École des Ponts ParisTech

Introduction to Dynamic Programming

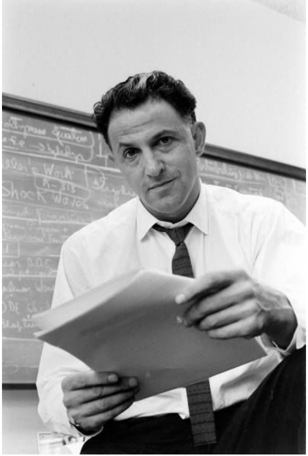
A bad and painless presentation

Emanuele Concas

emanuele.concas@enpc.fr

Outline

Who



- Invented by Richard Bellman
- RAND was hired by the USAF
- The SecDef was Wilson
- Need to hide the secret
  - Dynamic: it is dynamic
  - Programming: find an optimal program schedule

What

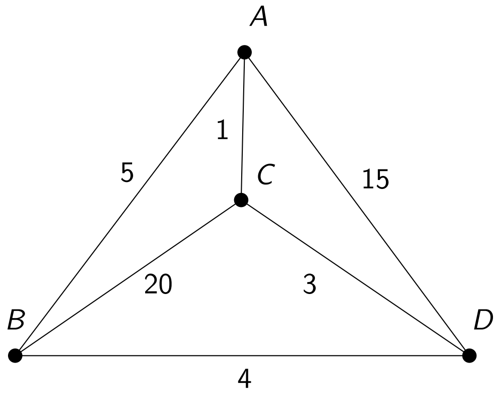
- Optimization approach
  - complex problem → sequence of simpler problem
- Decision are made in stage
- Decision may be not fully predictable

Key ideas

- A decision cannot be seen in isolation
  - Balance between cost now vs cost in the future
- Optimal subproblems
- Overlapping subproblems

