Chapter 4

Easy modelling problems

4.1 Compact storage of similar sequences

One practical problem encountered during the DNA mapping process is that of compactly storing extremely long DNA sequences of the same length which do not differ greatly. We consider here a simplified version of the problem with sequences of 2 symbols only (0 and 1). The *Hamming distance* between two sequences $a, b \in \mathbb{F}_2^n$ is defined as $\sum_{i=1}^{n} |a_i - b_i|$, i.e. the number of bits which should be flipped to transform a into b. For example, on the following set of 6 sequences below, the distance matrix is as follows:

1. `011100011101`
2. `101101011001`
3. `110100111001`
4. `101001111101`
5. `100100111101`
6. `010101011100`

	1	2	3	4	5	6
1	0	4	4	5	4	3
2	-	0	4	3	4	5
3	-	-	0	5	2	5
4	-	-	-	0	3	6
5	-	-	-	-	0	5
6	-	-	-	-	-	0

As long as the Hamming distances are not too large, a compact storage scheme can be envisaged where we only store one complete sequence and all the differences which allow the reconstruction of the other sequences. Explain how this problem can be formulated to find a spanning tree of minimum cost in a graph. Solve the problem for the instance given above. [*E. Amaldi, Politecnico di Milano*]

4.2 Communication of secret messages

Given a communication network the probability that a secret message is intercepted along a link connecting node i to j is p_{ij}. Explain how you can model the problem of broadcasting the secret message to every node minimizing the interception probability as a minimum spanning tree problem on a graph. [*E. Amaldi, Politecnico di Milano*]

4.3 Mixed production

A firm is planning the production of 3 products A_1,A_2,A_3. In a month production can be active for 22 days. In the following tables are given: maximum demands (units=100kg), price ($/100Kg), production costs (per 100Kg of product), and production quotas (maximum amount of 100kg units of product that would be produced in a day if all production lines were dedicated to the product).

Product	A_1	A_2	A_3
Maximum demand	5300	4500	5400
Selling price	\$124	\$109	\$115
Production cost	\$73.30	\$52.90	\$65.40
Production quota	500	450	550

1. Formulate an AMPL model to determine the production plan to maximize the total income.
2. Change the mathematical program and the AMPL model to cater for a fixed activation cost on the production line, as follows:

Product	A_1	A_2	A_3
Activation cost	\$170000	\$150000	\$100000

3. Change the mathematical program and the AMPL model to cater for both the fixed activation cost and for a minimum production batch:

Product	A_1	A_2	A_3
Minimum batch	20	20	16

[*E. Amaldi, Politecnico di Milano*]

4.4 Production planning

A firm is planning the production of 3 products A_1, A_2, A_3 over a time horizon of 4 months (january to april). Demand for the products over the months is as follows:

Demand	January	February	March	April
A_1	5300	1200	7400	5300
A_2	4500	5400	6500	7200
A_3	4400	6700	12500	13200

Prices, production costs, production quotas, activation costs and minimum batches (see Ex. 4.3 for definitions of these quantities) are:

Product	A_1	A_2	A_3
Unit prices	\$124	\$109	\$115
Activation costs	\$150000	\$150000	\$100000
Production costs	\$73.30	\$52.90	\$65.40
Production quotas	500	450	550
Minimum batches	20	20	16

There are 23 productive days in january, 20 in february, 23 in march and 22 in april. The activation status of a production line can be changed every month. Minimum batches are monthly.

Moreover, storage space can be rented at monthly rates of \$3.50 for A_1, \$4.00 for A_2 and \$3.00 for A_3. Each product takes the same amount of storage space. The total available volume is 800 units.

Write a mathematical program to maximize the income, and solve it with AMPL. [*E. Amaldi, Politecnico di Milano*]

4.5 Transportation

An Italian transportation firm should carry some empty containers from its 6 stores (in Verona, Perugia, Rome, Pescara, Taranto and Lamezia) to the main national ports (Genoa, Venice, Ancona, Naples, Bari). The container stocks at the stores are the following:

	Empty containers
Verona	10
Perugia	12
Rome	20
Pescara	24
Taranto	18
Lamezia	40

The demands at the ports are as follows:

	Container demand
Genoa	20
Venice	15
Ancona	25
Naples	33
Bari	21

Transportation is carried out by a fleet of lorries. The transportation cost for each container is proportional to the distance travelled by the lorry, and amounts to 30 euro / km. Every lorry can carry at most 2 containers. Distances are as follows:

	Genoa	Venice	Ancona	Naples	Bari
Verona	290 km	115 km	355 km	715 km	810 km
Perugia	380 km	340 km	165 km	380 km	610 km
Rome	505 km	530 km	285 km	220 km	450 km
Pescara	655 km	450 km	155 km	240 km	315 km
Taranto	1010 km	840 km	550 km	305 km	95 km
Lamezia	1072 km	1097 km	747 km	372 km	333 km

Write a mathematical program to find the minimal cost transportation policy and solve it with AMPL. [*E. Amaldi, Politecnico di Milano*]

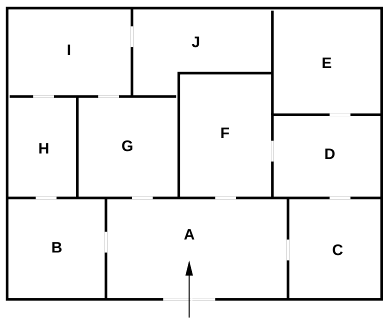
4.6 Project planning with precedences

A project consists of the following 7 activities, whose length in days is given in brackets: A (4), B (3), C (5), D (2), E (10), F (10), G (1). The following precedences are also given: $A \to G, D$; $E, G \to F$;

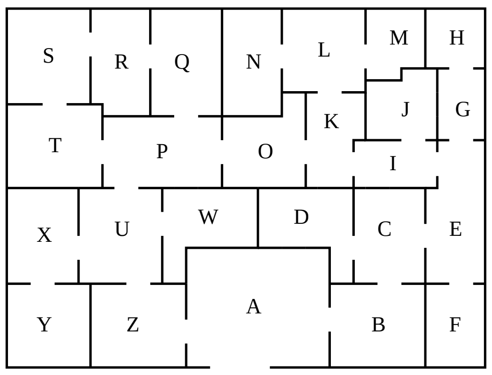
$D, F \to C$; $F \to B$. Each day of work costs 1000 euros; furthermore a special machinery must be rented from the beginning of activity A to the end of activity B at a daily cost of 5000 euros. Formulate this as an LP problem and suggest an algorithm for solving it. [*F. Malucelli, Politecnico di Milano*]

4.7 Museum guards

A museum director must decide how many guards should be employed to control a new wing. Budget cuts have forced him to station guards at each door, guarding two rooms at once. Formulate a mathematical program to minimize the number of guards. Solve the problem on the map below using AMPL.



