AMPL

A Mathematical Programming Language

The Diet Problem

Item / Nutrient	Protein	Vitamin C	Price
1. Hamburger	28	10	8
2. Sausage	17	0	4
3. Fries	3	15	2
4. Orange juice	1	120	3
Required amount	20	25	

$$\min 8x_1 + 4x_2 + 2x_3 + 3x_4$$

subject to

$$28x_1 + 17x_2 + 3x_3 + 1x_4 \geq 20$$

$$10x_1 + 0x_2 + 15x_3 + 120x_4 \geq 25$$

$$x_1, x_2, x_3, x_4 \in \{0,1\}$$

```
var x1 binary;
var x2 binary;
var x3 binary;
var x4 binary;

minimize TotalCost: 8 * x1 + 4 * x2 + 2 * x3 + 3 * x4;

subject to Protein: 28 * x1 + 17 * x2 + 3 * x3 + 1 * x4 >= 20;
subject to VitaminC: 10 * x1 + 0 * x2 + 15 * x3 + 120 * x4 >= 25;
```

The Diet Problem

Item / Nutrient	Protein	Vitamin C	Price
1. Hamburger	28	10	8
2. Sausage	17	0	4
3. Fries	3	15	2
4. Orange juice	1	120	3
Required amount	20	25	

$$\min 8x_1 + 4x_2 + 2x_3 + 3x_4$$

subject to

$$28x_1 + 17x_2 + 3x_3 + 1x_4 \geq 20$$

$$10x_1 + 0x_2 + 15x_3 + 120x_4 \geq 25$$

$$x_1, x_2, x_3, x_4 \in \{0,1\}$$

C:\Users\Desktop\Desktop\ampltutorial\ampl_mswin64\ampl.exe

```
ampl: option solver cplex;
ampl:
ampl: model C:\Users\Desktop\Desktop\ampltutorial\Diet1.mod;
ampl:
ampl: solve;
CPLEX 12.6.3.0: optimal integer solution; objective 9
3 MIP simplex iterations
0 branch-and-bound nodes
No basis.
ampl:
ampl: display x1, x2, x3, x4;
x1 = 0
x2 = 1
x3 = 1
x4 = 1

ampl:
```

The Diet Problem

Item / Nutrient	Protein	Vitamin C	Price
1. Hamburger	28	10	8
2. Sausage	17	0	4
3. Fries	3	15	2
4. Orange juice	1	120	3
Required amount	20	25	

$$\min 8x_1 + 4x_2 + 2x_3 + 3x_4$$

subject to

$$28x_1 + 17x_2 + 3x_3 + 1x_4 \geq 20$$

$$10x_1 + 0x_2 + 15x_3 + 120x_4 \geq 25$$

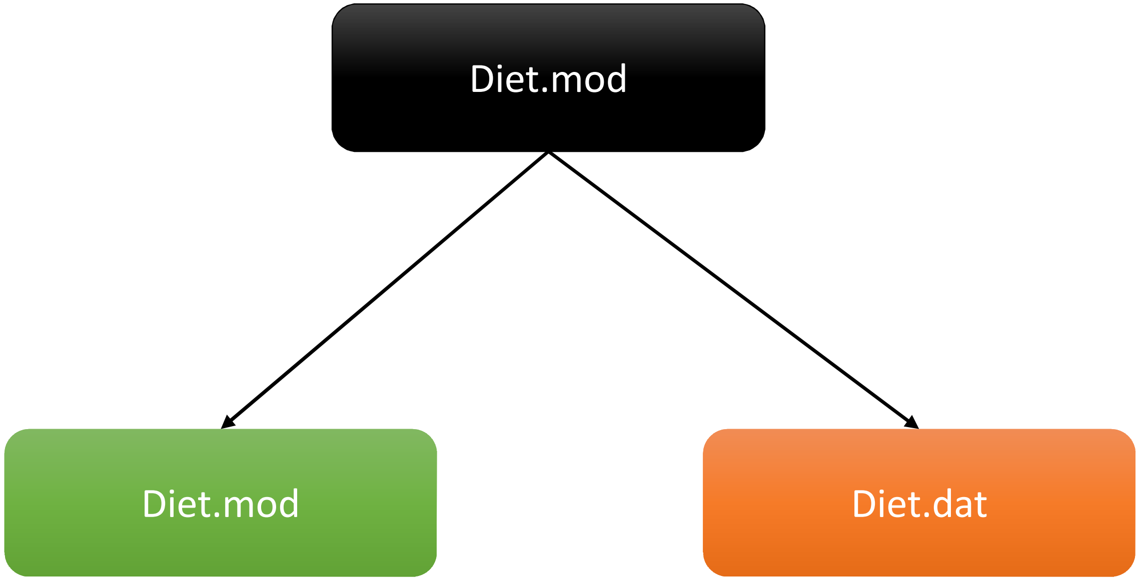
$$x_1, x_2, x_3, x_4 \in \{0,1\}$$

```
var x1 binary;
var x2 binary;
var x3 binary;
var x4 binary;

minimize TotalCost: 8 * x1 + 4 * x2 + 2 * x3 + 3 * x4;

subject to Protein: 28 * x1 + 17 * x2 + 3 * x3 + 1 * x4 >= 20;
subject to VitaminC: 10 * x1 + 0 * x2 + 15 * x3 + 120 * x4 >= 25;
```

