

Dynamic Programming

Cheng-Hung Wu

@NTUIE

Fall, 2017

Cheng-Hung Wu (@NTUIE)

Dynamic Programming

Fall, 2017

1 / 105

What is DP ?

- Dynamic Programming is an optimization procedure to solve problems requiring a sequence of inter-related decisions.
- Each decision transforms the current situation into a new situation.

What is DP ?

- Dynamic Programming is an optimization procedure to solve problems requiring a sequence of inter-related decisions.
- Each decision transforms the current situation into a new situation.

What do we want to cover ?

- Finite horizon deterministic dynamic programming
- Finite horizon stochastic dynamic programming

Cheng-Hung Wu

Dynamic Programming

Fall, 2017

2 / 105

Canonical Example :

Single item inventory problems : order a number of item each period in order to minimize expected (discounted) cost.

Cheng-Hung Wu

Dynamic Programming

Fall, 2017

2 / 105

Cheng-Hung Wu

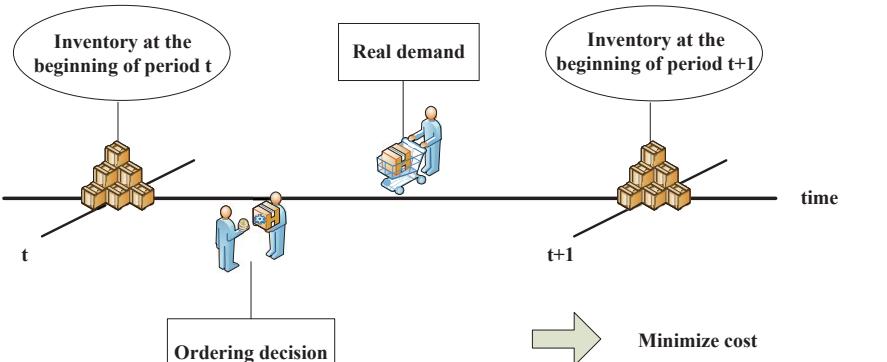
Dynamic Programming

Fall, 2017

3 / 105

Canonical Example :

Single item inventory problems : order a number of item each period in order to minimize expected (discounted) cost.



Characteristic of our example :

- (1) Must make ordering decision sequentially.
- (2) Decision are inter-related.
- (3) If I do not know exactly demand in the coming period, there are uncertainties.

We will see how to write equations for relationships between decisions.

For our single item inventory example :

Let X_t : inventory position at the beginning of period t .

D_t : demand during period t .

A_t : order placed at the beginning of period t . (order the item from supplier)

$$\Rightarrow X_{t+1} = X_t + A_t - D_t$$

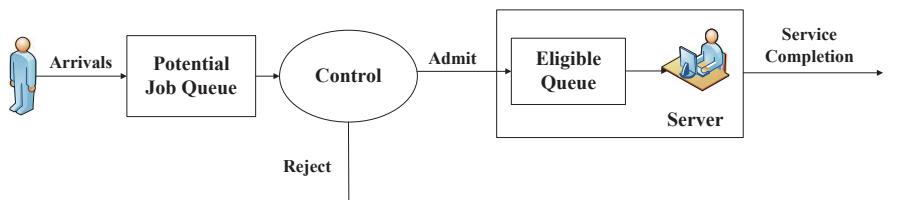
Can you write a state transition equation ?

$$\begin{aligned} X_{t+1} &= f(X_t, A_t, D_t) \\ &= X_t + A_t - D_t. \end{aligned}$$

(Deterministic case $D_t = d_t$ as constant)
(Stochastic case $D_t \sim F$)

Another example : queueing system

- Single server queue, batch arrivals
- Must decide how many (if any), to allow in your system.
 - Consider the web server
 - Holding cost $c(i)$, when there are i customer in the system.
 - Each served customer yields a reward of R .



For the queueing example :

Let

X_t = # of customers in the system at time t.

Y_t = # of customers seeking admittance at time t.

S_t = # of customers served during period t.

A_t = # of customers admitted.

$$X_{t+1} = X_t - S_t + A_t$$

Same equation as before ?

For the queueing example :

Let

X_t = # of customers in the system at time t.

Y_t = # of customers seeking admittance at time t.

S_t = # of customers served during period t.

A_t = # of customers admitted.

$$X_{t+1} = X_t - S_t + A_t$$

Same equation as before ? NO.

A_t is depend on Y_t .

However,

- decisions are still sequential.
- decisions are still inter-related.
- the interesting case is the stochastic case.

Reward Function :

$r(Z_n, A_n)$: reward function

$\bar{r}(Z_N)$: terminal reward

(Z_n : state, A_n : action)

Optimality Criterion :

For this course, we will consider for $0 < \alpha \leq 1$

$$V_\alpha^N(i) = \sum_{n=0}^{N-1} \alpha^n r(Z_n, A_n) + \bar{r}(Z_N)$$

(Starting at $Z_0 = i$)

($V_\alpha^N(i)$ is the total (discounted) reward over a N period planning horizon.)

If $\alpha = 1$. total reward problem

$\alpha < 1$. total discounted reward

In the stochastic case,

$$\hat{V}_\alpha^N(i) = E_i \left\{ \left[\sum_{n=0}^{N-1} \alpha^n r(Z_n, A_n) \right] + \alpha^N \bar{r}(Z_N) \right\}$$

Where $E_i = E(\cdot | Z_0 = i)$

Return to our queuing example, so we needed to know Y_t before we could decide what A_t to use ...

$Z_t = (X_t, Y_t)$ (2-dimensioned state space)

Routing :

Choose the time a vehicle leaves and the route it takes to get the destination.

N : Set of Nodes. (in a street network)

$A \subseteq N \times N$ (Directed Arcs)

$i \in N$ starting nodes.

