Mathematics for Decisions

AMPL problems

Romeo Rizzi, Alice Raffaele

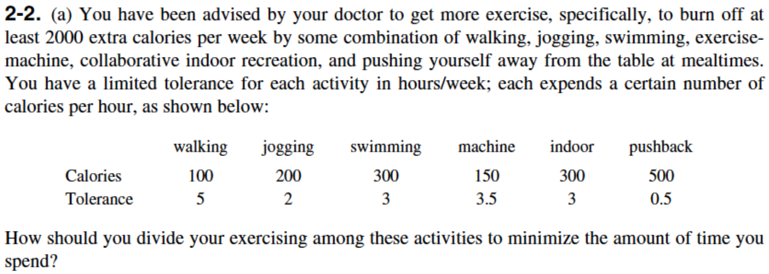
University of Verona

romeo.rizzi@univr.it, alice.raffaele@unitn.it

March 2020

Overview

Sports and workouts



2-2. (a) You have been advised by your doctor to get more exercise, specifically, to burn off at least 2000 extra calories per week by some combination of walking, jogging, swimming, exercise-machine, collaborative indoor recreation, and pushing yourself away from the table at mealtimes. You have a limited tolerance for each activity in hours/week; each expends a certain number of calories per hour, as shown below:

	walking	jogging	swimming	machine	indoor	pushback
Calories	100	200	300	150	300	500
Tolerance	5	2	3	3.5	3	0.5

How should you divide your exercising among these activities to minimize the amount of time you spend?

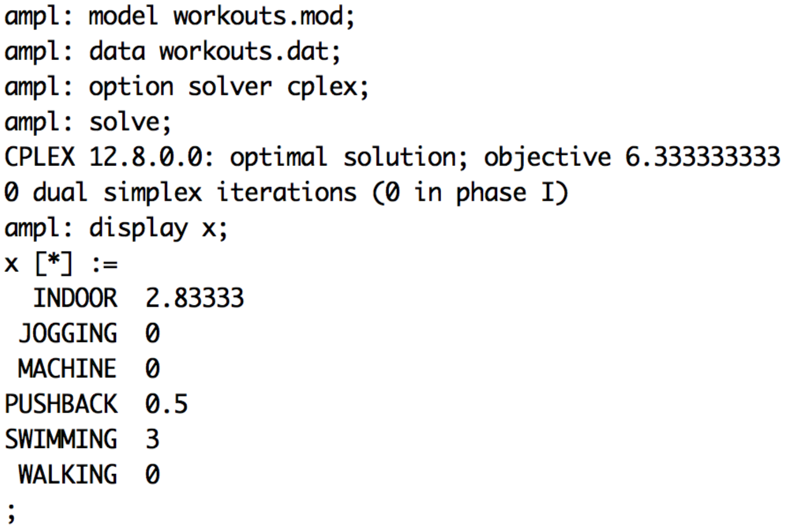
This problem is similar to the Diet one.

Sports and workouts - Formulation

- **Sets**: we can define W as the set of workouts.
- **Parameters**: for each $i \in W$, t_i is the tolerance and c_i the calories per hour.
- **Variables**: a variable for every workout, whose value indicates how many hours of that exercise we should practice every week: $x_i \geq 0, \forall\ i$ in W.
- **Constraints**:
  - For each workout i, we cannot perform more hours than the tolerance indicated: $x_i \leq t_i,\ \forall\ i \in W$.
  - We must perform the needed hours in order to burn off at least 2000 extra calories per week: $\sum_{i \in W} x_i \cdot c_i \geq 2000$.
- **Objective function**: we want to minimize the total amount of time: $\min \sum_{i \in W} x_i$.

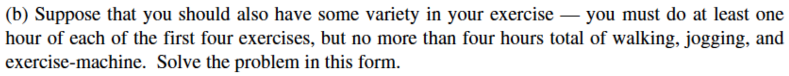
Sports and workouts - AMPL solution

AMPL



```
ampl: model workouts.mod;
ampl: data workouts.dat;
ampl: option solver cplex;
ampl: solve;
CPLEX 12.8.0.0: optimal solution; objective 6.333333333
0 dual simplex iterations (0 in phase I)
ampl: display x;
x [*] :=
  INDOOR  2.83333
 JOGGING  0
 MACHINE  0
PUSHBACK  0.5
SWIMMING  3
 WALKING  0
;
```

