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

May 15, 2018
Marco Chiarandini

DM559/DM545 – Linear and integer programming

Sheet 8, Spring 2018 [pdf format]

Exercise 1*

Manpower Planning. Given a set of workers and the need to cover a set of 15 working hours per day with a, possibly different, number of required persons as staff at each hour, decide the staff at each hour taking into consideration that each person works in shifts that cover 7 hours and hence a person starting in hour i contributes to the workload in hours $i, \ldots, i+6$ (e.g., a person starting in hour 3 contributes to the workload in hours 3,4,5,6,7,8,9).
Formulate the problem to determine the number of people required to cover the workload in mathematical programming terms.

Solution:

Decision Variables:

- $x_i \in \mathbb{Z}_0^+$: number of people starting work in hour $i (i = 1, \ldots, 15)$

Objective Function:

$$\min \sum_{i=1}^{15} x_i$$

Constraints:

- Demand:

$$\sum_{i=t-6}^{i=t} x_i \geq d_t \text{ for } t = 1, \ldots, 15$$

  The number of workers at time t is given by those that started at $t, t-1, \ldots, t-6$

- Bounds:

$$x_{-5}, \ldots, x_0 = 0$$

- Variables:

$$x_i \in \mathbb{Z}_0^+ \qquad i = 1, \ldots, 15$$

Shift scheduling. The administrators of a department of a urban hospital have to organize the working shifts of nurses maintaining sufficient staffing to provide satisfactory levels of health care. Staffing requirements at the hospital during the whole day vary from hour to hour and are reported in the table:

Hour	Staffing requirement
0 am to 6 am	2
6 am to 8 am	8
8 am to 11 am	5
11 am to 2 pm	7
2 pm to 4 pm	3
4 pm to 6 pm	4
6 pm to 8 pm	6
8 pm to 10 pm	3
10 pm to 12 pm	1

According to union agreements, nurses can work according to one of the seven shift patterns below each with its own cost

pattern	Hours of work	total hours	cost
1	0 am to 6 am	6	720 Dkk
2	0 am to 8 am	6	800 Dkk
3	6 am to 2 pm	8	740 Dkk
4	8 am to 4 pm	8	680 Dkk
5	2 pm to 10 pm	8	720 Dkk
6	4 pm to 12 pm	6	780 Dkk
7	6 pm to 12 pm	6	640 Dkk

The department administrators would like to identify the assignment of nurses to working shifts that meets the staffing requirements and minimizes the total cost.

Solution:

$$\min \quad 720x_1 + 800x_2 + 740x_3 + 680x_4 + 720x_5 + 780x_6 + 640x_7$$

$$0-6: \quad x_1 + x_2 \geq 2$$

$$6-8: \quad x_2 + x_3 \geq 8$$

$$8-11: \quad x_3 + x_4 \geq 5$$

$$11-14: \quad x_3 + x_4 \geq 7$$

$$14-16: \quad x_4 + x_5 \geq 3$$

$$16-18: \quad x_5 + x_6 \geq 4$$

$$18-20: \quad x_5 + x_6 + x_7 \geq 6$$

$$20-22: \quad x_5 + x_6 + x_7 \geq 3$$

$$22-24: \quad x_6 + x_7 \geq 1$$

$$x_1, x_2 \dots x_7 \geq 0 \text{ and integer.}$$

Exercise 2*

Give two MILP formulations for the chromatic number problem: given a graph find a coloring (i.e., a mapping from the natural numbers to the set of vertices) such that every pair of vertices connected by an edge receives different colors and the number of colors used is the least possible.