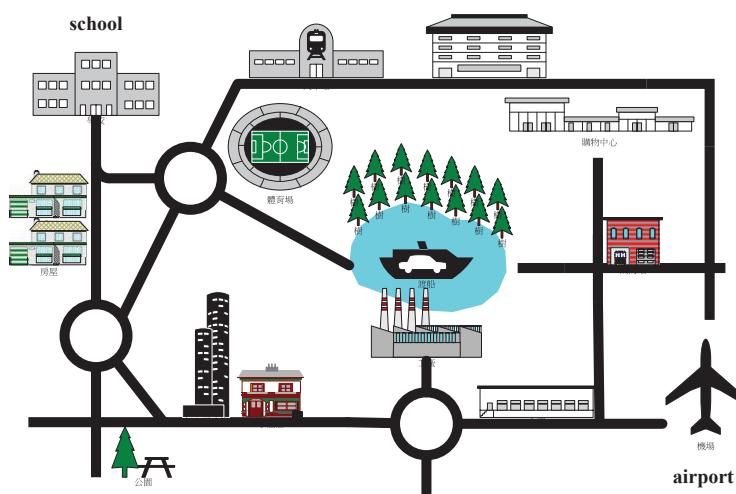
$r \subseteq N$ Goal set (maybe a singleton)

$c(n, n')$ = cost function for arc (n, n')

$c(n, t, n')$ = cost function depends on time

$\bar{c}(t)$ = terminal cost

EX (Routing) : Choose the time a vehicle leaves an origin and the route the vehicle takes to destination at a particular time (JIT).
(Kim,Lewis,White (2003))



If we are interested in minimizing the total cost

$$E_{i,t} \left\{ \sum_{k=1}^K c(n_k, t_k, n_{k+1}) + \bar{c}(t_K) \right\}$$

(t_k is the arrival time to Γ , and i is the initial node.)

So what do we need to know to make DP work ?

- state or current node
- current time (or stage, or period)
- planning horizon (how long do we look into the future)
- available decision (action, possibly state dependent)
- optimality criterion

For us,

- (1) Temporally additive cost functions (no products/non-after effect)
- (2) Finite planning horizons
- (3) Initially assume no uncertainties
- (4) Mathematically describable

What constitutes a solution ?

What constitutes a solution ?

We need to know:

- the optimal cost/profit
- the policy that achieves it.

EX 1 Inventory

- (a) Ordering quantity (inventory to hold) given current inventory position
- (b) Total cost

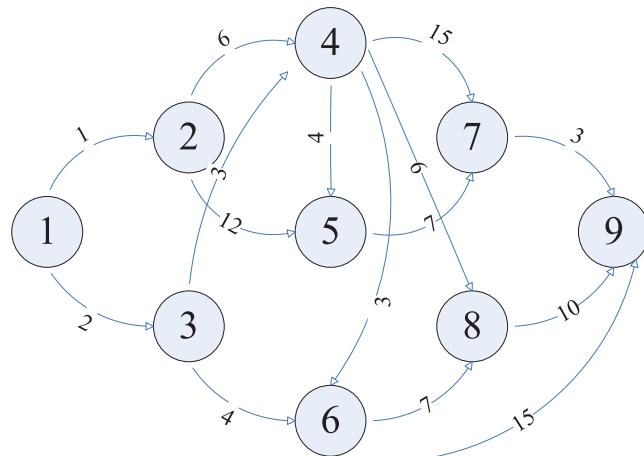
EX 2 Queuing Problem :

- (a) How many customers to admit (Given the current in system and the current batch size)
- (b) Total cost

Ex 3 Routing Problem

- (a) Optimal route to take and the departure time
- (b) Minimal cost from initial node i to Γ given the departure time

Min. path problem



Techniques used for "sequential decision problems"

- (1) In some cases , exhaustive search
- (2) Heuristic method
- (3) Dynamic Programming (DP)

Summary :

We will

- (1) Examine decision problems
 - (a) Deterministic and Stochastic
- (2) We must
 - (a) Model
 - (b) Define what constitute a solution
 - (c) Develop solution procedures

The prototypical sequential decision problem

- Shortest path problem
 - (1) In some cases , exhaustive search
 - (2) Heuristic method
 - (3) Dynamic Programming (DP)
- a How many different routes do we have in this shortest path problem?
- b How much calculation effort does it take?

It turns out exhaustive search require 55 adds and 12 comparisons, and there are 13 different paths.

The optimal routes 1 , 3 , 4 , 5 , 7 , 9 , cost 19.
Let's compare how to calculate the length of a path

(1) Suppose we compute the length of

1 , 2 , 4 , 6 , 8 , 9.

(2) Suppose we want to compute the length of

1 , 3 , 4 , 6 , 8 , 9.

We can reuse calculation of 4 , 6 , 8 , 9

(3) Similarly for

1 , 2 , 4 , 7 , 9

1 , 2 , 4 , 6 , 8 , 9

Backward and forward induction exploit these facts

some notation :

S: set of nodes

$T \subseteq S \times S$, directed arcs

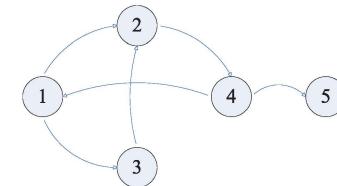
$SCS(i) = \{j : (i, j) \in T\}$

($SCS(i)$: successive set)

EX1 :

$$S = \{1, 2, \dots, 5\}$$

$$T = \{(1, 2), (1, 3), (2, 4), (3, 2), (4, 1), (4, 5)\}$$



Let t_{ij} be the length of (i, j)

Theorem

For path (i_1, i_2, \dots, i_n) , the path length = $\sum_{k=1}^{n-1} t_{i_k, i_{k+1}}$

Let's start with finite, acyclic networks

Fact : One can label finite , acyclic network
s.t. $(i, j) \in T$ is s.t. $i < j$

1. cyclic : There is a cycle in the graph.
 $SCS(4) = \{1, 5\}$

2. acyclic : There is no cycles in the graph.

Backward Induction (Backward DP)

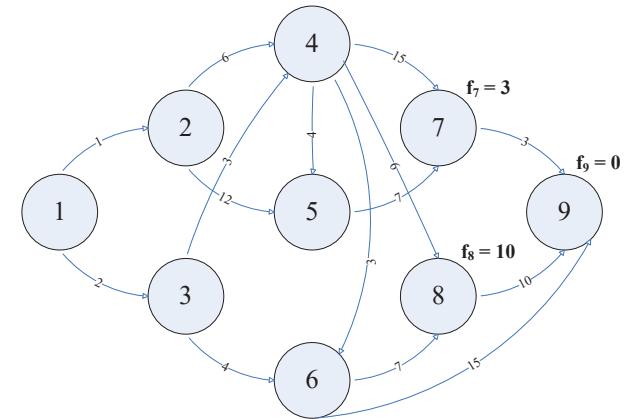
suppose $S = \{1, \dots, k\}$ and arcs are s.t. $i < j$

if $(i, j) \in T$
 (k : terminal node)