An example: Travelling salesman problem

- Shortest Hamiltonian circuit in a graph

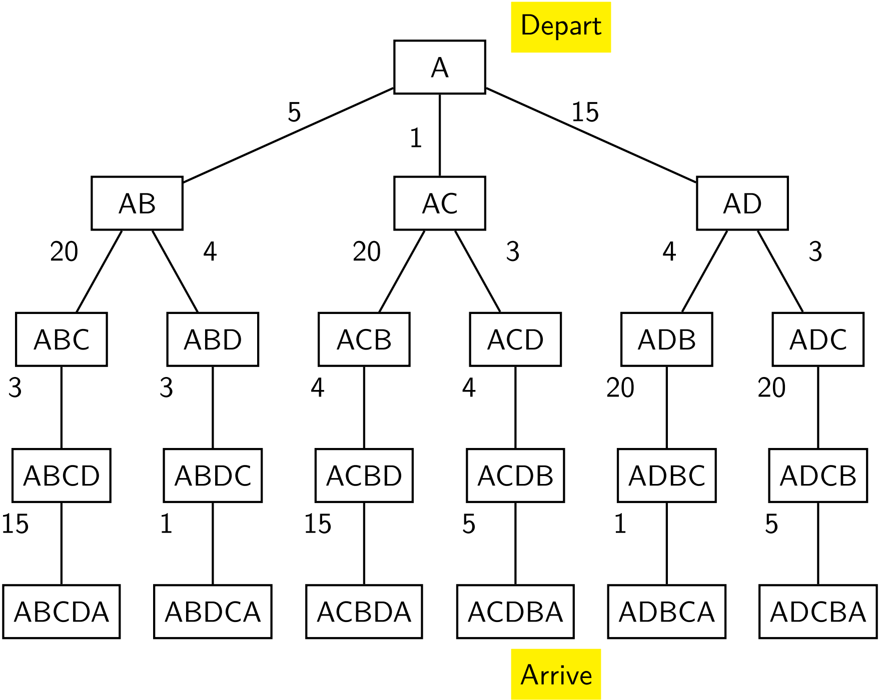


TSP brute-force

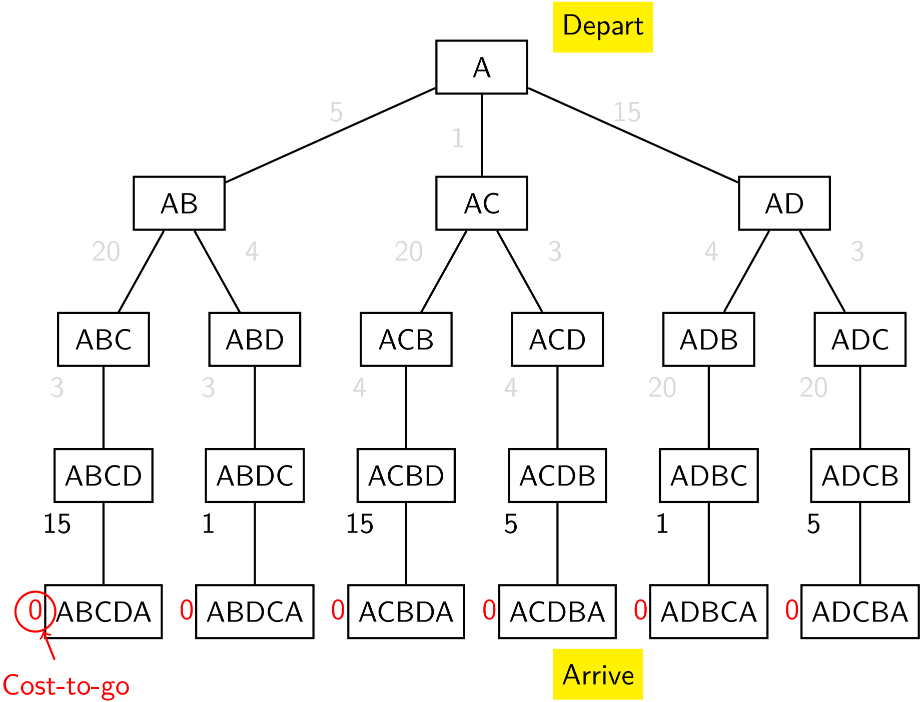
- Take all the possible combinations
- Calculate the cost

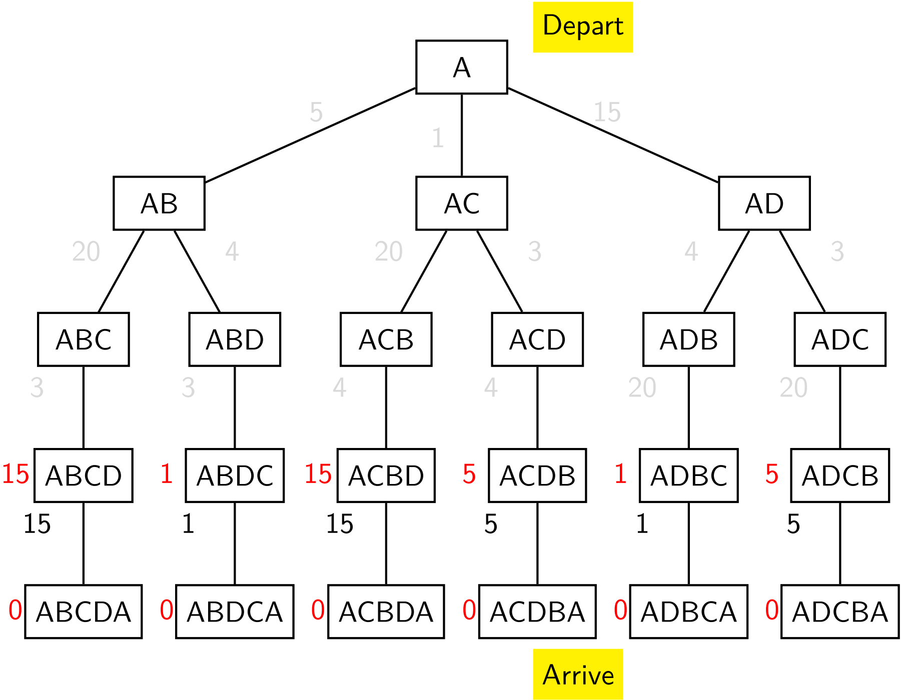
$$A \to B \to C \to D \to A$$
$$A \to B \to D \to C \to A$$
$$A \to C \to B \to D \to A$$
$$A \to C \to D \to B \to A$$
$$A \to D \to B \to C \to A$$
$$A \to D \to C \to B \to A$$