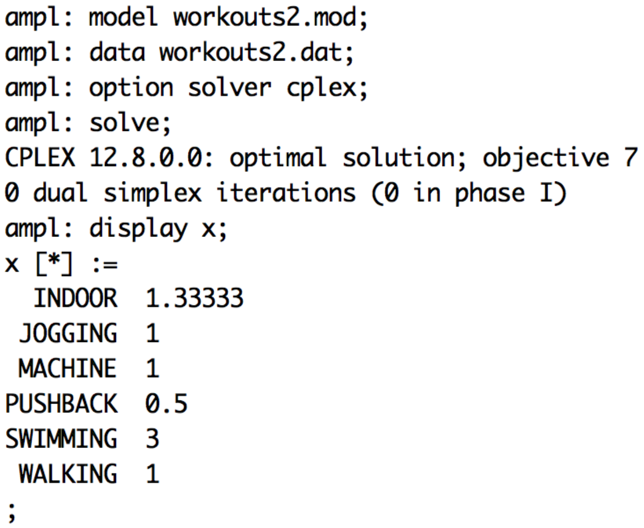
Sports and workouts - More variety


(b) Suppose that you should also have some variety in your exercise — you must do at least one hour of each of the first four exercises, but no more than four hours total of walking, jogging, and exercise-machine. Solve the problem in this form.

Additional constraints:

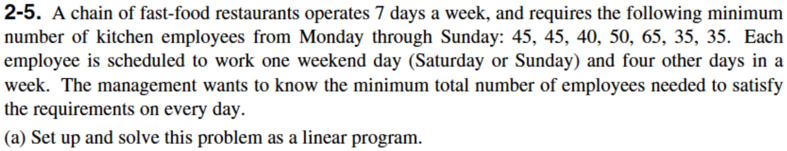
- $x_i \geq 1, \ i = WALKING, JOGGING, SWIMMING, MACHINE$
- $x_{WALKING} + x_{JOGGING} + x_{MACHINE} \leq 4$

Sports and workouts - AMPL solution (II)



```
ampl: model workouts2.mod;
ampl: data workouts2.dat;
ampl: option solver cplex;
ampl: solve;
CPLEX 12.8.0.0: optimal solution; objective 7
0 dual simplex iterations (0 in phase I)
ampl: display x;
x [*] :=
  INDOOR  1.33333
 JOGGING  1
 MACHINE  1
PUSHBACK  0.5
SWIMMING  3
 WALKING  1
;
```

Fast-food



2-5. A chain of fast-food restaurants operates 7 days a week, and requires the following minimum number of kitchen employees from Monday through Sunday: 45, 45, 40, 50, 65, 35, 35. Each employee is scheduled to work one weekend day (Saturday or Sunday) and four other days in a week. The management wants to know the minimum total number of employees needed to satisfy the requirements on every day.

(a) Set up and solve this problem as a linear program.

For instance, consider a Monday. Employees working that day will have one of the following schedules:

1. Monday-Tuesday-Wednesday-Thursday-Saturday
2. Monday-Tuesday-Wednesday-Thursday-Sunday
3. Monday-Tuesday-Wednesday-Friday-Saturday
4. Monday-Tuesday-Wednesday-Friday-Sunday
5. Monday-Tuesday-Thursday-Friday-Saturday
6. Monday-Tuesday-Thursday-Friday-Sunday
7. Monday-Wednesday-Thursday-Friday-Saturday
8. Monday-Wednesday-Thursday-Friday-Sunday

Fast-food - Formulation

Other possible schedules:

9. Tuesday-Wednesday-Thursday-Friday-Saturday
10. Tuesday-Wednesday-Thursday-Friday-Sunday

So we have 10 different schedules for our employees.
Let's model the problem with AMPL:

- **Sets**:
  - $S := \{1, \dots, 10\}$, the set of possible schedules;
  - $D := \{1, \dots, 7\}$, the set of days.
- **Parameters**:
  - The minimum number of employees per each day;
- **Variables**: $x_i, \forall\ i \in S$, number of employees with schedule i.
- **Constraints**: each day, the number of employees required must be satisfied, considering all employees that can work in that day according to their schedules.
  For example, Monday: $x_1 + x_2 + x_3 + \dots + x_8 \geq 45$

Fast-food - AMPL solution

AMPL

```
ampl: model fastfood.mod;
ampl: data fastfood.dat;
ampl: option solver cplex;
ampl: solve;
CPLEX 12.8.0.0: optimal solution; objective 70
6 dual simplex iterations (0 in phase I)
ampl: display x;
x [*] :=
 1   0
 2   0
 3   0
 4  20
 5  15
 6  15
 7  20
 8   0
 9   0
10   0
;
```

Set Covering

The Ministry of Health wants to build some orthopedic-specialized hospitals, able to serve, within the range of 200 kms, Latina, Lecce, Matera, Napoli, Potenza, Salerno e Roma. Here follows, for every place, the list of other cities far less then 200 kms:

- Latina: Latina, Napoli, Roma;
- Lecce: Lecce, Matera;
- Matera: Lecce, Matera, Potenza;
- Napoli: Latina, Napoli, Potenza, Salerno;
- Potenza: Matera, Napoli, Potenza, Salerno;
- Salerno: Napoli, Potenza, Salerno;
- Roma: Latina, Roma.

