

Dynamic programming

Dynamic programming

- Dynamic programming is a technique for making a **sequence of interrelated decisions**.
- Dynamic programming may be used to solve **Dynamic optimization problems**, where the problem is split into **stages**
- Each stage has a number of **states**, describing the possible conditions at which the system might be at that beginning of the stage
- The effect of the decision made at every stage is to transform the current state to a state associated with the next stage

The idea of the Dynamic programming

- Dynamic programming starts with **a small portion** (the last stage) of the original problem and finds its optimal solution.
- Next, Dynamic programming **enlarges the problem**, finding the current optimal solution from the preceding one (one stage is added)
- The stages are added one by one until the problem is solved.

Prototype example:

The fortune seeker problem

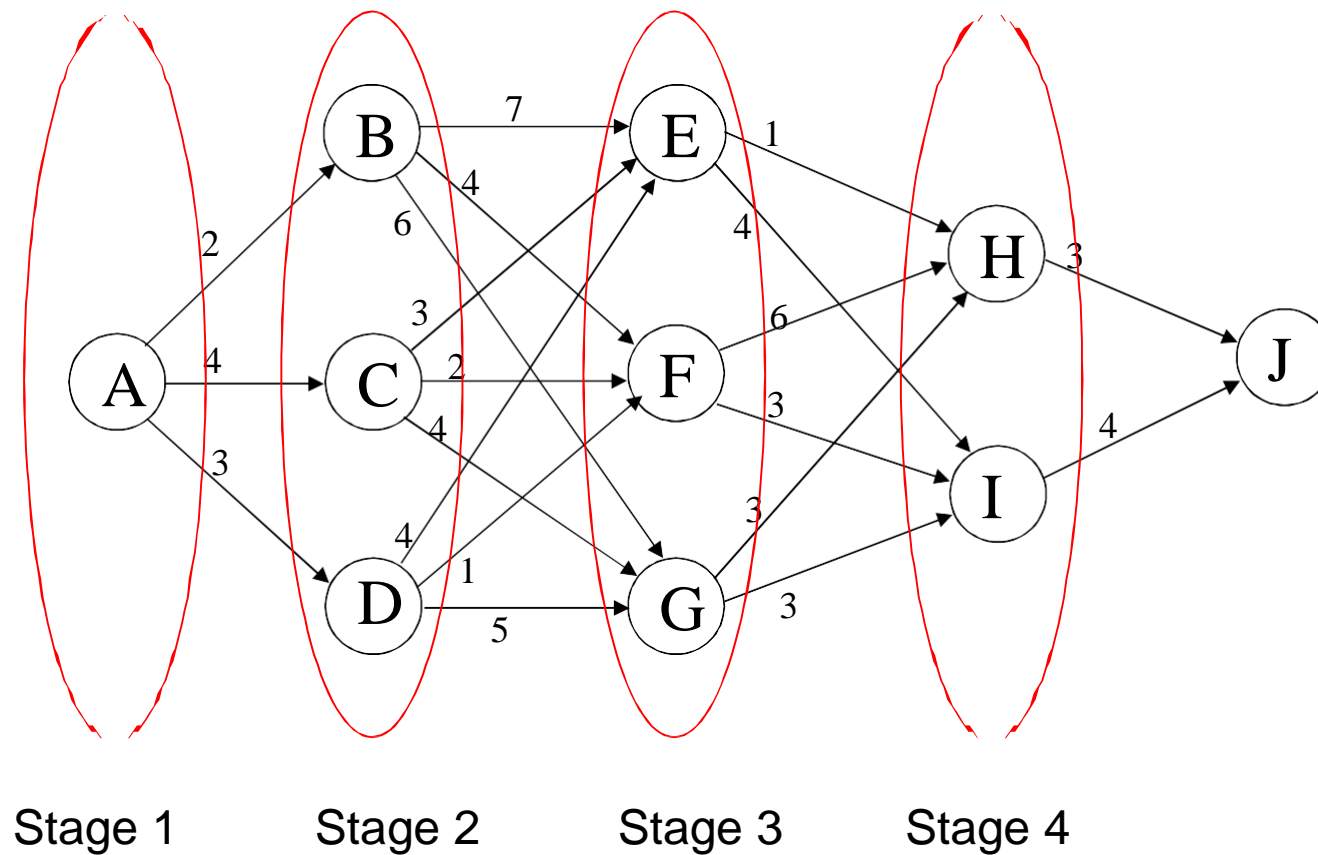
The fortune seeker is concerned about his safety.