Let $f_i = \min$ travel cost from i to k

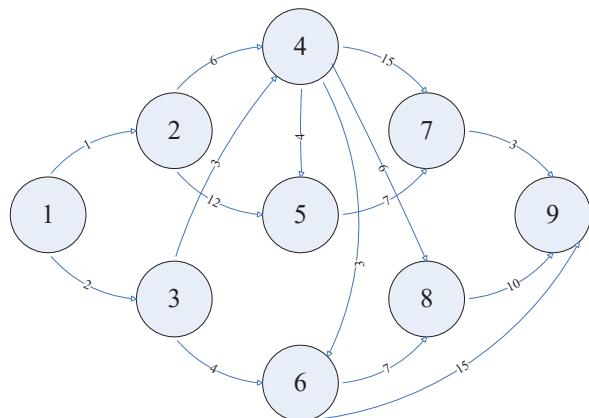
$$f_k = 0$$

Return to the Example



$$\begin{aligned} 2+, 1\text{ compare } f_6 &= 15 & \min(7 + 10, 15 + 0) \\ f_5 &= \\ f_4 &= \end{aligned}$$

Return to the Example

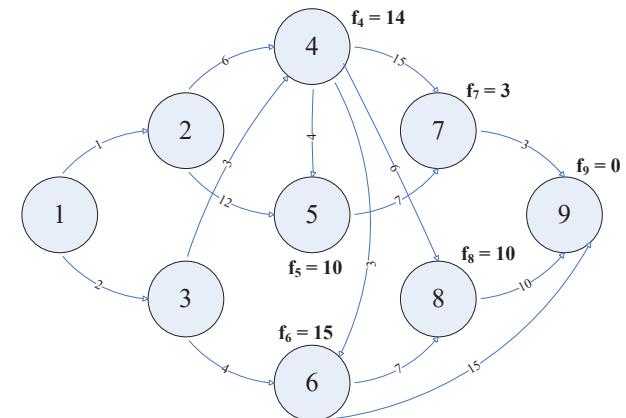


$$f_9 = 0$$

$$f_8 =$$

$$f_7 =$$

Return to the Example



$$f_3 =$$

$$f_2 =$$

$$f_1 =$$

Optimal Strategy is

$$= 1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 9$$

Total computational effort : 12 adds , 7 compare

Recall exhaustive search : 55 adds, 12 compare

How would we construct (dynamically) the shortest path tree ?

- As we compute the optimal costs , delete arcs that are not on the shortest path from each node to k.

Ex :

$$\begin{aligned}f_4 &= \min(4 + f_5, 3 + f_6, 15 + f_7, 7 + f_8) \\&= 4 + f_5\end{aligned}$$

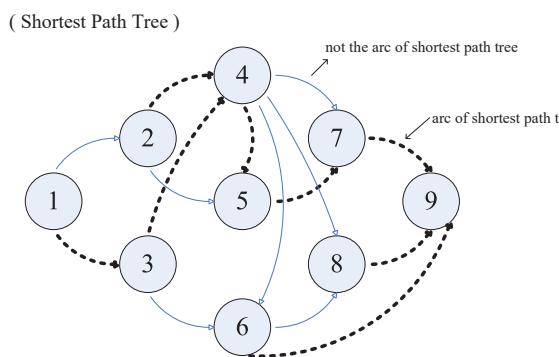
⇒ delete (4,6) , (4,7) , (4,8) from the original network to form the shortest tree from 4 to 9

(What if the cost is equal ?)

In general, optimality equation

$$f_i = \min\{t_{ij} + f_j : (i,j) \in T\}, f_k = 0$$

((i,j) ∈ T : reduce each sub. problem to the successive set SCS(i) of i)



Theorem

A policy tells you what to do when you are at a particular node or state

Example : Here is a "good" policy

i j

(If at node i , go to node j)

1	3
2	4
3	4
4	5
5	7
6	9
7	9
8	9

(shortest path tree is a good policy)

For acyclic graph with finite nodes :

Theorem

f_i is the minimum distance from i to sink

Optimality equation $f_i = \min\{t_{ij} + f_j : (i, j) \in T\}$
 $((i, j) \in T \text{ or } j \in SCS(i))$

Boundary condition $f_j = 0, \forall j \in \Gamma$

(This algorithm will lead to a "policy", which is now a set of if-then statement.)

A few notes :

- (1) D.P. offers a significant reduction in the # of operations v.s. exhaustive search
- (2) Sub-paths of optimal paths are optimal (principle of optimality)
- (3) D.P. gives both optimal cost and policy
- (4) Problem is decomposed into finding the solution of a set of smaller problem f_i

Two things,

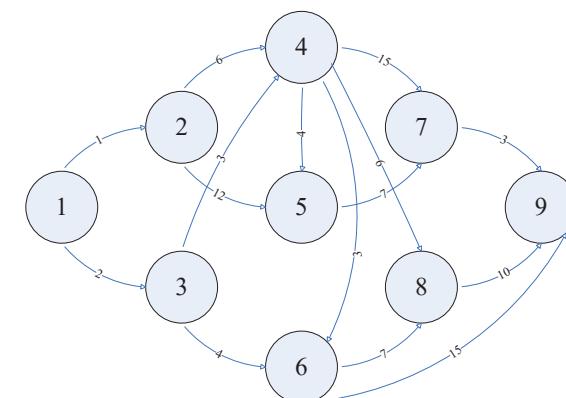
- (1) We will use backward induction more often
 (Because in stochastic case, forward induction won't work)
- (2) D.P. produces an optimal path and a minimum path cost (guaranteed)

It's possible , we could reduce the amount of work using heuristics

- possible at the loss of optimality

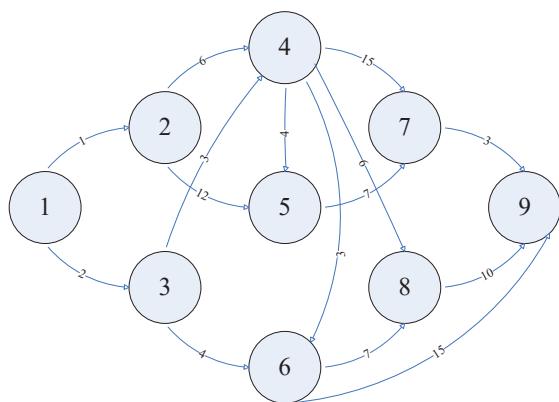
Ex:

A heuristic rule : go to the node with the lowest immediate cost (Myopic !)
forward :



Ex:

A heuristic rule : go to the node with the lowest immediate cost (Myopic !)
forward :