Also solve the problem on the following map.



[*P. Belotti, Carnegie Mellon University*]

4.8 Inheritance

A rich aristocrat passes away, leaving the following legacy:

- A Caillebotte picture: 25000$
- A bust of Diocletian: 5000$
- A Yuan dinasty chinese vase: 20000$

- A 911 Porsche: 40000$
- Three diamonds: 12000$ each
- A Louis XV sofa: 3000$
- Two very precious Jack Russell race dogs: 3000$ each (the will asserts that they may not be separated)
- A sculpture dated 200 A.D.: 10000$
- A sailing boat: 15000$
- A Harley Davidson motorbike: 10000$
- A piece of furniture that once belonged to Cavour: 13.000$,

which must be shared between the two sons. What is the partition that minimizes the difference between the values of the two parts? Formulate a mathematical program and solve it with AMPL. [*P. Belotti, Carnegie Mellon*]

4.9 Carelland

The independent state of Carelland mainly exports four goods: steel, engines, electronic components and plastics. The Chancellor of the Exchequer (a.k.a. the minister of economy) of Carelland wants to maximize exports and minimize imports. The unit prices on the world markets for steel, engines, electronics and plastics, expressed in the local currency (the Klunz) are, respectively: 500, 1500, 300, 1200. Producing 1 steel unit requires 0.02 engine units, 0.01 plastics units, 250 Klunz in other imported goods and 6 man-months of work. Producing 1 engine unit requires 0.8 steel units, 0.15 electronics units, 0.11 plastics units, 300 Klunz in imported goods and 1 man-year. One electronics unit requires: 0.01 steel units, 0.01 engine units, 0.05 plastics units, 50 Klunz in imported goods and 6 man-months. One plastics unit requires: 0.03 engine units, 0.2 steel units, 0.05 electronics units, 300 Klunz in imported goods and 2 man-years. Engine production is limited to 650000 units, plastics production to 60000 units. The total available workforce is 830000 each year. Steel, engines, electronics and plastics cannot be imported. Write a mathematical program that maximizes the gross internal product and solve the problem with AMPL. [*G. Carello, Politecnico di Milano*]

4.10 CPU Scheduling

10 tasks must be run on 3 CPUs at 1.33, 2 and 2.66 GHz (each processor can run only one task at a time). The number of elementary operations of the tasks (expressed in billions of instructions (BI)) is as follows:

process	1	2	3	4	5	6	7
BI	1.1	2.1	3	1	0.7	5	3

Schedule tasks to processors so that the completion time of the last task is minimized. Solve the problem with AMPL.

4.11 Dyeing plant

A fabric dyeing plant has 3 dyeing baths. Each batch of fabric must be dyed in each bath in the order: first, second, third bath. The plant must colour five batches of fabric of different sizes. Dyeing batch i in bath j takes a time s_{ij} expressed in hours in the matrix below:

$$\begin{pmatrix} 3 & 1 & 1 \\ 2 & 1.5 & 1 \\ 3 & 1.2 & 1.3 \\ 2 & 2 & 2 \\ 2.1 & 2 & 3 \end{pmatrix}.$$

Schedule the dyeing operations in the baths so that the ending time of the last batch is minimized.

4.12 Parking

On Dantzig Street cars can be parked on both sides of the street. Mr. Edmonds, who lives at number 1, is organizing a party for around 30 people, who will arrive in 15 cars. The length of the i-th car is λ_i, expressed in meters as follows:

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
λ_i	4	4.5	5	4.1	2.4	5.2	3.7	3.5	3.2	4.5	2.3	3.3	3.8	4.6	3

In order to avoid bothering the neighbours, Mr. Edmonds would like to arrange the parking on both sides of the street so that the length of the street occupied by his friends' cars should be minimum. Give a mathematical programming formulation and solve the problem with AMPL.

How does the program change if on exactly one of the street sides the cars should not occupy more than 15m?

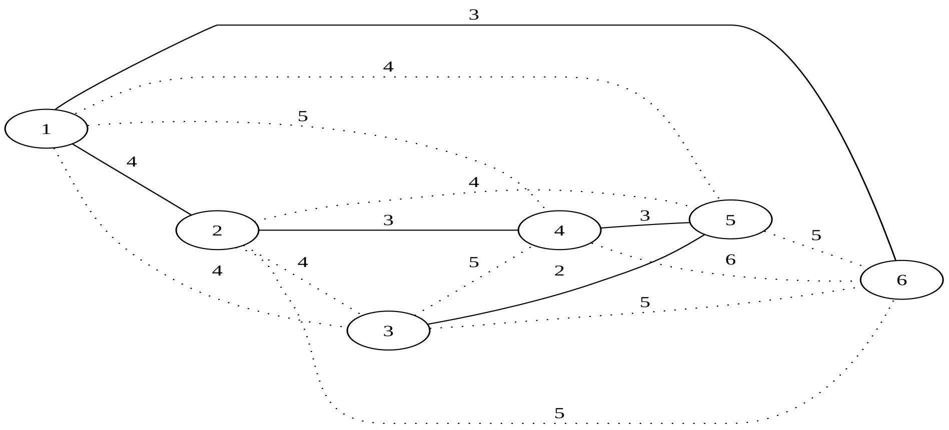
Chapter 11

Easy modelling problems: solutions

11.1 Compact storage of similar sequences: Solution

Consider a complete undirected graph $G = (V, E)$ where each vertex is a sequence and the weight of an edge $\{i, j\} \in E$ is given by the Hamming distance between sequence i and sequence j. To each edge $\{i, j\} \in E$ we also associate the sequence of bit flips necessary to transform sequence i into sequence j. A minimum cost spanning tree in G provides the most economical way to recover all possible sequences starting from only one of these sequences.

