Example Problems Solved By AMPL

Feng-Tien Yu

June 14, 2000

Contents

1 Using AMPL at CAEN

1.1 Setting Up AMPL Environment at CAEN

Before you can start using AMPL, you need to create a symbolic link to AMPL interactive solver by typing the following command line on any Sun workstation:

```
ln -s /afs/engin.umich.edu/group/engin/priv/ioe/
ampl/ampl_interactive ampl
```

Now you can run AMPL by simply typing *ampl* on any Sun work station. Note that AMPL only serves as an interface between your programs and various solvers. The default solver is CPLEX.

In most cases, you won't need other solvers. However if for some reasons you would like to use other solvers, you must create a symbolic link for the desired solver before you can use it. Currently three additional solvers, OSL, MINOS and ALPO, are available. These command lines would create symbolic links to various solvers.

```
ln -s /afs/engin.umich.edu/group/engin/priv/
ioe/ampl/solvers_SunOS/osl osl

ln -s /afs/engin.umich.edu/group/engin/priv/
ioe/ampl/solvers_SunOS/minos minos

ln -s /afs/engin.umich.edu/group/engin/priv/
ioe/ampl/solvers_SunOS/alpo alpo
```

1.2 Starting and Quitting AMPL

To invoke AMPL, type *ampl* in the directory where you created the symbolic link. Then type in AMPL statements in response to the `ampl:` prompt, until you leave AMPL by typing *quit*. Use *reset* to erase the previous model and read in another model.

2 A Product Mixture Problem

The nutritionist at a food research lab is trying to develop a new type of multi-grain flour. The grains that can be included have the following composition and price.

	% of Nutrient in Grain			
	1	2	3	4
Starch	30	20	40	25
Fiber	40	65	35	40
Protein	20	15	5	30
Gluten	10	0	20	5
Cost (cents/kg.)	70	40	60	80

Because of taste considerations, the percent of grain 2 in the mix cannot exceed 20, the percent of grain 3 in the mix has to be at least 30, and the percent of grain 1 in the mix has to be between 10 to 25.

The percent protein content in the flour must be at least 18, the percent gluten content has to be between 8 to 13, and the percent fiber content at most 50.

Find the least costly way of blending the grains to make the flour, subject to the constraints given.

2.1 LP Formulation

The decision variables are:

$$x_j = \text{Percent of grain } j \text{ in the flour, } j = 1 \text{ to } 4$$

The model is:

$$\begin{array}{ll} \text{Minimize cost} = & 70x_1 + 40x_2 + 60x_3 + 80x_4 \\ \\ \text{s. to} & x_1 + x_2 + x_3 + x_4 = 100 \\ & x_2 \leq 20, \quad x_3 \geq 30 \\ & 10 \leq x_1 \leq 25 \\ & 0.20x_1 + 0.15x_2 + 0.05x_3 + 0.30x_4 \geq 18 \\ & 8 \leq 0.10x_1 + 0.20x_3 + 0.05x_4 \leq 13 \\ & 0.40x_1 + 0.65x_2 + 0.35x_3 + 0.40x_4 \leq 50 \\ & x_j \geq 0 \text{ for all } j = 1 \text{ to } 4 \end{array}$$

2.2 AMPL Model File: grain.mod

```
set GRAIN;     # set of grains
set NUTRIENT;  # set of nutrients

param n_percent{GRAIN,NUTRIENT}>=0;
# percentage of nutrients in grains
param cost{GRAIN}>=0;
# cost of grains
param u_grain{GRAIN}>=0;
```

```
# upper bound for the percentage
# of each grain in flour
param l_grain{GRAIN}>=0;
# lower bound for the percentage
# of each grain in flour
param u_nutrient{NUTRIENT} >=0;
# upper bound for the percentage
# of each nutrient in flour
param l_nutrient{NUTRIENT} >=0;
# lower bound for the percentage
# of each nutrient in flour

var G_percent{GRAIN}>=0;
# percentage of each grain in flour

minimize Total_cost:
    sum{i in GRAIN} cost[i]*G_percent[i];
subject to Flour:
    sum{i in GRAIN} G_percent[i] = 100;
# total percentage should be 100%
subject to Grain_u{i in GRAIN}:
    G_percent[i] <= u_grain[i];
# grain percentage <= upper bound
subject to Grain_l{i in GRAIN}:
    G_percent[i] >= l_grain[i];
# grain percentage >= lower bound
subject to Nutri_u{j in NUTRIENT}:
    sum {i in GRAIN} G_percent[i] *
    n_percent[i,j] / 100 <= u_nutrient[j];
# nutrient percentage <= upper bound
subject to Nutri_l{j in NUTRIENT}:
    sum {i in GRAIN} G_percent[i] *
    n_percent[i,j] /100 >= l_nutrient[j];
# nutrient percentage >= lower bound
```