Suppose that you consider tens of nutrients instead of just protein and vitamin C;

and suppose also that the menu includes tens of items instead of just four.

The Diet Problem



Elements of a Mathematical Programming Model

1. Sets
2. Parameters
3. Variables
4. Constraints
5. Objective Function

1. Sets

In AMPL, **set** is a keyword used to define finite collections of elements.

These elements can be numerical or non-numerical, and frequently are used as **indices** (of parameters and variables).

```
set S := 1..5;

set S := { 1, 2, 3, 10 };

set S := { 'a', 'b', 'c', 'd' };

set S;

param n;
set S := 1..n;
```

2. Parameters

Parameters are the given constants of your mathematical programming model. AMPL keyword for parameters is **param**.

```
param budget;
param capacity := 500;
```

```
# p[j] is the processing time of job job
set J; #jobs
param p {J};
```

```
# p[i,j] is the processing time of job j
# on machine i
set I; #machines
set J; #jobs
param p {I, J};
```

2. Parameters

Data Validation: data types

```
# g[p] is the number of goals scored by
# player p
set P; #players
param g {P} integer;
```

```
# a[i,j] equals 1 if there exists a link
# between nodes i and j; 0 otherwise
set N; #nodes
param a {N,N} binary;
```

2. Parameters

Data Validation: bounds

```
# p[i,j] is the processing time of job j
# on machine i
set I; #machines
set J; #jobs
param p {I, J} >= 0, <= 120;
```

```
# b[i] is the budget of project i.
set I; #projects
param u {I}; #upper bound on project budget
param b {i in I} >= 0, <= u[i];
```

3. Variables

AMPL's keyword for decision variables is **var**.

Declaration of variables is only slightly different than that of the parameters.

3. Variables

Continuous Decision Variables

```
# x is the deviation of a certain measure from a
# constant value
var x;
```

```
# x[j] is the amount of chemical j used in a chemical
# mixture
set J; #chemicals
var x{J} >= 0;
```

```
# we want to ensure that at least 10mg of each
# chemical is used; and no chemical is used more than
# 100mg
set J; #chemicals
var x{J} >= 10, <= 100;
```

```
# or we want to restrict the use each chemical with a
# lower and upper bound specific to that chemical
set J; #chemicals
param lower{J};
param upper{J};
var x{j in J} >= lower[j], <= upper[j];
```

3. Variables

Discrete Decision Variables

```
# x[i] is the number of workers assigned to workstation i
set I; #workstations
var x{I} integer;
```

```
# actually, we know that it is nonnegative
set I; #workstations
var x{I} integer >= 0;
```

```
# y[j,k] equals 1 if job j precedes job k; and 0 otherwise
set J; #jobs
var y{J,J} integer >=0, <=1;
```

```
# or simply
set J; #jobs
var y{J,J} binary;
```

4. Constraints

Constraint declaration in AMPL has the following syntax:

subject to **constraintName {forall}: constraintExpression;**

Let x_i be the amount of money we invest on project i.