The instance in the exercise yields a minimum spanning tree having cost 15.



11.2 Communication of secret messages: Solution

The communication network is represented by a directed graph $G = (V, A)$. Each arc (i, j) is weighted by its probability $1 - p_{ij}$ that the the message is not intercepted along the arc. In order to broadcast the message to all nodes we want to find a subset of arcs which is connected, reaches all nodes, and has no cycle (otherwise the interception probability might increase). In other words, a spanning tree. The spanning tree T should maximize the chances that the message arrives at each node without interception, i.e.:

$$\max_{\text{all } T}\{\prod_{\{i,j\}\in T}(1-p_{ij}) \mid T \text{ spanning tree}\}. \tag{11.1}$$

Since the Prim (and Kruskal) algorithms for finding optimum spanning trees deal with the case when the cost of the tree is the sum of the costs of the edges, we cannot use those algorithms to solve the problem.

However, we can reformulate the problem by requiring the spanning tree T which maximizes the modified objective function $\log\prod_{\{i,j\}\in T}(1-p_{ij})$. This will change the value of the objective function associated to the solution but not the solution itself, since the log function is monotonic increasing.

$$\begin{aligned}
\log & \max_{\text{all } T}\{\prod_{\{i,j\}\in T}(1-p_{ij}) \mid T \text{ spanning tree}\} = \\
= & \max_{\text{all } T}\{\log\prod_{\{i,j\}\in T}(1-p_{ij}) \mid T \text{ spanning tree}\} = \\
= & \max_{\text{all } T}\{\sum_{\{i,j\}\in T}\log(1-p_{ij}) \mid T \text{ spanning tree}\}.
\end{aligned}$$

The latter is a "proper" minimum spanning tree problem on the graph G where each arc $(i,j)\in A$ is weighted by $\log(1-p_{ij})$, and can be solved using either Prim's algorithm.

11.3 Mixed production: Solution

11.3.1 Formulation

- *Indicex*: Let i be an index on the set $\{1,2,3\}$.
- *Parameters*:
  - P: number of production days in a month;
  - d_i: maximum market demand for product i;
  - v_i: selling price for product i;
  - c_i: production cost for product i;
  - q_i: maximum production quota for product i;
  - a_i: activation cost for the plant producing i;
  - l_i: minimum batch of product i.
- *Variables*:
  - x_i: quantity of product i to produce ($x_i \geq 0$);
  - y_i: activation status of product i (1 if active, 0 otherwise).
- *Objective function*:

$$\max\sum_i((v_i-c_i)x_i-a_iy_i)$$

- *Constraints*:
  1. (demand): for each i, $x_i \leq d_i$;
  2. (production): $\sum_i \frac{x_i}{q_i} \leq P$;
  3. (activation): for each i, $x_i \leq Pq_iy_i$;
  4. (minimum batch): for each i, $x_i \geq l_iy_i$;

11.3.2 AMPL model, data, run

```
# mixedproduction.mod

set PRODUCTS;

param days >= 0;
param demand { PRODUCTS } >= 0;
param price { PRODUCTS } >= 0;
param cost { PRODUCTS } >= 0;
param quota { PRODUCTS } >= 0;
param activ_cost { PRODUCTS } >= 0;    # activation costs
param min_batch { PRODUCTS } >= 0;     # minimum batches

var x { PRODUCTS } >= 0;               # quantity of product
var y { PRODUCTS } >= 0, binary;       # activation of production lines

maximize revenue: sum {i in PRODUCTS}
((price[i] - cost[i]) * x[i] - activ_cost[i] * y[i]);

subject to requirement {i in PRODUCTS}:
x[i] <= demand[i];

subject to production:
sum {i in PRODUCTS} (x[i] / quota[i]) <= days;

subject to activation {i in PRODUCTS}:
x[i] <= days * quota[i] * y[i];

subject to batches {i in PRODUCTS}:
        x[i] >= min_batch[i] * y[i];


# mixedproduction.dat

set PRODUCTS := A1 A2 A3 ;

param days := 22;
param : demand  price   cost  quota  activ_cost min_batch :=
     A1 5300    124     73.30   500      170000        20
     A2 4500    109     52.90   450      150000        20
     A3 5400    115     65.40   550      100000        16 ;


# mixedproduction.run

model mixedproduction.mod;
data mixedproduction.dat;
option solver cplexstudent;
solve;
display x;
display y;
```

11.3.3 CPLEX solution

.

```
CPLEX 7.1.0: optimal integer solution; objective 220690
```

```
5 MIP simplex iterations
0 branch-and-bound nodes
ampl: display x;
x [*] :=
A1     0
A2  4500
A3  5400
;

ampl: display y;
y [*] :=
A1  0
A2  1
A3  1
;
```

11.4 Production planning: Solution

11.4.1 Formulation

- *Indices*:
  - i: an index on the set $\pi = \{A_1, A_2, A_3\}$;
  - j: an index on the set $\mu = \{1, 2, 3, 4\}$.
- *Parameters*:
  - P_j: number of production days in month j;
  - d_{ij}: maximum demand for product i in month j;
  - v_i: selling price for product i;
  - c_i: production cost of product i;
  - q_i: maximum production quota of product i;
  - a_i: activation cost for production i;
  - l_i: minimum batch for production i;
  - m_i: storage cost for product i;
  - C: storage capacity in number of units.
- *Variables*:
  - x_{ij}: quantity of product i produced during month j;
  - w_{ij}: quantity of product i sold during month j;
  - z_{ij}: quantity of product i stocked during month j;
  - y_{ij}: activation status for production i: (1=active, 0=inactive).

