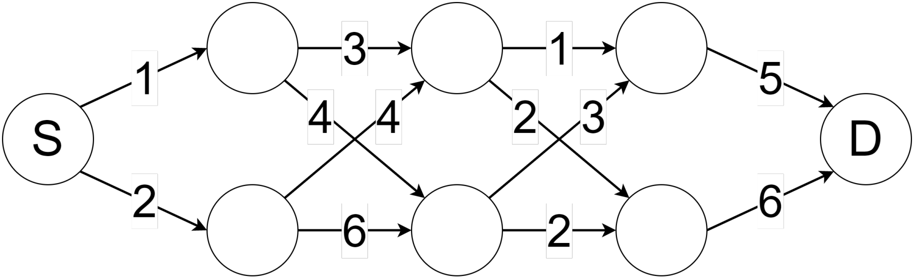
Lecture 1: Dynamic Programming (DP)

Course: Reinforcement Learning Theory
Instructor: Lei Ying
Department of EECS
University of Michigan, Ann Arbor

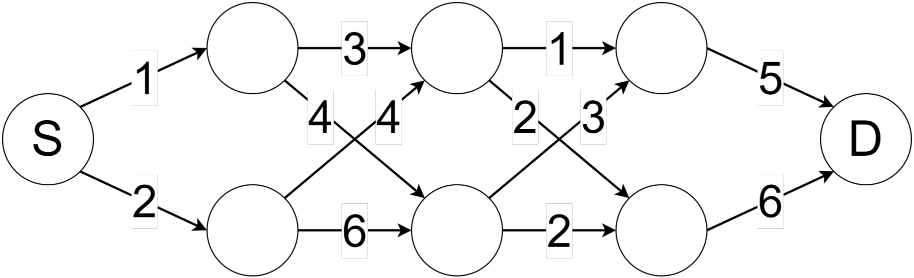
A Deterministic DP Example



Find the shortest path from S to D.

Exclusive search (forward search):
number of possible paths $= 2 \times 2 \times 2 = 8$

A Deterministic DP Example



Find the shortest path from S to D.

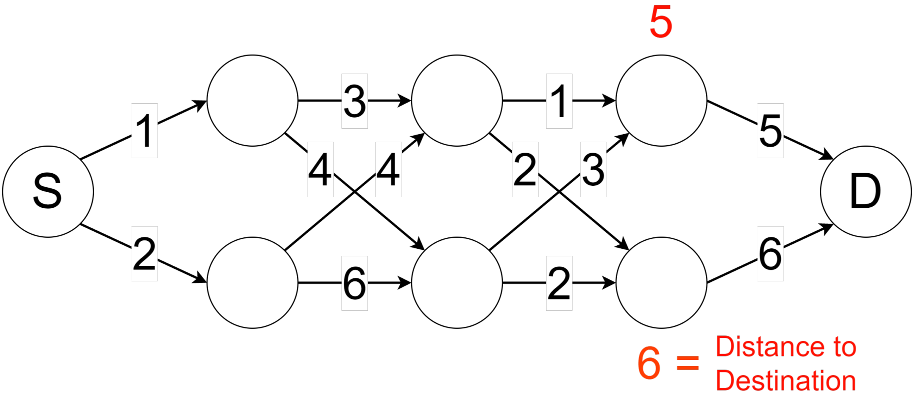
DP (backward search):
start from the destination

Deterministic DP Example

Find the shortest path from S to D.

DP: backward search.

- Calculate the shortest distance from a current node to destination D.
- Each calculation is a comparison of two choices (numbers)

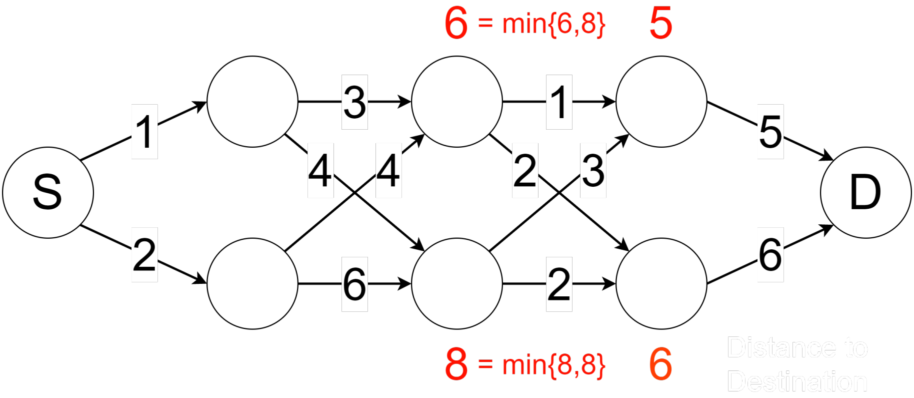


Deterministic DP Example

Find the shortest path from S to D.

DP: backward search.