We have a budget of b; and hence, sum of our investments cannot exceed b.

$$\sum_{i \in I} x_i \leq b$$

```
param b;
set I;
var x{I} >= 0;
subject to BudgetConstraint: sum{i in I} x[i] <= b;
```

4. Constraints

Consider a network flow problem.

We want to ensure that the difference between the flow leaving a node and the flow entering a node should be equal to the supply of that node.

Let x_{ij} be the amount of flow from node i to node j; and let b_i be the supply of node i.

$$\sum_{j \in N} x_{ij} - \sum_{k \in N} x_{ki} = b_i \qquad , \forall\, i \in N$$

```
set N;
param b{N};
var x{N,N} >= 0;
subject to FlowBalance {i in N}: sum{j in N} x[i,j] - sum{k in N} x[k,i] = b[i];
```

4. Constraints

Let c_j be the completion time of job j.

And let x_{jk} equal 1 if job j is processed before job k; and 0 otherwise.

Let parameter M be the big-M; and let parameter p_j be the processing time of job j.

We want to write the big-M constraints to avoid overlapping of tasks in a single machine scheduling problem.

$$M\,x_{jk} + c_j \geq c_k + p_j \qquad ;\forall\, j \in J, k \in J$$

$$M\,(1 - x_{jk}) + c_k \geq c_j + p_k \qquad ;\forall\, j \in J, k \in J$$

```
set J;
param p{J};
param M;
var x{J,J} binary;
var c{J} >= 0;
subject to Overlap1 {j in J, k in J}: M * x[j,k]       + c[j] >= c[k] + p[j];
subject to Overlap2 {j in J, k in J}: M * (1-x[j,k])   + c[k] >= c[j] + p[k];
```

4. Constraints

Let c_j be the completion time of job j.

And let x_{jk} equal 1 if job j is processed before job k; and 0 otherwise.

Let parameter M be the big-M; and let parameter p_j be the processing time of job j.

We want to write the big-M constraints to avoid overlapping of tasks in a single machine scheduling problem.

$$M\, x_{jk} + c_j \geq c_k + p_j \qquad ;\forall\, j \in J, k \in J \ni k > j$$

$$M\left(1 - x_{jk}\right) + c_k \geq c_j + p_k \qquad ;\forall\, j \in J, k \in J \ni k > j$$

```
set J;
param p{J};
param M;
var x{J,J} binary;
var c{J} >= 0;
subject to Overlap1 {j in J, k in J: k > j}: M * x[j,k]        + c[j] >= c[k] + p[j];
subject to Overlap2 {j in J, k in J: k > j}: M * (1-x[j,k])    + c[k] >= c[j] + p[k];
```

4. Constraints

Consider the same scheduling problem; however, this time we want to write the precedence constraints. We are given the parameter $prec_{jk}$ which is equal to 1 if job j has to be processed before job k; and 0 otherwise.

$$c_k \geq c_j + p_k \qquad ; \forall\, j \in J, k \in J \ \ni prec_{jk} = 1$$

```
set J;
param p{J};
param prec{J,J};
var c{J} >= 0;
subject to Precedence {j in J, k in K: prec[j,k]=1}: c[k] >= c[j] + p[k];
```

5. Objective Function

Objective function declaration in AMPL has the following syntax:

minimize/maximize **objectiveName**: **objectiveFunction**;

Let decision variable x_{ij} be the amount of flow from node i to node j.

Let parameter c_{ij} be the cost of unit flow from node i to node j.

$$\min \sum_{i \in N} \sum_{j \in N} c_{ij} \ x_{ij}$$

```
set N;
param c{N,N};
var x{N,N} >= 0;
minimize TotalCost: sum{i in N, j in N} x[i,j] * c[i,j];
```