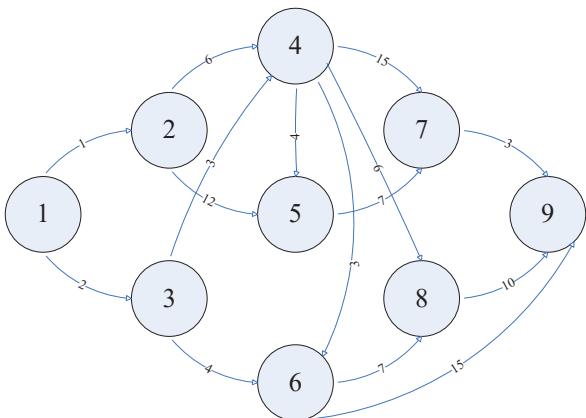


$1 \rightarrow 2 \rightarrow 4 \rightarrow 6 \rightarrow 8 \rightarrow 9$

total 27

this is the max length

backward :



$1 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 7 \rightarrow 9$

total 19

this is the min length

Both Forward and Backward Induction use something called "Recursive Fixing"

Consider how they work in Longest Route Problem

(1) Set $V_1 = 0, V_j = -\infty, j = 2, \dots, N$

Do for $j = 2, \dots, N$

Do for $i = 1, \dots, j-1$

$$V_j = \max\{V_j, V_i + t_{ij}\}$$

(If $(i, j) \notin T$ let $t_{ij} = -\infty$)

Note : Recursive Fixing examines each node and "fixes" the value at that node based on the "Ancestor" nodes.

Once we fix the nodes , they are fixed forever

Embedding : we embedded the problem of interest (Find f_1 or g_9) into a set of problems (finding $f_i = 1, \dots, 9$)

This is called the Recursive Feature.

Functional Equation (optimality equation)

Example: $f_i = \max_j \{t_{ij} + f_j, (i, j) \in T\}, f_\alpha = 0 \quad \forall \alpha \in \text{Destination}$

Recursive Fixing : The process of recursively determining the solution of the optimality equation

Principle of Optimality

- (1) All sub-paths of the optimal path are optimal
- (2) There exists an optimal policy (that is optimal for every state)
- (3) An optimal policy is s.t. whatever the initial state (node) and decision the remaining state and decisions must be optimal as well

$$g_2 = \max \{-4 + (0.9) \times 20, -2 + (0.75) \times 20\} \\ = \max \{14, 13\} = 14$$

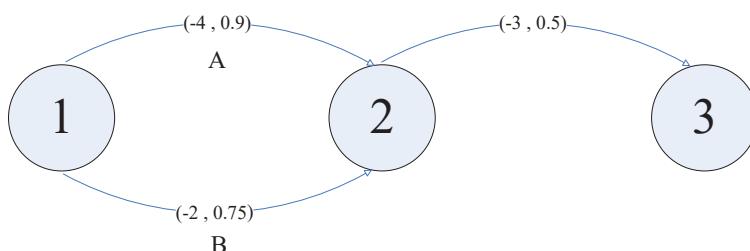
By route A $g_3 \equiv -4 - 3 + (0.9)(0.5)(20) = 2$
However, by route B, $-2 - 3 + (0.75)(0.5)(20) = 2.5$

Why? It turns out the objective is not additive , it is a mixture of add and multiplies.

(Note: we assume temporally additive cost/reward structure from the very beginning.)

What if the objective functions or costs are not additive? Suppose that there are 2 routes

If they arrive on time \Rightarrow Then you have \$20 bounds



First element = cost of travel

Second element = prob. of arriving on time

Pseudo code for Recursive Fixing

Pseudo code for Recursive Fixing (Backward / Min. Path)

- (1) Set $V_N = 0$, $V_j = \infty$, $j = 1, \dots, N-1$
- (2) Do for $i = N-1, \dots, 1$
Do for $j = i, \dots, N$
$$V_i = \min\{V_i, t_{ij} + V_j\}$$

Now , Resource Allocation

Example:

Deployment of Fighters

Bandwidth to various information classes

Worker to Jobs

Money to Investments

Capacity to Different Products (Product-Mix)

General Problem

Single resource allocation problem

- Suppose we have K units of a resource available
- This resource is allocated to N different commodities
(for now assume allocation in integer quantities)

What case do I need to know?

Producing/Using x_n units of/at commodity

- (1) Consumes $C_n(x_n)$ units of resource
- (2) Yields profit $P_n(x_n)$
- (3) Assume $x_n \leq B, \forall n \Rightarrow$ (upper bound)

Problem: Maximize Profit subject to resource constraints
 Mathematical Programming Formulation

$$\begin{aligned} \Rightarrow \text{Max } & \{P_1(x_1) + \dots + P_N(x_N)\} \\ \text{s.t. } & C_1(x_1) + \dots + C_N(x_N) \leq K \\ & x_n, \text{integer} \quad \forall n \\ & 0 \leq x_n \leq B \quad \forall n \end{aligned}$$

(If cost and constraints are linear and integer constraints, this is an LP or MILP)

The LP formulation

$$\begin{aligned} \text{Max } & \left\{ \sum_{n=0}^N P_n(x_n) \right\} \\ \text{s.t. } & \sum_{n=0}^N C_n(x_n) = K \\ & x_n \geq 0, n = 0, \dots, N \end{aligned}$$

I have made a few small change

(1) A 0th commodity akin to empty space

This is called the "Knapsack Problem" like placing objects in a Knapsack with limited space

Examples:

Allocate 5 medical teams to 3 countries to improve health education and training programs

Scenario: the world health council needs to determine how many (if any) to allocate to each country to maximize total effectiveness of the 5 teams
 Measure of Effectiveness : MOE , additional person-years of life

Estimated additional person-year of life (in multiple of 1000) for each country

# of teams	countries		
	1	2	3
0	0	0	0
1	50	20	45
2	70	45	70
3	80	75	90
4	100	110	105
5	130	150	120

Theorem

$f_n(y) = \max \text{ total "profit" obtained from allocating } y \text{ units of your resource to the "production" of commodities 1 through } n \text{ (n: stage)}$

