Lecture 11

Lecture 11

Previously in Opt 2 …

Example: Inventory planning problem

Period (k)	d_k	c_k	F_k	h_k
1	10	3	5	0.2
2	40	2	20	0.3
3	20	4	10	0.5
4	50	3	10	0.8
5	0			

Example: Inventory planning problem DP Formulation

1. Specify stages

stage k $\leftrightarrow$ period k

2. Specify states at each stage k

State $I \leftrightarrow$ inventory level I

(that is, I is the quantity in the inventory)

3. Specify allowable decisions at each state I and stage k

values x_k such that demand d_k can be satisfied

i.e., x_k such that $I + x_k \geq d_k$

Example: Inventory planning problem DP Formulation

1. Specify stages

   Stages: $k = 1, 2, 3, 4, 5$

2. Specify states at each stage k

   $S_k = \{0, 10, 20, \dots , 120\}$

   (S_k is the set of possible inventory levels at stage k)

3. Specify allowable decisions at each state I and stage k

   $Q_{I,k} = \{x \text{ in } \{0, 10, \dots, 120\} \mid I + x \geq d_k\}$

Example: Inventory planning problem DP Formulation

4. Word-description of optimal function to be solved at state I in stage k

   $f_k^*(I)$ = the minimum cost for satisfying demands in periods k, k+1, ..., T, T+1

5. Boundary conditions

   $$f_{T+1}^*(I) = 0 \text{ for all } I$$

6. Recurrence relation

$$f_k^*(I) = \min_{x_k \text{ in } Q_{k,I}} \left\{ c_k x_k + f_k 1_{\{x_k > 0\}} + h_k (I + x_k - d_k) + f_{k+1}^*(I + x_k - d_k) \right\}$$

Example: Inventory planning problem DP Formulation

4. Word-description of optimal function to be solved at state I in stage k

   $f_k^*(I)$ = the minimum cost for satisfying demands in periods k, k+1, …, T, T+1

5. Boundary conditions

   $$f_{T+1}^*(I) = 0 \text{ for all } I$$

6. Recurrence relation

$$\underbrace{\mathbf{f_k^*(I)}}_{\text{Min cost in periods k to T+1 given that inventory at start is } I} = \min_{x_k \text{ in } Q_{k,I}} \{c_k x_k + f_k 1_{\{x_k>0\}} + h_k(I + x_k - d_k) + f_{k+1}^*(I + x_k - d_k)\}$$

Example: Inventory planning problem DP Formulation

4. Word-description of optimal function to be solved at state I in stage k

   $f_k^*(I)$ = the minimum cost for satisfying demands in periods k, k+1, …, T, T+1

5. Boundary conditions

   $f_{T+1}^*(I) = 0$ for all I

6. Recurrence relation

$$f_k^*(I) = \min_{x_k \text{ in } Q_{k,I}} \left\{ \underbrace{\mathbf{c_k x_k + f_k 1_{\{x_k>0\}} + h_k (I + x_k - d_k)}}_{\text{Cost incurred in period k}} + f_{k+1}^*(I + x_k - d_k) \right\}$$

Example: Inventory planning problem DP Formulation

4. Word-description of optimal function to be solved at state I in stage k

   $f_k^*(I)$ = the minimum cost for satisfying demands in periods k, k+1, …, T, T+1

5. Boundary conditions

   $f_{T+1}^*(I) = 0$ for all I

6. Recurrence relation

$$f_k^*(I) = \min_{x_k \text{ in } Q_{k,I}} \left\{ c_k x_k + f_k 1_{\{x_k>0\}} + h_k (I + x_k - d_k) + \underbrace{\mathbf{f_{k+1}^*(I + x_k - d_k)}}_{\text{Min cost in periods k+1 to T+1}} \right\}$$

Example: Inventory planning problem DP Formulation

4. Word-description of optimal function to be solved at state I in stage k

$$f_k^*(I) = \text{the minimum cost for satisfying demands in periods } k, k+1, \ldots, T, T+1$$

5. Boundary conditions

$$f_{T+1}^*(I) = 0 \text{ for all } I$$

6. Recurrence relation

$$f_k^*(\underbrace{I}_{\text{Old inventory level}}) = \min_{x_k \text{ in } Q_{k,I}} \left\{ c_k x_k + f_k 1_{\{x_k>0\}} + h_k(I + x_k - d_k) + f_{k+1}^*(\underbrace{I + x_k - d_k}_{\text{New inventory level}}) \right\}$$

