Backward Induction (Part 2)

- Previously, two examples of dynamic decision problems.
  - Transportation. Shortest path. (deterministic DP)
  - Career planning over 10 years (stochastic DP)
- We numerically solved the two problems with <u>backward induction</u>.
- Today's goal:
  1. formally introduce the backward induction algorithm
  2. Work through another example to demonstrate the algorithm.
  3. (Look at the Python implementation of the BI a/g)

You should know more precisely about the algorithm, how it is applied, and implemented.

(Backward induction is sometimes known as the "dynamic programming" method ; but in our lectures, we follow optimization convention and use dynamic programming to denote a set of problems / models)

Define : x_t state of system at time t

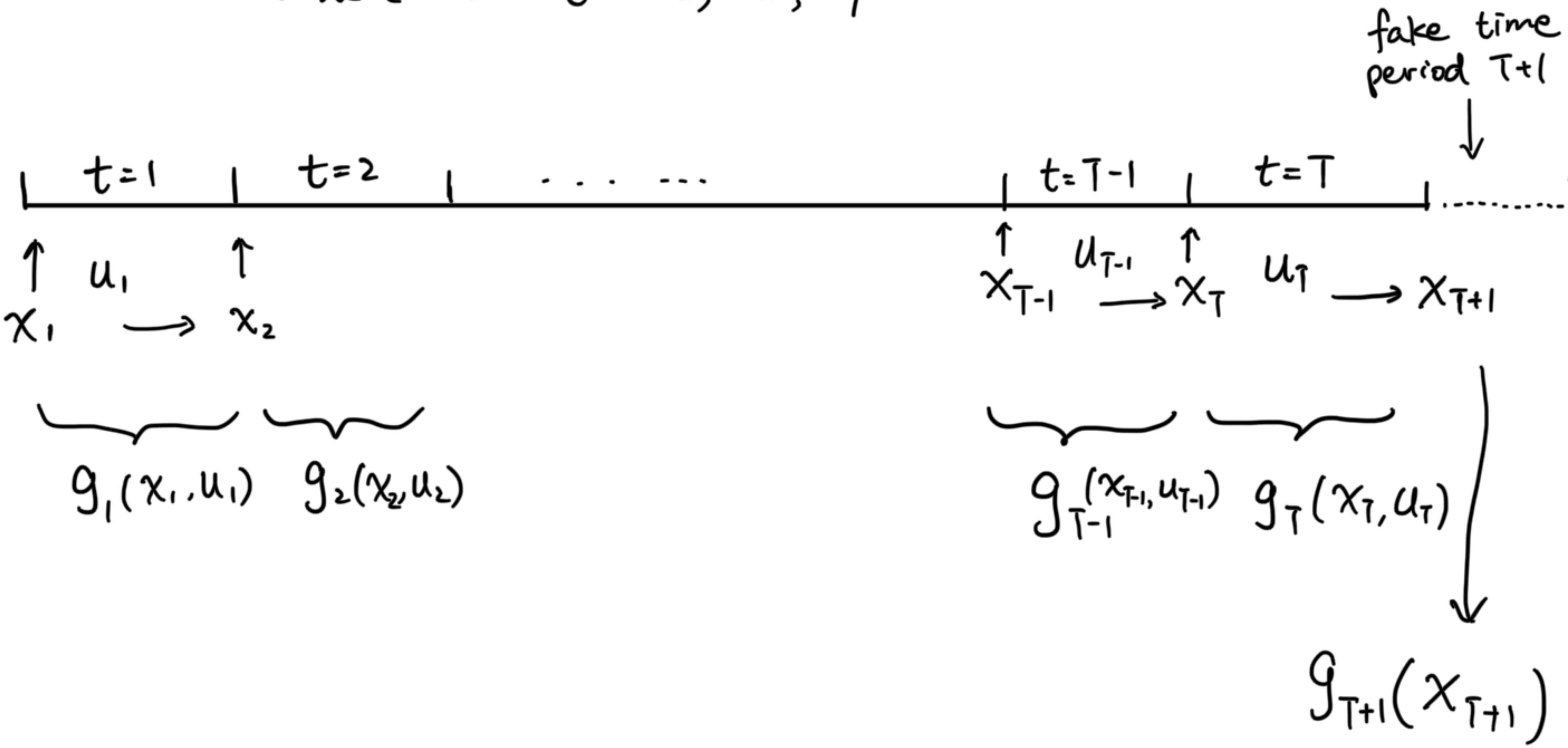
u_t action taken by agent at time t. (decision)

$f_t(x_t, u_t)$ transition of state from time t to $t+1$

$g_t(x_t, u_t)$, $t=1, \dots, T$ cost incurred at time t.