Exercise: The Diet Problem

Item / Nutrient	Protein	Vitamin C	Price
1. Hamburger	28	10	8
2. Sausage	17	0	4
3. Fries	3	15	2
4. Orange juice	1	120	3
Required amount	20	25	

$$\min 8x_1 + 4x_2 + 2x_3 + 3x_4$$

subject to

$$28x_1 + 17x_2 + 3x_3 + 1x_4 \geq 20$$

$$10x_1 + 0x_2 + 15x_3 + 120x_4 \geq 25$$

$$x_1, x_2, x_3, x_4 \in \{0,1\}$$

```
var x1 binary;
var x2 binary;
var x3 binary;
var x4 binary;

minimize TotalCost: 8 * x1 + 4 * x2 + 2 * x3 + 3 * x4;

subject to Protein: 28 * x1 + 17 * x2 + 3 * x3 + 1 * x4 >= 20;
subject to VitaminC: 10 * x1 + 0 * x2 + 15 * x3 + 120 * x4 >= 25;
```

Define the **sets**, **parameters**, **variables**, **constraints** and the **objective function** of the diet problem.

Your model file should not include any data.

Exercise: The Diet Problem: Diet2.mod

```
set N;                              #nutrients
set F;                              #foods
param p{F};                         #food price
param r{N};                         #required nutrient amount
param a{F,N};                       #amount of nutrients in foods
var b{F} binary;                    #binary buy/don't buy variable

subject to Nutrient {n in N}: sum{f in F} a[f,n] * b[f] >= r[n];

minimize TotalCost: sum{f in F} p[f] * b[f];
```

The data file: Diet1.dat

```
set F :=
    Hamburger
    Sausage
    Fries
    "Orange Juice";

set N :=
    Protein
    VitaminC;
```

```
param p :=
    Hamburger           8
    Sausage             4
    Fries               2
    "Orange Juice"      3;

param r :=
    Protein     20
    VitaminC    25;

param a:
                        Protein     VitaminC :=
  Hamburger             28          10
  Sausage               17          0
  Fries                 3           15
  "Orange Juice"        1           120;
```

The Diet Solution: Diet1.dat

C:\Users\Desktop\Desktop\ampltutorial\ampl_mswin64\ampl.exe

```
ampl: option solver cplex;
ampl:
ampl: model C:\Users\Desktop\Desktop\ampltutorial\Diet2.mod;
ampl:
ampl: data C:\Users\Desktop\Desktop\ampltutorial\Diet1.dat;
ampl:
ampl: solve;
CPLEX 12.6.3.0: optimal integer solution; objective 9
3 MIP simplex iterations
0 branch-and-bound nodes
No basis.
ampl:
ampl: display b;
b [*] :=
         Fries  1
     Hamburger  0
'Orange Juice'  1
       Sausage  1
;

ampl: _
```

The Diet Solution: Diet2.dat

C:\Users\Desktop\Desktop\ampltutorial\ampl_mswin64\ampl.exe

```
ampl: option solver cplex;
ampl:
ampl: model C:\Users\Desktop\Desktop\ampltutorial\Diet2.mod;
ampl:
ampl: data C:\Users\Desktop\Desktop\ampltutorial\Diet2.dat;
ampl:
ampl: solve;
CPLEX 12.6.3.0: optimal integer solution; objective 6.66
17 MIP simplex iterations
0 branch-and-bound nodes
No basis.
ampl:
ampl: display b;
b [*] :=
           '1% Lowfat Milk'  1
                  'Big Mac'  1
             Filet-O-Fish  1
            'Fries, small'  1
       'McGrilled Chicken'  0
 'McLean Deluxe w/ Cheese'  0
           'Orange Juice'  1
'Quarter Pounder w/ Cheese'  0
         'Sausage McMuffin'  1
;

ampl:
```

Exercise: $1|r_j|\sum w_j C_j$ with time-indexed variables

$$\min \sum_{j\in J}\sum_{t\in T} w_j\,(t+p_j)\,x_{jt}$$

subject to

$$\sum_{t\in T} x_{jt} = 1 \qquad ,\forall j\in J$$

$$\sum_{j\in J}\ \sum_{\substack{s\in T\ \ni\\ s\geq \max(0,t+1-p_j)\ and\ s\leq t}} x_{js} \leq 1 \qquad ,\forall t\in T\ \ni t\leq |T|-1$$