More generally, recursive relations look like:

$$f_k^*(\textbf{\textit{current state}}) = \min_{x_k \text{ in } Q_{k,I}} \left\{ \text{value/cost due to decision } x_k \, \mathbf{+} \, f_{k+1}^*(\textbf{\textit{new state due to decision }} \boldsymbol{x_k}) \right\}$$

More generally, recursive relations look like:

(Or another operation)

$$f_k^*(\textbf{\textit{current state}}) = \min_{x_k \text{ in } Q_{k,I}} \left\{ \text{value/cost due to decision } x_k + f_{k+1}^*(\textbf{\textit{new state due to decision }} \boldsymbol{x_k}) \right\}$$

Example: Inventory planning problem DP Formulation

7. Computation

Summary of DP Computation:

Possible States (i)	Stage 5	Stage 4		Stage 3		Stage 2		Stage 1	
	f*_5(i)	f*_4(i)	x*_4	f*_3(i)	x*_3	f*_2(i)	x*_2	f*_1(i)	x*_1
0	0					286	110		
10	0					266	100		
20	0					246	90		
30	0					226	80		
40	0								
50	0								
60	0								
70	0								
80	0								
90	0					100	0		
100	0					78	0		
110	0					46	0		

Example: Inventory planning problem DP Formulation

8. Trace back to find optimal solution

Summary of DP Computation:

	Stage 5	Stage 4		Stage 3		Stage 2		Stage 1	
Possible States (i)	f*_5(i)	f*_4(i)	x*_4	f*_3(i)	x*_3	f*_2(i)	x*_2	f*_1(i)	x*_1
0	0					286	110		
10	0					266	100		
20	0					246	90		
30	0					226	80		
40	0								
50	0								
60	0								
70	0								
80	0								
90	0					100	0		
100	0					78	0		
110	0					46	0		

Example: Inventory planning problem DP Formulation

8. Trace back to find optimal solution

Possible States (i)	Stage 5	Stage 4		Stage 3		Stage 2		Stage 1	
	f*_5(i)	f*_4(i)	x*_4	f*_3(i)	x*_3	f*_2(i)	x*_2	f*_1(i)	x*_1
0	0	160	50	250	20	286	110	321	10
10	0	130	40	210	10	266	100		
20	0	100	30	160	0	246	90		
30	0	70	20	135	0	226	80		
40	0	40	10	110	0	206	70		
50	0	0	0	85	0	186	60		
60	0			60	0	166	0 or 50		
70	0			25	0	144	0		
80	0					122	0		
90	0					100	0		
100	0					78	0		
110	0					46	0		

i>clicker question

Q: What is the optimal quantity to produce at each time period?

Possible States (i)	Stage 5	Stage 4		Stage 3		Stage 2		Stage 1	
	f*_5(i)	f*_4(i)	x*_4	f*_3(i)	x*_3	f*_2(i)	x*_2	f*_1(i)	x*_1
0	0	160	50	250	20	286	110	321	10
10	0	130	40	210	10	266	100		
20	0	100	30	160	0	246	90		
30	0	70	20	135	0	226	80		
40	0	40	10	110	0	206	70		
50	0	0	0	85	0	186	60		
60	0			60	0	166	0 or 50		
70	0			25	0	144	0		
80	0					122	0		
90	0					100	0		
100	0					78	0		
110	0					46	0		

A. $x_1 = 10, x_2 = 110, x_3 = 20, x_4 = 50$

B. $x_1 = 10, x_2 = 0, x_3 = 0, x_4 = 0$

C. $x_1 = 10, x_2 = 110, x_3 = 0, x_4 = 0$

D. $x_1 = 10, x_2 = 40, x_3 = 20, x_4 = 50$

E. $x_1 = 10, x_2 = 0, x_3 = 110, x_4 = 0$

Q: What is the optimal quantity to produce at each time period?