For example, if an hospital is built in Napoli, it would be able to serve Latina, Potenza and Salerno too, that are less than 200 kms far.
We want to decide in which of the seven places to build the hospitals, such that every city is served at least by one hospital, far less than 200 kms, and considering that two hospitals cannot be built in the same city.

Set Covering - Analysis (I)

- Given a certain number of places, we must choose where to build or install a set of elements (or buildings, like in this case).
- The information about the installation cost can be known (and the problem will be named *weighted*).
- Let $P = \{1, \ldots, p\}$ be the set of cities, while $B = \{1, \ldots, b\}$ is the set of potential buildings.
- Let $S_j \subseteq P$ be the set of regions served by the building j, $j \in B$, and c_j its cost of installation.

Set Covering - Analysis (II)

The problem is to determine the collection of subsets T such that:

$$\min_{T \subseteq B} \left\{ \sum_{j \in T} c_j : \cup_{j \in T} S_j = P \right\}.$$

The solution to the Set Covering Problem is the smallest sub-collection of S, whose union equals the universe P.

Set Covering - Formulation (I)

- **Sets**: we can define P as the set of places and B the set of potential buildings.
- **Parameters**: we know data about which places will be covered if a building is built $\rightarrow$ Incidence matrix A made just by 1 and 0 values (in this instance, the matrix is squared because it is possible to build an hospital in every place).

Place/Building	**LT**	**LE**	**MT**	**NA**	**PT**	**SA**	**RM**
LT	1	0	0	1	0	0	1
LE	0	1	1	0	0	0	0
MT	0	1	1	0	1	0	0
NA	1	0	0	1	1	1	0
PT	0	0	1	1	1	1	0
SA	0	0	0	1	1	1	0
RM	1	0	0	0	0	0	1

- **Variables**: a boolean variable for every building, whose value will be 1 if the hospital i is built, 0 otherwise: x_i, $\forall\ i$ in B.

Set Covering - Formulation (II)

Constraints:

- There must be at most one hospital in every place: $x_i \leq 1$, for every i in B; since our variables are boolean (0 or 1), this constraint is automatically satisfied.
- Every place must be covered by at least one hospital: $Ax \geq 1$, where A is the incidence matrix of cities and buildings and x is the column-vector composed of the binary variables associated to the buildings in B.
  For example, from the first line of the incidence matrix we know that

$$x_{LT} + x_{NA} + x_{RM} \geq 1.$$

Objective function: since we do not have information about the cost of building a hospital in a specific city, we just want to minimize the number of hospitals to build: $\min \sum_{i \in B} x_i$.

Formulation (III)

$$\begin{array}{lrcll} \min \sum_{i \in B} & x_i & & & \\ \text{s.t.} & Ax & \geq & 1 & \\ & x_i & \in & \{0,1\} & \forall i \in B \end{array}$$

Weighted version: if every building has its related cost of installation, a more general version of the problem could be modeled, where the purpose is not to minimize the number of buildings to build but the total installation costs:

$$\min \sum_{i \in B} c_i x_i$$

where c_i is the cost of building i, for every i in B.
The rest of the formulation does not need to be changed, because the cost only impacts the objective function.

Production / Lot Sizing

A company produces cubic meter glass fiber and would like to plan the production for the following six weeks. The production capacity is limited and this limit varies according to the considered week. Weekly demand is already known for the whole period. Production and warehousing costs also vary according to the week. Data are shown in the following table:

Week	**Capacity**	**Demand**	**Prod. Cost**	**Warehousing Cost**
1	140	100	5	0.20
2	100	120	8	0.30
3	110	100	6	0.20
4	100	90	6	0.25
5	120	120	7	0.30
6	100	110	6	0.40

Plan the production minimizing production and warehousing costs.

