Production Models: Maximizing Profits

- Robert Fourer, David M. Gay, and Brian Kernighan

-Presenting by Afzal Hossain

Production Problem

- Constraints on resources
- Make products within constraints
- Maximize profit from production
- Maximize C'x

$Ax <= b$

$x >= 0$

Toy Problem:

Tons per hour:	Bands	200	Total production
	Coils	140	hour: 40
Profit per ton:	Bands	$25	
	Coils	$30	
Maximum tons:	Bands	6000	
	Coils	4000	

XB=? XC=?

Maximize $25 * XB + 30 * XC$

Subject to $(1/200) * XB + (1/140) * XC <= 40$

$0 <= XB <= 6000$

$0 <= XC <= 4000$

Linear program in AMPL

Mathematical Notation:

Maximize	25* XB + 30* XC
Subject to	(1/200)* XB + (1/140)* XC <=40
	0<=XB<=6000
	0<=XC<=4000

What if we had 40 products rather than 2, more than 100 values will be added in the script

AMPL Notation:

```
var XB;
var XC;
maximize Profit: 25*XB + 30*XC;
subject to Time: (1/200)*XB + (1/140)*XC <=40;
subject to B_Limit: 0<=XB<=6000;
subject to C_Limit: 0<=XC<=4000;

ampl: model prod0.mod;

ampl: solve;
MINOS 5.5: optimal solution found.
2 iterations, objective 192000

ampl: display XB, XC;
XB=6000
XC=1400
ampl: quit;
```

Linear programming model

- **Sets**, like the products
- **Parameters**, like the production and profit rates
- **Variables**, whose value is to determine
- An **objective**, to be maximized or minimized
- **Constraints**, that the solution must satisfy

Given:
P, a set of products
a_j = tons per hour of product j, for each $j \in P$
b = hours available at the mill
c_j = profit per ton of product j, for each $j \in P$
u_j = maximum tons of product j, for each $j \in P$

Define variables: X_j = tons of product j to be made, for each $j \in P$

Maximize: $\sum_{j \in P} c_j X_j$

Subject to: $\sum_{j \in P} (1/a_j) X_j \leq b$

$0 \leq X_j \leq u_j$, for each $j \in P$

Figure 1-1: Basic production model in algebraic form.

steel.mod

```
set P;
param a { j in P};
param b;
param c { j in P};
param  u { j in P};

var X { j in P};
maximize Total_Profit: sum { j in P} c[j] * X[j];
subject to Time: sum { j in P} (1/a[j] * X[j]) <= b;
subject to Limit { j in P}: 0<= X[j] <=u[j];
```

steel.dat

```
set p:= bands coils;
param:      a     c     u:=
bands       200   25    6000
coils       140   30    4000;
param b:=40;
```

```
ampl: model steel.mod;
ampl: data steel.dat;
ampl: solve;
```

An improved model

steel.mod

```
set P;
param a { j in P};
param b;
param c { j in P};
param  u { j in P};

var X { j in P};
maximize Total_Profit: sum { j in P} c[j] * X[j];
subject to Time: sum { j in P} (1/a[j] * X[j]) <= b;
subject to Limit { j in P}: 0<= X[j] <=u[j];
```

steel.dat

```
set p:= bands coils;
param:        a       c       u:=
bands         200     25      6000
coils         140     30      4000;
param b:=40;
```

steel.mod

```
set PROD;
param rate {PROD}>0;
param avail >= 0;
param profit {PROD}>=0;
param  market {PROD}>=0;

var make {p in POD}>=0, <=market[p];
maximize Total_Profit: sum {p in PROD} profit[p] * make[p];
subject to Time: sum {p in PROD} (1/rate[p] * make[p]) <= avail;
```

steel.dat

```
set PROD:= bands coils;
param:  rate      profit    market:=
bands           200       25        6000
coils           140       30        4000;
param avail:=40;
```

steel.dat

```
set PROD:= bands coils plate;
param:  rate      profit    market:=
bands           200       25        6000
coils           140       30        4000
plate           160       29        3500;
param avail:=40;
```

Add new products without changing model

Add lower limits

steel.mod

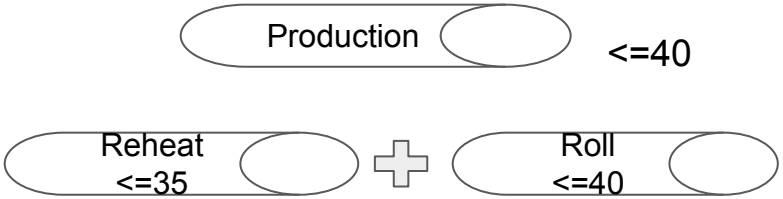
```
set PROD;
param rate {PROD}>0;
param avail >= 0;
param profit {PROD}>=0;
param  market {PROD}>=0;
param commit {PROD}>=0

var make {p in PROD}>=commit[p], <=market[p];
maximize Total_Profit: sum {p in PROD} profit[p] * make[p];
subject to Time: sum {p in PROD} (1/rate[p] * make[p]) <= avail;
```

steel.dat

```
set PROD:= bands coils plate;
param:      rate    profit   commit      market:=
bands       200     25       1000        6000
coils       140     30        500        4000
plate       160     29        750        3500;
param avail:=40;
```

Adding resource constraint




```
set PROD;   # products
set STAGE;  # stages

param rate {PROD,STAGE} > 0; # tons per hour in each stage
param avail {STAGE} >= 0;    # hours available/week in each stage
param profit {PROD};         # profit per ton

param commit {PROD} >= 0;    # lower limit on tons sold in week
param market {PROD} >= 0;    # upper limit on tons sold in week

var Make {p in PROD} >= commit[p], <= market[p]; # tons produced

maximize Total_Profit: sum {p in PROD} profit[p] * Make[p];

              # Objective: total profits from all products

subject to Time {s in STAGE}:
   sum {p in PROD} (1/rate[p,s]) * Make[p] <= avail[s];

              # In each stage: total of hours used by all
              # products may not exceed hours available
```

Figure 1-6a: Additional resource constraints (`steel4.mod`).

```
set PROD := bands coils plate;
set STAGE := reheat roll;

param rate:   reheat   roll :=
  bands         200    200
  coils         200    140
  plate         200    160 ;

param:     profit  commit  market :=
  bands      25     1000    6000
  coils      30      500    4000
  plate      29      750    3500 ;

param avail :=  reheat 35    roll    40 ;
```