Department of Mathematics and Computer Science
University of Southern Denmark, Odense

March 29, 2021
Marco Chiarandini

DM545/DM871 – Linear and integer programming

Sheet 8, Spring 2021

Starred exercises are more relevant for the exam.

Exercise 1* Scheduling on Uniform Parallel Machines

We consider scheduling a set J of jobs on M uniform parallel machines. Each job $j \in J$ has a processing requirement p_j (denoting the number of machine days required to complete the job), a release data r_j (representing the beginning of the day when job j become available for processing), and a due date $d_j \geq r_j + p_j$ (representing the beginning of the day by which the job must be completed). We assume that a machine can work on only one job at a time and that each job can be processed by at most one machine at a time. However we allow preemptions (ie, we can interrupt a job and process it on different machines on different days). The scheduling problem is to determine a feasible schedule that completes all jobs before their due dates or to show that no such schedule exists.
Formulate the feasible scheduling problem as a maximum flow problem.

Solution:
[AMO] pages 170-176.

Exercise 2* Tanker Scheduling Problem

A steamship company has contracted to deliver perishable goods between several different origin-destination pairs. Since the cargo is perishable the customers have specified precise dates (ie, delivery dates) when the shipments must reach their destinations. (The cargoes may not arrive early or late). The steamship company wants to determine the minimum number of ships needed to meet the delivery dates of the shiploads.
Formulate this problem as a maximum flow problem modeling the example in Table 1 with four shipments. Each shipment is a full shipload with the characteristics shown in Table 1. For example, as specified by the first row in this figure, the company must deliver one shipload available at port A and destined for port C on day 3.

ship-ment	origin	desti-nation	delivery date
1	Port A	Port C	3
2	Port A	Port C	8
3	Port B	Port D	3
4	Port B	Port C	6

	C	D
A	3	2
B	2	3

	A	B
C	2	1
D	1	2

Table 1: Data for the tanker scheduling problem: Left shipment characteristics; Center, shipment transit times; Right return times.

Solution:
[AMO] pages 170-176.

Exercise 3* Directed Chinese Postman Problem

Suppose a postman has to deliver mail along all the streets in a small town. Assume furthermore that on one-way streets the mail boxes are all on one side of the street, whereas for two-way streets, there are mail boxes on both sides of the street. For obvious reasons the postman wishes to minimize the distance he has to travel in order to deliver all the mail and return home to his starting point. Show how you can solve this problem using minimum cost flows. A similar model can be formulated for the Snow Plow problem or the Salt Spreding problem.

Solution:

The solution to this problem can be found on page 174 of J. Bang-Jensen and G. Gutin. Digraphs: Theory, Algorithms and Applications, Springer London, 2009 http://dx.doi.org/10.1007/978-1-84800-998-1_4.

In short, the solution to the Chinese postman problem is a min cost flow that traverses each arc at least once in a network where arcs are streets and nodes are street instersections.

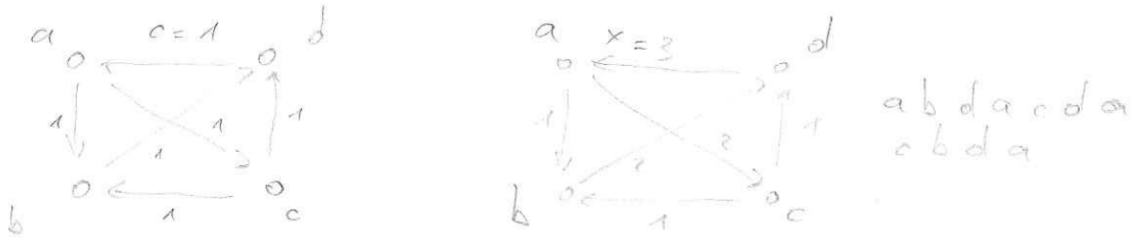
We must assume that the graph is strongly connected otherwise there is no solution (any closed walk is strongly connected.)

A digraph where there exists a walk that visits arcs exactly once is called Eulerian.

$$N = (V, A, \mathbf{l}, \mathbf{u} = \infty, \mathbf{c})$$

$$x_{ij} = \#\text{ of times arc is used}$$

The cost os a min cost circulation in N equals the cost of a Chinese postman walk in D.