Summary of DP Computation:

Possible States (i)	Stage 5	Stage 4		Stage 3		Stage 2		Stage 1	
	f*_5(i)	f*_4(i)	x*_4	f*_3(i)	x*_3	f*_2(i)	x*_2	f*_1(i)	x*_1
0	0	160	50	250	20	286	110	321	10
10	0	130	40	210	10	266	100		
20	0	100	30	160	0	246	90		
30	0	70	20	135	0	226	80		
40	0	40	10	110	0	206	70		
50	0	0	0	85	0	186	60		
60	0			60	0	166	0 or 50		
70	0			25	0	144	0		
80	0					122	0		
90	0					100	0		
100	0					78	0		
110	0					46	0		

A. $x_1 = 10, x_2 = 110, x_3 = 20, x_4 = 50$

B. $x_1 = 10, x_2 = 0, x_3 = 0, x_4 = 0$

C. $x_1 = 10, x_2 = 110, x_3 = 0, x_4 = 0$

D. $x_1 = 10, x_2 = 40, x_3 = 20, x_4 = 50$

E. $x_1 = 10, x_2 = 0, x_3 = 110, x_4 = 0$

Example: Inventory planning problem DP Formulation

8. Trace back to find optimal solution

Summary of DP Computation:

Possible States (i)	Stage 5	Stage 4		Stage 3		Stage 2		Stage 1	
	f*_5(i)	f*_4(i)	x*_4	f*_3(i)	x*_3	f*_2(i)	x*_2	f*_1(i)	x*_1
0	0	160	50	250	20	286	110	321	10
10	0	130	40	210	10	266	100		
20	0	100	30	160	0	246	90		
30	0	70	20	135	0	226	80		
40	0	40	10	110	0	206	70		
50	0	0	0	85	0	186	60		
60	0			60	0	166	0 or 50		
70	0			25	0	144	0		
80	0					122	0		
90	0					100	0		
100	0					78	0		
110	0					46	0		

DP Computation using AMPL

As usual, need:

1. Model file
2. Data file
3. Script file
   – Optional, but this can save you time from having to re-enter the following commands many times while debugging/re-running:

```
model myDP.mod;
data myDP.dat;
solve;
(etc.)
```

DP Computation using AMPL

1. **Model file**
   - We are not going to model a linear program here, so we won't have:

     ```
     var
     maximize …
     subject to …
     ```

   - Will use only sets and parameters:
     - The values of some sets/parameters are supplied by the data file
     - The values of some other sets/parameters are computed within the model file; not supplied by the data file

DP Computation using AMPL

2. **Data file**
   - Same as before!
   - Here, you specify values of parameters that are declared in the model file.

DP Computation using AMPL

2. **Data file**
   – Same as before!
   – Here, you specify values of parameters that are declared in the model file.
3. **Script file**
   – Same as before!
   – If you haven't been using scripts, you can start now!

DP Computation using AMPL

1. inventory.mod
2. inventory.dat
3. inventoryScript.txt

inventory.mod

```
param T;                          # number of periods
param c {k in 1..T};              # cost per unit in period k
param F {k in 1..T};              # fixed cost in period k
param h {k in 1..T};              # holding cost for inventory from k-1 to k
param d {k in 1..T};              # demand in period k

param MaxQuantity := sum{k in 1..T} d[k];

set allowableDecisions {k in 1..T, I in 0..MaxQuantity} :=
 {x in 0..MaxQuantity: d[k] <= I + x <= MaxQuantity};
 # quantity to produce at stage k, state I (inventory level)

param f {k in 1..T+1, I in 0..MaxQuantity} :=  # min cost in periods k through T+1
 if k=T+1 then 0 # no more demand to consider
 else
   min{x in allowableDecisions[k, I]}
(c[k]*x + F[k]*(if x <>0 then 1 else 0) + h[k]*(I + x - d[k]) + f[k+1, I + x - d[k]]);

set opt {k in 1..T, I in 0..MaxQuantity} :=   # optimal decisions
 {x in allowableDecisions[k, I]: f[k, I] = c[k]*x + F[k]*(if x <> 0 then 1 else 0) +
    h[k]*(I + x - d[k]) + f[k+1, I + x - d[k]]};

param DPvalue := f[1, 0];  # compute the optimal value
```