  All variables are constrained to be non-negative. y_{ij} are binary variables.
- *Objective function*:

$$\max \sum_{i=1}^{3} \left(v_i \sum_{j=1}^{4} w_{ij} - c_i \sum_{j=1}^{4} x_{ij} - m_i \sum_{j=1}^{4} z_{ij} - a_i \sum_{j=1}^{4} y_{ij} \right).$$

- *Constraints*:
    1. (requirement): for each i, j: $w_{ij} \leq d_{ij}$;
    2. (production): per each j: $\sum_{i=1}^{3} \frac{x_{ij}}{q_i} \leq P_j$;
    3. (balance): for each i, j: $z_{i,j-1} + x_{ij} = z_{ij} + w_{ij}$;
    4. (capacity): for each j: $\sum_{i=1}^{3} z_{ij} \leq C$;
    5. (activation): for each i, j: $x_{ij} \leq P_j q_i y_{ij}$;
    6. (minimum batch): for each i, j: $x_{ij} \geq l_i y_{ij}$;
    7. (december): for each i: $z_{i0} = 0$.

11.4.2 AMPL model, data, run

```
# productionplan.mod

set PRODUCTS;

param Months;

set MONTHS := 1..Months;
set MONTHS0 := MONTHS union {0};

param days{MONTHS} >= 0;
param demand { PRODUCTS, MONTHS } >= 0;
param price { PRODUCTS } >= 0;
param cost { PRODUCTS } >= 0;
param quota { PRODUCTS } >= 0;
param activation { PRODUCTS } >= 0;
param batch { PRODUCTS } >= 0;
param storage { PRODUCTS } >= 0;
param capacity >= 0;

var x { PRODUCTS, MONTHS } >= 0;
var w { PRODUCTS, MONTHS } >= 0;
var z { PRODUCTS, MONTHS0 } >= 0;
var y { PRODUCTS, MONTHS } >= 0, binary;

maximize revenue:
sum {i in PRODUCTS}
(price[i] * sum {j in MONTHS} w[i,j] -
 cost[i]   * sum {j in MONTHS} x[i,j] -
 storage[i]  * sum {j in MONTHS} z[i,j] -
                activation[i] * sum {j in MONTHS} y[i,j]) ;

subject to requirement {i in PRODUCTS, j in MONTHS}:
w[i,j] <= demand[i,j];

subject to production {j in MONTHS}:
sum {i in PRODUCTS} (x[i,j] / quota[i]) <= days[j];

subject to bilance {i in PRODUCTS, j in MONTHS}:
z[i,j-1] + x[i,j] = z[i,j] + w[i,j];

subject to capacitymag {j in MONTHS}:
sum {i in PRODUCTS} z[i,j] <= capacity;
```

```
subject to active {i in PRODUCTS, j in MONTHS}:
        x[i,j] <= days[j]*quota[i]*y[i,j];

subject to minbatch {i in PRODUCTS, j in MONTHS}:
        x[i,j] >= batch[i]*y[i,j];


# productionplan.dat

set PRODUCTS := A1 A2 A3 ;

param Months := 4 ;

param days :=
     1  23
     2  20
     3  23
     4  22 ;

param demand:  1       2          3          4    :=
        A1     5300   1200      7400        5300
        A2     4500   5400      6500        7200
        A3     4400   6700     12500       13200 ;

param : price   cost quota  activation batch storage :=
     A1     124 73.30 500    150000      20    3.5
     A2     109 52.90 450    150000      20    4
     A3     115 65.40 550    100000      16    3   ;

param capacity := 800 ;

let {i in PRODUCTS} z[i,0] := 0;
fix {i in PRODUCTS} z[i,0];


# productionplan.run

model productionplan.mod;
data productionplan.dat;
option solver cplexstudent;
solve;
option display_round 4;
display revenue;
display x;
display y;
quit;
```

11.4.3 CPLEX solution

```
CPLEX 7.1.0: optimal integer solution; objective 1581550
47 MIP simplex iterations
0 branch-and-bound nodes
guadagno = 1581550.0000

x :=
A1 1    6100.0000
A1 2       0.0000
A1 3       0.0000
```

```
A1 4       0.0000
A2 1       0.0000
A2 2    3518.1818
A2 3       0.0000
A2 4       0.0000
A3 1    4400.0000
A3 2    6700.0000
A3 3   12650.0000
A3 4   12100.0000 ;

y :=
A1 1   1.0000
A1 2   0.0000
A1 3   0.0000
A1 4   0.0000
A2 1   0.0000
A2 2   1.0000
A2 3   0.0000
A2 4   0.0000
A3 1   1.0000
A3 2   1.0000
A3 3   1.0000
A3 4   1.0000 ;
```

11.5 Transportation: Solution

11.5.1 Formulation

- *Indices*:
  - i: index on the set $\{1, \ldots, M\}$ (stores);
  - j: index on the set $\{1, \ldots, P\}$ (ports);
- *Parameters*:
  - m_i: availability (in number of containers) at i-th store;
  - r_j: demand at j-th port;
  - d_{ij}: distance between store i and port j;
  - C: unit transportation cost (per km).
- *Variables*:
  - x_{ij}: number of containers sent from store i to port j;
  - y_{ij}: number of lorries travelling from store i to port j;

  All variables are constrained to be non-negative.
- *Objective function*:

$$\min \sum_{i=1}^{M} \sum_{j=1}^{P} C d_{ij} y_{ij}$$

- *Constraints*:
  1. (store availability) for each $i \leq M$: $\sum_{j=1}^{P} x_{ij} \leq m_i$;
  2. (port demand) for each $j \leq P$: $\sum_{i=1}^{M} x_{ij} \geq r_j$;
  3. (lorry capacity) for each $i \leq M$, $j \leq P$, $2y_{ij} \geq x_{ij}$.