Objective Find $f_N(K)$ and an allocation policy $(x_1^*, x_2^*, \dots, x_N^*)$

Boundary Conditional

$$f_1(y) = \max_{x_1} P_1(x_1)$$

$$0 \leq x_1 \leq B$$

$$0 \leq x_1 \leq y$$

Optimality Equation's

$$f_n(y) = \max_{x_n} \{P_n(x_n) + f_{n-1}(y - c_n(x_n))\}$$

$$0 \leq x_n \leq B$$

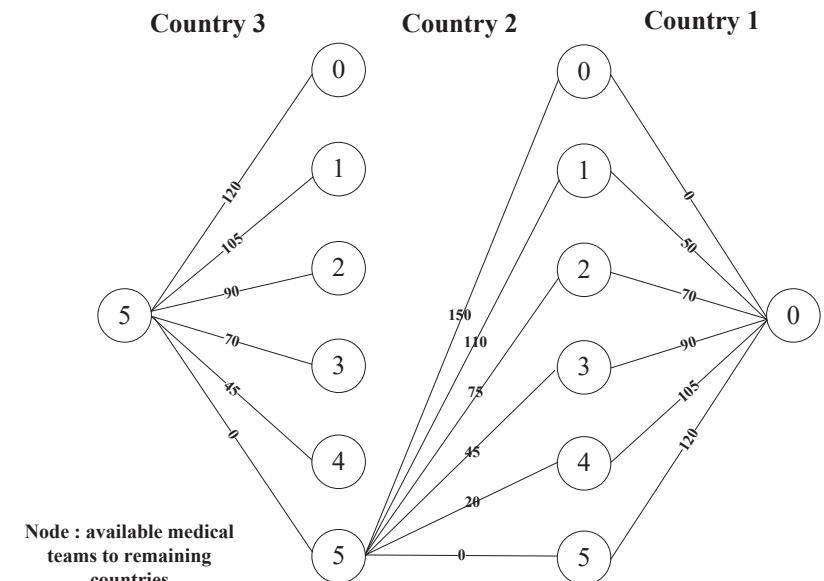
$$0 \leq c_n(x_n) \leq y$$

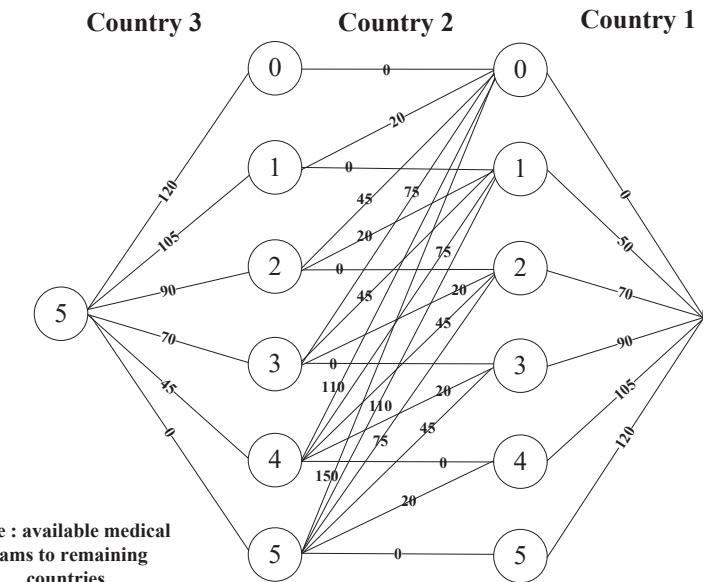
$$x_n : \text{integer}$$

Solution Procedure

Recursive Fixing —Backward Induction

Procedure : determine $f_1(y) \quad \forall y$
then $f_2(y) \quad \forall y$
.
.
.
 $f_N(y) \quad \forall y$





Recall our example:

Let $x_n = \#$ of medical teams allocated to country n

$$\begin{aligned} \text{Max } & \sum_{n=1}^3 P_n(x_n) \\ \text{s.t. } & \sum_{n=1}^3 x_n = 5, \quad x_n \geq 0, \text{ int.} \\ & (c_n(x_n) = x_n) \end{aligned}$$

$f_n(y) = \max$ estimated additional person years of life achieved by allocation y team to country $1, \dots, n$

we know $y \leq 5$ and y is integer, Then

$$f_n(y) = \max\{P_n(x_n) + f_{n-1}(y - x_n); x_n \leq y, x_n \in \mathbb{Z}^+\}$$

$$f_1(y) = \max\{P_1(x_1); x_1 \leq y, x_1 \in \mathbb{Z}^+\}$$

We start with $f_1(y)$

y	$f_1(y)$	x_1^*
0	0	0
1	1	
2	2	
3	3	
4	4	
5	5	

$$f_1(y) = \max\{P_1(x_1); x_1 \leq y, x_1 \in \mathbb{Z}^+\}$$

We start with $f_1(y)$

y	$f_1(y)$	x_1^*
0	0	0
1	50	1
2	70	2
3	80	3
4	100	4
5	130	5

$$f_2(y) = \max\{P_2(x_2) + f_1(y - x_2)\}$$

$$f_2(y) = \max\{P_2(x_2) + f_1(y - x_2)\}$$

x_2	0	1	2	3	4	5	$f_2(y)$	x_2^*
y	0	50	20				0	0
	1	70	70	45			50	0
	2	80	90	95	75		70	0 or 1
	3	100	100	115	125	110	95	2
	4	130	120	125	145	160	125	3
	5						160	4

I

$$f_2(y) = \max\{P_2(x_2) + f_1(y - x_2)\}$$

x_2	0	1	2	3	4	5	$f_2(y)$	x_2^*
y	0	1	2	3	4	5		
0								
1								
2								
3								
4								
5								
I								

$$f_3(y) = \max\{P_3(x_3) + f_2(y - x_3)\}$$

x_3	0	1	2	3	4	5	$f_3(y)$	x_3^*
y	0	1	2	3	4	5		
0								
1								
2								
3								
4								
5								
I								

$$f_3(y) = \max\{P_3(x_3) + f_2(y - x_3)\}$$

x_3	0	1	2	3	4	5	$f_3(y)$	x_3^*
y	160	170	165	160	155	120	170	1

More resource allocation examples:

- An airline needs to allocate space to various customer classes
 - Make it easy ,all seats are the same size

To formulate the problem as DP what information do we need?

- \$ earned from seat of each class
- Available space in a plane (total number of seats)
- Number of seat types

$f_n(y) = \max$ profit from a y -seat plane and classes $1, \dots, n$ of customers.

Suppose you have some stock and some prospective buyers ...

- How much stocks do you have -K units
 - How many buyer do you have -N Buyers
 - How much is earned for each buyer? $-P_n(x_n)$ Mamer("1984")
- $f_n(y) = \max$ total profit from y units of stock to n buyers

Can you write math program to solve the max total reward problem ?
 Yes, but difficult to solve. Knapsack problems with integer decision variables are known to be NP-complete problems. Dynamic programming is a possible way to reduce the computational complexity.