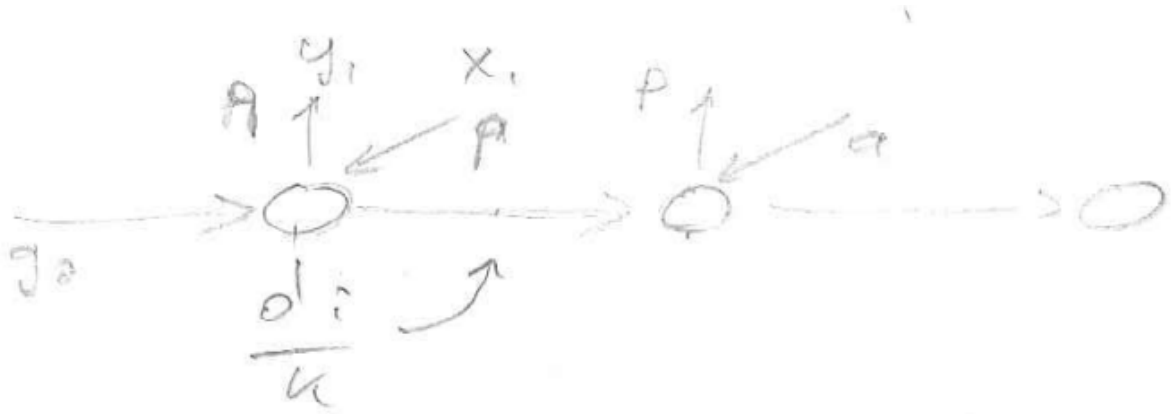


Exercise 4*

The production plan of a factory for the next year is to produce d_t units of product per month t, $t = 1, \ldots, 12$. Each worker can produce k units of product in a month. The monthly salary is equal to s. Employing and firing personnel has costs: precisely, employing one person costs p while firing one costs q. Assuming that initially there are g_0 workers, determine the number of workers that must be present during every month such that the demand is always satisfied and the overall costs of salary, employment, and firing are minimized.

Solution:

It is possible to model the employment and firing of workers as a flow in a network.



Exercise 5* Warehousing of Seasonal Products

A company manufactures multiple products. The products are seasonal with demand varying weekly, monthly, or quarterly. To use its work-force and capital equipment efficiently, the company wishes to "smooth" production, storing pre-season production to supplement peak-season production. The company has a warehouse with fixed capacity R that it uses to store all the products it produces. Its decision problem is to identify the production levels of all the products for every week, month, or quarter of the year that will permit it to satisfy the demands incurring the minimum possible production and storage costs.

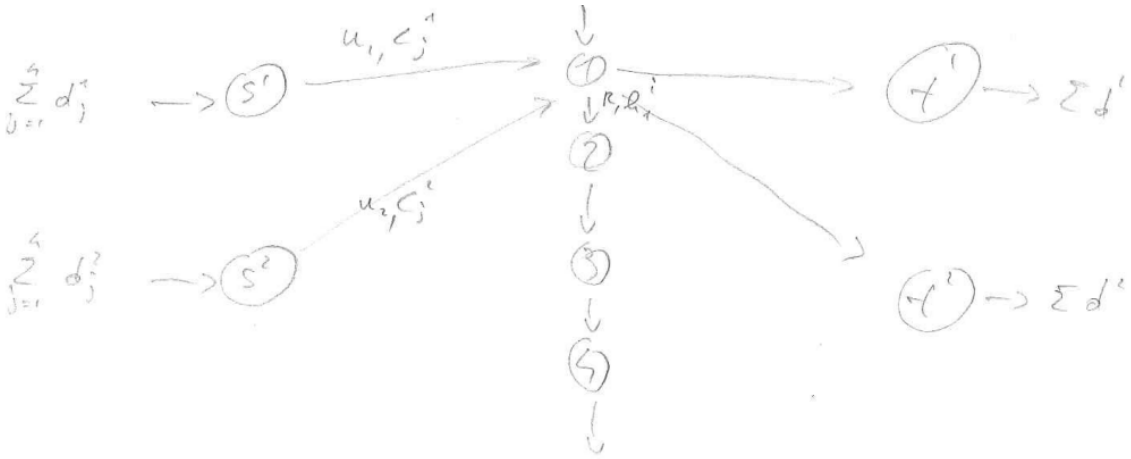
We can represent this warehousing problem as a relevant generalization of the min cost network flow problem encountered in the course.

For simplicity, consider a situation in which the company makes two products and then it needs to schedule its production for each of the next four quarters of the year. Let d_j^1 and d_j^2 denote the demand for products 1 and 2 in quarter j. Suppose that the production capacity for the jth quarter is u_j^1 and u_j^2, and that the per unit cost of production for this quarter is c_j^1 and c_j^2. Let h_j^1 and h_j^2 denote the storage (holding) costs per unit of the two products from quarter j to quarter $j+1$.

Represent graphically the network in the two products four periods case and write the Linear Programming formulation of the problem. Which network flows problem models this application? If all input data are integer, will the solution be integer?

Solution:

We can model this problem as a multicommodity flow in a network. The network has 4 nodes, one for each quarter, two target nodes for each quarter (one per product), two nodes for each plant and for each quarter (one per product) and two global sources for the two products.



See page 655 of [AMO].