Life insurance policy is based on a careful evaluation of the safety. The safest route should be the one with the cheapest total life insurance.

In fact, the problem is a **Shortest path problem**.

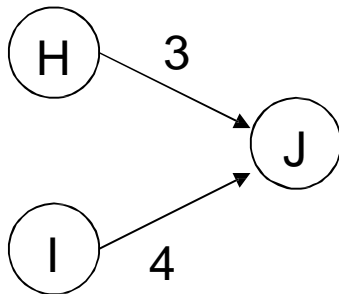
The fortune seeker problem

The road system and costs



The fortune seeker: stage 4

- When the fortune seeker has only one stage to go (**stage 4**), the life insurance cost is found explicitly:



This column is the **output** of stage 4

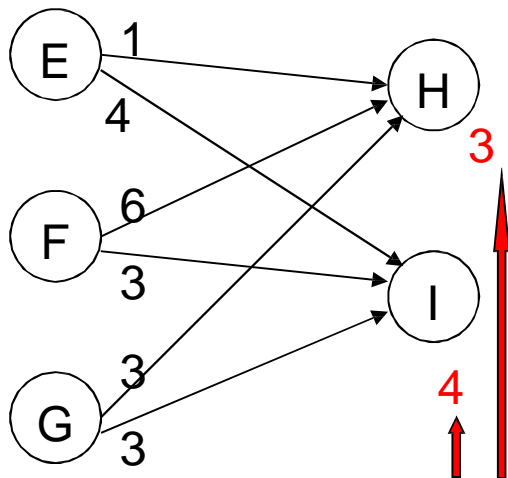


| X_3 | $F_3(x)$ | X_4 |
|-------|----------|-------|
| H | 3 | J |
| I | 4 | J |

The fortune seeker: stage 3

How to find the minimal cost from E to J, if the minimal costs H→J and I→J are already known (the output of the previous stage)?

$$F_2(X_2) = \min C(X_2 \rightarrow X_3) + F_3(X_3)$$



| X_2 | X_3 | $F_2(x)$ |
|-------|-------|---------------|
| E | H | $EH + F_3(H)$ |
| | I | $EI + F_3(I)$ |

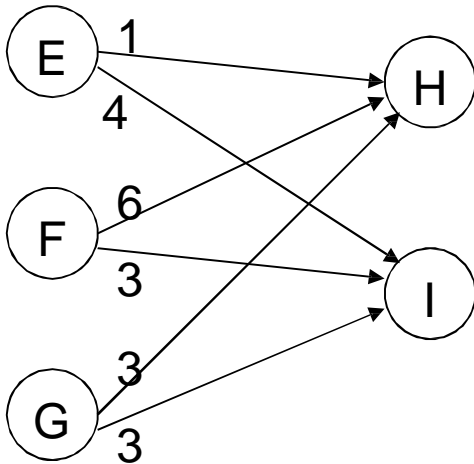
These are **outputs** of the previous stage 4

The fortune seeker: stage 3

Other nodes are done similarly:

$$F_2(X_2) = \min C(X_2 \rightarrow X_3) + F_3(X_3)$$

This column is the **output** of stage 3

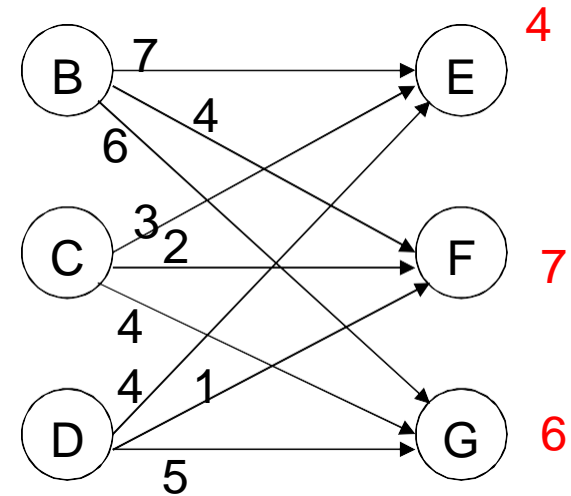


| X_2 | H | I | $F_2(X_2)$ | X_3 |
|-------|-----|-----|------------|-------|
| E | 1+3 | 4+4 | 4 | H |
| F | 6+3 | 3+4 | 7 | I |
| G | 3+3 | 3+4 | 6 | H |

The fortune seeker: stage 2

- Next, stage 2 is added to the problem.

$$F_1(X_1) = \min C(X_1 \rightarrow X_2) + F_2(X_2)$$

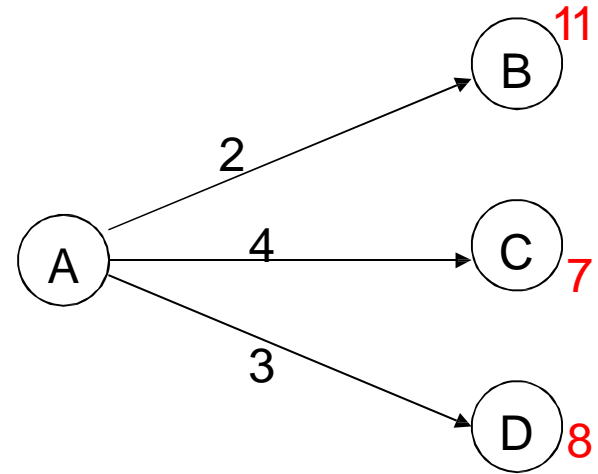


| X_1 | E | F | G | $F_1(X_1)$ | X_2^* |
|-------|-----|-----|-----|------------|---------|
| B | 7+4 | 4+7 | 6+6 | 11 | E, F |
| C | 3+4 | 2+7 | 4+6 | 7 | E |
| D | 4+4 | 1+7 | 5+6 | 8 | E, F |

The fortune seeker: stage 1

- Finally, stage 1 is added to the problem.
- $F_0(X_0) = \min C(X_0 \rightarrow X_1) + F_1(X_1)$

The whole problem has been solved.

