EE365: Deterministic Finite State Control

Deterministic optimal control

Shortest path problem

Dynamic programming

Examples

Outline

Deterministic optimal control

Shortest path problem

Dynamic programming

Examples

Deterministic dynamic system

$$x_{t+1} = f_t(x_t, u_t), \quad t = 0, 1, \ldots$$

- t is time period or epoch
- $x_t \in \mathcal{X}_t$ is state
- $u_t \in \mathcal{U}_t$ is input, action, or control
  (variation: $u_t \in \mathcal{U}_t(x_t)$, *i.e.*, $\mathcal{U}_t$ depends on x_t)
- $f_t : \mathcal{X}_t \times \mathcal{U}_t \to \mathcal{X}_{t+1}$ is state transition function
- initial state x_0 is given
- common special case: f_t, $\mathcal{X}_t$, $\mathcal{U}_t$ do not depend on t
  called **time-invariant** (TI) system

Deterministic optimal control

$$\begin{array}{ll} \text{minimize} & J = \sum_{t=0}^{T-1} g_t(x_t, u_t) + g_T(x_T) \\ \text{subject to} & x_{t+1} = f_t(x_t, u_t), \quad t = 0, \ldots, T-1 \end{array}$$

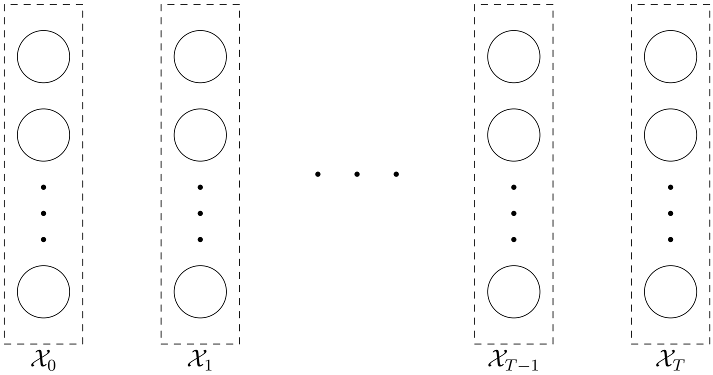
- variables are $x_1, \ldots, x_T,\ u_0, \ldots, u_{T-1}$; x_0 is given
- $g_t : \mathcal{X}_t \times \mathcal{U}_t \to \mathbf{R} \cup \{\infty\}$ is stage cost function
- $g_T : \mathcal{X}_T \to \mathbf{R} \cup \{\infty\}$ is terminal cost function
- infinite values of stage and terminal costs encode (state/action) constraints
- just an optimization problem (trivial information pattern)
- called TI when dynamic system is TI and g_t doesn't depend on t

Outline

Finite state/action deterministic control

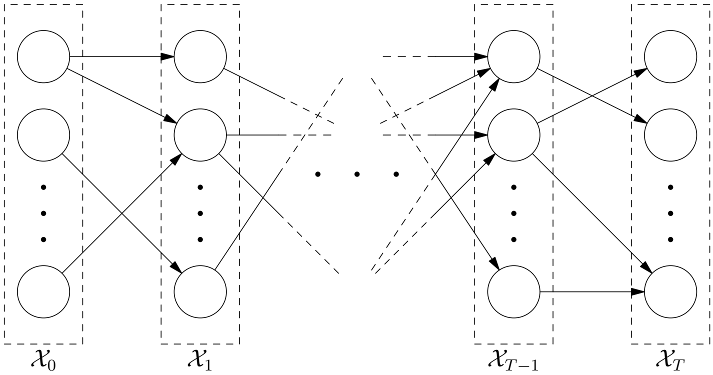
- now suppose $\mathcal{X}_t$, $\mathcal{U}_t$ are finite
- create **unrolled graph**
  - vertices are $\mathcal{X}_0 \cup \cdots \cup \mathcal{X}_T$ (separate copies for each t)
  - directed edges labeled by u_t from x_t to $x_{t+1} = f_t(x_t, u_t)$ (can have multple edges from x_t to x_{t+1})
  - edge weights are $g_t(x_t, u_t)$; nodes in $\mathcal{X}_T$ have weights $g_T(x_T)$
  - a sequence of actions is a path through the unrolled graph, starting at x_0 and ending in $\mathcal{X}_t$
  - associated objective J is (total, weighted) path length