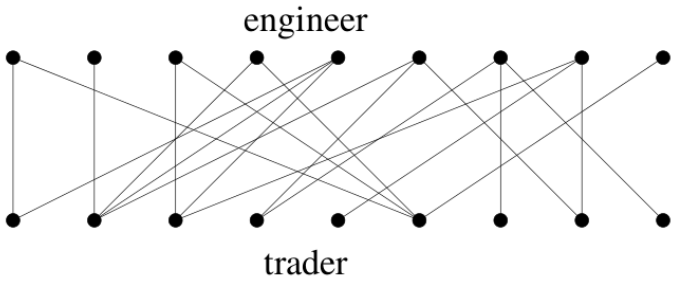


Figure 1:

Exercise 6[Goe11]

A managing director has to launch the marketing of a new product. Several candidate products are at his disposal and he has to choose the best one. Hence, he let each of these products be analysed by a team made of an engineer and a trader who write a review together. The teams are made along the graph in Figure 1; each edge corresponds to a product and its endvertices to the engineer and trader examining it.

a) How many people at least does the managing director gather in order to have the report on all the products? (The report can be given by either the engineer or the trader.)

b) Assuming now that the report must be done jointly by an engineering and a trader, and that each engineer and trader can be occupied with only one candidate product, give a polynomial time algorithm to identify which products will for sure not have the possibility to obtain a report.

Solution:

a) This is an application of the vertex cover problem and its strong duality with maximum matching.
b) This can be done by finding all maximum matching of the graph. The edges that are never in a matching are those that will be never reviewed. We could solve $|E|$ linear programs formulations of the max matching problem for bipartite graphs in each of which a different edge is enforced to be in the solution. Other, more efficient methods based on direct algorithms exist.

Exercise 7 [DT97]

Suppose that in a minimum cost flow problem restrictions are placed on the total flow leaving a node k, i.e.

$$\underline{\theta}_k \leq \sum_{(k,j)\in E} x_{kj} \leq \overline{\theta}_k$$

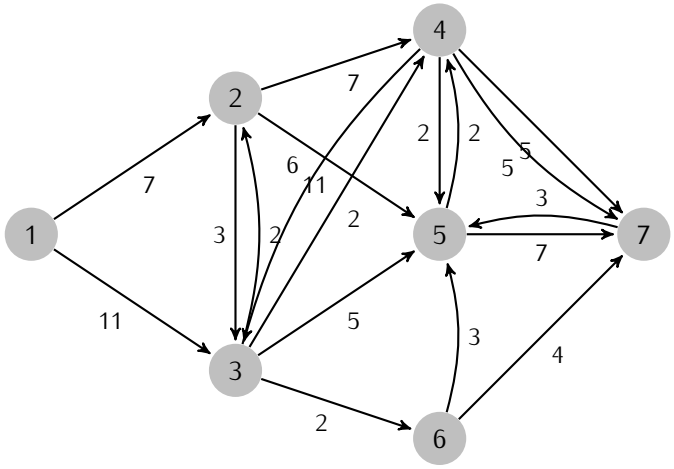
Show how to modify these restrictions to convert the problem into a standard cost flow problem.

Solution:

Bounds on nodes are not possible in our model of min cost flow. However, we can transform the network by splitting the vertex in two and introducing an arc between the new vertices with the given bounds.

Exercise 8*

Given the Network in Figure 2, determine the max flow and indicate the min cut.




```
\begin{tikzpicturehevea}[scale=0.9, auto,swap]
\tikzstyle{vertex}=[circle,fill=black!25,minimum size=20pt,inner sep=0pt]
\tikzstyle{selected vertex} = [vertex, fill=red!24]
\tikzstyle{edge} = [draw,thick,-]
\tikzstyle{arc} = [draw,thick,->,shorten >=1pt,>=stealth']
\tikzstyle{arcl} = [draw,thick,->,shorten >=1pt,>=stealth',bend left=25]
\tikzstyle{arcr} = [draw,thick,->,shorten >=1pt,>=stealth']
\tikzstyle{weight} = [font=\small]
\tikzstyle{selected edge} = [draw,line width=5pt,-,red!50]
\tikzstyle{ignored edge} = [draw,line width=5pt,-,black!20]

   % First we draw the vertices
  \foreach \pos/\name in {{(0,3)/1}, {(3,1)/3}, {(3,5)/2},
                          {(6,0)/6}, {(6,3)/5}, {(6,6)/4}, {(9,3)/7}}
       \node[vertex] (\name) at \pos {$\name$};
   % Connect vertices with edges and draw weights
   \foreach \source/ \dest /\weight in {
     1/2/7, 1/3/11, 2/3/3, 3/6/2,
     2/4/7, 2/5/11, 3/4/2, 3/5/5,
     4/7/5, 4/5/2, 5/7/7, 6/7/4}
   \path[arcr] (\source) -- node[weight] {$\weight$} (\dest);
   \foreach \source/ \dest /\weight in {
     3/2/2, 4/3/6, 4/7/5, 5/4/2, 7/5/3, 6/5/3}
       \path[arcl,bend right] (\source) edge [bend right=15] node[weight] {$\weight$} (\
           dest);
\end{tikzpicturehevea}
```

Figure 2: Find the maximum flow from 1 to 7. Numbers on arcs are capacity values. [In preparation for the exam, below the graph you find the excerpt of latex code to produce the picture. You can use it to experiment whether its use is fast enough for an exam session.]