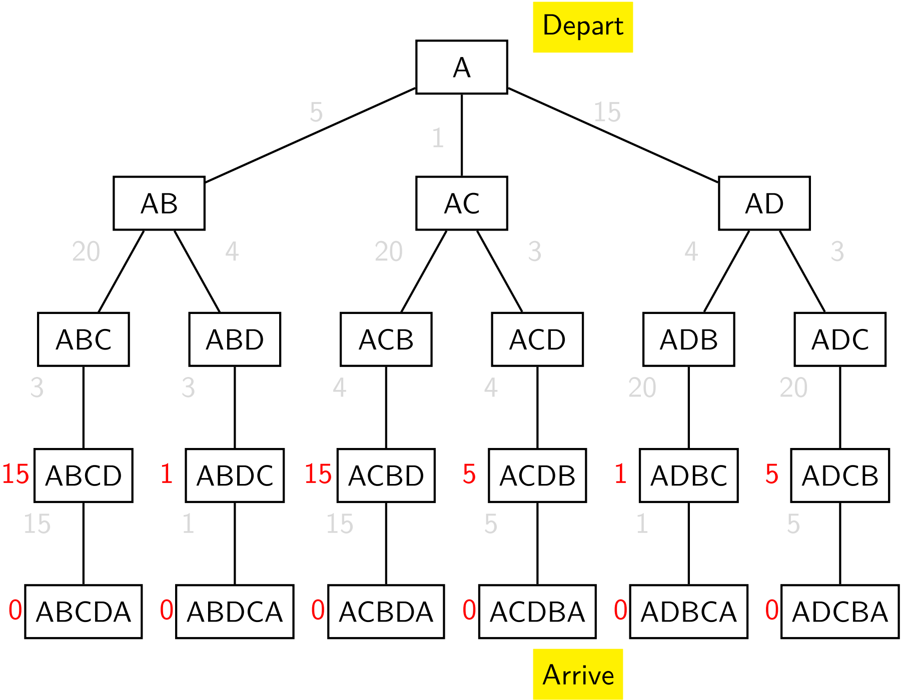
- Overlapping subproblems

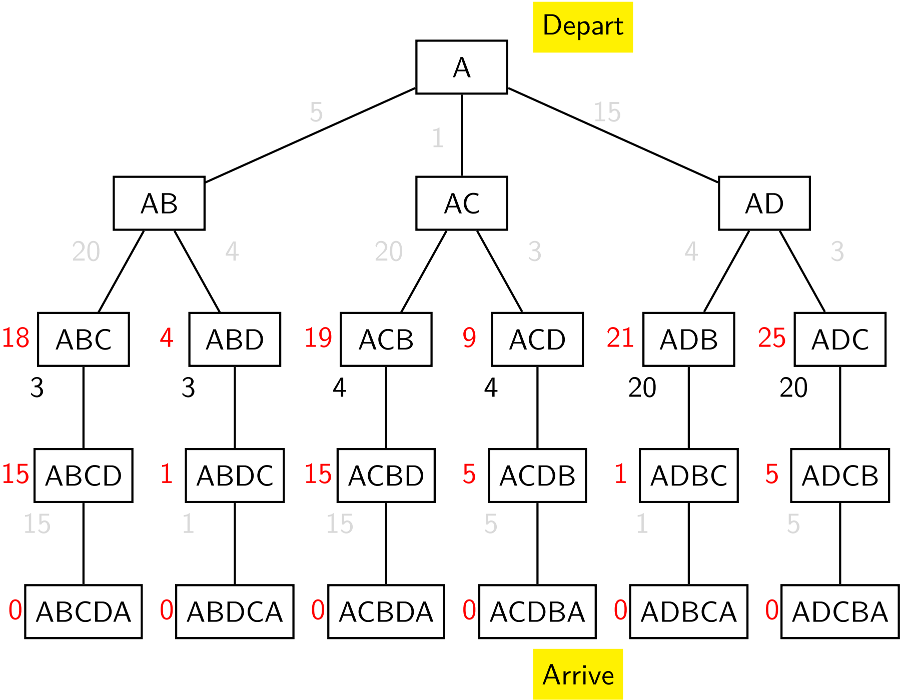


Depart
A
5
1
15
AB
AC
AD
20
4
20
3
4
3
ABC
ABD
ACB
ACD
ADB
ADC
3
3
4
4
20
20
ABCD
ABDC
ACBD
ACDB
ADBC
ADCB
15
1
15
5
1
5
ABCDA
ABDCA
ACBDA
ACDBA
ADBCA
ADCBA
Arrive