2.3 AMPL Data File: grain.dat

```
set GRAIN := G1 G2 G3 G4 ;
set NUTRIENT :=
Starch Fiber Protein Gluten;
param n_percent :
Starch Fiber Protein Gluten :=
G1 30 40 20 10
G2 20 65 15  0
G3 40 35  5 20
G4 25 40 30  5 ;
```

```
param cost :=
G1 70 G2 40 G3 60 G4 80 ;
param u_grain :=
G1 25 G2 20 G3 100  G4 100 ;
param l_grain :=
G1 10 G2 0  G3 30 G4 0 ;
param u_nutrient :=
Starch  100   Fiber  50
Protein 100   Gluten 13 ;
param l_nutrient :=
Starch  0   Fiber 0
Protein 18  Gluten 8 ;
```

2.4 AMPL Solutions

```
draw% ampl
ampl: model grain.mod;
ampl: data grain.dat;
ampl: solve;
CPLEX 6.0: optimal solution; objective 6450
3 iterations (0 in phase I)
ampl: display G_percent;
G_percent [*] :=
G1  15
G2  20
G3  30
G4  35
;

ampl: display G_percent.dual;
G_percent.dual [*] :=
G1  0
G2  0
G3  0
G4  0
;

ampl: display G_percent.lb, G_percent.ub, G_percent.slack;
:  G_percent.lb G_percent.ub G_percent.slack    :=
G1       10           25             5
G2        0           20             0
G3       30          100             0
G4        0          100            35
;

ampl: display Flour;
```

```
Flour = 50

ampl: display Grain_u, Grain_l;
:  Grain_u Grain_l    :=
G1       0          0
G2     -25          0
G3       0          5
G4       0          0
;

ampl: display Nutri_u, Nutri_l;
:       Nutri_u Nutri_l    :=
Fiber         0             0
Gluten        0             0
Protein       0           100
Starch        0             0
;

ampl: display Grain_u.lb, Grain_u.body, Grain_u.ub;
:   Grain_u.lb Grain_u.body Grain_u.ub    :=
G1   -Infinity       15          25
G2   -Infinity       20          20
G3   -Infinity       30         100
G4   -Infinity       35         100
;

ampl: display Grain_l.lb, Grain_l.body, Grain_l.ub;
:  Grain_l.lb Grain_l.body Grain_l.ub    :=
G1      10          15       Infinity
G2       0          20       Infinity
G3      30          30       Infinity
G4       0          35       Infinity
;

ampl: display Nutri_u.lb, Nutri_u.body, Nutri_u.ub;
:        Nutri_u.lb Nutri_u.body Nutri_u.ub    :=
Fiber     -Infinity     43.5          50
Gluten    -Infinity      9.25         13
Protein   -Infinity     18           100
Starch    -Infinity     29.25        100
;

ampl: display Nutri_l.lb, Nutri_l.body, Nutri_l.ub;
:       Nutri_l.lb Nutri_l.body Nutri_l.ub    :=
Fiber         0        43.5       Infinity
Gluten        8         9.25      Infinity
```

```
Protein      18        18         Infinity
Starch        0        29.25      Infinity
;


ampl: quit;
draw%
```

2.5 Redirect Outputs

To direct the output to a file rather than the screen, add a > and the name of the file. To open a file and append output to whatever is already there (rather than overwriting), use >> instead of >. Once a file is open, subsequent use of > and >> have the same effect.

```
ampl: model grain.mod;
ampl: data grain.dat;
ampl: solve;
CPLEX 6.0: optimal solution; objective 6450
3 iterations (0 in phase I)
ampl: print 'OBJECTIVES:' > ampl.out;
ampl: display Total_cost >> ampl.out;
ampl: print 'VARIABLES:' >> ampl.out;
ampl: display G_percent >> ampl.out;
ampl: print 'CONSTRAINTS (Dual Values):' >> ampl.out;
ampl: show constraints;

constraints:   Flour   Grain_l   Grain_u   Nutri_l   Nutri_u
ampl: display Flour >> ampl.out;
ampl: display Grain_l >> ampl.out;
ampl: display Grain_u >> ampl.out;
ampl: display Nutri_l >> ampl.out;
ampl: display Nutri_u >> ampl.out;
ampl: close ampl.out;
ampl: quit;
```