Production / Lot Sizing - Analysis (I)

- We need a multi-period model that keeps in consideration the storage (i.e., products in the warehouse waiting to be sold).
- The production planning during a determined time horizon is one of the classical problem of Operations Research. The warehouse is needed when the production capacity is limited, for a certain period, therefore it becomes essential to plan to have storage in order to be able to satisfy the demand.
- In this case, there is a single product to be produced during a time horizon of weeks $\{1, \ldots, N\}$. For every period $i \in \{1, \ldots, N\}$, we know the production capacity p_i, the related demand d_i and also the production cost c_i and the warehousing cost w_i per single unit produced and left in the warehouse. We need to determine how many units of cubic meter glass fiber produce, satisfying the demand of every week i and minimizing the sum of production costs and warehousing costs in the whole period.

Production / Lot Sizing - Analysis (II)

- Looking to numerical data, we can see that during Week 2 the production capacity is 100, whereas the demand is 120: we can already say that, during Week 2, some items produced but not sold in Week 1 will be used to cover customers' needs in Week 2.
- But let's forget about data for now and just focus on the formulation.

Production / Lot Sizing - Formulation (i)

- **Sets**: we can define a set for the production periods (in this exercise they are weeks, but abstracting will allow us to consider every possible time interval, such as hours, days, months, etc): $S := \{1, \dots, N\}$.
- **Parameters**: as said before, we have data about:
  - $production\_capacity := \{p_1, p_2, \dots, p_N\}$
  - $demand := \{d_1, d_2, \dots, d_N\}$
  - $production\_cost := \{c_1, c_2, \dots, c_N\}$
  - $warehousing\_cost := \{w_1, w_2, \dots, w_N\}$
- **Variables**: we need to know how many units have been produced in every period and how many units end up in the warehouse. We indicate the former with x and the latter with y:
  - $x := \{x_1, x_2, \dots, x_N\}$
  - $y := \{y_1, y_2, \dots, y_N\}$

Production / Lot Sizing - Formulation (ii)

Constraints: Obviously x and y are related between each other.

- We can assume that $y_0 = 0$, since there is no mention in the problem text of any initial storage $\rightarrow$ Therefore in the first period $y_1 = x_1 - d_1$.
- In all other periods we also need to count the storage left from previous periods: $y_i = y_{i-1} + x_i - d_i$.

Moreover, defining x, we need to consider the production capacity limit: $x_n \leq p_n$, for every period $\{1, \dots, N\}$.

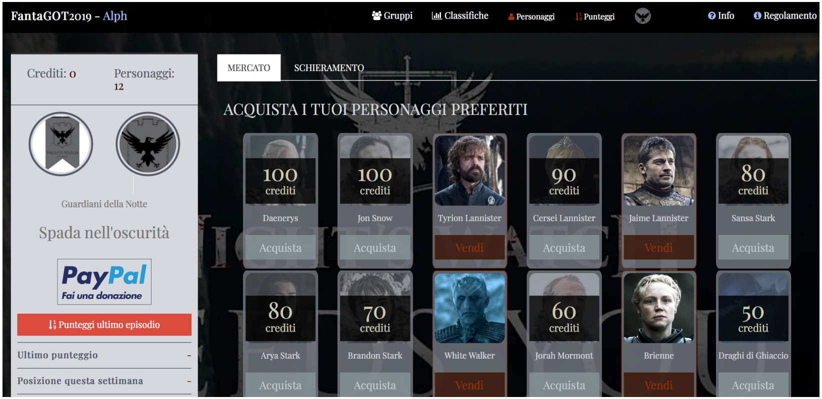
Objective function: we want to minimize the total costs, given by the sum of the total production costs and the total warehousing costs: $\sum_{i=1}^{N}(x_i c_i + y_i w_i)$.

Production / Lot Sizing - Formulation (iii)

The complete formulation is the following:

$$
\begin{array}{llcll}
\min \sum_{i=1}^{N} & (x_i c_i + y_i w_i) & & & \\
\text{s.t.} & y_1 & = & x_1 - d_1 & \\
 & y_i & = & y_{i-1} + x_i - d_i & \forall i \geq 2 \in S \\
 & x_i & \leq & p_i & \forall i \in S \\
 & x_i, y_i & \geq & 0 & \forall i \in S
\end{array}
$$

Fantasy GoT


FantaGOT2019 - Alph

Gruppi Classifiche Personaggi Punteggi Info Regolamento

Crediti: 0 Personaggi: 12

Guardiani della Notte

Spada nell'oscurità

PayPal Fai una donazione

Punteggi ultimo episodio

Ultimo punteggio -

Posizione questa settimana -

MERCATO SCHIERAMENTO

ACQUISTA I TUOI PERSONAGGI PREFERITI

100 crediti	100 crediti		90 crediti		80 crediti
Daenerys	Jon Snow	Tyrion Lannister	Cersei Lannister	Jaime Lannister	Sansa Stark
Acquista	Acquista	Vendi	Acquista	Vendi	Acquista
80 crediti	70 crediti		60 crediti		50 crediti
Arya Stark	Brandon Stark	White Walker	Jorah Mormont	Brienne	Draghi di Ghiaccio
Acquista	Acquista	Vendi	Acquista	Vendi	Acquista