11.5.2 AMPL model, data, run

```
# transportation.mod

set STORES;
set PORTS;

param availability { STORES } >= 0;
param demand { PORTS } >= 0;
param distance { STORES, PORTS } >= 0;
param costkm >= 0;

var x { STORES, PORTS } >= 0;
var y { STORES, PORTS } >= 0, integer;

minimize cost:
sum {i in STORES, j in PORTS} costkm * distance[i,j] * y[i,j];


subject to avail {i in STORES}:
sum {j in PORTS} x[i,j] <= availability[i];

subject to request {j in PORTS}:
sum {i in STORES} x[i,j] >= demand[j];

subject to lorrycap {i in STORES, j in PORTS}:
        2*y[i,j] >= x[i,j];


# transportation.dat

set STORES := Verona Perugia Rome Pescara Taranto Lamezia;
set PORTS := Genoa Venice Ancona Naples Bari;

param availability :=
        Verona   10
        Perugia  12
        Rome     20
        Pescara  24
        Taranto  18
        Lamezia  40 ;

param demand :=
        Genoa    20
        Venice   15
        Ancona   25
        Naples   33
        Bari     21 ;

param distance :
                 Genoa  Venice  Ancona   Naples  Bari :=
        Verona    290     115    355      715     810
        Perugia   380     340    165      380     610
        Rome      505     530    285      220     450
        Pescara   655     450    155      240     315
        Taranto  1010     840    550      305      95
        Lamezia  1072    1097    747      372     333 ;

param costkm := 300;
```

```
# transportation.run

model transportation.mod;
data transportation.dat;
option solver cplexstudent;
solve;
option display_round 4;
display cost;
display x;
display y;
```

11.5.3 CPLEX solution

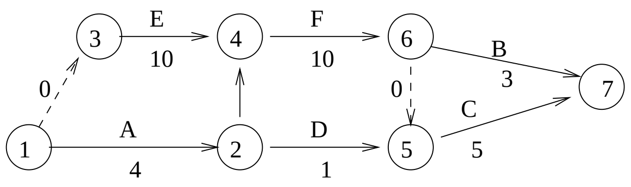
```
CPLEX 7.1.0: optimal integer solution; objective 4685100
70 MIP simplex iterations
0 branch-and-bound nodes
costo = 4685100.0000

x [*,*]
:         Ancona     Bari     Genova    Napoli    Venezia    :=
Lamezia    0.0000    4.0000    0.0000   26.0000    0.0000
Perugia    1.0000    0.0000    6.0000    0.0000    5.0000
Pescara   24.0000    0.0000    0.0000    0.0000    0.0000
Roma       0.0000    0.0000   14.0000    6.0000    0.0000
Taranto    0.0000   17.0000    0.0000    1.0000    0.0000
Verona     0.0000    0.0000    0.0000    0.0000   10.0000
;

y [*,*]
:         Ancona     Bari    Genova   Napoli   Venezia    :=
Lamezia    0.0000    2.0000   0.0000   13.0000   0.0000
Perugia    1.0000    0.0000   3.0000    0.0000   3.0000
Pescara   12.0000    0.0000   0.0000    0.0000   0.0000
Roma       0.0000    0.0000   7.0000    3.0000   0.0000
Taranto    0.0000    9.0000   0.0000    1.0000   0.0000
Verona     0.0000    0.0000   0.0000    0.0000   5.0000
;
```

11.6 Project planning with precedences: Solution

The precedence graph $G = (V, A)$ (which associates to each arc an activity) is as follows.



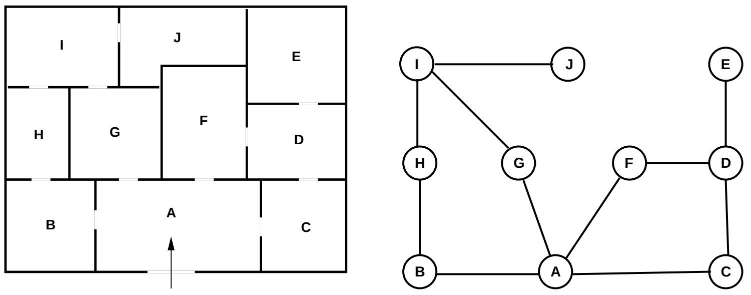
To each vertex $i \in V$ we associate a variable t_i (the starting time of the activities represented by arcs in $\bar{\delta}^+(i)$. The mathematical programming formulation of the problem is:

$$\begin{aligned} \min \quad & t_7 - t_1 + 5000(t_4 - t_2) \\ & t_i + d_{ij} \leq t_j \quad \forall\, (i,j) \in A, \end{aligned}$$

where d_{ij} is the cost of the arc (i,j).

11.7 Museum guards: Solution

The problem can be formalized by representing each museum room by a vertex $v \in V$ of an undirected graph $G = (V, E)$. There is an edge between two vertices if there is a door leading from one room to the other; this way, edges represent the possibility of there being a guard on a door. We want to choose the smallest subset $F \subseteq E$ of edges *covering* all vertices, i.e. such that for all $v \in V$ there is $w \in V$ with $\{v, w\} \in F$.



To each $\{i,j\} \in E$ we associated a binary variable x_{ij} is assigned the value 1 if there is a guard on the door represented by edge $\{i,j\}$ and 0 otherwise.