Unrolled graph



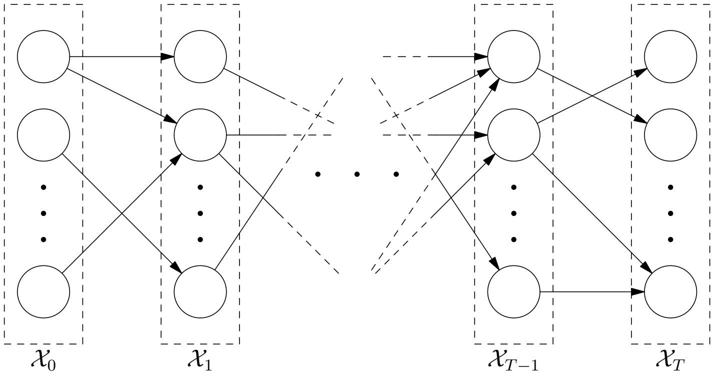
vertex set is $\mathcal{X}_0 \cup \cdots \cup \mathcal{X}_T$

Unrolled graph



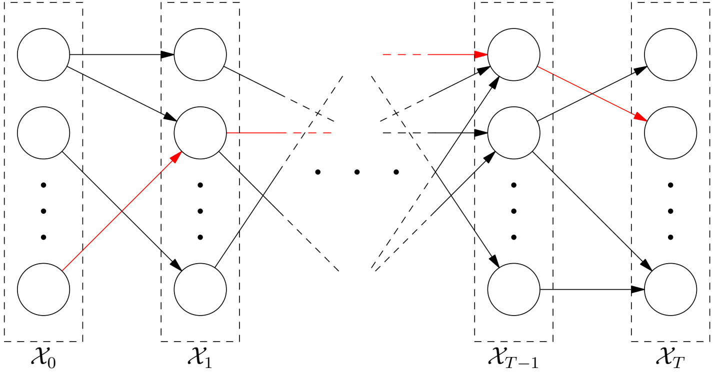
directed edges, labeled by u_t, from x_t to $x_{t+1} = f_t(x_t, u_t)$

Unrolled graph



edge weights are $g_t(x_t, u_t)$; nodes in $\mathcal{X}_T$ have weights $g_T(x_T)$

Unrolled graph



a sequence of actions is a path through the unrolled graph

Deterministic control via shortest path

- control/action sequence is a path through the unrolled graph
- J is total path weight
- deterministic optimal control problem is a **shortest path problem**
- many methods to solve; we'll focus on one, that we'll see later: **dynamic programming** (DP)

Outline

Value function

- define **tail problem** from $x_t = z$ as

$$\begin{array}{ll} \text{minimize} & J_t = \sum_{\tau=t}^{T-1} g_\tau(x_\tau, u_\tau) + g_T(x_T) \\ \text{subject to} & x_{\tau+1} = f_\tau(x_\tau, u_\tau), \quad \tau = t, \ldots, T-1 \\ & x_t = z \end{array}$$

  with optimal value $V_t(z)$

- $V_t : \mathcal{X}_t \to \mathbf{R} \cup \{\infty\}$ is called the **optimal value function**, **cost-to-go function**, or **Bellman function**
- $V_t(z)$ is the minimum cost-to-go if we are in state z at time t
- $V_T(z) = g_T(z)$
- $J^\star = V_0(x_0)$

Dynamic programming recursion

- optimal action in terms of current state x, value function:

$$u_t \in \underset{u \in \mathcal{U}_t}{\operatorname{argmin}} (g_t(x, u) + V_{t+1}(f_t(x, u)))$$

- in words: in state x at time t, optimal current action minimizes
  - immediate cost $g_t(x, u)$, plus
  - optimal cost from where you land, $V_{t+1}(f_t(x, u))$
- value function recursion:

$$V_t(x) = \min_{u \in \mathcal{U}_t} (g_t(x, u) + V_{t+1}(f_t(x, u)))$$

  - gives V_t in terms of V_{t+1} (and g_t, f_t)

Dynamic programming

backward recursion for value function, optimal policy:

- $V_T(x) = g_T(x)$ for $x \in \mathcal{X}_T$
- for $t = T-1, \ldots, 0$,
  - $\mu_t(x) \in \mathop{\mathrm{argmin}}_{u \in \mathcal{U}_t}(g_t(x,u) + V_{t+1}(f_t(x,u)))$ for $x \in \mathcal{X}_t$
  - $V_t(x) = g_t(x, \mu_t(x)) + V_{t+1}(f_t(x, \mu_t(x)))$ for $x \in \mathcal{X}_t$
- cost is $\sum_{t=0}^{T} |\mathcal{X}_t||\mathcal{U}_t|$ operations ($T|\mathcal{X}||\mathcal{U}|$ in TI case)

What DP gives you

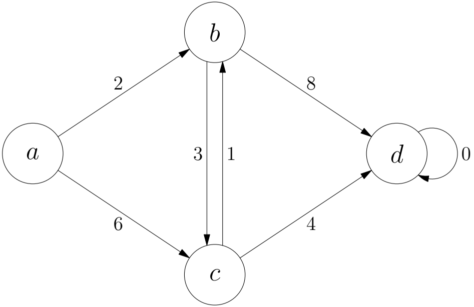
- DP gives **optimal policy** $\mu_t : \mathcal{X}_t \to \mathcal{U}_t,\ t = 0, \ldots, T-1$
- optimal $u_0, \ldots, u_{T-1},\ x_1, \ldots, x_T$ given by recursion

$$u_t = \mu_t(x_t), \quad x_{t+1} = f_t(x_t, u_t), \quad t = 0, \ldots, T-1$$

- in fact, DP gives solution for **any** intial state x_0

Example

find shortest path from a to d (without loss of generality, of length 3)



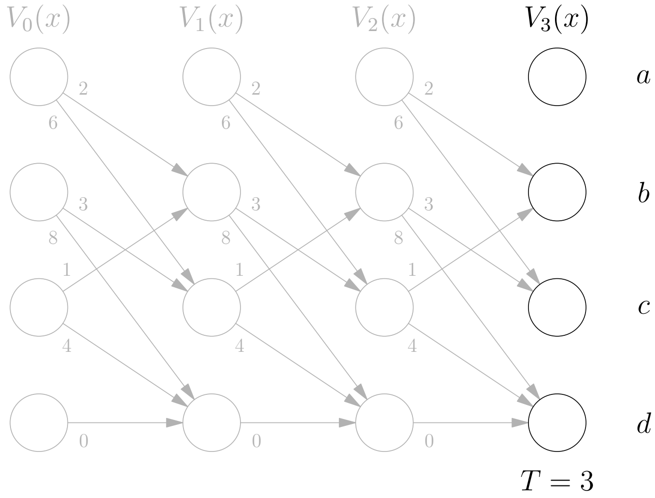
(unique) solution is evidently $a \to b \to c \to d$

Example

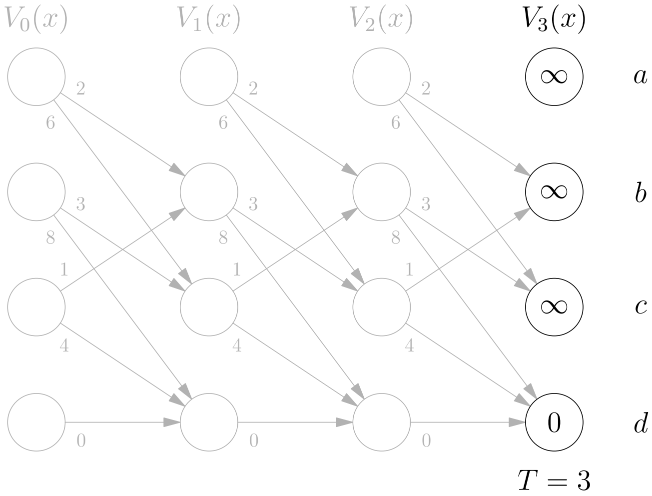
as deterministic optimal control problem:

- $\mathcal{X} = \{a, b, c, d\}$, $x_0 = a$
- $\mathcal{U}(x) = \text{successors}(x)$
- $f(x, u) = u$
- $T = 3$
- $g(x, u)$ is given weight on edge (x, u)
- $g_T(x) = \begin{cases} \infty & x = a, b, c \\ 0 & x = d \end{cases}$ (enforces terminal constraint $x_3 = d$)

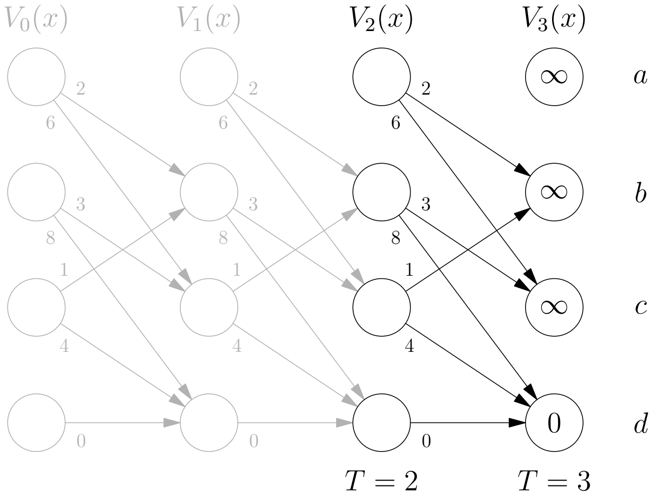
Example



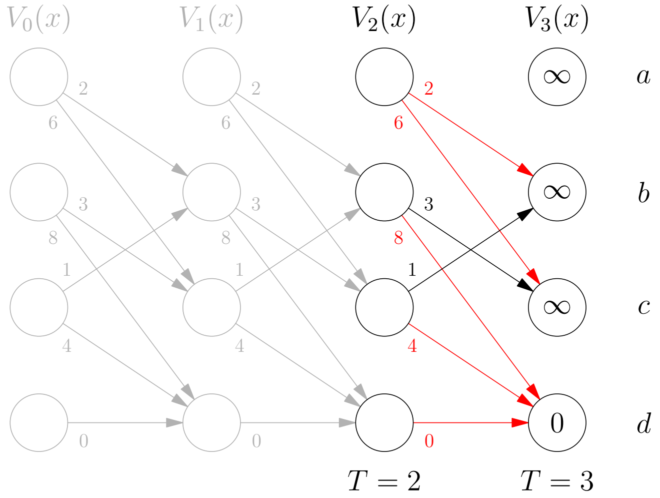
Example



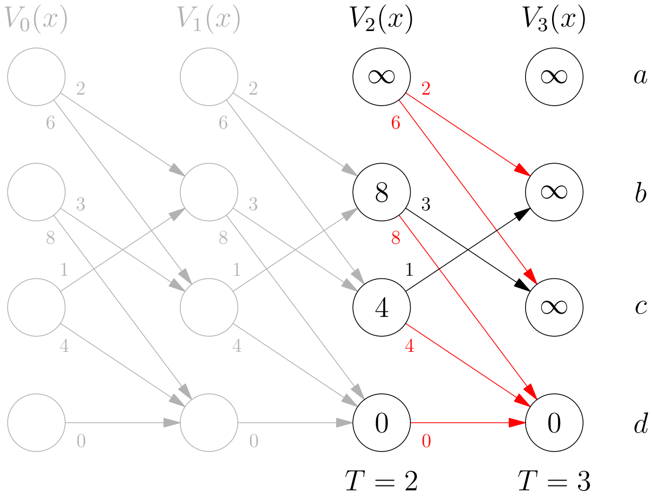
Example



Example

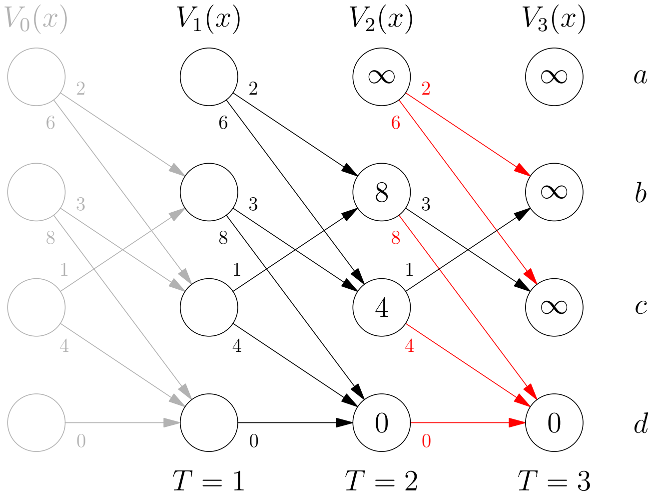