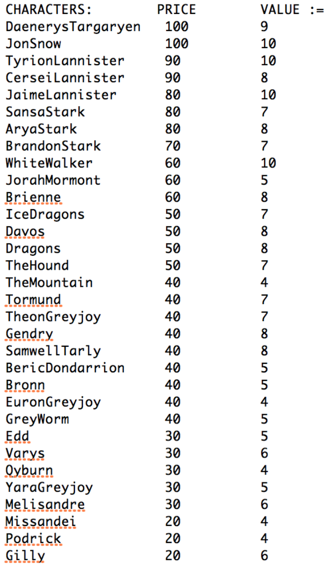
Fantasy GoT (I)

The new season of Game of Thrones is about to begin. Why not enjoying it more, playing with friends and relatives to some *Fantasy GoT*?

Once signed in to its website, **Fantasy GoT** allows you to build your own team, choosing twelve characters among the ones in the series of *A Song of Ice and Fire*.

Every character is assigned with a *price*, more or less proportional to its relevance in the series, and with a *value*, usually established according to his/her behavior in previous seasons.

Fantasy GoT (II)



```
CHARACTERS:         PRICE         VALUE :=
DaenerysTargaryen    100          9
JonSnow              100          10
TyrionLannister      90           10
CerseiLannister      90           8
JaimeLannister       80           10
SansaStark           80           7
AryaStark            80           8
BrandonStark         70           7
WhiteWalker          60           10
JorahMormont         60           5
Brienne              60           8
IceDragons           50           7
Davos                50           8
Dragons              50           8
TheHound             50           7
TheMountain          40           4
Tormund              40           7
TheonGreyjoy         40           7
Gendry               40           8
SamwellTarly         40           8
BericDondarrion      40           5
Bronn                40           5
EuronGreyjoy         40           4
GreyWorm             40           5
Edd                  30           5
Varys                30           6
Qyburn               30           4
YaraGreyjoy          30           5
Melisandre           30           6
Missandei            20           4
Podrick              20           4
Gilly                20           6
```

Fantasy GoT (III)

Obviously, as in every fantasy game you can find online, you cannot buy all the strongest characters because of a *limited budget*, for example 600 credits.

How to build your team? There can be several goals:

- Maximizing the total value of chosen characters;
- Using all available budget.

Knapsack - Analysis (I)

Classic problem of combinatorics optimization:

- **INPUT:** two natural numbers n e B; a set of n items, each one of them described by a weight w_i and a value v_i, for each $i = 1, \ldots, n$;
- **OUTPUT:** a subset S of the items to minimize/maximize a certain objective function, such that the sum of the weights of the selected items does not exceed the capacity B.

A lot of practical applications:

- The Beagle Boys enters in Uncle Scrooge's safety vault, but they cannot bring all the gold bars because of their small truck: its capacity is limited to just 4.2 quintals...
- Burning a CD with limited space, choosing tracks from a list of .mp3 songs, each one given with its size.
- Etc.

Knapsack - Analysis (II)

- **Variables**: one binary variable for each item $i = 1, \dots, n$: $x_i = 1$ if the item is inserted into S; 0 otherwise.
- **Objective function**:
  - Filling the knapsack as most as possible: $\min B - \sum_i w_i \cdot x_i$
  - Maximizing the total value: $\max \sum_i v_i \cdot x_i$
- **Constraints**:
  - Limited capacity: $\sum_i x_i \cdot w_i \leq B$;
  - Maximum number of objects: $\sum_i x_i \leq 12$

    (Note: usually this last constraint is not required, but in FantasyGoT it is)

Knapsack - Analysis (III)

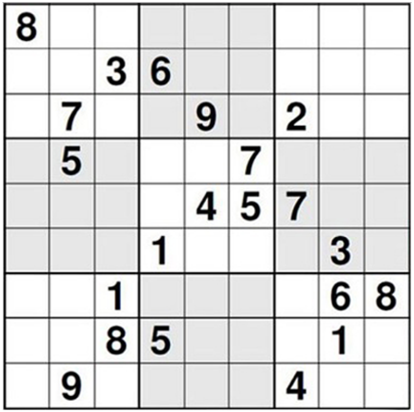
Model for maximizing the total value:

$$\begin{array}{lll} & \max \sum_{i=1}^{n} x_i \cdot v_i & \\ \text{s.t.} & \sum_{i=1}^{n} x_i \cdot w_i \leq B & \\ & \sum_{i=1}^{n} x_i \leq MAX\_NB & \\ & x_i \in \{0,1\} & \forall i = \{1,\ldots,n\} \end{array}$$