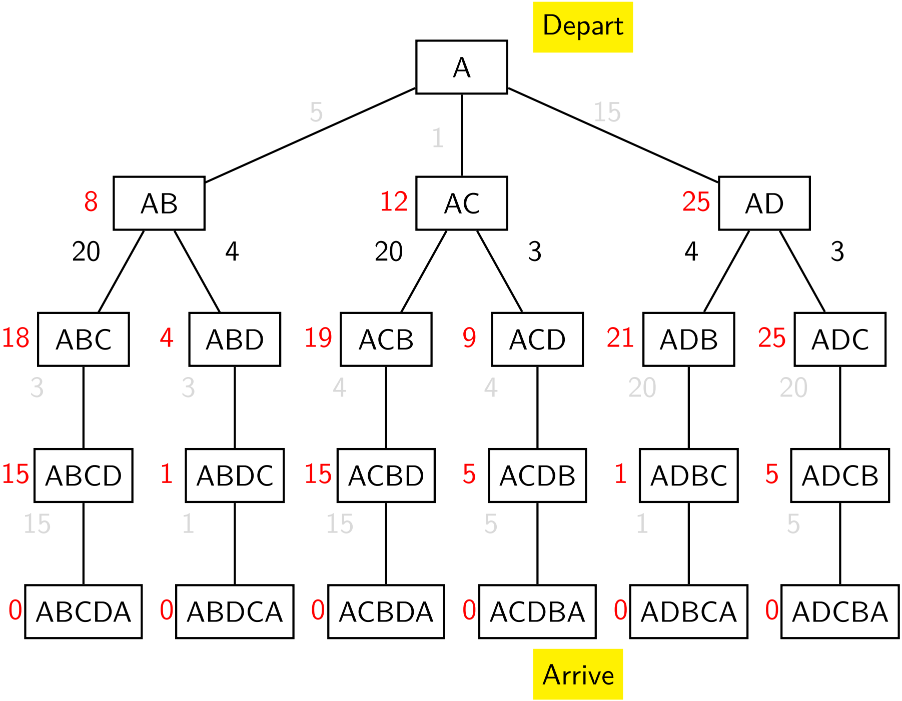

Depart
A
5
1
15
AB
AC
AD
20
4
20
3
4
3
ABC
ABD
ACB
ACD
ADB
ADC
3
3
4
4
20
20
ABCD
ABDC
ACBD
ACDB
ADBC
ADCB
15
1
15
5
1
5
0
ABCDA
0
ABDCA
0
ACBDA
0
ACDBA
0
ADBCA
0
ADCBA
Cost-to-go
Arrive


Depart
A
5
1
15
AB
AC
AD
20
4
20
3
4
3
ABC
ABD
ACB
ACD
ADB
ADC
3
3
4
4
20
20
15
ABCD
1
ABDC
15
ACBD
5
ACDB
1
ADBC
5
ADCB
15
1
15
5
1
5
0
ABCDA
0
ABDCA
0
ACBDA
0
ACDBA
0
ADBCA
0
ADCBA
Arrive