Example



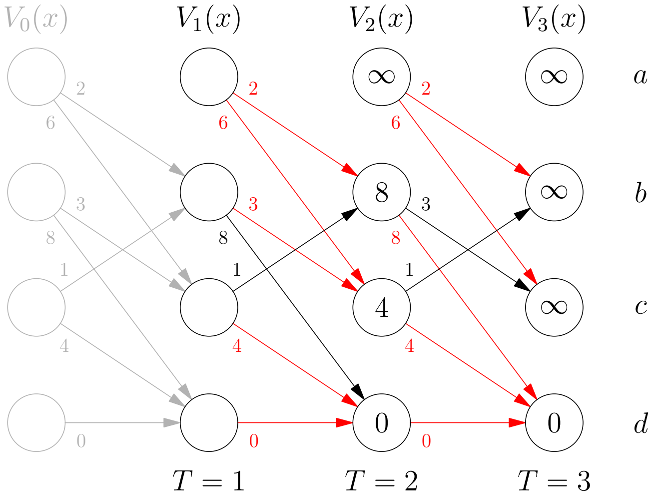
$V_0(x)$
$V_1(x)$
$V_2(x)$
$V_3(x)$
2
6
3
8
1
4
0
2
6
3
8
1
4
0
∞
8
4
0
2
6
3
8
1
4
0
∞
∞
∞
0
a
b
c
d
$T = 2$
$T = 3$

Example



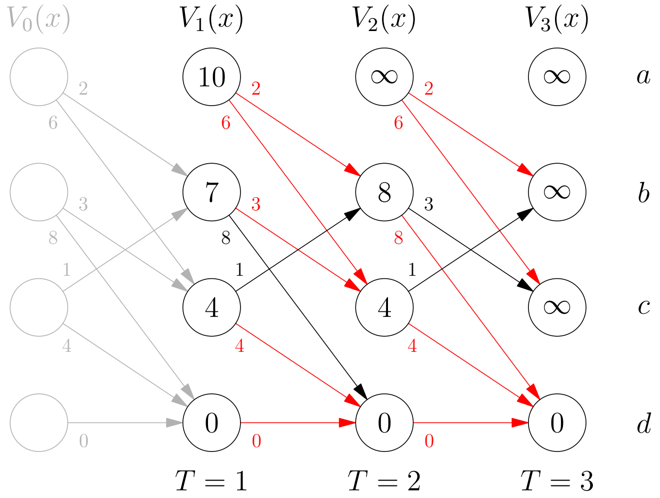
$V_0(x)$
$V_1(x)$
$V_2(x)$
$V_3(x)$
∞
∞
∞
∞
8
4
0
0
a
b
c
d
2
6
3
8
1
4
0
$T = 1$
$T = 2$
$T = 3$

Example

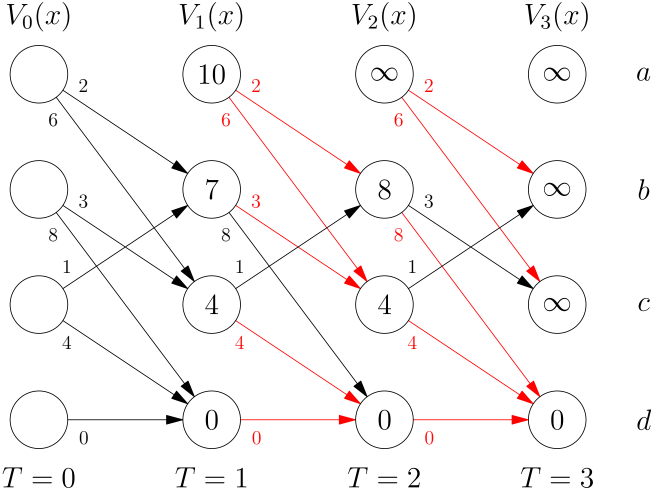


$V_0(x)$
$V_1(x)$
$V_2(x)$
$V_3(x)$
∞
∞
∞
∞
8
4
0
0
a
b
c
d
2
6
3
8
1
4
0
$T=1$
$T=2$
$T=3$