Knapsack - Complexity and possible algorithms

- The corresponding decision problem (*Can you obtain a total value of V without exceeding the capacity B?*) is NP-complete;
- The optimization problem is NP-hard and currently there is no polynomial algorithm able to solve it;
- **Approaches**:
  - *Brute force*: computing all possible subsets S of the n items and comparing their values costs $O(2^n)$;
  - *Heuristics* (to obtain good solutions):
    - Ordering items according to their value (decreasing);
    - Ordering items according to their weight (increasing);
    - Ordering items according to their ratio value/weight (decreasing - Martello & Toth, 1990).
  - *Dynamic programming*: pseudo-polynomial.

Sudoku



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

How to model this game as
an Integer Linear Programming problem?

Sudoku - Analysis (I)

- **Variables**: $x_{i,j,k} = 1$ if $(i,j) = k$;
  0 otherwise.
- **Objective function**: $0 \rightarrow$ It is a *feasibility* problem, we don't care about maximizing or minimizing anything.
- **Constraints**:
  - Each cell has to be filled:

$$\sum_{k=1}^{9} x_{i,j,k} = 1, \forall i \in \{1,\ldots,9\}, \forall j \in \{1,\ldots,9\}$$

  - Each digit from 1 to 9 appears exactly once in each row:

$$\sum_{j=1}^{9} x_{i,j,k} = 1, \forall i \in \{1,\ldots,9\}, \forall k \in \{1,\ldots,9\}$$

  - Each digit from 1 to 9 appears exactly once in each columns:

$$\sum_{i=1}^{9} x_{i,j,k} = 1, \forall j \in \{1,\ldots,9\}, \forall k \in \{1,\ldots,9\}$$

Sudoku - Analysis (II)

- **Constraints**:
  - Each digit from 1 to 9 appears exactly once in each square:

$$\sum_{j=(3p-2)}^{3p} \sum_{i=(3q-2)}^{3q} x_{i,j,k} = 1, \forall p \in \{1,2,3\}, q \in \{1,2,3\}, k \in \{1,\cdots,n$$

  - Information already known:

$$x_{i,j,k} = 1, \forall i,j,k \in DATA$$

Triangle

7

3 8

8 1 0

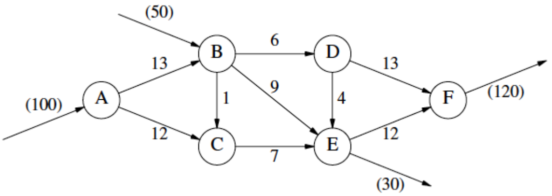
2 7 4 4

4 5 2 6 5

Modeling the problem in order to compute the biggest sum of number you can obtain following a path from the top of the triangle until its basis. At each step, you can diagonally go down to the right or to the left, picking only one number.

Network Transshipment

15-1. The following diagram can be interpreted as representing a network transshipment problem:



The arrows into nodes A and B represent supply in the indicated amounts, 100 and 50; the arrows out of nodes E and F similarly represent demand in the amounts 30 and 120. The remaining arrows indicate shipment possibilities, and the numbers on them are the unit shipping costs. There is a capacity of 80 on every arc.

Can you find the minimum cost flow that satisfies the demands exiting from nodes E and F?

Network Transshipment - Formulation

- **Sets**:
  - NODES := A B C D E F;
  - ARCS := {AB, AC, BC, BD, BE, CE, DE, DF, EF}, all arcs in the network (except supplies and demands); they are defined **within** NODES **cross** NODES;
- **Parameters**:
  - supplies{NODES}, the possible input sources;
  - demands{NODES}, the possible output targets;
  - arc_capacity{ARCS} := 80;
  - costs{ARCS}, the unit shipping cost for each arc in A.
- **Variables**: a variable for each arc, whose value indicates the amount passing through it: $0 \leq x_{ij} \leq 80$.
- **Constraints**: at each node, flow conservation.
  For example, node B: $50 + x_{AB} = x_{BD} + x_{BC} + x_{BE}$.