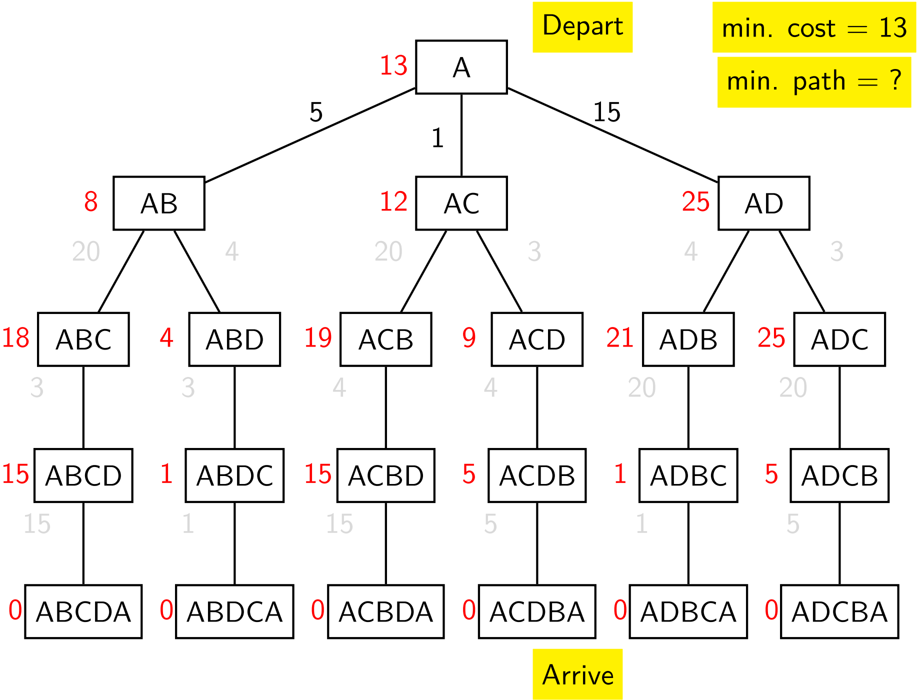

Depart
A
5
1
15
AB
AC
AD
20
4
20
3
4
3
ABC
ABD
ACB
ACD
ADB
ADC
3
3
4
4
20
20
15
ABCD
1
ABDC
15
ACBD
5
ACDB
1
ADBC
5
ADCB
15
1
15
5
1
5
0
ABCDA
0
ABDCA
0
ACBDA
0
ACDBA
0
ADBCA
0
ADCBA
Arrive

