ \ k = t, \ldots, N$$

- $V_t(z)$ is the minimum LQR cost-to-go starting from state z at time t
- $V_0(x_0)$ is the minimum LQR cost

Dynamic Programming Principle

Q: suppose we know $V_{t+1}(z)$. What is the optimal choice for u_t?

Example: Path Optimization



Showing the value function is quadratic

Showing the value function is quadratic

Summary of LQR via DP

- step 1: `set` $P_N = Q_f$
- step 2: `for` $t = N, \ldots, 1,$ `do` :

$$P_{t-1} = Q + A^\top P_t A - A^\top P_t B(R + B^\top P_{t+1} B)^{-1} B^\top P_t A$$

- step 3: `for` $t = N, \ldots, 1,$ `set` :

$$K_t = -(R + B^\top P_{t+1} B)^{-1} B^\top P_{t+1} A$$

- step 4: `for` $t = N, \ldots, 1,$ `set` :

$$u_t^\star := K_t x_t$$