Solution:

See https://dm871.github.io/assets/net_flow.html

Exercise 9*

Consider the following IP problem:

$$\begin{array}{ll} \max & 4x_1 + 7x_2 \\ s.t. & x_1 + 3x_2 \leq 12 \\ & 4x_1 + 6x_2 \leq 27 \\ & 4x_1 + 2x_2 \leq 20 \\ & x_1, x_2 \geq 0, x_1, x_2 \in \mathbb{Z} \end{array} \tag{1}$$

Subtask a

Give a heuristic primal bound and describe how you determined it.

Solution:

$x = [0, 0]$ is feasible because it satisfies all constraints and has value $z = 0$. This is a lower bound to the optimal solution.

Subtask b

Write the LP relaxation (2lp) of (2) to obtain a dual bound. Explain the relation between the optimal solution of (2lp) and the optimal solution of (2).

Solution:

We relax x_1 and x_2. The problem (2lp) becomes:

$$\begin{array}{ll} \max & z_{LP} = 4x_1 + 7x_2 \\ s.t. & x_1 + 3x_2 \leq 12 \\ & 4x_1 + 6x_2 \leq 27 \\ & 4x_1 + 2x_2 \leq 20 \\ & x_1, x_2 \geq 0 \end{array} \tag{2}$$

(2lp) gives an upper bound to the problem (2).

Subtask c

Write the first simplex tableau of (2lp) and indicate which variables constitute a basic solution. Call s_1, s_2, s_3 the slack variables.

Subtask d

Explain which variable leaves the basis and which variable enters the basis in the first iteration of the simplex algorithm with largest coefficient pivot rule. Show that the answer would be the same if, instead, the largest increase pivot rule was used.

Subtask e

After a number of iterations the tableau is the following:

x1	x2	s1	s2	s3	-z	b
0	1	2/3	-1/6	0	0	7/2
1	0	-1	1/2	0	0	3/2
0	0	8/3	-5/3	1	0	7
0	0	-2/3	-5/6	0	1	-61/2

Argue that an optimal solution for (2lp) has been found and give for it the value of x_1 and x_2 together with its objective function value. Report the optimality gap for (2) at this stage.

Subtask e

Show how you can reconstruct the tableau at the previous point by just knowing that x_2, x_1 and s_3 are in basis and that:

$$A_B^{-1} = \begin{bmatrix} 2/3 & -1/6 & 0 \\ -1 & 1/2 & 0 \\ 8/3 & -5/3 & 1 \end{bmatrix}.$$

Solution:

```
import numpy as np
from fractions import Fraction as f
A=np.array([[ 1, 3,0],[4, 6,0],[4,2,1]])
# print np.linalg.inv(A)
A_1= np.array([[f(2,3),f(-1,6),0],[f(-1),f(1,2),0],[f(8,3),f(-5,3),1]])
print np.dot(A[:,[1,0,2]],A_1)
```

Subtask f

From the second row of the last tableau derive a Gomory cut and write it in the space of the original variables.
Argue shortly that the cut is a valid inequality for (2) and that it will make the current optimal solution of (2lp) infeasible.

Subtask g

Introduce the cut in the tableau and explain how the solution algorithm will continue. Indicate the new pivot and explain how you found it. (You do not need to carry out the simplex iteration.)

Subtask h

After the introduction of the cut the tableau of the optimal solution to the new LP problem is the following.

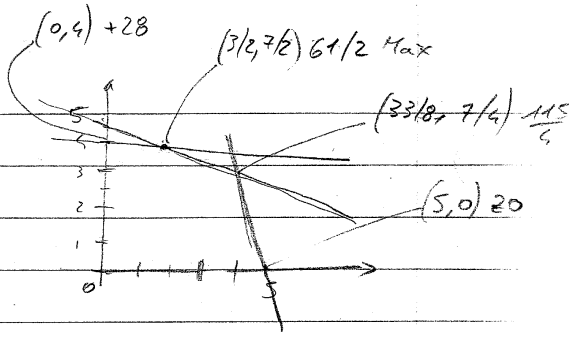
x1	x2	s1	s2	s3	s4	-z	b
0	1	2/3	0	0	-1/3	0	11/3
0	0	0	1	0	-2	0	1
0	0	8/3	0	1	-10/3	0	26/3
1	0	-1	0	0	1	0	1
0	0	-2/3	0	0	-5/3	1	-89/3

Explain how the solution process would continue from this stage by branch and bound. Define the next branching and indicate what can be done in each open node.

Solution:

TASK 1

$$\max z = 4x_1 + 7x_2$$
$$x_1 + 3x_2 \le 12$$
$$4x_1 + 6x_2 \le 27$$
$$4x_1 + 2x_2 \le 20$$



(a) (0,0) is feasible $\Rightarrow z = 0 = LB$

(c) (d)

x_1	x_2	s_1	s_2	s_3	$-z$	b	b_i/a_{i1}	b_i/a_{i2}	$c_j \cdot b_i/a_{i1}$
1	3	1	0	0	0	12	12	4	$7 \cdot 4 = 28$
4	6	0	1	0	0	27	$27/4 = 8$	$\frac{27}{6} \simeq 4$	
4	2	0	0	1	0	20	5	10	$5 \cdot 4 = 20$
4	7	0	0	0	1	0			