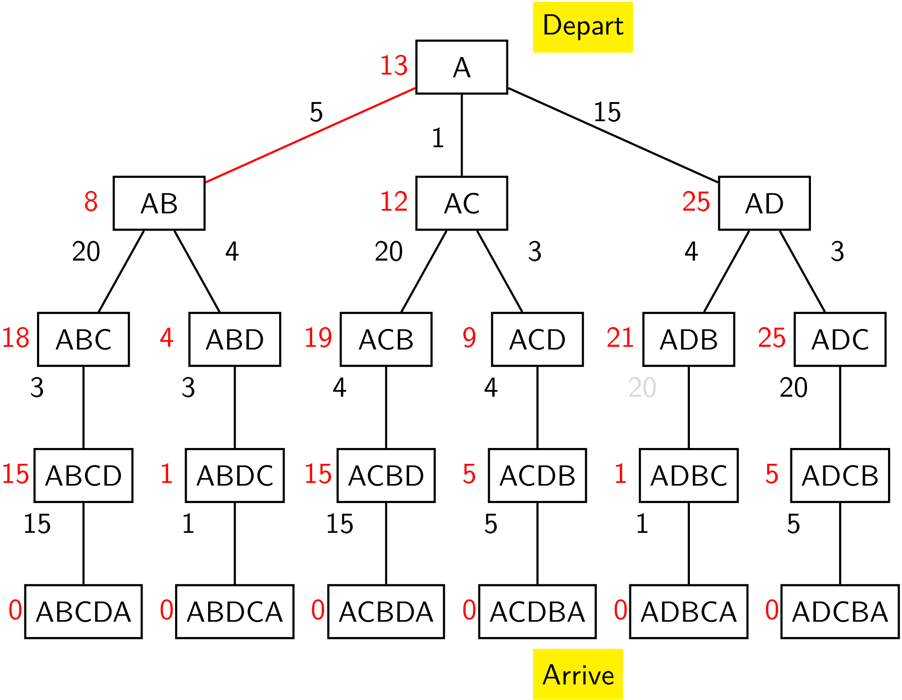


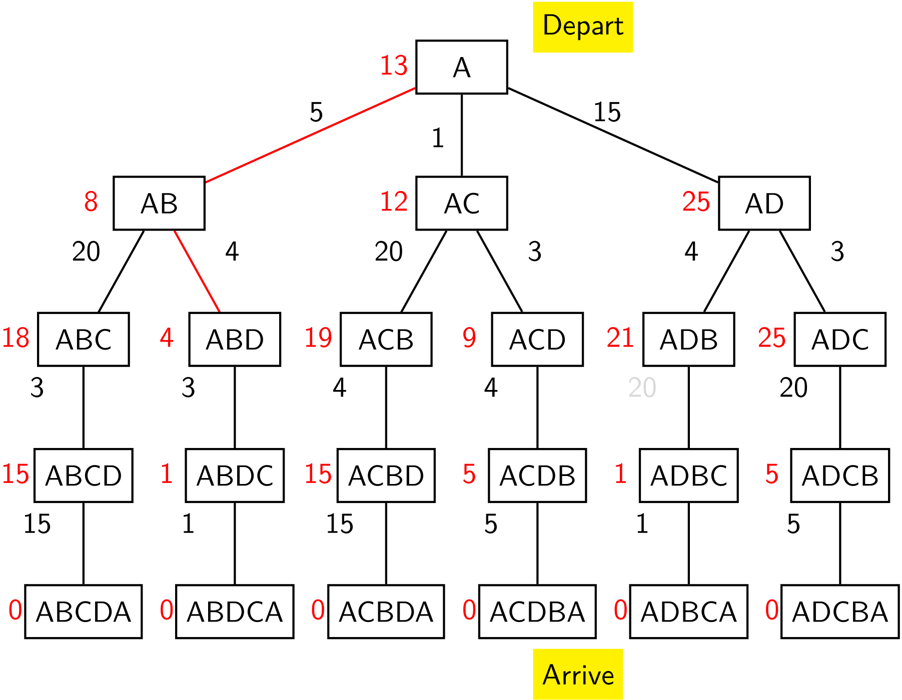
Depart
A
5
1
15
AB
AC
AD
20
4
20
3
4
3
18
ABC
4
ABD
19
ACB
9
ACD
21
ADB
25
ADC
3
3
4
4
20
20
15
ABCD
1
ABDC
15
ACBD
5
ACDB
1
ADBC
5
ADCB
15
1
15
5
1
5
0
ABCDA
0
ABDCA
0
ACBDA
0
ACDBA
0
ADBCA
0
ADCBA
Arrive

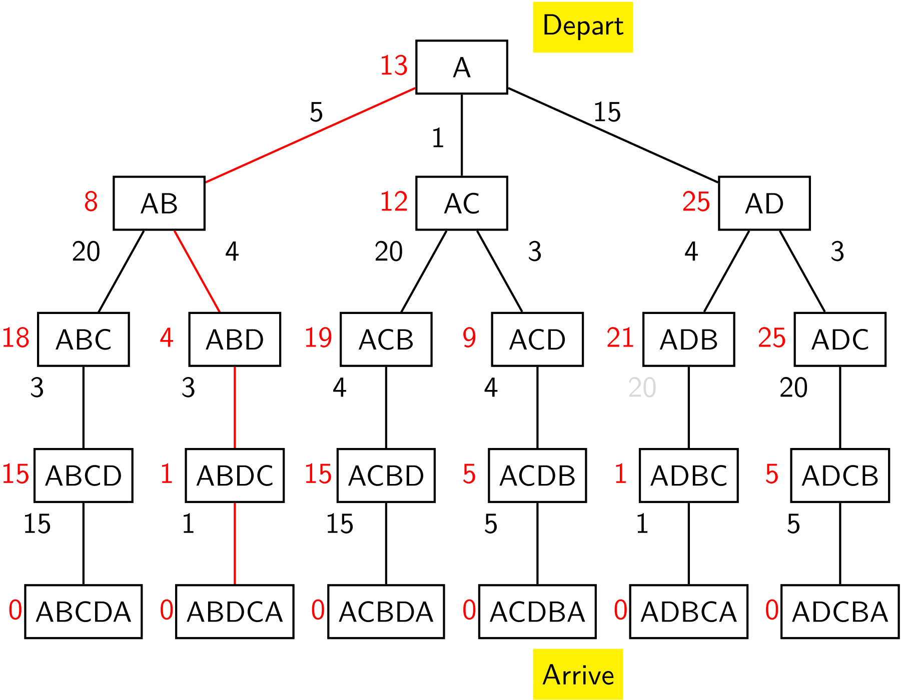


Depart
A
5
1
15
8
AB
12
AC
25
AD
20
4
20
3
4
3
18
ABC
4
ABD
19
ACB
9
ACD
21
ADB
25
ADC
3
3
4
4
20
20
15
ABCD
1
ABDC
15
ACBD
5
ACDB
1
ADBC
5
ADCB
15
1
15
5
1
5
0
ABCDA
0
ABDCA
0
ACBDA
0
ACDBA
0
ADBCA
0
ADCBA
Arrive