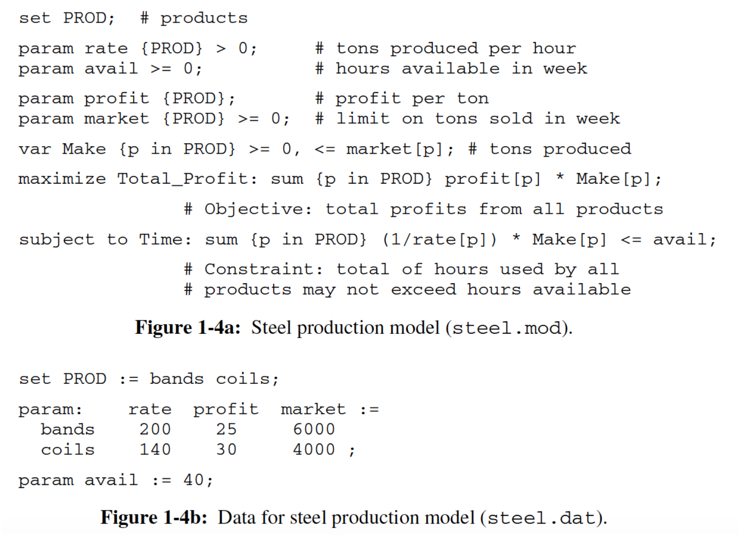
Network Transshipment - AMPL solution

AMPL

```
ampl: model transshipment2.mod;
ampl: data transshipment2.dat;
ampl: option solver cplex;
ampl: solve;
CPLEX 12.8.0.0: optimal solution; objective 3710
2 dual simplex iterations (1 in phase I)
ampl: display x;
x :=
A B   20
A C   80
B C    0
B D   70
B E    0
C E   80
D E    0
D F   70
E F   50
;
```

Steel

Consider the following AMPL problem:



```
set PROD;  # products

param rate {PROD} > 0;      # tons produced per hour
param avail >= 0;           # hours available in week

param profit {PROD};        # profit per ton
param market {PROD} >= 0;   # limit on tons sold in week

var Make {p in PROD} >= 0, <= market[p]; # tons produced

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

               # Objective: total profits from all products

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

               # Constraint: total of hours used by all
               # products may not exceed hours available
```

Figure 1-4a: Steel production model (`steel.mod`).

```
set PROD := bands coils;

param:    rate  profit  market :=
  bands    200    25     6000
  coils    140    30     4000 ;

param avail := 40;
```

Figure 1-4b: Data for steel production model (`steel.dat`).

Steel T - Multiperiod (I)

Now, suppose we want to consider T periods. How should we modify the model and the data file?

```
set PROD;      # products
param T > 0;   # number of weeks

param rate {PROD} > 0;           # tons per hour produced
param inv0 {PROD} >= 0;          # initial inventory
param avail {1..T} >= 0;         # hours available in week
param market {PROD,1..T} >= 0;   # limit on tons sold in week

param prodcost {PROD} >= 0;      # cost per ton produced
param invcost {PROD} >= 0;       # carrying cost/ton of inventory
param revenue {PROD,1..T} >= 0;  # revenue per ton sold

var Make {PROD,1..T} >= 0;       # tons produced
var Inv {PROD,0..T} >= 0;        # tons inventoried
var Sell {p in PROD, t in 1..T} >= 0, <= market[p,t]; # tons sold

maximize Total_Profit:
   sum {p in PROD, t in 1..T} (revenue[p,t]*Sell[p,t] -
      prodcost[p]*Make[p,t] - invcost[p]*Inv[p,t]);

               # Total revenue less costs in all weeks

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

               # Total of hours used by all products
               # may not exceed hours available, in each week

subject to Init_Inv {p in PROD}:  Inv[p,0] = inv0[p];

               # Initial inventory must equal given value

subject to Balance {p in PROD, t in 1..T}:
   Make[p,t] + Inv[p,t-1] = Sell[p,t] + Inv[p,t];

               # Tons produced and taken from inventory
               # must equal tons sold and put into inventory
```

Steel T - Multiperiod (II)

And here's the corresponding data file:

```
param T := 4;
set PROD := bands coils;

param avail :=  1 40  2 40  3 32  4 40 ;

param rate :=  bands 200   coils 140 ;
param inv0 :=  bands  10   coils   0 ;

param prodcost :=  bands 10    coils  11 ;
param invcost  :=  bands  2.5  coils   3 ;

param revenue:   1     2     3     4 :=
       bands    25    26    27    27
       coils    30    35    37    39 ;

param market:    1     2     3     4 :=
       bands  6000  6000  4000  6500
       coils  4000  2500  3500  4200 ;
```