Example

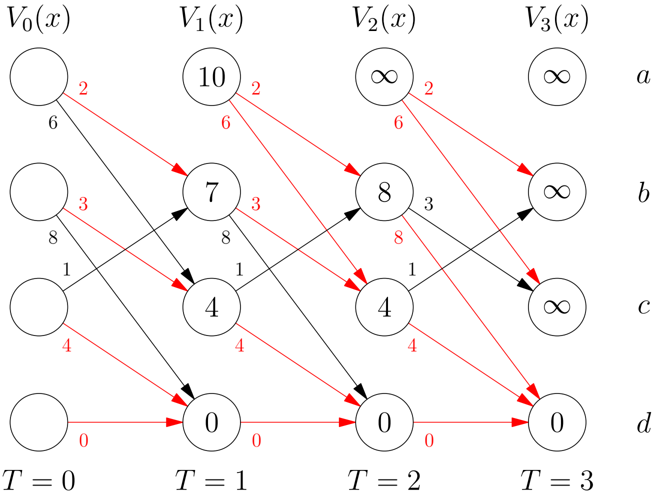

$V_0(x)$
$V_1(x)$
$V_2(x)$
$V_3(x)$
10
∞
∞
7
8
∞
4
4
∞
0
0
0
a
b
c
d
$T = 1$
$T = 2$
$T = 3$

Example

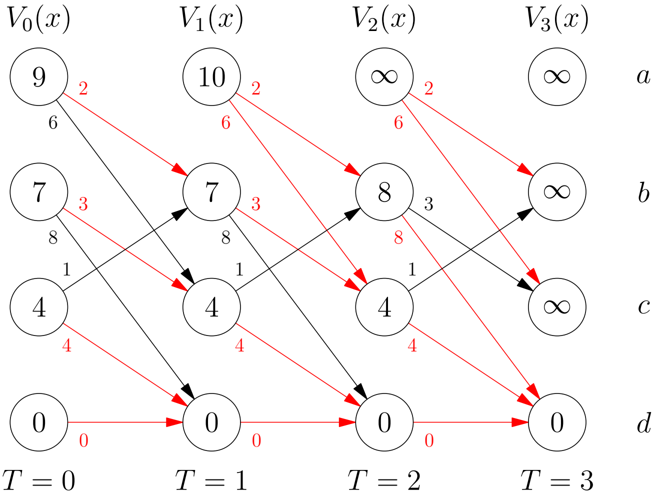


$V_0(x)$
$V_1(x)$
$V_2(x)$
$V_3(x)$
10
∞
∞
a
7
8
∞
b
4
4
∞
c
0
0
0
d
2
6
3
8
1
4
0
$T = 0$
$T = 1$
$T = 2$
$T = 3$

Example

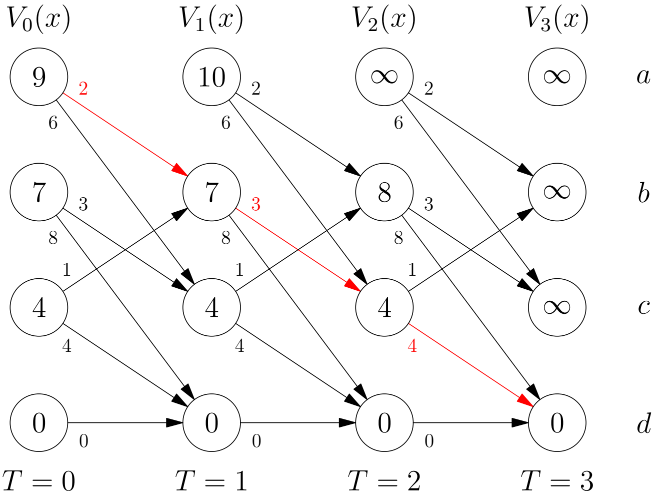

$V_0(x)$
$V_1(x)$
$V_2(x)$
$V_3(x)$
10
∞
∞
7
8
∞
4
4
∞
0
0
0
a
b
c
d
2
6
3
8
1
4
0
$T = 0$
$T = 1$
$T = 2$
$T = 3$

Example



$V_0(x)$
$V_1(x)$
$V_2(x)$
$V_3(x)$
9
10
∞
∞
a
7
7
8
∞
b
4
4
4
∞
c
0
0
0
0
d
2
6
3
8
1
4
0
$T = 0$
$T = 1$
$T = 2$
$T = 3$

Example



Why tail problems and a backward recursion?

answer: for deterministic control problem, there's no reason ...

