MS-E2148 Dynamic optimization

Lecture 6

Contents

- Dynamic programming problem and terminology
- Dynamic Programming (DP) algorithm
- Material Bertsekas 1.2 and 1.3

Basic problem

Bertsekas, Ch 1.1.-1.2.

- Discrete time, dynamic system

$$x_{k+1} = f_k(x_k, u_k, w_k) \qquad k = 0, 1, ..., N - 1$$

- k is *step* or *time* or some index representing recursion
- State $x_k \in S_k$ is some relevant factor of the system
  - Initial state x_0, final state x_N
- Control $u_k \in C_k$
- Random parameter $w_k \in D_k$ can be noise, *disturbance* or *error*

Basic problem
Example

$$x_{k+1} = (1 + r)x_k - u_k - w_k$$

- State x_k is *wealth* at stage k
- Control u_k is *consumption* at stage k
- Random parameter w_k is *loss* at stage k
- Constant r is the interest rate

Basic problem
State

- State x_k at stage k contains **all** relevant information in the past that can be used in making the future decisions

- Example: The altitude of Boeing 747 is relevant information when considering the optimal landing. Knowing what the altitude was at stage $t - 34$ does not help
- Example: the balance of your account today is relevant when paying your rent the next morning but the balance week ago is not

Basic problem
Control and noise

$$x_{k+1} = f_k(x_k, u_k, w_k) \qquad k = 0, 1, ..., N-1$$

- Let us restrict the control at stage k to belong to some set that depends on the state at stage k: $u_k \in U_k(x_k) \subset C_k$
- The noise can be characterized by a probability distribution $P_k(\cdot|x_k, u_k)$ that is independent of the past values $w_{k-1}, ..., w_0$
- This is called a *stochastic* system

Basic problem
Control law

- Let us examine a control law $\pi = \{\mu_0, ..., \mu_{N-1}\}$ where μ_k maps the states x_k into controls $u_k = \mu_k(x_k)$, $\mu_k(x_k) \in U_k(x_k) \subset C_k$ for all $x_k \in S_k$ and k
- These control laws are *admissible* or feasible and they belong to a set of control laws Π.
- Control laws are also called *policy*.

- The states x_k that can be obtained at stage k and with admissible control laws are called *reachable states*

Basic problem
Cost

- Assume that the objective is to minimize *the additive cost* at stage k that depends on the state, control and the noise: $g_k(x_k, u_k, w_k)$
- By using admissible control law π
- Then the expected cost, when x_0 is fixed, is

$$J_\pi(x_0) = E\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\}$$

Basic problem

Cost

- The optimization means that we search for the optimal π^* from Π:

$$J_{\pi^*}(x_0) = \min_{\pi \in \Pi} J_\pi(x_0)$$

- The cost $J_\pi(x_0)$ is then the "**objective function**" and the control law π is the "**decision variable**"

Basic problem

The value of information

- The **closed-loop** (feedback) controls depend on the state for each k: $u_k = \mu_k(x_k)$
  - The dependency of state is important especially in stochastic problems
- The **open-loop** controls depend only on the initial state x_0: $u_k = \mu_k(x_0)$,
  - Could you fly Boeing 747 like this?
- Claim: the closed-loop control can never give worse performance than the open-loop control

Basic problem

The system

- Discrete-time system
- Noise independent of past values
- Constrained controls
- Additive cost
- Optimization over (closed-loop) control laws

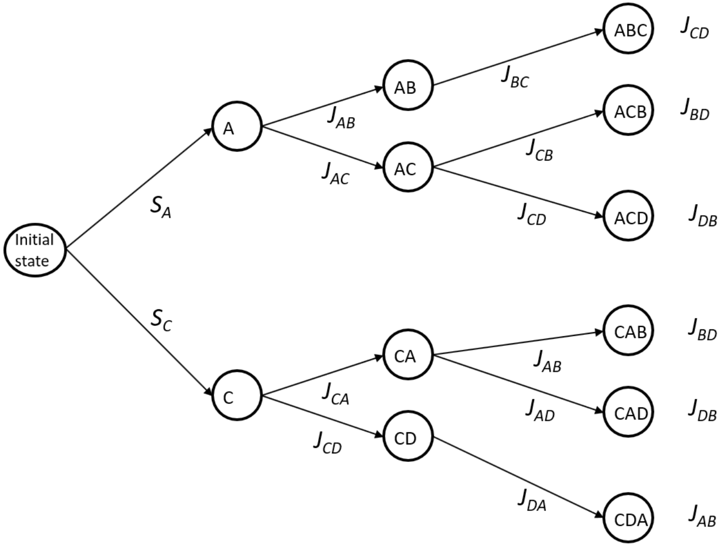
Example: deterministic scheduling

- Machine has four operations A,B,C and D
- B can only be done if A has been done, and D can only be done after C
- We know the cost J_{mn} that incurs when we switch from operation m to n, and what are the initial costs S_A and S_C.
- The costs are additive, e.g., the cost of doing operations ACDB is $S_A + J_{AC} + J_{CD} + J_{DB}$