Steel T - Sensitivity analysis

```
ampl: model steelT.mod; data steelT.dat;
ampl: option solver cplex;
ampl: option cplex_options 'sensitivity';

ampl: solve;
CPLEX 8.0.0: sensitivity
CPLEX 8.0.0: optimal solution; objective 515033
16 dual simplex iterations (0 in phase I)

suffix up OUT;
suffix down OUT;
suffix current OUT;
```

Steel T - Variables

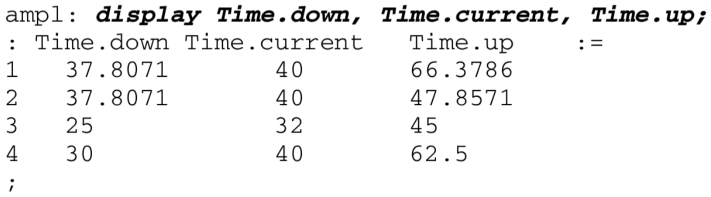
Suffix .current indicates the objective function coefficient in the current problem, while .down and .up give the smallest and largest values of the objective coefficient for which the current LP basis remains optimal.

```
ampl: display Sell.down, Sell.current, Sell.up;
:         Sell.down Sell.current     Sell.up      :=
bands 1    23.3            25       1e+20
bands 2    25.4            26       1e+20
bands 3    24.9            27          27.5
bands 4    10              27          29.1
coils 1    29.2857         30          30.8571
coils 2    33              35       1e+20
coils 3    35.2857         37       1e+20
coils 4    35.2857         39       1e+20
;
```

Note: values of -1e+20 and 1e+20 in the .down and .up column correspond to what CPLEX calls *-infinity* and *+infinity* in its tables.

Steel T - Constraints

The interpretation is similar except that it applies to a constraint's constant term (i.e., the right-hand-side value)



```
ampl: display Time.down, Time.current, Time.up;
: Time.down Time.current   Time.up    :=
1   37.8071         40       66.3786
2   37.8071         40       47.8571
3   25              32       45
4   30              40       62.5
;
```

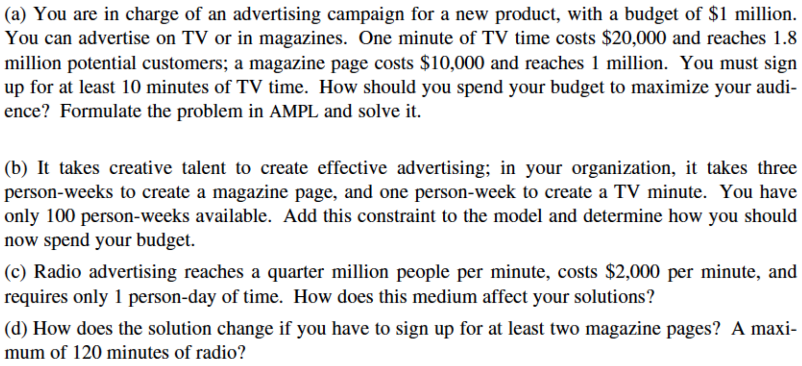
Cocoa Purchase Planning

Our chocolate factory buys different kind of cocoa beans from three suppliers and then sells chocolate bars to retailers.
For next winter, we'll produce four new kind of chocolate bars and we have already received orders to be filled, as given in the following table:

Kind	**Orders**	**S1**		**S2**		**S3**	
		Capacity	**Price**	**Capacity**	**Price**	**Capacity**	**Price**
Milk	80	40	5	45	6	30	4
White	70	25	4	30	5	25	6
Dark	120	70	8	75	7	30	10
Ruby	90	50	10	30	15	30	12
Total		150		160		90	

Determine the optimal cocoa purchase planning minimizing total costs.

Advertisement



(a) You are in charge of an advertising campaign for a new product, with a budget of \$1 million. You can advertise on TV or in magazines. One minute of TV time costs \$20,000 and reaches 1.8 million potential customers; a magazine page costs \$10,000 and reaches 1 million. You must sign up for at least 10 minutes of TV time. How should you spend your budget to maximize your audience? Formulate the problem in AMPL and solve it.

(b) It takes creative talent to create effective advertising; in your organization, it takes three person-weeks to create a magazine page, and one person-week to create a TV minute. You have only 100 person-weeks available. Add this constraint to the model and determine how you should now spend your budget.

(c) Radio advertising reaches a quarter million people per minute, costs \$2,000 per minute, and requires only 1 person-day of time. How does this medium affect your solutions?

(d) How does the solution change if you have to sign up for at least two magazine pages? A maximum of 120 minutes of radio?