Largest coefficient $\rightsquigarrow 7$

Largest increase $\rightsquigarrow 7$

1/3	1	1/3	0	0	0	4
2	0	−2	1	0	0	3
10/3	0	−2/3	0	1	0	12
−5/3	0	7/3	0	0	1	28

(e)

0	1	2/3	−1/6	0	0	7/2
1	0	−1	1/2	0	0	3/2
0	0	8/3	−5/2	1	0	7
0	0	−2/3	−5/6	0	1	−61/2

$$GAP = \frac{61/2 - 0}{61/2} = 1$$

(g)

I row: $\frac{2}{3}s_1 + \frac{5}{6}s_2 \ge \frac{1}{2}$ $\qquad \frac{1}{2}s_1 \ge \frac{1}{2}$

$$\frac{2}{3}(12 - x_1 - 3x_2) + \frac{5}{6}(27 - 4x_1 - 6x_2) \ge \frac{1}{2}$$
$$8 - \frac{2}{3}x_1 - 2x_2 + \frac{135}{6} - \frac{10}{3}x_1 - 5x_2 \ge \frac{1}{2}$$
$$-4x_1 - 7x_2 \ge \frac{1}{2} - 8 - \frac{135}{6}$$

II row: $\frac{1}{2}s_2 \ge \frac{1}{2}$ $\qquad$ (i)-form

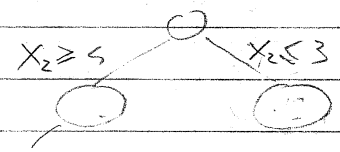
$$\frac{1}{2}(27 - 4x_1 - 6x_2) \ge 1$$
$$-4x_1 - 6x_2 \ge -26$$
$$-2x_1 - 3x_2 \ge -13$$
$$2x_1 + 3x_2 \le 13 \qquad \text{(ii)-form}$$

(g) Introducing the i-form we have a basis but infeasible.
Introducing the ii-form we do elementary row operations to arrive to a canonical form with infeasible basis.
We use dual simplex:

0	0	0	$\boxed{-1/2}$	0	0	1	$-1/2$
0	1	$2/3$	$-1/6$	0	0	0	$7/2$
1	0	-1	$1/2$	0	0	0	$3/2$
0	0	$8/3$	$-5/3$	1	0	0	7
0	0	$-2/3$	$-5/6$	0	1	0	$-61/2$

1) pivot < 0
2) row with b term negative
3) col that $\min \left|\frac{c_j}{a_{ij}}\right|$ $\Rightarrow$ $-\frac{1}{2}$ is pivot

(h) Using branch and bound on we have



one could use heuristics again at each node

the LP requires a dual-simplex step
sol will be (0,4) of val 28

sol will be (2,3) of val 29

Exercise 10

This is a continutation of the Factory Planning problem from the computer lab class Sheet 4, Exercise 3. The setting is the multiperiod problem discussed in tasks 2.

Task 3 Implement the single-period model (Task 1) in a SpreadSheet. Implement the multi-period model (Task 2) in Python and Gurobi. Solve the problem on the data given.

- Report and comment relevant information from the execution of the solver.
- Report the production plan, that is, how much of each product should the factory produce in the months.
- Indicate which resource capacity could be convinient to increase in some months and the impact that such increase would have on the total profit.

Solution:

The implementation in Python is show in the Figure 3
The optimal policy yields a total profit of 93715.18 Euro and it is shown below:

```
def solve(data):
    m = Model("fpmm")
    m.setParam(GRB.param.Method, 0)

    ######### BEGIN: Write here your models

    x={}
    for i in data.products:
        for (t_int, t_string) in enumerate(data.months):
            x[i,t_int]=m.addVar(lb=0.0,ub=GRB.INFINITY,obj=0.0,vtype=GRB.CONTINUOUS,name="
                x_%s_%s" % (i,t_int))

    s={}
    for i in data.products:
        for (t_int, t_string) in enumerate(data.months):
            s[i,t_int]=m.addVar(lb=0.0,ub=GRB.INFINITY,obj=0.0,vtype=GRB.CONTINUOUS,name="
                s_%s_%s" % (i,t_int))

    h={}
    for i in data.products:
        for (t_int, t_string) in enumerate(data.months):
            h[i,t_int]=m.addVar(lb=0.0,ub=100,obj=0.0,vtype=GRB.CONTINUOUS,name="h_%s_%s" %
                 (i,t_int))


    m.update()

    m.setObjective(quicksum(data.profits[i0]*s[i1,t_int]-0.5*h[i1,t_int]
                          for (i0,i1) in enumerate(data.products)
                          for (t_int,t_string) in enumerate(data.months)),
                  GRB.MAXIMIZE)

    # machine capacities
    c={}
    for j in data.machines:
        for (t_int, t_string) in enumerate(data.months):
            c[j,t_string]=m.addConstr(quicksum(data.coeff[j,i]*x[i,t_int] for i in data.
                products) <= 384*(data.capacity[j]-data.maintainance[j,t_string]),"cap_%s"
                % j)

    # mass balance
    for i in data.products:
        for (t_int, t_string) in enumerate(data.months):
            if t_int==0:
                m.addConstr(x[i,t_int]==s[i,t_int]+h[i,t_int],"bal0_%s_%s" % (i,t_int))
            else:
                m.addConstr(h[i,t_int-1]+x[i,t_int]==s[i,t_int]+h[i,t_int],"bal_%s_%s" % (i
                    ,t_int))

    for i in data.products:
        for (t_int, t_string) in enumerate(data.months):
            m.addConstr(s[i,t_int]<=data.market_limits[t_string, i],"market_limits_%s_%s" %
                 (i,t_int) )

    for i in data.products:
        m.addConstr(h[i,5]>=50)

    ######### END
```