- we could just as well have worked with initial problems (from $\tau = 0$ to $\tau = t$) instead of tail problems
- would yield forward recursion for W_t (min cost-from-start)
- for solving the stochastic control problem, however, DP will need to run backward

Outline

Hidden Markov state estimation

- $z_t \in \{1, \ldots, n\}$, $t = 0, \ldots, T$ is a Markov chain with
  - transition probabilities $P_{ij} = \mathbf{Prob}(z_{t+1} = j \mid z_t = i)$
  - initial distribution $\pi_j = \mathbf{Prob}(z_0 = j)$
- $y_t \in \{1, \ldots, m\}$, $t = 0, \ldots, T$ is a set of measurements related to z_t by conditional probabilities $Q_{ik} = \mathbf{Prob}(y_t = k \mid z_t = i)$
- we don't know the state sequence $z_0, \ldots, z_T$, but we do know the measurements $y_0, \ldots, y_T$ (and the probabilities P_{ij}, π_j, Q_{ik})
- so we will estimate $z_0, \ldots, z_T$ based on the measurements $y_0, \ldots, y_T$

Maximum a posteriori state estimation

- maximum a posteriori (MAP) estimate of $z_0, \ldots, z_T$, denoted $\hat{z}_0, \ldots, \hat{z}_T$, maximizes $\mathbf{Prob}(z_0, \ldots, z_T \mid y_0, \ldots, y_T)$
- same as maximizing (over $z_0, \ldots, z_T$)

$$\begin{aligned}
&\mathbf{Prob}(z_0, \ldots, z_T)\, \mathbf{Prob}(y_0, \ldots, y_T \mid z_0, \ldots, z_T) \\
&= \left(\mathbf{Prob}(z_0) \prod_{t=0}^{T-1} \mathbf{Prob}(z_{t+1} \mid z_t) \right) \left(\prod_{t=0}^{T} \mathbf{Prob}(y_t \mid z_t) \right) \\
&= \pi_{z_0} \left(\prod_{t=0}^{T-1} P_{z_t, z_{t+1}}\, Q_{z_t, y_t} \right) Q_{z_T, y_T}
\end{aligned}$$

- equivalently, minimize the negative logarithm

$$-\log \pi_{z_0} - \sum_{t=0}^{T-1} \log(P_{z_t, z_{t+1}}\, Q_{z_t, y_t}) - \log Q_{z_T, y_T}$$

- a bad method: evaluate this expression for all n^{T+1} sequences

MAP Markov state estimation as deterministic control problem

- state space $\mathcal{X} = \{1, \ldots, n\}$, action space $\mathcal{U} = \mathcal{X}$
- state-transition function $f(x_t, u_t) = u_t$
- time horizon T
- stage cost $g_t(x_t, u_t) = -\log(P_{x_t, u_t} Q_{x_t, y_t})$
- terminal cost $g_T(x_T) = -\log Q_{x_T, y_T}$
- initial cost $g_T(x_0) = -\log \pi_{x_0}$

an efficient method for MAP estimation of Markov state sequence:

- use DP to find V_0, optimal policy
- find optimal x_0 by minimizing $-\log \pi_{x_0} + V_0(x_0)$