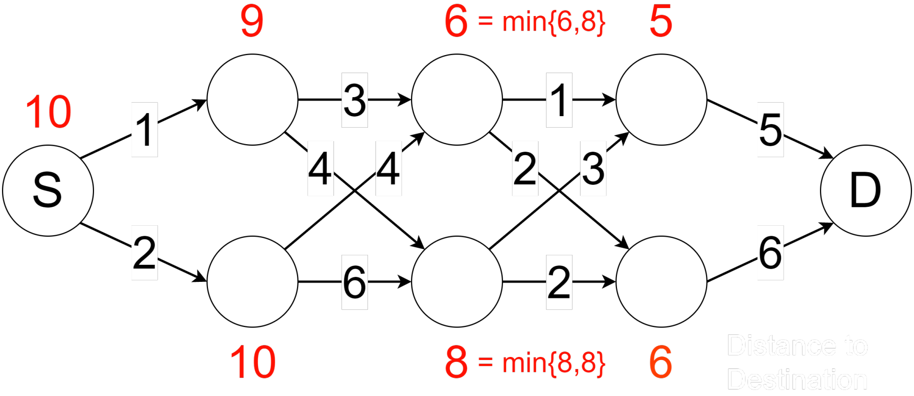
- Calculate the shortest distance from a current node to destination D.
- Each calculation is a comparison of two choices (numbers)



Deterministic DP Example

Find the shortest path from S to D.
DP: backward search.

- Calculate the shortest distance from a current node to destination D.
- Each calculation is a comparison of two choices (numbers)



1 + 2 + 2 + 1 = 6 comparisons

Deterministic DP

The computational complexity of exclusive search versus DP. Suppose the problem has T stages. Then

- Exclusive search: exponential in T
- Dynamic programming: linear in T

Deterministic DP: The shortest path problem

How to find the shortest distance to a node (say node i) from any other node?

Deterministic DP: The shortest path problem

How to find the shortest distance to a node (say node i) from any other node?

Dynamic Programming Principle

$\mathcal{N}(j)$: set of outgoing neighbors of node j
$c(j,k)$: distance of edge from node j to node k (denoted by (j,k))

$$d(j,i) = \min_{k \in \mathcal{N}(j)} c(j,k) + d(k,i)$$

Deterministic DP: The shortest path problem

How to find the shortest distance to a node (say node i) from any other node?

Dynamic Programming Principle

$\mathcal{N}(j)$: set of outgoing neighbors of node j
$c(j,k)$: distance of edge from node j to node k (denoted by (j,k))

$$d(j,i) = \min_{k \in \mathcal{N}(j)} c(j,k) + d(k,i)$$

1. "$\leq$ proof": a shortest path j to i has to go through $\mathcal{N}(j)$,

$$d(j,i) \leq c(j,k) + d(k,i) \quad \forall k \in \mathcal{N}(j)$$

Deterministic DP: The shortest path problem

How to find the shortest distance to a node (say node i) from any other node?

Dynamic Programming Principle

$\mathcal{N}(j)$: set of outgoing neighbors of node j
$c(j,k)$: distance of edge from node j to node k (denoted by (j,k))

$$d(j,i) = \min_{k \in \mathcal{N}(j)} c(j,k) + d(k,i)$$

1. "$\geq$ proof": if $P_{j \to i}$ (path from j to i) is a shortest path and goes through $k^* \in \mathcal{N}(j)$, the subpath to i from k^* on $P_{j \to i}$ is a shortest path from k^* to i,

$$d(j,i) = c(j,k^*) + d(k^*,i) \geq \min_{k \in \mathcal{N}(j)} c(j,k) + d(k,i).$$

We therefore conclude: $d(j,i) = \min_{k \in \mathcal{N}(j)} c(j,k) + d(k,i)$.

Finite horizon deterministic DP

System equation:

$$x_{k+1} = f_k(x_k, u_k), \quad k = 0, 1, \ldots, N-1$$

$\underline{x_k\text{: system state at time } k}$, $\underline{u_k\text{: control at time } k}$

Finite horizon deterministic DP

System equation:

$$x_{k+1} = f_k(x_k, u_k), \quad k = 0, 1, \ldots, N-1$$

x_k: system state at time k, u_k: control at time k

In the shortest path problem,
x_k: on the top or bottom node at stage k
u_k: move up or down at stage k

Finite horizon deterministic DP

For given initial state x_0, the cost over control sequence $(u_0, u_1, \ldots, u_{N-1})$ is defined to be

$$J(x_0; u_0, \ldots, u_{N-1}) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k)$$

Finite horizon deterministic DP

For given initial state x_0, the cost over control sequence $(u_0, u_1, \ldots, u_{N-1})$ is defined to be

$$J(x_0; u_0, \ldots, u_{N-1}) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k)$$

Optimal cost and optimal control:

$$J^*(x_0) = \min_{u_0, \ldots, u_{N-1}} J(x_0; u_0, \ldots, u_{N-1})$$

Finite horizon deterministic problem

Backward computation:

$$J_N^*(x_N) = g_N(x_N). \qquad \text{(trivial)}$$

In the shortest path problem, we have

$$g_N(x_N) = 0.$$

Finite horizon deterministic problem

Backward computation:

$$J_N^*(x_N) = g_N(x_N). \qquad \text{(trivial)}$$

According to the DP principle,

$$J_{N-1}^*(x_{N-1}) = \min_{u_{N-1}} \left(g_{N-1}(x_{N-1}, u_{N-1}) + J_N^*(f(x_{N-1}, u_{N-1}))\right),$$

where we assume $f_k(\cdot) = f(\cdot)$ for all K. And in general,

$$J_k^*(x_k) = \min_{u_k} \left(g_k(x_k, u_k) + J_{k+1}^*(f(x_k, u_k))\right).$$

Reference

- Chapter 1.1 of Dimitri P. Bertsekas, *Reinforcement Learning and Optimal Control*, Athena Scientific, 2019. Slides and lectures available at `https://web.mit.edu/dimitrib/www/RLbook.html`

Acknowledgements: I would like to thank Alex Zhao for helping prepare the slides, and Honghao Wei and Zixian Yang for correcting typos/mistakes.