Example: deterministic scheduling

- Let us choose as state the operations that are performed
  - For example: the state at stage $k = 2$ is one of the following: AB, AC, CA, or CD
- The task is to choose the order of three operations (the last is determined by these)
- Initial state is either operation A or operation C
- The control determines what is the next operation

Example: deterministic scheduling



Example: deterministic scheduling

- The problem is *deterministic* and not stochastic, since the next state $k + 1$ is determined when the state and control are known at stage k
- The number of states is *finite*

The principle of optimality

- Richard Bellman 1957
- Let $\pi^* = \{\mu_0^*, \mu_1^*, ..., \mu_{N-1}^*\}$ be the optimal control law for the basic problem
- At stage i and state x_i, we consider a subproblem to minimize the **cost-to-go**

$$E\left\{ g_N(x_N) + \sum_{k=i}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\} \tag{1}$$

- The minimization gives the optimal continuation cost from that state to the end
- Then the control law $\{\mu_i^*, \mu_{i+1}^*, ..., \mu_{N-1}^*\}$ is optimal for this subproblem

The principle of optimality

- Intuitively: if the control were not optimal, then we could decrease the cost-to-go (1) by switching to some other control law at stage i
- The principle gives the property: the future controls must be optimal for the current state regardless of the initial state and the past controls

The principle of optimality

- In other words: if the optimal route from Espoo to Kuopio is to drive through Heinola and then Mikkeli, then the route from Heinola to Kuopio through Mikkeli must also be optimal. The route from Heinola to Kuopio through Jyväskylä cannot be optimal in that case.
- NOTE: Additivity of cost is necessary!

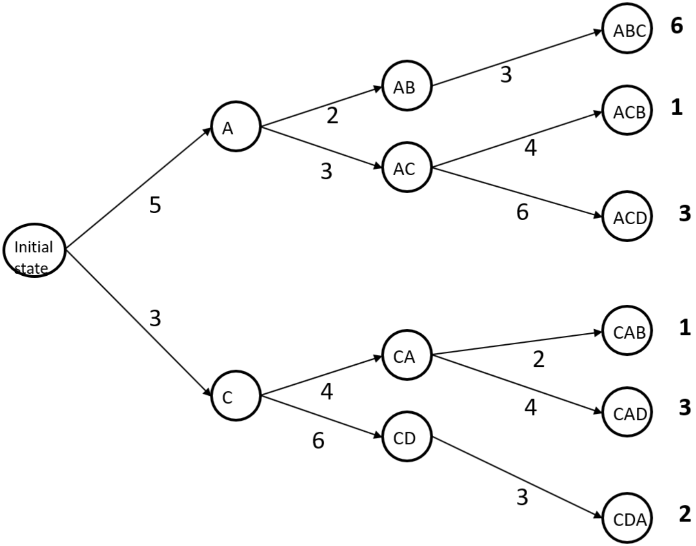
DP-algorithm

- How do you find the optimal control law?
  $\Rightarrow$ Use the principle of optimality to the cost-to-go (1), starting from the last stage and propagating back to the first stage
- **Dynamic Programming algorithm** (DP-algorithm)