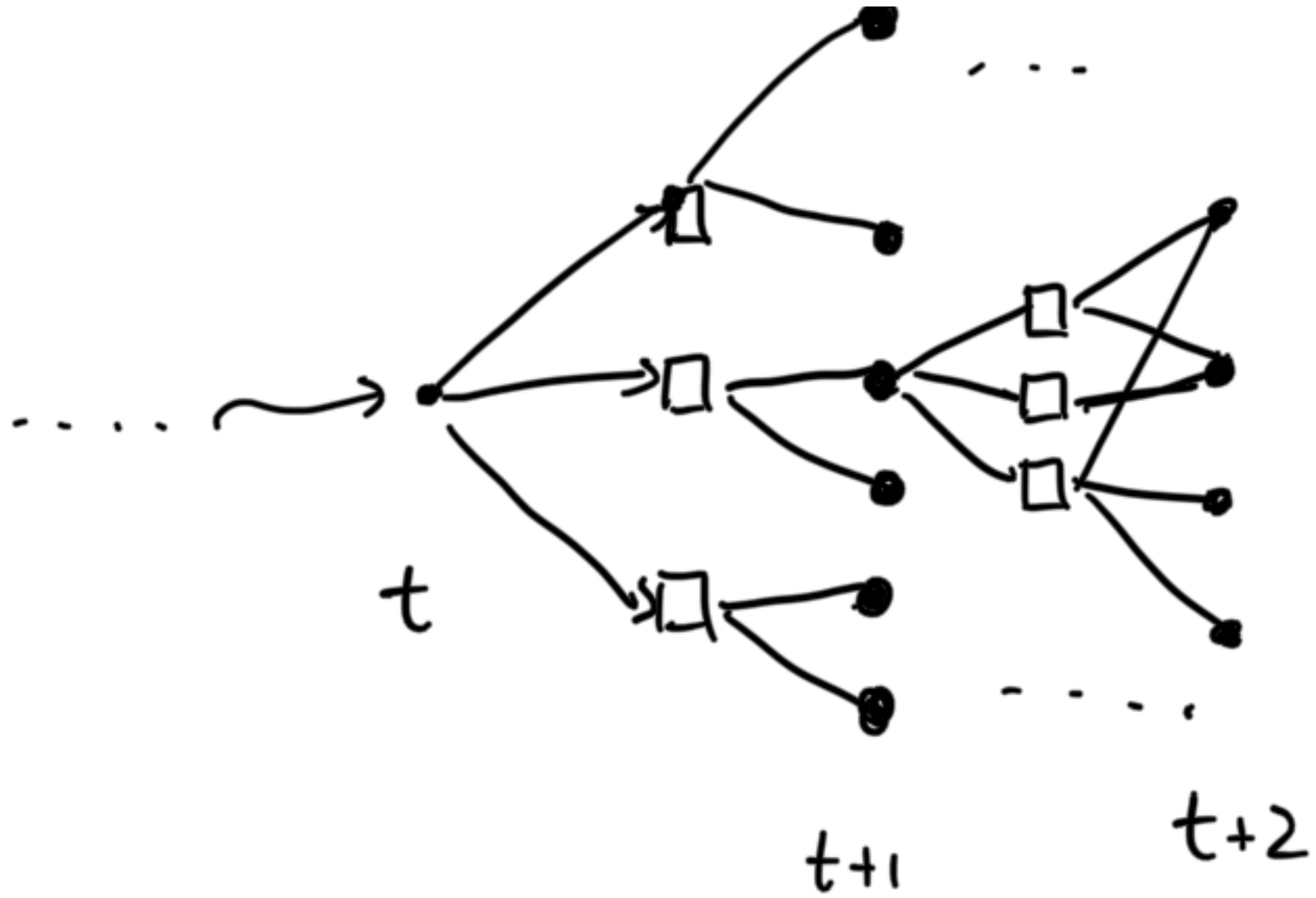
$g_{T+1}(x_{T+1})$ terminal cost.