And the optimality equation ?
 $(c_n(x_n), d_n(x_n)) \in Z^+$, and $x_n \in Z^+$
 $f_n(y, z) = \text{Max}_{x_n \in A} \{ P_n(x_n) + f_{n-1}(y - c_n(x_n), z - d_n(x_n)) \}$

where,
 $A \equiv \{x_n \mid c_n(x_n) \leq y, d_n(x_n) \leq z, x_n \leq B, x_n : \text{non-negative integer}\}$

Boundary Condition: $f_1(y, z) = \max_{x \in Z} (P_1(x_1))$
 So in theory , we could solve this problem
 But now our state space is larger, we might have dimensional problems.
 $\{1, 2, \dots\} \times \{0, 1, \dots, K\} \times \{0, 1, \dots, L\}$
 $(\{1, 2, \dots\} \times \{0, 1, \dots, K\} \times \{0, 1, \dots, L\}) : \# \text{ of states}$

In general as additional resources are introduced ,we must add more state variables,

$/$: type of resource

Y : is the # of each resource

There are Y^I states to consider for each time period (decision epoch)
 This is the so-called "curse of dimensionality"
 (classic work Gallego and Van Ryzin)

A state is the amount of information that is sufficient to

- (1) Describe transitions
- (2) Enable induction
- (3) Enough to make decisions

Examples:

- (1) Med Team :
- (2) Seat Allocation :
- (3) Stock Example :

Examples:

- (1) Med Team : must know # of countries left ,# of teams remaining
"y"
- (2) Seat Allocation :
- (3) Stock Example :

Examples:

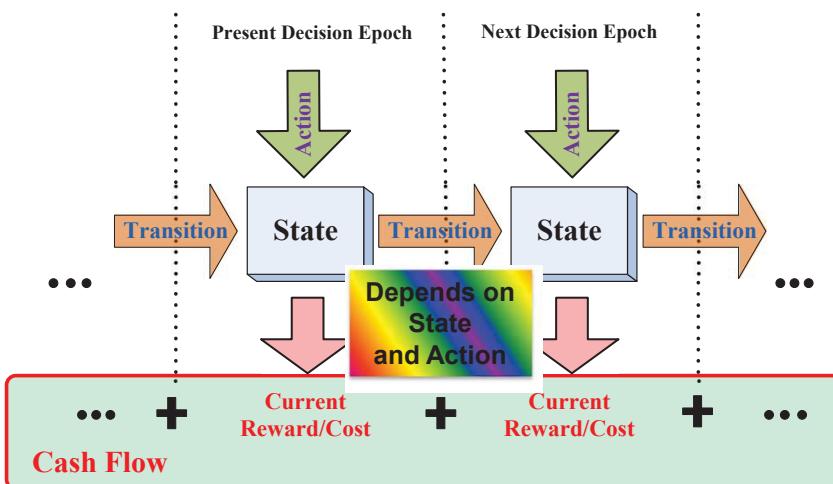
- (1) Med Team : must know # of countries left ,# of teams remaining
"y"
- (2) Seat Allocation : # of remaining customer classes, # of seats remaining
- (3) Stock Example :

Examples:

- (1) Med Team : must know # of countries left ,# of teams remaining
"y"
 - (2) Seat Allocation : # of remaining customer classes, # of seats remaining
 - (3) Stock Example : # of buyers,# of stocks remaining
(state:# of teams remaining y,# of seats remaining,# of stocks remaining)
(stage/time: # countries/ customer/ buyers)
- Product-Mix Example:

With each state comes an action set, from which we make choice,

- Our choice cause a state change which we observe and continue



5 parts to make our dynamic programming model complete

$$\{T, S, A_s, t_n(s_n, a_n), r_n(s_n, a_n)\}$$

T : set of decision epochs, $T = \{1, \dots, N\}$ the time horizon

S : state space, could divide S into $S = S_1 \times S_2 \times \dots \times S_{n+1}$ (not usually done)

$A \equiv \bigcup_{s \in S} A_s \doteq$ action space

$t_n(s_n, a_n)$ (How nodes gets from current state to next): transition structure, when in state s_n , choosing action $a_n \in A_{s_n}$, causes us to move to state $t_n(s_n, a_n) = s_{n+1}$

$r_n(s_n, a_n)$: reward earned or cost accrued from choosing action a_n in state s_n

Example: A company that makes fighter planes has one fighter on hand at the beginning of the current month. Orders for this month and for next three months are 1; 2; 1; 0 fighters, respectively. The company wants to have one fighter on hand at the beginning of the fifth month, which means that a total of 4 fighters must be manufactured over the next 4 months. Orders for a particular month may be filled from that month's production or from inventory. The problem is to find a production schedule that satisfies demand and minimizes the total costs. The cost of producing 0, 1, or 2 fighters in a given month is 10, 17, and 20, respectively. The cost of having 0, 1, or 2 machines in inventory at the start of a month is 0, 3, and 7, respectively.

- Draw a network whose shortest path is the best production schedule.
- Find the best production schedule.

Now, let's move to stochastic dynamic decision problems.

Recall

Definition

A policy π that uses the same decision rule, for each epochs is called stationary

Let π^{SD} , stationary deterministic

What is the smallest set of policies ?

(1) Stationary deterministic

Which one would you not expect to be enough for finite horizon problems?

- Stationary (can not always keep)
- Deterministic
- Markovian

Definition

Suppose we specify an action for every state at a particular decision epoch
This is called a decision rule

A decision rule " d_n " at epoch n , is then a function from state space to action set $d_n : S \rightarrow A$

Let's redefine our optimality equation:

Let $\pi = \{d_1, d_2, \dots, d_N\}$ be a sequence of decision rules

$$f(s, \pi) = \sum_{n=1}^N r_n(s_n, d_n(s_n)) + \bar{r}_{N+1}(s_{N+1})$$

, where $s_1 = s$, $\bar{r}(s_{N+1})$: terminal reward function

We are in search of $g(s) = \sup_{\pi} \{f(s, \pi)\}$

Inventory Control :

- We must order Inventory for N periods to meet (stochastic demand)

Let, s_k = inventory position at the beginning of k^{th} period

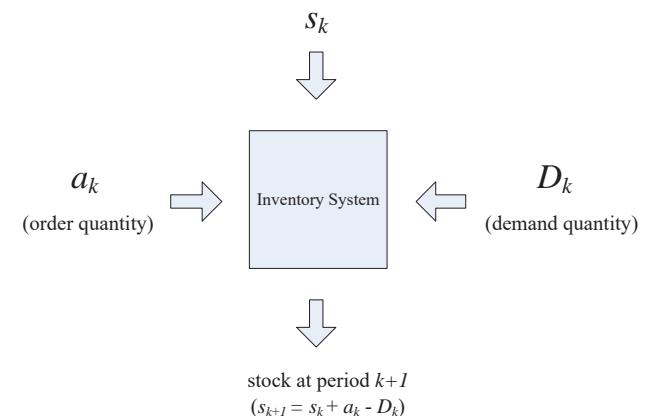
a_k = order quantity in period k (immediately delivery)