The following is the output file *ampl.out*.

```
OBJECTIVES:
Total_cost = 6450

VARIABLES:
G_percent [*] :=
G1  15
G2  20
G3  30
```

```
G4  35
;

CONSTRAINTS (Dual Values):
Flour = 50

Grain_l [*] :=
G1  0
G2  0
G3  5
G4  0
;

Grain_u [*] :=
G1    0
G2  -25
G3    0
G4    0
;

Nutri_l [*] :=
  Fiber    0
 Gluten    0
Protein  100
 Starch    0
;

Nutri_u [*] :=
  Fiber  0
 Gluten  0
Protein  0
 Starch  0
;
```

2.6 Batch Operations

After a model has been under development for a while, you may find that you are typing the same series of commands again and again. To speed things up, you may put the commands into a file, and run them all automatically by typing *include* and the filena me. The following is an example batch file called *grain.run*.

```
model grain.mod;
data grain.dat;
solve;
```

```
print 'OBJECTIVES:' > ampl.out;
display Total_cost >> ampl.out;
print 'VARIABLES:' >> ampl.out;
display G_percent >> ampl.out;
print 'CONSTRAINTS (Dual Values):' >> ampl.out;
display Flour >> ampl.out;
display Grain_l >> ampl.out;
display Grain_u >> ampl.out;
display Nutri_l >> ampl.out;
display Nutri_u >> ampl.out;
close ampl.out;
```

Type *include grain.run*; in AMPL to run this batch file. The outputs would be redirected to the file *ampl.out*.

```
putt% ampl
ampl: include grain.run;
CPLEX 6.0: optimal solution; objective 6450
3 iterations (0 in phase I)
ampl: quit;
putt%
```

The following is the output file *ampl.out*:

```
OBJECTIVES:
Total_cost = 6450

VARIABLES:
G_percent [*] :=
G1  15
G2  20
G3  30
G4  35
;

CONSTRAINTS (Dual Values):
Flour = 50

Grain_l [*] :=
G1  0
G2  0
G3  5
G4  0
;
```

```
Grain_u [*] :=
G1    0
G2  -25
G3    0
G4    0
;

Nutri_l [*] :=
  Fiber    0
 Gluten    0
Protein  100
 Starch    0
;

Nutri_u [*] :=
  Fiber  0
 Gluten  0
Protein  0
 Starch  0
;
```

3 A Production Planning Problem

A nonferrous metals company makes four different alloys from two metals. The requirements are given below. Find the optimal product mix that maximizes gross revenue.

	proportion of metal in alloy				Availability
Metal	1	2	3	4	per day
1	0.5	0.6	0.3	0.1	25 tons
2	0.5	0.4	0.7	0.9	40 tons
Alloy price ($/ton)	750	650	1200	2200	

3.1 AMPL Model File: Nonferrous.model

```
set METAL;
set ALLOY;
param avail {METAL} >= 0;
param portion {METAL,ALLOY} >= 0;
```

```
param price {ALLOY} >= 0;
var Prod {ALLOY} >= 0;
maximize revenue: sum {j in ALLOY} Prod[j] * price[j];
subject to metal_avail {i in METAL}:
sum {j in ALLOY} portion[i,j] * Prod[j] <= avail[i];
```

3.2 AMPL Data File: Nonferrous.data

```
set METAL := M1 M2 ;
set ALLOY := A1 A2 A3 A4 ;
param avail := M1 25 M2 40;
param portion : A1 A2 A3 A4 :=
M1 0.5 0.6 0.3 0.1
M2 0.5 0.4 0.7 0.9 ;
param price :=
A1 750 A2 650 A3 1200 A4 2200 ;
```

3.3 AMPL Solutions

We now solve this problrm by using AMPL (draw% is the prompt on the workstation on which this problem is solved).

```
draw% ampl
ampl: model Nonferrous.model;
ampl: data Nonferrous.data;
ampl: solve;
CPLEX 6.0: optimal solution; objective 97777.77778
1 iterations (0 in phase I)

ampl: display Prod;
Prod [*] :=
A1   0
A2   0
A3   0
A4  44.4444
;

ampl: display metal_avail;
metal_avail [*] :=
M1     0
M2  2444.44
;

ampl: quit;
draw%
```

4 A Purchase Planning Problem

An American textiles marketing firm buys shirts from manufacturers and sells them to clothing retailers. For the next season they are considering four styles with the total orders to be filled as given in the following table (the unit K = kilo, i.e., one thousand shirts). They are considering 3 manufacturers, M_1, M_2, M_3 who can make these styles in quantities and prices ($/shirt) as given below.