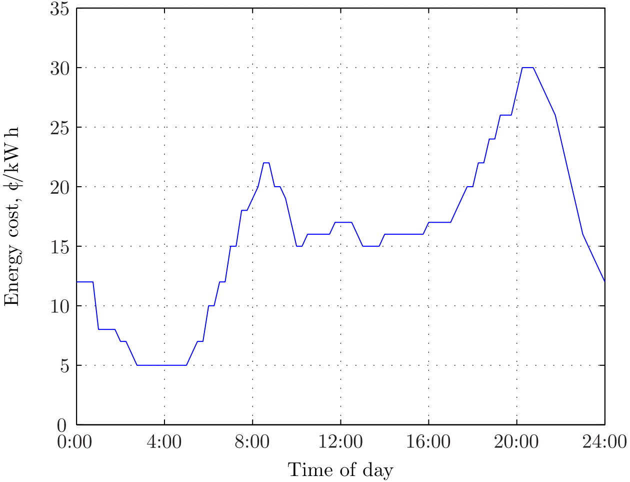
Appliance scheduling

- appliance (say, dishwasher) has five cycles (each 15 min)

cycle		power
1	prewash	1.5 kW
2	main wash	2.0 kW
3	rinse 1	0.5 kW
4	rinse 2	0.5 kW
5	dry	1 kW

- cycles must be run in order, possibly with idle periods
- electricity price varies (in 15 min periods)
- find cheapest cycle schedule starting at 17:00 and ending at 24:00

Electricity price

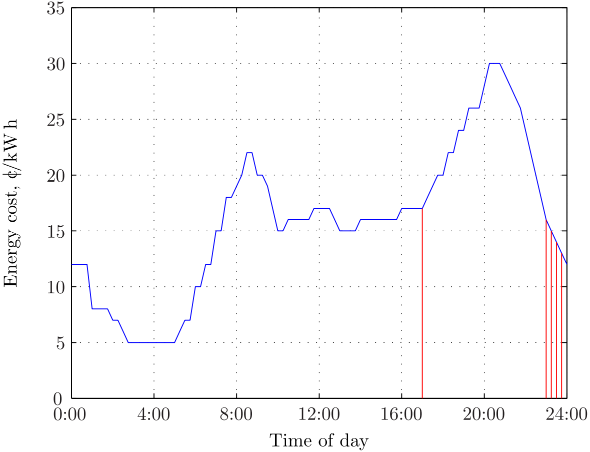


Appliance scheduling

as deterministic optimal control problem

- t gives 15 min periods; $t = 0$ is 17:00–17:15, $T = 28$ is 24:00–24:15
- $\mathcal{X} = \{0, \ldots, 5\}$; x_t is number of cycles completed; $x_0 = 0$
- $\mathcal{U}(x) = \{0, 1\}$ (wait, run) for $x = 0, \ldots, 4$; $\mathcal{U}(5) = \{0\}$ (done)
- state-transition function: $x_{t+1} = f(x_t, u_t) = x_t + u_t$
- stage cost: $g(x_t, u_t) = (1/4) c_t p_{x_t+1} u_t$
  - c_t is electricity cost in period t
  - p_i is power of cycle i
- terminal cost: $g_T(x_T) = 0$ for $x_T = 5$; ∞ otherwise

Appliance scheduling



the cheapest cycle schedule is shown in red

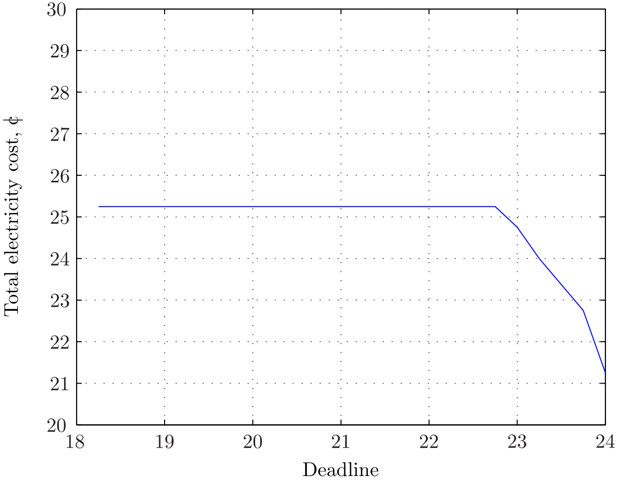
Appliance scheduling

now suppose we change the time horizon...

the optimal cost is

- infinite for $T < 5$
- monotonically decreasing as a function of T

Appliance scheduling



Appliance scheduling

optimal policy: run appliance during the marked time periods