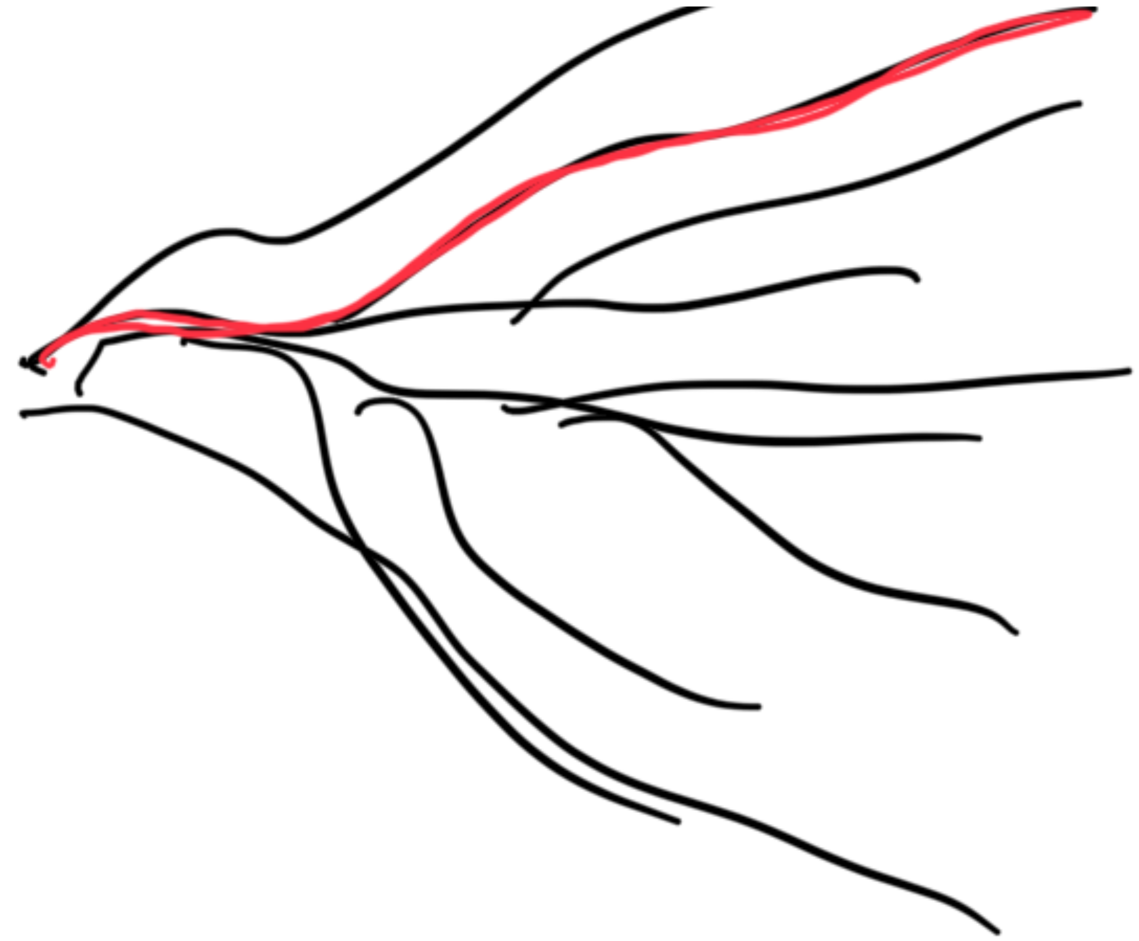
where : $t = 1, \dots, T$



$\boxed{\text{Cost-to-go function: } J_t(x_t)}$

The minimal total cost from time t all the way to the end of the system evolution, if we start from x_t





For $t = 1, \dots, T$:

$$J_t(x_t) = \min_{u_t, u_{t+1}, \dots, u_T} \; g_t(x_t, u_t) + g_{t+1}(x_{t+1}, u_{t+1}) + \cdots + g_T(x_T, u_T) + g_{T+1}(x_{T+1})$$

where:

$$x_{t+1} = f_t(x_t, u_t)$$

$$x_{t+2} = f_{t+1}(x_{t+1}, u_{t+1})$$

$$\vdots$$

For $T+1$:

$$J_{T+1}(x_{T+1}) = g_{T+1}(x_{T+1})$$

Simpler Definition: (Recursive Definition)

$$\begin{cases} J_{T+1}(x_{T+1}) = g_{T+1}(x_{T+1}) & \text{for all } x_{T+1} \\ \underbrace{J_t(x_t)}_{\substack{\text{Cost-to-go,} \\ \text{cost from now} \\ \text{to future.}}} = \min_{u_t} \Big(\underbrace{g_t(x_t, u_t)}_{\substack{\text{Current} \\ \text{cost}}} + \underbrace{J_{t+1}(f_t(x_t, u_t))}_{\text{future cost from } t+1} \Big) \end{cases}$$

Now for Stochastic Settings:

(State transition and cost ftns can both be random.)

$$
\begin{cases}
J_t(x_t) = \min\limits_{u_t} \mathbb{E}\left[g_t(x_t, u_t) + J_{t+1}(x_{t+1}) \right] \\
\qquad\qquad\qquad\qquad \text{for } t=1, \dots, T, \quad \text{for all } x_t \\
J_{T+1}(x_{T+1}) = \mathbb{E}\left[g_{T+1}(x_{T+1}) \right]
\end{cases}
$$

Now we are ready to define the backward Induction algorithm.

(pseudo code)

State space : set of all feasible states

BI-ALG :

capital "X"

INPUT: $X_1, X_2, \dots, X_T, X_{T+1}$ (State spaces)

$g_1, g_2, \dots, g_{T+1}$

$f_1, f_2, \dots, f_T$

For x_{T+1} in X_{T+1} :

For x_{T+1} in X_{T+1}:

$$J_{T+1}(x_{T+1}) = \mathbb{E}\left[g_{T+1}(x_{T+1}) \right]$$

For $t = T, T-1, \dots, 1$:

For x_t in X_t:

$$J_t(x_t) = \min_{u_t} \mathbb{E}\left[g_t(x_t, u_t) + J_{t+1}(f_t(x_t, u_t)) \right]$$

At the end of the algorithm, we have $J_1(x_1)$ for any $x_1 \in X_1$