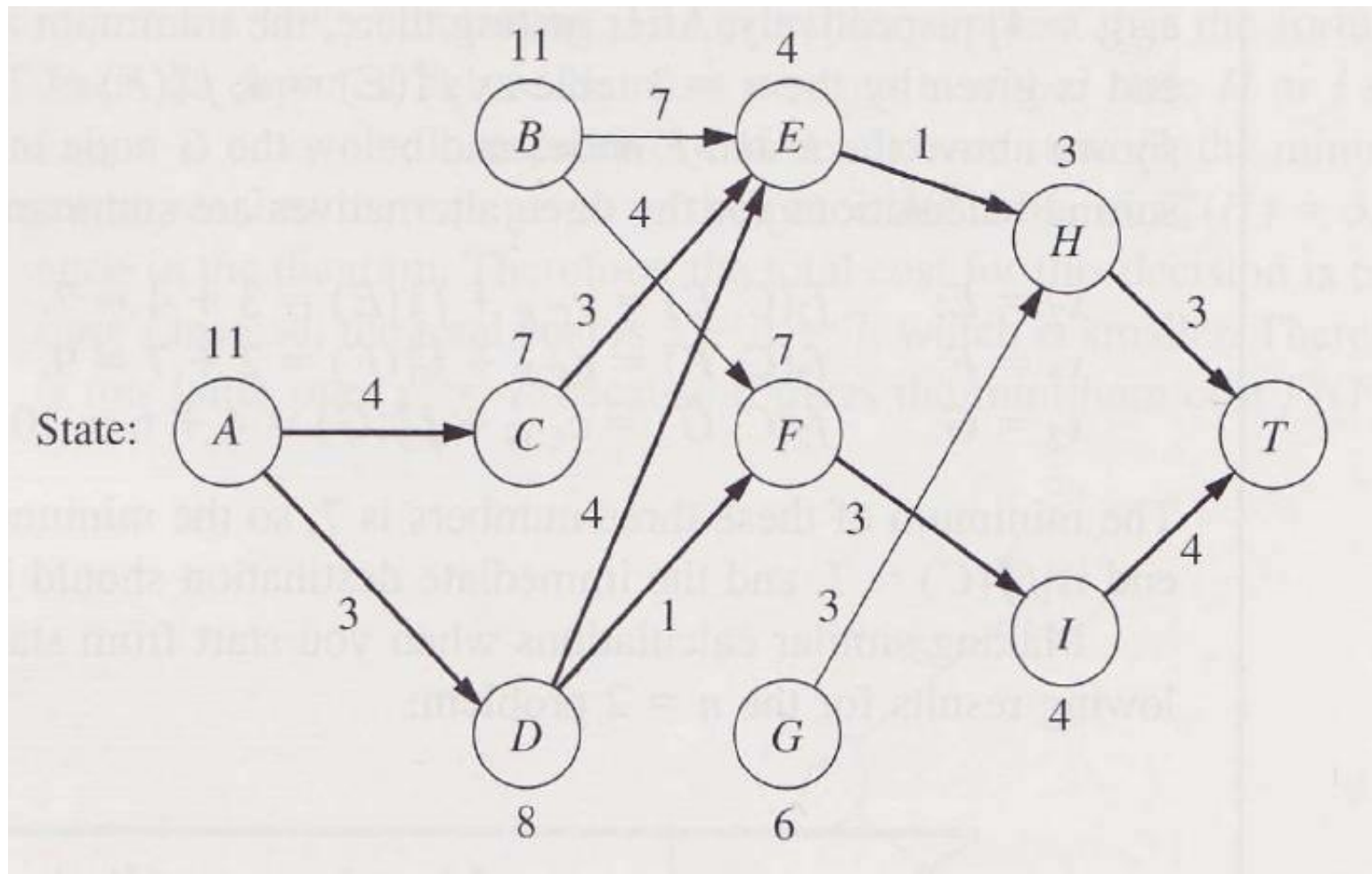


| X ₀ | B | C | D | $F_0(X_0)$ | X_1^* |
|----------------|------|-----|-----|------------|---------|
| A | 2+11 | 4+7 | 3+8 | 11 | C, D |

The fortune seeker: the solution

- The minimal costs to go from any node to the destination J are found in functions $F_4(X_4)$, $F_3(X_3)$, $F_2(X_2)$, $F_1(X_1)$.
- The selection of the optimal decision (arc) at every stage can be found in the last column of the tables.

The fortune seeker: the solution



Characteristics of the Dynamic programming problems

Characteristics of the Dynamic programming problems

- There is **no standard math formulation** of the problem. Rather, Dynamic programming is a general type of **approach to problem solving**, and its equations can be adopted to fit many problems.
- The problem is divided into **stages**, and a **decision is required at each stage**. Often the problem can be considered as a Dynamic optimization problem.

States and decisions

- Each stage has a number of **states** (for instant in the fortune seeker problem nodes are states). The number of state can be either finite or infinite.
- The effect of the current **decision** at each stage is **to transform the current state to a state at the next stage** (the fortune seeker's decision as to his next destination led him to a state at the next stage).

The role of the states

- Knowledge of the current state of the system conveys all the information about its previous behavior necessary for determining the optimal policy henceforth
- Any problem lacking this property cannot be solved as a dynamic programming problem

Solving procedure 1

- The procedure begins by finding the optimal policy for **the last stage**. The solution of the last stage problem is usually trivial.
- The solution procedure starts at the end and **moves backward** stage by stage until it finds the optimal policy starting at the initial stage.