Depart
min. cost = 13
min. path = ?
13 A
5
1
15
8 AB
12 AC
25 AD
20
4
20
3
4
3
18 ABC
4 ABD
19 ACB
9 ACD
21 ADB
25 ADC
3
3
4
4
20
20
15 ABCD
1 ABDC
15 ACBD
5 ACDB
1 ADBC
5 ADCB
15
1
15
5
1
5
0 ABCDA
0 ABDCA
0 ACBDA
0 ACDBA
0 ADBCA
0 ADCBA
Arrive


Depart
13 A
5
1
15
8 AB
12 AC
25 AD
20
4
20
3
4
3
18 ABC
4 ABD
19 ACB
9 ACD
21 ADB
25 ADC
3
3
4
4
20
20
15 ABCD
1 ABDC
15 ACBD
5 ACDB
1 ADBC
5 ADCB
15
1
15
5
1
5
0 ABCDA
0 ABDCA
0 ACBDA
0 ACDBA
0 ADBCA
0 ADCBA
Arrive


Depart
13
A
5
1
15
8
AB
12
AC
25
AD
20
4
20
3
4
3
18
ABC
4
ABD
19
ACB
9
ACD
21
ADB
25
ADC
3
3
4
4
20
20
15
ABCD
1
ABDC
15
ACBD
5
ACDB
1
ADBC
5
ADCB
15
1
15
5
1
5
0
ABCDA
0
ABDCA
0
ACBDA
0
ACDBA
0
ADBCA
0
ADCBA
Arrive



Depart
A
13
5
1
15
AB
8
AC
12
AD
25
20
4
20
3
4
3
ABC
18
ABD
4
ACB
19
ACD
9
ADB
21
ADC
25
3
3
4
4
20
20
ABCD
15
ABDC
1
ACBD
15
ACDB
5
ADBC
1
ADCB
5
15
1
15
5
1
5
ABCDA
0
ABDCA
0
ACBDA
0
ACDBA
0
ADBCA
0
ADCBA
0
Arrive

Types of Dynamic Programming

Exact DP

- Can formulate any optimization problem
- Deterministic and stochastic
- Combinatorial (discrete)
- One decision maker and two player games
- Finite or infinite states
- BUT:
  - Curse of dimensionality
  - Need for a mathematical model

Types of Dynamic Programming (cont.)

Curse of dimensionality

- Linear in time
- Exponential in the states
- Exponential in the decision
- Exponential in the disturbances

Example (w/o disturbances)

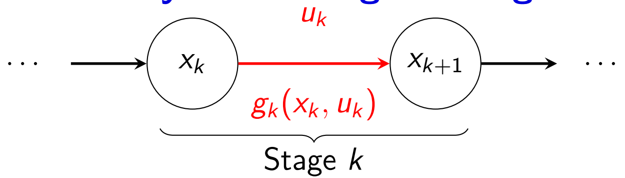
- 5 states, 5 controls with a horizon of 50
- State and controls discretized in 100 values (100^5)
- Flops $= \mathcal{O}(50 * 10^{10} * 10^{10}) \approx \mathcal{O}(10^{22})$
- It takes 31709 years with a PC with 10^{10} flops/s

Types of Dynamic Programming (cont.)

Approximate DP

- Approximation
- Simulation
- Very problem dependent

Deterministic Dynamic Programming



Deterministic problem

- System that evolves according to equations

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

where $u_k \in U_k(x_k)$

- Cost function that is accumulated in time

$$J(x_0; u_0, \ldots, u_{N-1}) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k) \tag{2}$$

Deterministic Dynamic Programming (cont.)

Deterministic problem (cont.)