inventory.mod

```
param T;                    # number of periods
param c {k in 1..T};        # cost per unit in period k
param F {k in 1..T};        # fixed cost in period k
param h {k in 1..T};        # holding cost for inventory from k-1 to k
param d {k in 1..T};        # demand in period k

param MaxQuantity := sum{k in 1..T} d[k];

set allowableDecisions {k in 1..T, I in 0..MaxQuantity} :=
 {x in 0..MaxQuantity: d[k] <= I + x <= MaxQuantity};
 # quantity to produce at stage k, state I (inventory level)

param f {k in 1..T+1, I in 0..MaxQuantity} :=  # min cost in periods k through T+1
 if k=T+1 then 0 # no more demand to consider
 else
   min{x in allowableDecisions[k, I]} (
c[k]*x + F[k]*(if x <>0 then 1 else 0) + h[k]*(I + x - d[k]) + f[k+1, I + x - d[k]]);

set opt {k in 1..T, I in 0..MaxQuantity} :=   # optimal decisions
 {x in allowableDecisions[k, I]: f[k, I] = c[k]*x + F[k]*(if x <> 0 then 1 else 0) +
    h[k]*(I + x - d[k]) + f[k+1, I + x - d[k]]};

param DPvalue := f[1, 0];  # compute the optimal value
```

Values are provided by inventory.dat

inventory.mod

```
param T;                          # number of periods
param c {k in 1..T};              # cost per unit in period k
param F {k in 1..T};              # fixed cost in period k
param h {k in 1..T};              # holding cost for inventory from k-1 to k
param d {k in 1..T};              # demand in period k

param MaxQuantity := sum{k in 1..T} d[k];
```

Max quantity considered

```
set allowableDecisions {k in 1..T, I in 0..MaxQuantity} :=
 {x in 0..MaxQuantity: d[k] <= I + x <= MaxQuantity};
 # quantity to produce at stage k, state I (inventory level)

param f {k in 1..T+1, I in 0..MaxQuantity} :=  # min cost in periods k through T+1
 if k=T+1 then 0 # no more demand to consider
 else
   min{x in allowableDecisions[k, I]}
(c[k]*x + F[k]*(if x <>0 then 1 else 0) + h[k]*(I + x - d[k]) + f[k+1, I + x - d[k]]);

set opt {k in 1..T, I in 0..MaxQuantity} :=   # optimal decisions
 {x in allowableDecisions[k, I]: f[k, I] = c[k]*x + F[k]*(if x <> 0 then 1 else 0) +
    h[k]*(I + x - d[k]) + f[k+1, I + x - d[k]]};

param DPvalue := f[1, 0];  # compute the optimal value
```

inventory.mod

```
param T;                          # number of periods
param c {k in 1..T};              # cost per unit in period k
param F {k in 1..T};              # fixed cost in period k
param h {k in 1..T};              # holding cost for inventory from k-1 to k
param d {k in 1..T};              # demand in period k

param MaxQuantity := sum{k in 1..T} d[k];

set allowableDecisions {k in 1..T, I in 0..MaxQuantity} :=
 {x in 0..MaxQuantity: d[k] <= I + x <= MaxQuantity};
 # quantity to produce at stage k, state I (inventory level)

param f {k in 1..T+1, I in 0..MaxQuantity} :=  # min cost in periods k through T+1
 if k=T+1 then 0 # no more demand to consider
 else
   min{x in allowableDecisions[k, I]}
(c[k]*x + F[k]*(if x <>0 then 1 else 0) + h[k]*(I + x - d[k]) + f[k+1, I + x - d[k]]);

set opt {k in 1..T, I in 0..MaxQuantity} :=   # optimal decisions
 {x in allowableDecisions[k, I]: f[k, I] = c[k]*x + F[k]*(if x <> 0 then 1 else 0) +
    h[k]*(I + x - d[k]) + f[k+1, I + x - d[k]]};

param DPvalue := f[1, 0];  # compute the optimal value
```