Solving procedure 2

- A recursive relationship that identifies the solution for stage n , given the optimal solution for stage $n+1$, is available:

$$F_n(X_n) = \min_{X_{n+1}} (C(X_n, X_{n+1}) + F_{n+1}(X_{n+1}))$$

- The first term $C(x_n, X_{n+1})$ is the current costs associated with the current stage
 - The second term $F_{n+1}(X_{n+1})$ are the costs associated with the future stages
- Both the minimum cost $F_n(X_n)$ and the optimal policy $X_{n+1}^*(X_n)$ are defined.

The optimal policy

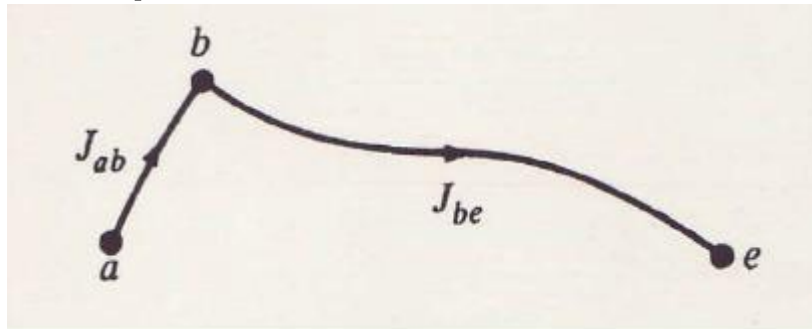
- The solution procedure is designed to find an **optimal policy** for the overall problem, i.e. a prescription of the optimal policy decision at **each** stage for **each** of the possible states.
- The optimal sequence of the decisions is found by searching the iteration results in the forward direction .

The principle of optimality

- An optimal policy has the property that whatever the initial state and initial decision are, the remaining decision must constitute an optimal policy with regard to the state resulting from the first decision

The principle of optimality

- The optimal path is shown in the figure:



The first decision (made at a) results in segment $a-b$ with cost J_{ab} and the remaining decisions yield segment $b-e$ at a cost of J_{be} .

- Assertion: If $a-b-e$ is the optimal path from a to e , then $b-e$ is the optimal path from b to e .

An example: Distributing
medical teams

Example: Distributing Medical teams to countries

- The world health council has 5 medical teams available to allocate them among three undeveloped countries to improve their medical care.
- How many teams should be allocated to each country to maximize the total efficiency?

Distributing Medical teams to countries

| | Extra person-years | | |
|---------|--------------------|---------|-----|
| Medical | of life, 1000 | | |
| teams | | Country | |
| | 1 | 2 | 3 |
| 0 | 0 | 0 | 0 |
| 1 | 45 | 20 | 50 |
| 2 | 70 | 45 | 70 |
| 3 | 90 | 75 | 80 |
| 4 | 105 | 110 | 100 |
| 5 | 120 | 150 | 130 |

There is no even **a fixed sequence of decisions.**

However, Dynamic programming is efficiently applied to this problem.

Distributing Medical teams to countries

- Three decision variables are available: x_1 , x_2 , x_3 . They will be considered as **three stages** of the problem.
- What is the 'state' in the distributing medical teams problem? Which information about the previous stages is needed to determine the optimal policy hereafter?

The number of still available medical teams is the 'state' (or the number of distributed teams may be used as well)

Distributing Medical teams to countries: stage 3

| | Extra person-years | | |
|---------|--------------------|---------|-----|
| Medical | of life, 1000 | | |
| teams | | Country | |
| | 1 | 2 | 3 |
| 0 | 0 | 0 | 0 |
| 1 | 45 | 20 | 50 |
| 2 | 70 | 45 | 70 |
| 3 | 90 | 75 | 80 |
| 4 | 105 | 110 | 100 |
| 5 | 120 | 150 | 130 |

| State | Minimal cost | Decision |
|-------|--------------|----------|
| s_3 | $F_3(s_3)$ | x_3 |
| 0 | 0 | 0 |
| 1 | 50 | 1 |
| 2 | 70 | 2 |
| 3 | 80 | 3 |
| 4 | 100 | 4 |
| 5 | 130 | 5 |