11.7.1 Formulation

- *Parameters.* $G = (V, A)$: graph description of the museum topology.
- *Variables.* x_{ij}: 1 if edge $\{i,j\} \in E$ is to be included in F, 0 otherwise.
- *Objective function*
$$\min \sum_{\{i,j\} \in E} x_{ij}$$
- *Constraints.* (Vertex cover): $\sum\limits_{j \in V : \{i,j\} \in E} x_{ij} \geq 1 \quad \forall i \in V.$

11.7.2 AMPL model, data, run

```
# museum.mod

param n >= 0, integer;
set V := 1..n;
set E within {V,V};
var x{E} binary;
```

```
minimize cost : sum{(i,j) in E} x[i,j];
subject to vertexcover {i in V} :
  sum{j in V : (i,j) in E} x[i,j] + sum{j in V : (j,i) in E} x[j,i] >= 1;
```

```
# museum.dat

param n := 10;
set E :=
  1 2
  1 3
  1 6
  1 7
  2 8
  3 4
  4 5
  7 9
  8 9
  9 10 ;
```

```
# museum.run

model museum.mod;
data museum.dat;
option solver cplexstudent;
solve;
display cost;
display x;
```

11.7.3 CPLEX solution

```
CPLEX 7.1.0: optimal integer solution; objective 6
2 MIP simplex iterations
0 branch-and-bound nodes
cost = 6

x :=
1 2    0
1 3    1
1 6    1
1 7    1
2 8    1
3 4    0
4 5    1
7 9    0
8 9    0
9 10   1
;
```

11.8 Inheritance: Solution

The problem may be formalized as follows: given a set A of n elements each with an evaluation function $v : A \to \mathbb{R}$, we want to find a partition of A in A_1, A_2 such that

$$|v(A_1) - v(A_2)| = |\sum_{a\in A_1} v(a) - \sum_{a\in A_2} v(a)|$$

is minimum. This is known as the SUBSET-SUM problem.

It can be modelled using mathematical programming by introducing binary variables x_a, y_a for each $a \in A$, such that $x_a = 1$ and $y_a = 0$ if object a is assigned to brother x, and $x_a = 0$ and $y_a = 1$ if a is assigned to y. We naturally need the constraint

$$\forall a \in A \quad (x_a + y_a = 1).$$

The objective function to be minimized is:

$$\min |\sum_{a\in A_1} v_a x_a - \sum_{a\in A_2} v_a y_a|,$$

which ensures that the inheritance is split between the two brothers as fairly as possible. Because of the absolute value, this formulation is nonlinear.

Let $V = \sum_{a\in A} v(a)$ be the total value of the inheritance. The SUBSET-SUM can also be described as follows:

- maximize the inheritance assigned to one of the brothers with the constraint that it should not exceed $V/2$;
- minimize the inheritance assigned to one of the brothers with the constraint that it should not be less than $V/2$.

This interpretation gives us two integer linear programming formulations:

$$\left.\begin{array}{rlcl} \max & \sum\limits_{a\in A} v_a x_a & & \\ \text{s.t.} & \sum\limits_{a\in A} v_a x_a & \leq & \frac{V}{2} \sum\limits_{a\in A} v_a \\ \forall a \in A & x_a \in \{0,1\} & & \end{array}\right\}$$

$$\left.\begin{array}{rlcl} \min & \sum\limits_{a\in A} v_a x_a & & \\ \text{s.t.} & \sum\limits_{a\in A} v_a x_a & \geq & \frac{V}{2} \sum\limits_{a\in A} v_a \\ \forall a \in A & x_a \in \{0,1\} & & \end{array}\right\}$$

11.8.1 AMPL model, data, run

```
# subsetsum.mod

param n;
param v {1..n};
param V := sum {i in 1..n} v [i];
var x {1..n} binary;
minimize cost: sum {i in 1..n} v[i] * x[i];
subject to limit: sum {i in 1..n} v [i]* x[i] >= 0.5 * V;
```

```
# subsetsum.dat

param n := 13;
param: v :=
  1 25000
  2 5000
  3 20000
  4 40000
  5 12000
  6 12000
  7 12000
  8 3000
  9 6000
  10 10000
  11 15000
  12 10000
  13 13000;
```

```
# subsetsum.run

model subsetsum.mod;
data subsetsum.dat;
option solver cplexstudent;
solve;
display cost;
display x;
```

11.8.2 CPLEX solution

```
CPLEX 8.1.0: optimal integer solution; objective 92000
7 MIP simplex iterations
0 branch-and-bound nodes
cost = 92000

x [*] :=
 1  1
 2  0
 3  1
 4  0
 5  0
 6  0
 7  0
 8  1
 9  1
10  0
11  1
12  1
13  1
;
```

11.9 Carelland: Solution

Miximize the profits (exported quantities - produced quantities) subject to the constraints on production, amount of work and balance between produced and exported products.

11.9.1 Formulation

Parameters:

- P: set of products;
- H: total available amount of work (man-years);
- M_i maximum possible production for product $i \in P$;
- p_i market price for product $i \in P$;
- m_i amount of raw materials necessary to manufacture a unit of product $i \in P$;
- h_i amount of work required to manufacture a unit of product $i \in P$;

Variabili:

- x_a, x_m, x_p, x_e: produced units of steel, engines, plastics and electronics
- y_a, y_m, y_p, y_e: exported units of steel, engines, plastics and electronics.

Model:

$$\begin{aligned}
\max \sum_{i\in P} p_i y_i - \sum_{i \in P} m_i x_i & \\
\sum_{i\in P} h_i x_i \le H & \\
x_i \le M_i & \qquad \forall i \in P \\
y_a + 0.8x_m + 0.01x_e + 0.2x_p = x_a & \\
y_m + 0.02x_a + 0.01x_e + 0.03x_p = x_m & \\
y_e + 0.15x_m + 0.05x_p = x_e & \\
y_p + 0.01x_a + 0.11x_m + 0.05x_e = x_p & \\
x_i, y_i \ge 0 & \qquad \forall i \in P
\end{aligned}$$