Set of allowable decisions at each stage k, state *I*

inventory.mod

```
param T;                        # number of periods
param c {k in 1..T};            # cost per unit in period k
param F {k in 1..T};            # fixed cost in period k
param h {k in 1..T};            # holding cost for inventory from k-1 to k
param d {k in 1..T};            # demand in period k

param MaxQuantity := sum{k in 1..T} d[k];

set allowableDecisions {k in 1..T, I in 0..MaxQuantity} :=
 {x in 0..MaxQuantity: d[k] <= I + x <= MaxQuantity};
 # quantity to produce at stage k, state I (inventory level)

param f {k in 1..T+1, I in 0..MaxQuantity} :=  # min cost in periods k through T+1
 if k=T+1 then 0 # no more demand to consider
 else
   min{x in allowableDecisions[k, I]}
(c[k]*x + F[k]*(if x <>0 then 1 else 0) + h[k]*(I + x - d[k]) + f[k+1, I + x - d[k]]);

set opt {k in 1..T, I in 0..MaxQuantity} :=   # optimal decisions
 {x in allowableDecisions[k, I]: f[k, I] = c[k]*x + F[k]*(if x <> 0 then 1 else 0) +
    h[k]*(I + x - d[k]) + f[k+1, I + x - d[k]]};

param DPvalue := f[1, 0];  # compute the optimal value
```

Computation of $f_k^*(I)$ at each stage k, each state I

inventory.mod

```
param T;                          # number of periods
param c {k in 1..T};              # cost per unit in period k
param F {k in 1..T};              # fixed cost in period k
param h {k in 1..T};              # holding cost for inventory from k-1 to k
param d {k in 1..T};              # demand in period k

param MaxQuantity := sum{k in 1..T} d[k];

set allowableDecisions {k in 1..T, I in 0..MaxQuantity} :=
 {x in 0..MaxQuantity: d[k] <= I + x <= MaxQuantity};
 # quantity to produce at stage k, state I (inventory level)

param f {k in 1..T+1, I in 0..MaxQuantity} :=  # min cost in periods k through T+1
 if k=T+1 then 0 # no more demand to consider
 else
   min{x in allowableDecisions[k, I]}
(c[k]*x + F[k]*(if x <>0 then 1 else 0) + h[k]*(I + x - d[k]) + f[k+1, I + x - d[k]]);

set opt {k in 1..T, I in 0..MaxQuantity} :=   # optimal decisions
 {x in allowableDecisions[k, I]: f[k, I] = c[k]*x + F[k]*(if x <> 0 then 1 else 0) +
    h[k]*(I + x - d[k]) + f[k+1, I + x - d[k]]};

param DPvalue := f[1, 0];  # compute the optimal value
```

Keeping track of optimal decision, x_k*, at each stage k, each state I

inventory.mod

```
param T;                        # number of periods
param c {k in 1..T};            # cost per unit in period k
param F {k in 1..T};            # fixed cost in period k
param h {k in 1..T};            # holding cost for inventory from k-1 to k
param d {k in 1..T};            # demand in period k

param MaxQuantity := sum{k in 1..T} d[k];

set allowableDecisions {k in 1..T, I in 0..MaxQuantity} :=
 {x in 0..MaxQuantity: d[k] <= I + x <= MaxQuantity};
 # quantity to produce at stage k, state I (inventory level)

param f {k in 1..T+1, I in 0..MaxQuantity} :=  # min cost in periods k through T+1
 if k=T+1 then 0 # no more demand to consider
 else
   min{x in allowableDecisions[k, I]}
(c[k]*x + F[k]*(if x <>0 then 1 else 0) + h[k]*(I + x - d[k]) + f[k+1, I + x - d[k]]);

set opt {k in 1..T, I in 0..MaxQuantity} :=   # optimal decisions
 {x in allowableDecisions[k, I]: f[k, I] = c[k]*x + F[k]*(if x <> 0 then 1 else 0) +
    h[k]*(I + x - d[k]) + f[k+1, I + x - d[k]]};

param DPvalue := f[1, 0];  # compute the optimal value
```

Determining the optimal cost: $f_1^*(0)$

inventory.dat

```
param T := 4; # number of periods

param: d c F h :=
1    10  3  5 0.2
2    40  2 20 0.3
3    20  4 10 0.5
4    50  3 10 0.8;
```

inventoryScript.txt

```
reset;
model Opt2Models/Lec11/inventory.mod;
data Opt2Models/Lec11/inventory.dat;
display DPvalue;
```

In the “sw” console:

```
sw: ampl
ampl: include Opt2Models/Lec11/inventoryScript.txt;
DPvalue = 321

ampl:
```