D_k : bounded demand during period k (Assume i.i.d. for all periods)

c : per unit production cost

$H(i)$: holding cost for $i > 0$, backlogging cost for $i < 0$

A picture, stock at period k



$$\begin{aligned} s_{k+1} &= s_k + a_k - D_k \\ c(a_k) + h(s_k + a_k - D_k) & \\ (c(a_k) : \text{cost of order}, s_k + a_k - D_k : \text{holding cost}) \end{aligned}$$

5 Key Elements for any stochastic dynamic programming model:

$$\{T, S, A, t_n(s_n, a_n), r_n(s_n, a_n)\}$$

T: time, epoch

S: state space

A: action space

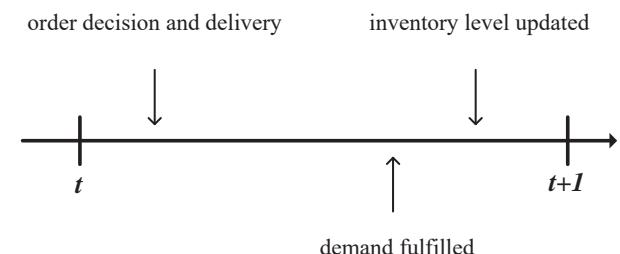
$t_n(s_n, a_n)$: How nodes get from current state to next state

$r_n(s_n, a_n)$: reward/cost function

- Order decision and delivery. (beginning of each month)
- Demands are filled before the end of each month
- If demand exceeds inventory, sales are lost. (no backlogging)
- Revenue, cost and demand are stationary (do not vary from month to month)
- Warehouse capacity M

$$\Rightarrow s_{t+1} = \max\{s_t + a_t - D_t, 0\}, \text{ (inventory level)}$$

$$= [s_t + a_t - D_t]^+$$



5 Key Elements for any stochastic dynamic programming model:

T: set of decision epochs = {1, 2, ..., N}

S: state space

$$A: \bigcup_{s \in S} A_s$$

$t_n(s_n, a_n)$: transition structure, in state s_n choosing a_n causes us to move to s_{n+1}

$r_n(s_n, a_n)$: reward earned or cost accrued from choosing a_n in state s_n

$$O(a_t) : \text{ordering cost}$$

$$r_t(s_t, a_t, s_{t+1}) = -O(a_t) - h(s_t + a_t) + f(s_t + a_t - s_{t+1})$$

ordering cost

$$O(a_t) = \begin{cases} K + c(a_t) & , \quad \text{if } a_t > 0 \\ 0 & , \quad \text{if } a_t = 0 \end{cases}$$

$$r_N(s) = g(s), \text{ (t=N, terminal reward)}$$

$$h(s_t + a_t) : \text{holding cost}$$

A Numerical example :

$$p(\text{demand} = j) = p_j$$

$$p_j = \begin{cases} \frac{1}{4}, & \text{if } j = 0 \\ \frac{1}{2}, & \text{if } j = 1 \\ \frac{1}{4}, & \text{if } j = 2 \end{cases}$$

$$f(s_t + a_t - s_{t+1}) = 8(s_t + a_t - s_{t+1}), \text{ revenue}$$

M = 3, warehouse capacity

g(u) = 0, salvage value, after the final period

N = 3, # of periods

h(u) = u, inventory cost

K = 4, ordering cost (fixed)

c(u) = 2u, ordering cost (variable)

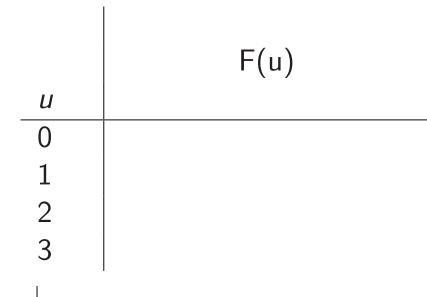
Let $u = s_t + a_t$, (inventory level after ordering decision)

Define

$$\begin{aligned} F(u) &= E[f(s_t + a_t - s_{t+1})] \\ &= \sum_{j=0}^{u-1} f(j)p_j + f(u)q_u, \quad \text{where } q_u = \sum_{j=u}^{\infty} p_j \end{aligned}$$

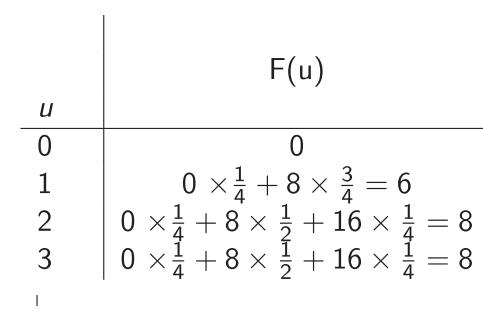
$$F(u) = E[f(s_t + a_t - s_{t+1})]$$

$$= \sum_{j=0}^{u-1} f(j)p_j + f(u)q_u, \quad \text{where } q_u = \sum_{j=u}^{\infty} p_j$$



$$F(u) = E[f(s_t + a_t - s_{t+1})]$$

$$= \sum_{j=0}^{u-1} f(j)p_j + f(u)q_u, \quad \text{where } q_u = \sum_{j=u}^{\infty} p_j$$



$p_j(j|s, a) = ?$ Can we create an transition probability table as well?

$$\begin{aligned} E\{r_t(s_t, a_t, s_{t+1})\} &= E\{-O(a_t) - h(s_t + a_t) + f(s_t + a_t - s_{t+1})\} \\ &= -O(a_t) - h(s_t + a_t) + E[f(s_t + a_t - s_{t+1})] \end{aligned}$$

		$r_t(s, a)$		
	$a=0$	1	2	3
$s_t=0$				
1				
2				
3				

	s_{t+1}	0	1	2	3
$s_t + a_t$					
0					
1					
2					
3					

$p_j(j|s, a) = ?$ Can we create an transition probability table as well?

$$\begin{aligned} E\{r_t(s_t, a_t, s_{t+1})\} &= E\{-O(a_t) - h(s_t + a_t) + f(s_t + a_t - s_{t+1})\} \\ &= -O(a_t) - h(s_t + a_t) + E[f(s_t + a_t - s_{t+1})] \end{aligned}$$