DP-algorithm: deterministic scheduling

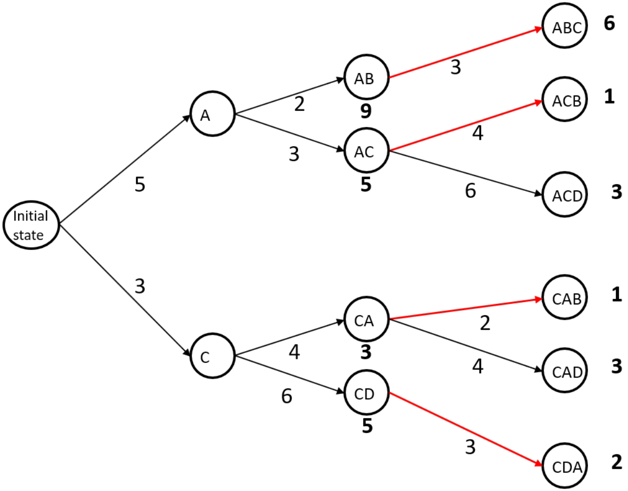
- State: the route this far
- E.g. if the route CABD is optimal
  - it is optimal to go to B after CA, and from state C to state A
- In DP algorithm we compute
  - all subproblems of length two,
  - all subproblems of length three,
  - and finally the original problem of length four
- By moving backwards in time, we can compute the optimal control law from the subproblems and the cost from the cost-to-go values

DP-algorithm: deterministic scheduling



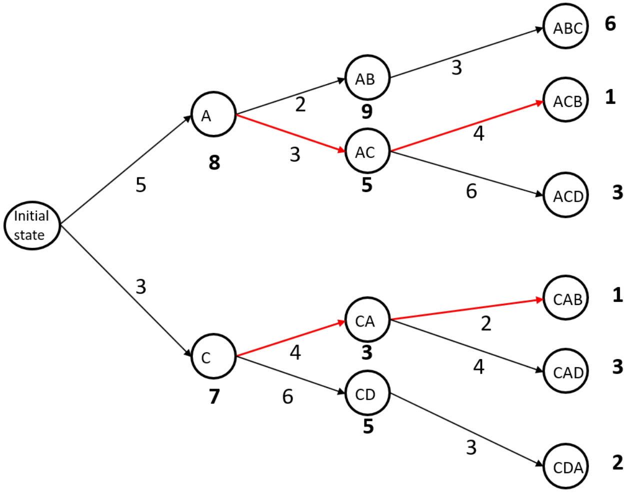
DP-algorithm: deterministic scheduling

Subproblems of length two



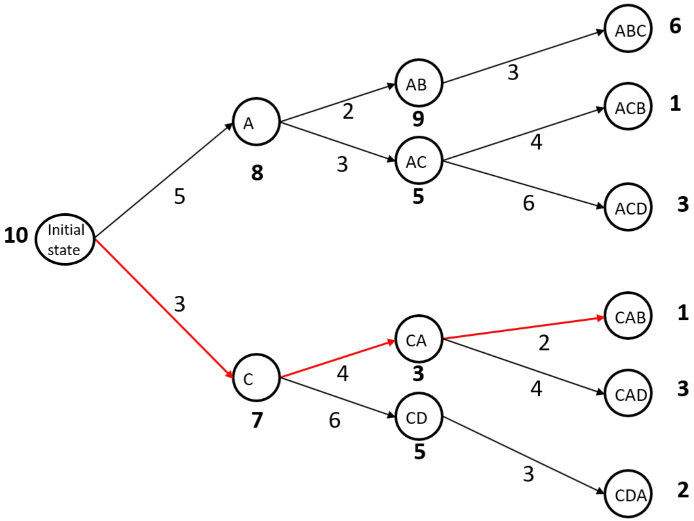
DP-algorithm: deterministic scheduling

Subproblems of length three



DP-algorithm: deterministic scheduling

The original problem with length four



Example: inventory control

- Order the product in N stages so that the stochastic demand is satisfied and the expected cost is minimized

- x_k is the amount of the product at the beginning of stage k
- u_k is the order quantity at stage k that arrives immediately and satisfies $u_k \geq 0$
- w_k is the stochastic demand at stage k; $w_0, w_1, ..., w_{N-1}$ are independent of each other, but the demand may depend on the state and/or the control

Example: inventory control

- The equation for the inventory is

$$x_{k+1} = x_k + u_k - w_k$$

- Negative x_k means that the demand is not met, positive means there is extra inventory

Example: inventory control

- The total cost over N periods is

$$E\left\{\sum_{k=0}^{N-1}(cu_k + r(x_k + u_k - w_k))\right\}$$

- The cost is made of two components:
  a. Ordering cost cu_k (c is a unit cost)
  b. Inventory cost $r(x_k + u_k - w_k)$, which can be positive or negative (r is a suitable function)
- How do you interpret the parameters c and r?
- What about the final cost $g_N(x_N)$?

Example: inventory control

- The control law $\pi = \{\mu_k(x_k) | k = 0, 1, ..., N-1\}$ gives a description how to act in different cases: "if you observe the inventory x_k at stage k, order $\mu_k(x_k)$"
- Is this closed or open-loop control?