$$x_{jt}=0 \qquad ,\forall j\in J, t\in T\ \ \ni t< r_j$$

$$x_{jt}\in\{0,1\} \qquad ,\forall j\in J, t\in T$$

Exercise: $1|r_j|\sum w_j C_j$ with time-indexed variables

```
param maxT;
set J; #jobs
set T := 0..maxT; #time intervals


param p{J};
param r{J};
param w{J};
var x{J,T} binary;


minimize WeightedCompTime:
   sum{j in J, t in T} w[j] * (t + p[j]) * x[j,t];

subject to C1 {j in J}: sum{t in T} x[j,t] = 1;

subject to C2 {t in T: t <= card(T) - 1}:
   sum{j in J, s in T: s >= max(0, t + 1 - p[j]) && s <= t} x[j,s] <= 1;

subject to C3 {j in J, t in T: t < r[j]}: x[j,t] = 0;
```

Exercise: $1|r_j|\sum w_j C_j$ with time-indexed variables

Solve the following three instances:

- Sch1.dat (7 jobs)
- Sch2.dat (35 jobs)
- Sch3.dat (100 jobs)

with the model:

- Sch.mod.

Exercise: $1|r_j|\sum w_j C_j$ with time-indexed variables (7 jobs)

C:\Users\Desktop\Desktop\ampltutorial\ampl_mswin64\ampl.exe

```
ampl: option solver cplex;
ampl:
ampl: model C:\Users\Desktop\Desktop\ampltutorial\Sch.mod;
ampl:
ampl: data C:\Users\Desktop\Desktop\ampltutorial\Sch1.dat;
ampl:
ampl: solve;
CPLEX 12.6.3.0: optimal integer solution; objective 3771
89 MIP simplex iterations
0 branch-and-bound nodes
No basis.
ampl:
ampl: display {j in J}: sum{t in T} t*x[j,t];
sum{t in T} t*x[1,t] = 150

sum{t in T} t*x[2,t] = 0

sum{t in T} t*x[3,t] = 18

sum{t in T} t*x[4,t] = 73

sum{t in T} t*x[5,t] = 47

sum{t in T} t*x[6,t] = 30

sum{t in T} t*x[7,t] = 57

ampl:
```

Exercise: $1|r_j|\sum w_j C_j$ with time-indexed variables (35 jobs)

C:\Users\Desktop\Desktop\ampltutorial\ampl_mswin64\ampl.exe

```
ampl: option solver cplex;
ampl: model C:\Users\Desktop\Desktop\ampltutorial\Sch.mod;
ampl: data C:\Users\Desktop\Desktop\ampltutorial\Sch2.dat;
ampl:
ampl: option cplex_options 'mipdisplay 2';
ampl:
ampl: solve;
CPLEX 12.6.3.0: mipdisplay 2
Found incumbent of value 71661.960000 after 0.01 sec. (7.35 ticks)
MIP Presolve eliminated 3 rows and 13 columns.
MIP Presolve modified 1 coefficients.
Reduced MIP has 1033 rows, 28011 columns, and 615958 nonzeros.
Reduced MIP has 28011 binaries, 0 generals, 0 SOSs, and 0 indicators.
MIP Presolve eliminated 3 rows and 13 columns.
MIP Presolve modified 1 coefficients.
Reduced MIP has 1030 rows, 27998 columns, and 615700 nonzeros.
Reduced MIP has 27998 binaries, 0 generals, 0 SOSs, and 0 indicators.
Probing time = 0.06 sec. (14.14 ticks)
Clique table members: 1030.
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 16 threads.
Root relaxation solution time = 0.63 sec. (691.64 ticks)

        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*     0+    0                       71661.9600        0.0000           100.00%
      0     0    54515.0390   191    71661.9600    54515.0390       37   23.93%
*     0+    0                       60129.2300    54515.0390             9.34%
      0     0    54587.7261   251    60129.2300      Cuts: 28      524    9.22%
      0     0    54598.5982   206    60129.2300      Cuts: 13      710    9.20%
*     0+    0                       55316.8100    54598.5982             1.30%
      0     0    54616.4704   194    55316.8100   Cliques: 16      874    1.27%
```