Distributing Medical teams to countries: stage 2

| | Extra person-years | | |
|---------------|--------------------|-----|-----|
| Medical teams | of life, 1000 | | |
| | Country | | |
| | 1 | 2 | 3 |
| 0 | 0 | 0 | 0 |
| 1 | 45 | 20 | 50 |
| 2 | 70 | 45 | 70 |
| 3 | 90 | 75 | 80 |
| 4 | 105 | 110 | 100 |
| 5 | 120 | 150 | 130 |

| State | Minimal cost | Decision |
|-------|--------------|----------|
| s_3 | $F_3(s_3)$ | x_3 |
| 0 | 0 | 0 |
| 1 | 50 | 1 |
| 2 | 70 | 2 |
| 3 | 80 | 3 |
| 4 | 100 | 4 |
| 5 | 130 | 5 |

| | X_2 | | | | | | | |
|-------|-------|-----|-----|-----|-----|-----|-------|-------|
| s_2 | 0 | 1 | 2 | 3 | 4 | 5 | F_2 | x_2 |
| 0 | 0 | | | | | | 0 | 0 |
| 1 | 50 | 20 | | | | | 50 | 0 |
| 2 | 70 | 70 | 45 | | | | 70 | 0,1 |
| 3 | 80 | 90 | 95 | 75 | | | 95 | 2 |
| 4 | 100 | 100 | 115 | 125 | 110 | | 125 | 3 |
| 5 | 130 | 120 | 125 | 145 | 160 | 150 | 160 | 4 |

Distributing Medical teams to countries: stage 1

| | Extra person-years | | |
|---------------|--------------------|-----------|-----------|
| Medical teams | of life, 1000 | | |
| | Country 1 | Country 2 | Country 3 |
| 0 | 0 | 0 | 0 |
| 1 | 45 | 20 | 50 |
| 2 | 70 | 45 | 70 |
| 3 | 90 | 75 | 80 |
| 4 | 105 | 110 | 100 |
| 5 | 120 | 150 | 130 |

| | X_1 | | | | | | | |
|-------|-------|-----|-----|-----|-----|-----|-------|-------|
| s_1 | 0 | 1 | 2 | 3 | 4 | 5 | F_1 | x_1 |
| 5 | 160 | 170 | 165 | 160 | 155 | 120 | 170 | 1 |

| s_2 | F_2 | x_2 |
|-------|-------|-------|
| 0 | 0 | 0 |
| 1 | 50 | 0 |
| 2 | 70 | 0,1 |
| 3 | 95 | 2 |
| 4 | 125 | 3 |
| 5 | 160 | 4 |

The optimal objective is 170 thousands of man-years

The optimal distribution is 1, 3, 1.

Example: Scheduling employment levels

- The workload for a LOCAL JOB SHOP is subject to seasonal fluctuation. Employment will not be permitted to fall below the levels given in the table:

| Season | Spring | Summer | Autumn | Winter | Spring |
|--------------|--------|--------|--------|--------|--------|
| Requirements | 255 | 220 | 240 | 200 | 255 |

- Any employment above this level costs 2000 USD per person per month.
- Hiring and firing costs are $200 \text{ USD} * \text{square of the difference of the employment levels}$.

Conclusions

- Dynamic programming is a technique for making **a sequence of interrelated decisions**.
 - Dynamic optimization, where the problem can be naturally spited into sequential stages
 - Provides a great computational benefits over exhaustive enumeration, especially for large problems

Conclusions

- Dynamic programming can be adopted to:
 - Both discrete and continuous time problems
 - Finite and infinite number of states
 - Several state variables and decision variables at a single stage (but the limitation is the Curse of dimensionality)

Conclusions

- Dynamic programming can not be applied, if the optimal solution for the rest of the problem **depends on** the decisions made at the previous stages
- Dynamic programming is applied, if the part of the optimal solution is the optimal solution for correspondent part of the original problem

Conclusions

- The main limitation is the Dynamic Programming is the **Curse of Dimensionality**.

It makes Dynamic programming technique practically infeasible even for moderate number of state variables and/or decision variables used at a stage