Example: inventory control

- The task is to find the best control law that minimizes the cost:
  $\pi^* = \arg\min_\pi \{J_\pi(x_0)\}$, where

$$J_\pi(x_0) = E\left\{\sum_{k=0}^{N-1} \left(cu_k + r(x_k + u_k - w_k)\right)\right\}$$

s.t.

$$x_{k+1} = x_k + u_k - w_k$$

DP-algorithm: inventory control

- The subproblems of length one: at stage $N-1$ the inventory is x_{N-1}. Choose control u_{N-1} that minimizes the ordering cost and the expected inventory cost:

$$J_{N-1}(x_{N-1}) = \min_{u_{N-1}\geq 0}\left[cu_{N-1} + E_{w_{N-1}}\left\{r(x_{N-1}+u_{N-1}-w_{N-1})\right\}\right]$$

- When this control is determined, we get the last part of the optimal control law $\mu^*_{N-1}(x_{N-1})$

DP-algorithm: inventory control

- The subproblems of length two: at stage $N-2$ the inventory is x_{N-2}. Choose control u_{N-2} to minimize:
  (the cost at stage $N-2$) + (the cost at stage $N-1$ s.t. we use the optimal control for the subproblem μ^*_{N-1}), i.e.,

$$J_{N-2}(x_{N-2}) = \min_{u_{N-2}\geq 0} \Big[cu_{N-2} + E_{w_{N-2}}\big\{ r(x_{N-2}+u_{N-2}-w_{N-2}) \\ + J_{N-1}(x_{N-2}+u_{N-2}-w_{N-2}) \big\} \Big]$$

- Note: $x_{N-1} = x_{N-2} + u_{N-2} - w_{N-2}$
- When this control is determined, we also get the second last part of the optimal control law $\mu^*_{N-2}(x_{N-2})$

DP-algorithm: inventory control

- The subproblems of length $N-k$: at stage k the inventory is x_k. Choose control u_k that minimizes:

$$J_k(x_k) = \quad \min_{u_k \geq 0} \Big[c u_k + E_{w_k} \big\{ r(x_k + u_k - w_k) \\ + J_{k+1}(x_k + u_k - w_k) \big\} \Big] \qquad (2)$$

- The functions $J_k(x_k)$ describe the optimal expected cost for subproblems that *start* at stage k; these functions are computed recursively backwards in time starting from stage $N-1$ and ending to stage 0
- The optimal cost for the problem is $J_0(x_0)$. The optimal control law will be determined for the minimization problem (2) for each k, and "initial" stage $x_k \in S_k$.

DP-algorithm

- To solve the optimal $J_k^*(x_k)$ and $u_k^* = \mu_k^*(x_k)$, the right-hand side minimization problem should have a solution for all x_k.
- In the inventory control, this is satisfied, e.g., when the "penalty term" is function $r(\cdot) = (x_k + u_k - w_k)^2$
- Minimization problem $\min_{u_k}[L(x_k, u_k)]$ is solved using the **first-order conditions**: the partial derivative is set to zero: $\partial L / \partial u_k = 0$ and from this we solve $u_k^* = \mu_k^*(x_k)$
- In DP-algorithm we have N of these minimization problems

DP-algorithm
Formulation

- For each initial state x_0 of the basic problem, the optimal cost $J^*(x_0)$ is $J_0(x_0)$, which we get at the last step of the computation:
$$J_N(x_N) = g_N(x_N), \tag{3}$$
$$J_k(x_k) = \min_{u_k \in U_k(x_k)} E_{w_k}\Big\{g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k))\Big\}, \tag{4}$$
$k = 0, 1, ..., N-1$
- If $u_k^* = \mu_k^*(x_k)$ minimizes the right-hand side of (4) for each x_k and k, the control law $\pi^* = \{\mu_0^*, ..., \mu_{N-1}^*\}$ is optimal

DP-algorithm: usage

- In ideal case, the DP-algorithm produces a closed-form solution
- Unfortunately, in many cases the analytic solution is not possible and only numerical solutions are available
- For small problems (e.g. $k = 0, 1, 2, 3$, the state and the controls are finite) the cost-to-go and the optimal control law can be computed in a table (see exercises)

Summary

- Discrete-time system
- Independent random variables
- Admissible controls
- Additive cost
- Closed-loop control
- Principle of optimality
- Cost-to-go
- DP-algorithm