Figure 3:

```
Optimize a model with 79 rows, 126 columns and 288 nonzeros
Coefficient statistics:
  Matrix range     [1e-02, 1e+00]
  Objective range  [5e-01, 1e+01]
  Bounds range     [6e+01, 1e+03]
  RHS range        [5e+01, 2e+03]
Presolve removed 74 rows and 110 columns
Presolve time: 0.00s
Presolved: 5 rows, 16 columns, 21 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    8.0175000e+04   0.000000e+00   5.300000e+01      0s
       9    9.3715179e+04   0.000000e+00   0.000000e+00      0s

Solved in 9 iterations and 0.00 seconds
Optimal objective  9.371517857e+04

x[i,t]=
      january     february        march        april          may         june
1         500.0        700.0          0.0        200.0          0.0        550.0
2 888.571428571        600.0          0.0        300.0        100.0        550.0
3         382.5        117.5          0.0        400.0        600.0          0.0
4         300.0          0.0          0.0        500.0        100.0        350.0
5         800.0        500.0          0.0        200.0       1100.0          0.0
6         200.0        300.0        400.0          0.0        300.0        550.0
7           0.0        250.0          0.0        100.0        100.0          0.0

s[i,t]=
      january     february        march        april          may         june
1         500.0        600.0        100.0        200.0          0.0        500.0
2 888.571428571        500.0        100.0        300.0        100.0        500.0
3         300.0        200.0          0.0        400.0        500.0         50.0
4         300.0          0.0          0.0        500.0        100.0        300.0
5         800.0        400.0        100.0        200.0       1000.0         50.0
6         200.0        300.0        400.0          0.0        300.0        500.0
7           0.0        150.0        100.0        100.0          0.0         50.0

h[i,t]=
      january     february        march        april          may         june
1           0.0        100.0          0.0          0.0          0.0         50.0
2           0.0        100.0          0.0          0.0          0.0         50.0
3          82.5          0.0          0.0          0.0        100.0         50.0
4           0.0          0.0          0.0          0.0          0.0         50.0
5           0.0        100.0          0.0          0.0        100.0         50.0
6           0.0          0.0          0.0          0.0          0.0         50.0
7           0.0        100.0          0.0          0.0        100.0         50.0
```

Information on the reduced costs of the variables s gives information on which change in price should be made to increase production of a product in a month in which it is not produced.

```
s[i,t].rc= (reduced costs)
      january     february        march        april          may         june
1           0.0          0.0          0.0          0.0          0.0          0.0
2           0.0          0.0          0.0          0.0          0.0          0.0
3           0.0          0.0          0.0          0.0          0.0          0.0
4           0.0          0.0          0.0          0.0          0.0          0.0
5           0.0          0.0          0.0          0.0          0.0          0.0
6           0.0          0.0          0.0          0.0          0.0          0.0
7 -1.28571428571          0.0          0.0          0.0          0.0          0.0
```

It seems that to make profitable producing the product 7 in January we should increase its price by 1.28. The fact that the reduced costs of other variables s that are zero in the solution are also zero is somehow suspicious. This might hint at the fact that the solution found is not unique!

Information on the value of acquiring new machines can be obtained from the *marginal values* of the appropriate constraints (ie, the dual variables). The value of an extra hour in the particular month when a particular type of machine is used to capacity is given below:

```
c[i,t]= (marginal values)
      january     february        march        april          may         june
grinder 8.57142857143          0.0          0.0          0.0          0.0          0.0
vdrill           0.0          0.0          0.0          0.0          0.0          0.0
hdrill           0.0        0.625          0.0          0.0          0.0          0.0
borer            0.0          0.0        200.0          0.0          0.0          0.0
planer           0.0          0.0          0.0          0.0          0.0        800.0
```

Marginal values of non binding constraints are zero since there is no gain in profit by augmenting the capacity of those constraints. The positive values represent the increase in profit achiavable by a unitary increase in capacity of that resource. It seems that increasing the capacity of planer in June would yield a considerable increase in the total profit. Increasing the borer in March would be also worth.

It is instructive to discuss the way in which multi-period models should be used. Such a model is usually run with the first period relating to the present times and subsequent periods relating to the future. As a result only the operating decisions suggested by the model for the present month are put into action. Operating decisions for future months will probably only be taken as provisional. After a further month (or the appropriate time period) has elapsed the model will be rerun with updated data and the first period applying to the new present period. In this way a multi-period model is in constant use as both an operating tool for the present and a provisional planning tool for the future.