11.9.2 AMPL model, data, run

```
# carelland.mod

set PRODUCTS;

param p {PRODUCTS} >= 0;
param HMan >=0;
param Max {PRODUCTS} >=0;
param m {PRODUCTS} >= 0;
param h {PRODUCTS} >= 0;
param a {PRODUCTS, PRODUCTS} >=0;


var x { PRODUCTS } >= 0;
var y { PRODUCTS } >= 0;
```

```
maximize klunz:
sum {i in PRODUCTS} (p[i]*y[i] - m[i]*x[i]);

subject to limit{i in PRODUCTS}:
x[i] <= Max[i];

subject to work:
sum {i in PRODUCTS} h[i]*x[i]<=HMan;


subject to balance{i in PRODUCTS} :
y[i] + sum{j in PRODUCTS}(a[j,i]*x[j]) = x[i];
```

```
# carelland.dat

set PRODUCTS := steel plastics electronics engines;

param HMan:= 830000;
param :             p          m     h        Max     :=
     steel        500        250     0.5     2000000
     plastics    1200        300     2         60000
     electronics  300         50     0.5      650000
     engines     1500        300     1       2000000  ;

param a:       steel           plastics         electronics        engines :=
steel           0                0.01             0                0.02
plastics        0.2              0                0.05             0.03
electronics     0.01             0.05             0                0.01
engines         0.8              0.11             0.15             0 ;
```

```
# carelland.run

model carelland.mod;
data carelland.dat;
option solver cplexstudent;
solve;
display profit;
display x;
display y;
```

11.9.3 CPLEX solution

```
CPLEX 8.1.0: optimal solution; objective 435431250
9 dual simplex iterations (6 in phase I)
klunz = 435431000

x [*] :=
electronics   74375
    engines  475833
   plastics   60000
      steel  393958
;
```

```
y [*] :=
electronics       0
    engines  465410
   plastics       0
      steel     547.917
;
```

11.10 CPU Scheduling: Solution

- *Indices*:
  - i, j: indices on a set P of tasks;
  - k: index on a set C of CPUs.
- *Parameters*:
  - b_i: number of BI (billion instructions) in task i;
  - s_k: speed of CPU k in GHz;
  - $W_{\max}$: upper bound for completion time of all tasks.
- *Variables*:
  - $x_i \geq 0$: starting time of task i;
  - $y_i \in \mathbb{Z}_+$: CPU ID to which task i is assigned;
  - $z_{ik} = 1$ if task i is assigned to CPU k, 0 otherwise;
  - $\sigma_{ij} = 1$ if task i ends before task j starts, 0 otherwise;
  - $\varepsilon_{ij} = 1$ if task i is executed on a CPU having lower ID than task j;
  - $L_i \geq 0$: length of task i;
  - $W \geq 0$: completion time of all tasks.
- *Objective function*:

$$\min W$$

- *Constraints*:
  - (lengths) $\forall i \in P \ (L_i = \sum_{k \in C} \frac{b_i}{s_k} z_{ik})$;
  - (times) $\forall i \in P \ (t_i + L_i \leq W)$
  - (assignment) $\forall i \in P \ (\sum_{k \in C} z_{ik} = 1)$;
  - (cpudef) $\forall i \in P \ (y_i = \sum_{k \in C} k z_{ik})$
  - (horizontal non-overlapping) $\forall i \neq j \in P \ (x_j - x_i - L_i - (\sigma_{ij} - 1) W_{\max} \geq 0)$
  - (vertical non-overlapping) $\forall i \neq j \in P \ (y_j - y_i - 1 - (\varepsilon_{ij} - 1)|P| \geq 0)$
  - (at least one position) $\forall i \neq j \in P \ (\sigma_{ij} + \sigma_{ji} + \varepsilon_{ij} + \varepsilon_{ji} \geq 1)$
  - (horizontal: at most one) $\forall i \neq j \in P \ (\sigma_{ij} + \sigma_{ji} \leq 1)$
  - (vertical: at most one) $\forall i \neq j \in P \ (\varepsilon_{ij} + \varepsilon_{ji} \leq 1)$

11.10.1 AMPL model, data, run

```
# cpuscheduling.mod

param p > 0, integer;
param c > 0, integer;
set P := 1..p;
set C := 1..c;

param b{P} >= 0;
param s{C} >= 0;
param Wmax default sum{i in P} b[i] / (min{k in C} s[k]);

var x{P} >= 0;
var y{P} >= 0;
var z{P,C} binary;
var sigma{P,P} binary;
var epsilon{P,P} binary;
var L{P} >= 0;
var W >= 0;

minimize makespan: W;

subject to lengths{i in P} : L[i] = sum{k in C} (b[i] / s[k]) * z[i,k];

subject to times{i in P} : x[i] + L[i] <= W;

subject to assignment{i in P} : sum{k in C} z[i,k] = 1;

subject to cpudef{i in P} : y[i] = sum{k in C} k * z[i,k];

subject to hnonoverlapping{i in P, j in P : i != j} :
  x[j] - x[i] - L[i] - (sigma[i,j] - 1) * Wmax >= 0;

subject to vnonoverlapping{i in P, j in P : i != j} :
  y[j] - y[i] - 1 - (epsilon[i,j] - 1) * p >= 0;

subject to atleastone{i in P, j in P : i != j} :
  sigma[i,j] + sigma[j,i] + epsilon[i,j] + epsilon[j,i] >= 1;

subject to hatmostone{i in P, j in P : i != j} :
  sigma[i,j] + sigma[j,i] <= 1;

subject to vatmostone{i in P, j in P : i != j} :
  epsilon[i,j] + epsilon[j,i] <= 1;


# cpuscheduling.dat

param p := 7;
param c := 3;

param : b :=
1 1.1
2 2.1
3 3.0
4 1.0
5 0.7
6 5.0
```

```
7 3.0 ;

param : s :=
1 1.33
2 2.00
3 2.66 ;


# cpuscheduling.run

model cpuscheduling.mod;
data cpuscheduling.dat;
option solver cplexstudent;
solve;
display makespan;
for{k in C} {
  printf "CPU %d : ", k;
  for{i in P : z[i,k] = 1} {
    printf "[%d:%f] ", i, x[i];
  }
  printf "\n";
}
```

11.10.2 CPLEX solution

```
CPLEX 8.1.0: optimal integer solution; objective 2.781954887
40175 MIP simplex iterations
8463 branch-and-bound nodes
makespan = 2.78195

CPU 1 : [5:0.000000] [7:0.526316]
CPU 2 : [1:1.731955] [3:0.000000] [4:2.281955]
CPU 3 : [2:0.000000] [6:0.789474]
```