Style	Orders	M_1		M_2		M_3	
		Capacity	Price	Capacity	Price	Capacity	Price
1	200K	100K	$8	80K	$6.75	120K	$9
2	150K	80K	10	60K	10.25	100K	10.50
3	90K	75K	11	50K	11	75K	10.75
4	70K	60K	13	40K	14	50K	12.75
Capacity for all styles together		275K		150K		220K	

The manufacturing facilities of M_3 are located within US territories, so there are no limits on how many shirts can be ordered from M_3. But manufacturers M_1 and M_2 are both located in the same foreign country, and there is a limit (quota) of 350K shirts that can be imported from both of them put together. Find the cheapest way to meet the demand, ignoring the integer requirements on the number of shirts of any style purchased from any manufacturer.

4.1 AMPL Model File: shirt.model

```
set MANUFACTURER;
set STYLE;
set FOREIGN within MANUFACTURER;
param capacity {STYLE,MANUFACTURER} >= 0;
param capacity_total {MANUFACTURER} >= 0;
param price {STYLE,MANUFACTURER} >= 0;
param order {STYLE} >= 0;
param quota >= 0;
var Buy {STYLE, MANUFACTURER} >= 0;
minimize cost: sum {i in STYLE, j in MANUFACTURER}
               Buy[i,j] * price[i,j];
subject to
max_order {j in MANUFACTURER}:
sum {i in STYLE} Buy[i,j]  <= capacity_total[j];
```

```
subject to
max_import :
sum {i in STYLE, j in FOREIGN} Buy[i,j]  <= quota;
subject to
max_order_style {i in STYLE, j in MANUFACTURER}:
Buy[i,j] <= capacity[i,j];
subject to demand {i in STYLE}:
sum {j in MANUFACTURER} Buy[i,j]  >= order[i];
```

4.2 AMPL Data File: shirt.data

```
set MANUFACTURER := M1 M2 M3;
set STYLE := S1 S2 S3 S4 ;
set FOREIGN := M1 M2;
param capacity : M1 M2 M3 :=
S1 100 80 120
S2 80 60 100
S3 75 50 75
S4 60 40 50 ;
param capacity_total := M1 275 M2 150 M3 220;
param price : M1 M2 M3 :=
S1  8   6.75   9
S2 10  10.25  10.50
S3 11  11     10.75
S4 13  14     12.75 ;
param order := S1 200 S2 150 S3 90 S4 70;
param  quota:= 350;
```

4.3 AMPL Solutions

```
draw% ampl
ampl: model  shirt.model;
ampl: data  shirt.data;
ampl: solve;
CPLEX 6.0: optimal solution; objective 4910
16 iterations (9 in phase I)

ampl: display Buy;
Buy :=
S1 M1   100
S1 M2    80
S1 M3    20
S2 M1    80
```

```
S2 M2    55
S2 M3    15
S3 M1     0
S3 M2    15
S3 M3    75
S4 M1    20
S4 M2     0
S4 M3    50
;

ampl: display max_import;
max_import = -0.25

ampl: display max_order_style;
max_order_style :=
S1 M1   -0.75
S1 M2   -2
S1 M3    0
S2 M1   -0.25
S2 M2    0
S2 M3    0
S3 M1    0
S3 M2    0
S3 M3   -0.5
S4 M1    0
S4 M2    0
S4 M3   -0.5
;

ampl: display demand;
demand [*] :=
S1   9
S2  10.5
S3  11.25
S4  13.25
;

ampl: quit;
draw%
```

5 A Multi-Period LP Problem

Important applications in production planning for planning production, storage, marketing of product over a planning horizon.

Production capacity, demand, selling price, production cost, may all vary from period to period.

AIM: To determine how much to produce, store, sell in each period; to max. net profit over entire planning horizon.

YOU NEED Variables that represent how much material is in storage at end of each period, and a material balance constraint for each period.