A further point of importance in multi-period models concerns what happens at the end of the last time period in the model. If the stocks at the end of the last period which occur in constraints (??) are included simply as variables the optimal solution will almost always decide that they should be zero. From the point of view of the model this would be sensible as it would be the minimum cost or maximum profit solution. In a practical situation, however, the model is unrealistic since operations will almost certainly continue beyond the end of the last period and stocks would probably not be allowed to run right down. One possible way out is to set the final stocks to constant values representing sensible final levels. It could be argued that the operating plans for the final period will be very provisional anyway and any inaccuracy that far ahead not serious. An alternative approach which is sometimes adopted is to value the final stocks in some way, i.e. give the appropriate variables positive profits in a maximization model or negative costs in minimization model. In effect such a valuation would cause the optimal solution to suggest producing final stocks to sell if it appeared profitable. Although the organization might never consider the possibility of selling off final stocks, the fact that they had been given realistic valuations would cause them to come out at sensible levels.

Task 4 Here, instead of stipulating when each machine is down for maintenance, it is desired to find the best month for each machine to be down.
Each machine must be down for maintenance in one month of the six apart from the grinding machines, only two of which need to be down in any six months.
Extend the model that correctly addressed tasks 2 and 3 to allow it to make these extra decisions.

- How many variables did you need to add? What is the domain of these variables?
- Has the matrix of the problem a similar structure to the one of the point above?
- Is the solution from Task 3 a valid solution to this problem? What information can it bear in this new case?
- Implement and solve the model in Python and Gurobi. After how many nodes in the branch and bound tree is the optimal solution found? And after how many is it proven optimal?
- How much worth is the extra flexibility of choosing when to place downtimes?

Solution:

The extra decisions that this task requires over the factory planning problem requires the use of integer programming.

The integer variables that we need to add are y_{jt}, that is, the number of machines of type j down for maintenance in month t. Depending on the type of machine these variables will have different upper bounds (as defined in the first sentence of the problem description. There are 30 such variables.

The model will change in the machine capacity constraints. Instead of the previous values we will now have: $384(c_j - y_{jt})$. In addition we need the constraints on the maintainance expressed at the beginning of this task:

$$\sum_{t=1}^{6} y_{jt} = \begin{cases} 2 & j = GR, VD \\ 3 & j = HD \\ 1 & j = BO \\ 1 & j = PL \end{cases}$$

The rest remains the same. The new model is:

$$(??), (??) - (??) \tag{3}$$

$$\sum_i a_{ij} x_{it} \leq 384(c_j - y_{j,t}) \qquad j \in \{GR, VD, HD, BR, PL\}, t = 1 \ldots, 6 \tag{4}$$

$$\sum_{t=1}^{6} y_{j,t} = c_j \qquad j \in \{VD, HD, BR, PL\} \tag{5}$$

$$\sum_{t=1}^{6} y_{GR,t} = 2 \qquad j \in \{VD, HD, BR, PL\} \tag{6}$$

$$y_{j,t} \in \mathbb{Z}_0^+ \qquad j \in \{GR, VD, HD, BR, PL\}, t = 1 \ldots, 6 \tag{7}$$

$$\tag{8}$$

An alternative formulation is possible using a $0-1$ variable to indicate for each machine whether it is down for maintenance in a particular month or not. Such a formulation would have more variables and suffer the drawback of producing equivalent alternate solutions in the tree search of the branch and bound.

The solution at point B is not a feasible solution because the maintainances are less than those required here. If the numbers of month in maintainance per machine was the same, then the solution to point B would be a feasible solution but not optimal, hence a primal bound, here a lower bound.

The implementation in Python is given in Figure 4.