11.11 Dyeing plant: Solution

- *Indices*:
  - i, j: index on the set L of fabric batches;
  - k: index on the set $V = \{1, \dots, v\}$ of dyeing baths;
- *Parameters*:
  - s_{ik}: time necessary to dye batch i in bath k;
  - M: upper bound to completion time of last bath.
- *Variables*:
  - $t_{ik} \geq 0$: starting time for dyeing batch i in bath k;
  - $T \geq 0$: completion time for last batch;
  - $y_{ijk} = 1$ if batch i is to be dyed before batch j in bath k, 0 otherwise.
- *Objective function*:

$$\min T$$

- *Constraints*:

- (sequential) $\forall i \in L, k \in V \smallsetminus \{v\}$ $(t_{ik} + s_{ik} \le t_{i(k+1)})$;
- (last bath) $\forall i \in L$ $(t_{iv} + s_{iv} \le T)$;
- (non overlapping) $\forall i, j \in L, k \in V, i \ne j$ $(t_{ik} + s_{ik} \le t_{jk} + M(1 - y_{ijk}))$;
- (disjunction) $\forall i, j \in L, k \in V, i \ne j$ $(y_{ijk} + y_{jik} = 1)$.

11.11.1 AMPL model, data, run

```
# dyeing.mod

param l >= 1;
param v >= 1;

set L := 1..l;
set V := 1..v;
set V0 := 1..v-1;

param s{L,V} >= 0;
param M default sum{i in L, k in V} s[i,k] ;

var t{L,V} >= 0;
var T >= 0;
var y{L,L,V} binary;

minimize makespan : T;

subject to sequential{i in L, k in V0} : t[i,k] + s[i,k] <= t[i,k+1];

subject to lastbath{i in L} : t[i,v] + s[i,v] <= T;

subject to nonoverlap{i in L, j in L, k in V : i != j} :
  t[i,k] + s[i,k] <= t[j,k] + M * (1 - y[i,j,k]);

subject to disjunction{i in L, j in L, k in V : i != j} :
  y[i,j,k] + y[j,i,k] = 1;


# dyeing.dat

param l := 5;
param v := 3;

param s :  1   2   3 :=
1         3.0 1.0 1.0
2         2.0 1.5 1.0
3         3.0 1.2 1.3
4         2.0 2.0 2.0
5         2.1 2.0 3.0 ;


# dyeing.run

model dyeing.mod;
data dyeing.dat;
option solver cplexstudent;
solve;
display makespan;
for {i in L} {
  printf "batch %d : ", i;
  for {k in V} {
    printf "[%f] ", t[i,k];
  }
  printf "\n";
}
```

11.11.2 CPLEX solution

```
CPLEX 8.1.0: optimal integer solution; objective 14.1
1618 MIP simplex iterations
362 branch-and-bound nodes
makespan = 14.1

batch 1 : [9.100000] [12.100000] [13.100000]
batch 2 : [7.100000] [9.100000] [12.100000]
batch 3 : [4.100000] [7.100000] [8.300000]
batch 4 : [2.100000] [4.100000] [9.600000]
batch 5 : [0.000000] [2.100000] [4.100000]
```

11.12 Parking: Solution

- *Indices*:
  - i: index on the set $N = \{1, \ldots, n\}$ of cars;
  - j: index on the set $M = \{1, 2\}$ of car lines (one per street side).
- *Parameters*:
  - λ_i: length of car i;
  - L: upper bound on the car line length;
  - μ: upper bound for the sum of car lengths.
- *Variables*:
  - $x_{ij} = 1$ if i is parked on line j and 0 otherwise;
  - $t_j \geq 0$: length of car line j;
  - $y_j = 1$ if $t_j \leq L$ and 0 otherwise.
- *Objective function*:
$$\min \max_{j \in M} t_j.$$
- *Constraints*:
  - (car line length definition) $\forall j \in M\ (t_j = \sum_{i \in N} \lambda_i x_{ij})$;
  - (assignment of cars to lines) $\forall i \in N\ (\sum_{j \in M} x_{ij} = 1)$;
  - (constraint disjunction) $\forall j \in M\ (t_j - L \leq \mu(1 - y_j))$;
  - (constraint on one line only) $(\sum_{j \in M} y_j = 1)$.

11.12.1 AMPL model, data, run

```
# parking.mod

param n > 0;
param m > 0;

set N := 1..n;
set M := 1..m;

param lambda{N} >= 0;
param mu := sum{i in N} lambda[i];
param L >= 0;

var x{N,M} binary;
var t{N} >= 0;
var y{M} binary;
var T >= 0;

minimize minmaxobj: T;

subject to minmax {j in M} : T >= t[j];

subject to carlinedef {j in M} :
  t[j] = sum{i in N} lambda[i] * x[i,j];

subject to assignment {i in N} : sum{j in M} x[i,j] = 1;

subject to disjunction {j in M} : t[j] - L <= mu * (1 - y[j]);

subject to onelineonly : sum{j in M} y[j] = 1;
```

```
# parking.dat

param n := 15;
param m := 2;
param L := 15;

param : lambda :=
1  4.0
2  4.5
3  5.0
4  4.1
5  2.4
6  5.2
7  3.7
8  3.5
9  3.2
10 4.5
11 2.3
12 3.3
13 3.8
14 4.6
15 3.0 ;
```

```
# parking.run

model parking.mod;
data parking.dat;
option solver cplexstudent;
solve;
display minmaxobj;
for {j in M} {
  printf "line %d (length = %f) : ", j, sum{i in N : x[i,j] = 1} lambda[i];
  for {i in N : x[i,j] = 1} {
    printf "%d ", i;
  }
  printf "\n";
}
```

11.12.2 CPLEX solution

```
CPLEX 8.1.0: optimal integer solution; objective 42.1
56 MIP simplex iterations
50 branch-and-bound nodes
minmaxobj = 42.1

line 1 (length = 42.100000) : 2 3 4 5 6 7 8 11 13 14 15
line 2 (length = 15.000000) : 1 9 10 12
```