C:\Users\Desktop\Desktop\ampltutorial\ampl_mswin64\ampl.exe

```
ampl: display {j in J, t in T: x[j,t] = 1} x[j,t];
x[j,t] [*,*]
:    0   4   9  18  24  47  79  95 115 125 141 171 182 217 236 245 266 294 326 :=
2    .   .   .   .   .   .   .   .   .   .   .   1   .   .   .   .   .   .   .
4    .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   1   .   .
6    .   .   .   .   .   .   .   .   .   .   .   .   .   1   .   .   .   .   .
7    .   .   .   .   .   .   .   .   .   .   .   .   .   .   1   .   .   .   .
8    .   .   .   1   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
9    .   1   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
11   .   .   .   .   .   .   .   .   1   .   .   .   .   .   .   .   .   .   .
14   1   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
17   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   1   .
18   .   .   .   .   .   .   .   .   .   1   .   .   .   .   .   .   .   .   .
20   .   .   .   .   .   .   1   .   .   .   .   .   .   .   .   .   .   .   .
21   .   .   .   .   .   1   .   .   .   .   .   .   .   .   .   .   .   .   .
23   .   .   1   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .
25   .   .   .   .   1   .   .   .   .   .   .   .   .   .   .   .   .   .   .
27   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   1
31   .   .   .   .   .   .   .   .   .   .   .   .   1   .   .   .   .   .   .
33   .   .   .   .   .   .   .   1   .   .   .   .   .   .   .   .   .   .   .
34   .   .   .   .   .   .   .   .   .   .   .   .   .   .   .   1   .   .   .
35   .   .   .   .   .   .   .   .   .   .   1   .   .   .   .   .   .   .   .

:  351 380 405 437 443 472 504 526 552 577 609 642 677 705 731 760     :=
1    .   .   .   .   .   .   .   .   1   .   .   .   .   .   .   .
3    .   .   .   .   .   .   .   .   .   .   .   .   .   1   .   .
5            1
```

Exercise: $1|r_j|\sum w_j C_j$ with time-indexed variables (100 jobs)

C:\Users\Desktop\Desktop\ampltutorial\ampl_mswin64\ampl.exe

```
ampl: option solver cplex;
ampl: option cplex_options 'mipdisplay 2';
ampl:
ampl: option show_stats 1;
ampl:
ampl: model C:\Users\Desktop\Desktop\ampltutorial\Sch.mod;
ampl: data C:\Users\Desktop\Desktop\ampltutorial\Sch3.dat;
ampl:
ampl: solve;

Presolve eliminates 64318 constraints and 64318 variables.
Adjusted problem:
185782 variables, all binary
2601 constraints, all linear; 4097778 nonzeros
        100 equality constraints
        2501 inequality constraints
1 linear objective; 185782 nonzeros.
```

```
CPLEX 12.6.3.0: mipdisplay 2
Found incumbent of value 539974.520000 after 0.05 sec. (48.97 ticks)
MIP Presolve eliminated 7 rows and 66 columns.
MIP Presolve modified 8 coefficients.
Reduced MIP has 2594 rows, 185716 columns, and 4096084 nonzeros.
Reduced MIP has 185716 binaries, 0 generals, 0 SOSs, and 0 indicators.
MIP Presolve eliminated 24 rows and 169 columns.
Reduced MIP has 2570 rows, 185547 columns, and 4090174 nonzeros.
Reduced MIP has 185547 binaries, 0 generals, 0 SOSs, and 0 indicators.
Probing time = 0.30 sec. (60.86 ticks)
Clique table members: 2568.
MIP emphasis: balance optimality and feasibility.
MIP search method: dynamic search.
Parallel mode: deterministic, using up to 16 threads.
Root relaxation solution time = 3.59 sec. (4374.31 ticks)

        Nodes                                         Cuts/
   Node  Left     Objective  IInf  Best Integer    Best Bound    ItCnt     Gap

*     0+    0                      539974.5200        0.0000           100.00%
      0     0   420158.4038   518   539974.5200   420158.4038       41   22.19%
*     0+    0                      514589.8400   420158.4038            18.35%
      0     0   420294.0867   608   514589.8400      Cuts: 89     1187   18.32%
```