The solution is shown below.

```
Presolve removed 0 rows and 14 columns

Root relaxation: objective 1.164550e+05, 75 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 116455.000    0   13 -175.00000 116455.000      -     -    0s
H    0     0                     92755.000000 116455.000  25.6%     -    0s
H    0     0                     107841.66667 116455.000  7.99%     -    0s
     0     0 111669.725    0    7 107841.667 111669.725  3.55%     -    0s
H    0     0                     108855.00000 111669.725  2.59%     -    0s
     0     0 109317.158    0    6 108855.000 109317.158  0.42%     -    0s
     0     0     cutoff    0      108855.000 108855.000  0.00%     -    0s

Cutting planes:
  Gomory: 3
  Implied bound: 15
  MIR: 5

Explored 0 nodes (132 simplex iterations) in 0.01 seconds
```

```
def solve(data):
    m = Model("fpmm")
    m.setParam(GRB.param.Method, 0)

    ######### BEGIN: Write here your models
    x={}
    for i in data.products:
        for (t_int, t_string) in enumerate(data.months):
            x[i,t_int]=m.addVar(lb=0.0,ub=GRB.INFINITY,obj=0.0,vtype=GRB.CONTINUOUS,name="
                x_%s_%s" % (i,t_int))

    s={}
    for i in data.products:
        for (t_int, t_string) in enumerate(data.months):
            s[i,t_int]=m.addVar(lb=0.0,ub=GRB.INFINITY,obj=0.0,vtype=GRB.CONTINUOUS,name="
                s_%s_%s" % (i,t_int))

    h={}
    for i in data.products:
        for (t_int, t_string) in enumerate(data.months):
            h[i,t_int]=m.addVar(lb=0.0,ub=100,obj=0.0,vtype=GRB.CONTINUOUS,name="h_%s_%s" %
                 (i,t_int))

    y={}
    for j in data.machines:
        for (t_int, t_string) in enumerate(data.months):
            y[j,t_int]=m.addVar(lb=0.0,ub=data.capacity[j],obj=0.0,vtype=GRB.INTEGER,name="
                y_%s_%s" % (j,t_int))

    m.update()

    m.setObjective(quicksum(data.profits[i0]*s[i1,t_int]-0.5*h[i1,t_int]
                          for (i0,i1) in enumerate(data.products)
                          for (t_int,t_string) in enumerate(data.months)),
                  GRB.MAXIMIZE)

    # machine capacities
    c={}
    for j in data.machines:
        for (t_int, t_string) in enumerate(data.months):
            c[j,t_string]=m.addConstr(quicksum(data.coeff[j,i]*x[i,t_int] for i in data.
                products) <= 384*(data.capacity[j]-y[j,t_int]),"cap_%s" % j)

    # maintainances
    for j in data.machines:
        if j == "grinder":
            m.addConstr(quicksum(y[j,t_int] for (t_int, t_string) in enumerate(data.months)
                )==2,"maintainance_%s" % j)
        else:
            m.addConstr(quicksum(y[j,t_int] for (t_int, t_string) in enumerate(data.months)
                )==data.capacity[j],"maintainance_%s" % j)

    # mass balance
    for i in data.products:
        for (t_int, t_string) in enumerate(data.months):
            if t_int==0:
                m.addConstr(x[i,t_int]==s[i,t_int]+h[i,t_int],"bal0_%s_%s" % (i,t_int))
            else:
                m.addConstr(h[i,t_int-1]+x[i,t_int]==s[i,t_int]+h[i,t_int],"bal_%s_%s" % (i
                    ,t_int))

    for i in data.products:
        for (t_int, t_string) in enumerate(data.months):
            m.addConstr(s[i,t_int]<=data.market_limits[t_string, i],"market_limits_%s_%s" %
                 (i,t_int) )

    for i in data.products:
        m.addConstr(h[i,5]>=50)
    ######### END
```

```
Thread count was 4 (of 8 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective 1.088550000000e+05, best bound 1.088550000000e+05, gap 0.0%
x[i,t]=
      january     february        march        april          may         june
1         500.0        600.0        400.0          0.0          0.0        550.0
2        1000.0        500.0        700.0          0.0        100.0        550.0
3         300.0        200.0        100.0          0.0        500.0        150.0
4         300.0          0.0        100.0          0.0        100.0        350.0
5         800.0        400.0        600.0          0.0       1000.0       1150.0
6         200.0        300.0        400.0          0.0        300.0        550.0
7         100.0        150.0        200.0          0.0          0.0        110.0

s[i,t]=
      january     february        march        april          may         june
1         500.0        600.0        300.0        100.0          0.0        500.0
2        1000.0        500.0        600.0        100.0        100.0        500.0
3         300.0        200.0          0.0        100.0        500.0        100.0
4         300.0          0.0          0.0        100.0        100.0        300.0
5         800.0        400.0        500.0        100.0       1000.0       1100.0
6         200.0        300.0        400.0          0.0        300.0        500.0
7         100.0        150.0        100.0        100.0          0.0         60.0

h[i,t]=
      january     february        march        april          may         june
1           0.0          0.0        100.0          0.0          0.0         50.0
2           0.0          0.0        100.0          0.0          0.0         50.0
3           0.0          0.0        100.0          0.0          0.0         50.0
4           0.0          0.0        100.0          0.0          0.0         50.0
5           0.0          0.0        100.0          0.0          0.0         50.0
6           0.0          0.0          0.0          0.0          0.0         50.0
7           0.0          0.0        100.0          0.0          0.0         50.0

y[j,t]=
      january     february        march        april          may         june
grinder           0.0          0.0          0.0          2.0          0.0          0.0
vdrill            0.0          0.0          0.0          1.0          1.0          0.0
hdrill            0.0          2.0          0.0          0.0          0.0          1.0
borer             0.0          0.0          0.0          1.0          0.0          0.0
planer            0.0          0.0          0.0          1.0          0.0          0.0
```

The optimal solution is found at the root node after the addition of cutting planes. The new total profit is 108855 Euro and shows that the added flexibility is worth: $108855 - 93715.18 = 15140$ Euro!

References

[DT97] George B. Dantzig and Mukund N. Thapa. *Linear Programming*. Springer, 1997.

[Goe11] Michel X. Goemans. Lecture notes on bipartite matching. http://www-math.mit.edu/~goemans/18433S11/matching-notes.pdf, 2011.