Planning Horizon = 6 periods.
Storage warehouse capacity: 3000 tons.
Storage cost: \$2/ton from one period to next.
Initial stock: 500 tons. Desired final stock: 500 tons.
Demand of each period must be fulfilled in that same period.

Period	Prod. cost	Prod. capacity	Demand (tons)	Sell price
1	20 \$/ton	1500	1100 tons	180 \$/ton
2	25	2000	1500	180
3	30	2200	1800	250
4	40	3000	1600	270
5	50	2700	2300	300
6	60	2500	2500	320

5.1 AMPL Model File: multi.mod

```
set PERIOD ordered;

param sales_period;
param production_cost {PERIOD};
param production_capacity {PERIOD};
param demand {PERIOD};
param sale_price {PERIOD};
param storage_capacity;
param holding_cost;
param initial_stock;
param final_stock;

var Production {PERIOD} >= 0;
var Storage {PERIOD} >= 0;

maximize Profit:
sum {i in PERIOD} demand[i] * sale_price[i]
- sum {i in PERIOD} Production[i] * production_cost[i]
- sum {i in PERIOD: ord(i) < sales_period} Storage[i] * holding_cost;

subject to Storage_Capacity{i in PERIOD}:
Storage[i] <= storage_capacity;
```

```
subject to Production_Capacity{i in PERIOD}:
Production[i] <= production_capacity[i];

subject to Balance{i in PERIOD: ord(i)> 1 }:
Production[i] + Storage[prev(i)] - demand[i] - Storage[i] = 0;

subject to Initial_Balance:
Production[first(PERIOD)]+ initial_stock-demand[first(PERIOD)]
- Storage[first(PERIOD)] = 0;

subject to Final_Balance:
Storage[last(PERIOD)] = final_stock;
```

5.2 AMPL Data File: multi.dat

```
set PERIOD :=  P1 P2 P3 P4 P5 P6;

param sales_period:= 6;
param : production_cost production_capacity demand sale_price :=
P1  20 1500 1100 180
P2  25 2000 1500 180
P3  30 2200 1800 250
P4  40 3000 1600 270
P5  50 2700 2300 300
P6  60 2500 2500 320;

param storage_capacity := 3000;

param holding_cost := 2;

param initial_stock := 500;

param final_stock := 500;
```

5.3 AMPL Solution

```
draw% ampl
ampl: model multi.mod;
ampl: data multi.dat;
ampl: solve;
CPLEX 6.0: optimal solution; objective 2446800
```

```
7 iterations (3 in phase I)
ampl: display Production;
Production [*] :=
P1  1500
P2  2000
P3  2200
P4  2800
P5  2300
P6     0
;

ampl: display Storage;
Storage [*] :=
P1   900
P2  1400
P3  1800
P4  3000
P5  3000
P6   500
;

ampl: display Storage_Capacity;
Storage_Capacity [*] :=
P1  0
P2  0
P3  0
P4  8
P5  8
P6  0
;

ampl: display Production_Capacity;
Production_Capacity [*] :=
P1  14
P2  11
P3   8
P4   0
P5   0
P6   0
;

ampl: display  Storage_Capacity.body, Storage_Capacity.ub;
:  Storage_Capacity.body Storage_Capacity.ub    :=
P1           900                 3000
P2          1400                 3000
P3          1800                 3000
```

```
P4          3000                    3000
P5          3000                    3000
P6           500                    3000
;

ampl: display Production_Capacity.body, Production_Capacity.ub;
:  Production_Capacity.body Production_Capacity.ub    :=
P1            1500                      1500
P2            2000                      2000
P3            2200                      2200
P4            2800                      3000
P5            2300                      2700
P6               0                      2500
;

ampl: display Storage.rc, Production.rc;
:  Storage.rc Production.rc    :=
P1      0           0
P2      0           0
P3      0           0
P4      0           0
P5      0           0
P6      0           0
;

ampl: display Profit;
Profit = 2446800

ampl: quit;
draw%
```

Note that a solver reports only one optimal dual price or reduced cost to AMPL, however, which may be the rate in either direction. Therefore, a dual price or reduced cost can give you only one slope on the piecewise-linear curve of objective values. Hence these quantities should be used as only an initial quide to the objective's sensitivity to certain variable or constraint bounds. If the sensitivity is very important to your application, you can make a series of runs with different bound settings.