		$r_t(s, a)$		
	$a=0$	1	2	3
$s_t=0$	0	-1	-2	-5
1	5	0	-3	×
2	6	-1	×	×
3	5	×	×	×

	s_{t+1}	$p_j(j s, a)$			
	$s_t + a_t$	0	1	2	3
$s_t + a_t$					
0		1	0	0	0
1		$\frac{3}{4}$	$\frac{1}{4}$	0	0
2		$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$	0
3		0	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$

The Stochastic Inventory Model

Because the model parameters are stationary, we drop the index on the transition probability and the reward.

$$r_t(s_t, a_t) \rightarrow r(s, a) \quad p_t(j|s_t, a_t) \rightarrow p(j|s, a)$$

$r(s, a)$: the current reward/cost from choosing action a in state s

The Stochastic Inventory Model

Define $v_t(s, a)$ by $v_t(s, a) = r(s, a) + \sum_{j \in S} p(j|s, a)v_{t+1}^*(j)$

$$v_t^*(s) = \max_{a \in A_s} \{v_t(s, a)\}$$

or

$$v_t^*(s) = \max_{a \in A_s} \{r(s, a) + \sum_{j \in S} p(j|s, a)v_{t+1}^*(j)\}$$

$\sum_{j \in S} p(j|s, a)v_{t+1}^*(j)$: the expected future reward from choosing action a in state s and change to state j next stage

The Stochastic Inventory Model

The algorithm finds: $v_t^*(s) = \max_{a \in A_s} \{r(s, a) + \sum_{j \in S} p(j|s, a)v_{t+1}^*(j)\}$

Pseudo code of algorithm

1. Set $v_{N+1}^*(s) = 0$, $s = 0, 1, 2, \dots, S$
2. Do for $t = N, \dots, 1$
 - Do for $s = 0, 1, \dots, S$
 - Do for $a = 0, 1, \dots, A$

$$v_t^*(s, a) = \max \{v_t^*(s, a), r(s, a) + \sum_{j \in S} p(j|s, a)v_{t+1}^*(j)\}$$

We implement the backward induction algorithm as follows.

1. Set $t=4$ and $v_4^*(s) = r_4(s) = 0$, $s = 0, 1, 2, 3$
2. Since $t \neq 1$, continue. Set $t = 3$ and

$$\begin{aligned} v_3^*(s) &= \max_{a \in A_s} \{r(s, a) + \sum_{j \in S} p(j|s, a)v_4^*(j)\}, \quad s = 0, 1, 2, 3 \\ &= \max_{a \in A_s} \{r(s, a)\} \end{aligned}$$

The quantities $v_3^*(s, a)$, $v_3^*(s)$ and $A_{s,3}^*$ are summarized in the following table with "x" denoting infeasible actions.

s	$v_3^*(s, a)$				$v_3^*(s)$	$A_{s,3}^*$
	a=0	a=1	a=2	a=3		
0	0	-1	-2	-5	0	0
1	5	0	-3	x	5	0
2	6	-1	x	x	6	0
3	5	x	x	x	5	0

3. Since $t \neq 1$, continue. Set $t = 2$ and

$v_2^*(s) = \max_{a \in A_s} \{v_2^*(s, a)\}$, where, for example,

$$\begin{aligned} v_2^*(0, 2) &= r(0, 2) + p(0|0, 2)v_3^*(0) + p(1|0, 2)v_3^*(1) + p(2|0, 2)v_3^*(2) \\ &\quad + p(3|0, 2)v_3^*(3) \\ &= -2 + \left(\frac{1}{4}\right) \times 0 + \left(\frac{1}{2}\right) \times 5 + \left(\frac{1}{4}\right) \times 6 + 0 \times 5 \\ &= 2 \end{aligned}$$

The quantities $v_2^*(s, a)$, $v_2^*(s)$ and $A_{s,2}^*$ are summarized in the following table.

s	$v_2^*(s, a)$				$v_2^*(s)$	$A_{s,2}^*$
	a=0	a=1	a=2	a=3		
0	0	$\frac{1}{4}$	2	$\frac{1}{2}$	2	2
1	$\frac{25}{4}$	4	$\frac{5}{2}$	\times	$\frac{25}{4}$	0
2	10	$\frac{9}{2}$	\times	\times	10	0
3	$\frac{21}{2}$	\times	\times	\times	$\frac{21}{2}$	0

4. Since $t \neq 1$, continue. Set $t=1$ and

$v_1^*(s) = \max_{a \in A_s} \{v_1^*(s, a)\}$

The quantities $v_1^*(s, a)$, $v_1^*(s)$, and $A_{s,1}^*$ are summarized in the following table.

s	$v_1^*(s, a)$				$v_1^*(s)$	$A_{s,1}^*$
	a=0	a=1	a=2	a=3		
0	0	$\frac{33}{16}$	$\frac{66}{16}$	$\frac{67}{16}$	$\frac{67}{16}$	3
1	$\frac{129}{16}$	$\frac{98}{16}$	$\frac{99}{16}$	\times	$\frac{129}{16}$	0
2	$\frac{194}{16}$	$\frac{131}{16}$	\times	\times	$\frac{194}{16}$	0
3	$\frac{227}{16}$	\times	\times	\times	$\frac{227}{16}$	0

5. Since $t = 1$, stop.

This algorithm yields the optimal expected total reward function $v_4^*(s)$ and the optimal policy $\pi^* = (d_1^*(S), d_2^*(s), d_3^*(s))$, which is tabulated below. Note in this example that the optimal policy is unique.

s	$d_i^*(S)$			$v_4^*(s)$
	$d_1^*(S)$	$d_2^*(S)$	$d_3^*(S)$	
0	3	2	0	$\frac{67}{16}$
1	0	0	0	$\frac{129}{16}$
2	0	0	0	$\frac{194}{16}$
3	0	0	0	$\frac{227}{16}$

The quantity $v_t^*(s)$ gives the expected total reward obtained using the optimal policy when the inventory at the start of the month t is s_t units. The optimal policy has a particular simple form; if at the start of month 1 the inventory is 0 units, order three units, otherwise do not order; If at the start of month 2 the inventory is two units, order two units, otherwise do not order. And do not order in month 3 for any inventory level.

$$d_1^*(S_1) = \begin{cases} 0 & s > 0 \\ 3 & s = 0 \end{cases}$$

Policies of this form are optimal in inventory models in which the ordering cost has the form

$$O(u) = \begin{cases} 0 & u = 0 \\ K + cu & u > 0 \end{cases}$$

where $K > 0$ and $c > 0$, the shortage and holding cost $h(u)$ is convex, and backlogging of unfilled orders is permitted. A proof that this type of policies are optimal under these assumption may be based on backward induction.