- Minimize (2), over all control sequences that satisfy the constraint

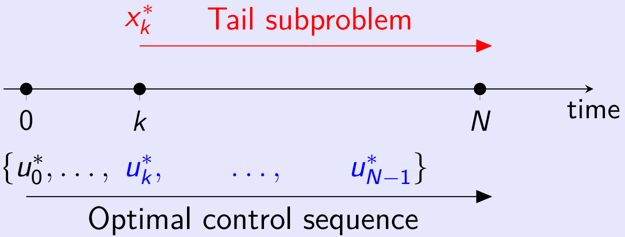
$$J^*(x_0) = \min_{\substack{u_k \in U_k(x_k) \\ k=0,\dots,N-1}} J(x_0, u_0, \dots, u_{N-1}) \qquad (3)$$

Formalize TSP

- States = partial tours (e.g. A, AB, ABC, ...)
- Controls = cities component (e.g. A, B, C, D)
- Constraints = visit all cities one time
- Cost function = cost between two cities

Principle of Optimality

Principle of Optimality



The tail of an optimal sequence is optimal for the tail subproblem

Principle of Optimality (cont.)

Example

- We want to go from Paris to Milano by car
- The optimal path pass through Lyon

$$\text{Paris} \rightarrow \text{Lyon} \rightarrow \text{Milano}$$

Tail optimal

Optimal sequence

- If we start our journey from Lyon, the optimal route to Milano is the same?

Idea

The optimal cost function can be constructed backwards

Algorithm

Algorithm idea

- Solve all the tail subproblems for all the states, using the solution of the previously sub-subproblems
- Given a x_k: take every $u_k \in U_k(x_k)$ and resolve the tail subproblem at state $x_{k+1} = f_k(x_k, u_k)$. We get the cost $g_k(x_k, u_k)$
- Minimize the cost by optimizing over the u_k

Algorithm (cont.)

Algorithm

- Start with final stage N

$$J_N^*(x_N) = g_N(x_N), \qquad \forall x_N \tag{4}$$

- For $k = 0, \dots, N-1$ and $\forall x_k$

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} \left[g_k(x_k, u_k) + J_{k+1}^*(f_k(x_k, u_k)) \right]$$

- We go backward and resolve the tail subproblems

$$J_N(x_N), J_{N-1}(x_{N-1}), \dots, J_0(x_0) = J_0^*(x_0)$$

Optimal control sequence

What we have

- We obtained the optimal cost function (min. cost in TSP)
- We need to find the optimal controls (route in TSP)

Optimal control sequence (cont.)

Costruct the optimal control sequence

- Start from x_0, the first optimal control is

$$u_0^* \in \arg \min_{u_0 \in U_0(x_0)} [g_0(x_0, u_0) + J_1^*(f_0(x_0, u_0))]$$

- We find $x_1^* = f_0(x_0, u_0^*)$
- We continue going forward for $k = 1, 2, \ldots, N-1$

$$u_k^* \in \arg \min_{u_k \in U_k(x_k^*)} [g_k(x_k^*, u_k) + J_{k+1}^*(f_k(x_k^*, u_k))]$$

- We find $x_{k+1}^* = f_k(x_k^*, u_k^*)$

Recap algorithms

Offline vs Online

- Offline: we compute the optimal cost functions with the Bellman equation
- Online: we compute the optimal control sequence when needed

Reminder

- The state space could be too large
- The cost function could be difficult to compute
- We can extend to approximate value function $\tilde{J}_k$

Extensions

Stochastic DP

- States affected by random parameter
- More difficult than deterministic

Infinite horizon

- Infinite number of stages
- Stationary system

Stochastic partial state information

- States not perfectly known
- Very difficult to solve

More about DP

[1] D. Bertsekas. *Reinforcement Learning and Optimal Control*. Athena Scientific optimization and computation series. Athena Scientific, 2019. ISBN: 9781886529397.

[2] D. Bertsekas. *Reinforcement Learning Course at ASU, Spring 2022*. URL: `https://www.youtube.com/playlist?list=PLmH30BG15SIoXhxLldoio0BhsIY84YMDj`.

[3] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. 3rd. Vol. I. Athena Scientific, 2005. ISBN: 9781886529434.

[4] Stuart Dreyfus. “Richard Bellman on the Birth of Dynamic Programming”. In: *Operations Research* 50.1 (2002), pp. 48–51. DOI: `10.1287/opre.50.1.48.17791`.

This presentation was very inspired by all of these works