Solution:

Graph coloring $G = (V, E)$ Γ set of colors

- Formulation 1

Let $x_{ik} = \begin{cases} 1 & \text{if vertex } i \in V \text{ gets color } k \in \Gamma \\ 0 & \text{else} \end{cases}$

$y_k = \begin{cases} 1 & \text{if color } k \text{ is used} \\ 0 & \text{else} \end{cases}$

$$\min \sum_{k \in \Gamma} y_k$$

$$(1) \quad x_{ik} + x_{jk} \leq y_k \qquad \forall\, ij \in E$$

$$x_{ik} \in \mathbb{B} \quad \forall\, i \in V,\ k \in \Gamma$$

$$y_k \in \mathbb{B} \quad \forall\, k \in \Gamma$$

It is possible to enhance (1) with clique constraints:

$$\sum_{i \in C} x_{ik} \leq y_k \qquad \forall\, C \in \text{cliques}$$

- Formulation 2

Let I be the set of all independent sets in the graph G

Let $y_i = \begin{cases} 1 & \text{for } i \in I \text{ if the set is selected} \\ 0 & \text{else} \end{cases}$

$$\min \sum_{i \in I} y_i$$

$$\sum_{i \in I} a_{ij}\, y_i \geq 1 \qquad \forall\, j \in V$$

$$y_i \in \mathbb{B} \qquad \forall\, i \in I$$

$a_{ij} = \begin{cases} 1 \text{ if } i \text{ contains node } j \\ 0 \text{ else} \end{cases}$

There are exponentially many independent sets in general, hence the model has exponential number of variables, nevertheless it is possible to use a delayed variable generation and solve this problem efficiently.

Exercise 3*

Formulate the Minimum Spanning Tree problem as a mathematical programming problem.

Solution:

A tree is a connected graph with no cycles, and a spanning tree is a subgraph in G that is a tree and includes all vertices of G i.e. V. The minimum spanning tree (MST) problem calls for finding a spanning tree whose total cost is minimum. The total cost is measured as the sum of the costs of all the edges in the tree.
A cut $C = \{X, X'\}$ is a partition of the set of vertices into two subsets. For $X \subset V$, we denote the set of arcs crossing the cut $\delta_{X,X'} = \{(u, v) \in E : u \in X, v \in V \setminus X\}$.
For a spanning tree of n vertices we need $n - 1$ edges. Furthermore, for each possible cut C at least one of the edges crossing the cut must be in the solution.
Cut set formulation of MST:

$$\begin{aligned} \min & \sum w_e x_e \\ & \sum_{e \in E} x_e = n - 1 \\ & \sum_{e \in \delta_X} x_e \geq 1 && \forall \emptyset \subset X \subset V \\ & x_e \in \{0, 1\} \end{aligned}$$

Sub tour elimination formulation of the MST:

$$\begin{aligned} \min & \sum w_e x_e \\ & \sum_{e \in E} x_e = n - 1 \\ & \sum_{e \in \gamma_S} x_e \leq |S| - 1 && \forall \emptyset \subset S \subset V \\ & x_e \in \{0, 1\} \end{aligned}$$

where $\gamma_S = \{ij \in E \mid i \in S, j \in S\}$.
More om MST in sec. 3.5 of [Wo] (where it is called maximum weight Tree) and on page 23 of [CL].

Exercise 4* Consider the polyhedron $P \subseteq R^2$ described by the following inequalities and depicted in Figure 1:

$$\begin{array}{rcrcr} 2x & - & y & \leq & 4 \\ 2x & + & 3y & \leq & 12 \\ & & y & \leq & 3 \\ 3x & + & 2y & \geq & 6 \\ x & & & \leq & 3 \\ 3/2x & + & y & \leq & 45/8 \\ 2x & + & 3y & \leq & 10 \end{array}$$

Which inequality is a face? Which is a facet? Which is redundant?

Solution:

A constraint $\mathbf{a}_0\mathbf{x} \leq b_0$ dominates another $\mathbf{a}_1\mathbf{x} \leq b_1$ if for $\mathbf{x} \geq \mathbf{0}$ the first constraint is closer to the feasibility region. This can be formalized more precisely, but we do not see it here.
A constraint $\mathbf{a}_0\mathbf{x} \leq b_0$ is redundant if there are $k \geq 2$ constraints whose linear combination dominates $\mathbf{a}_0\mathbf{x} \leq b_0$.

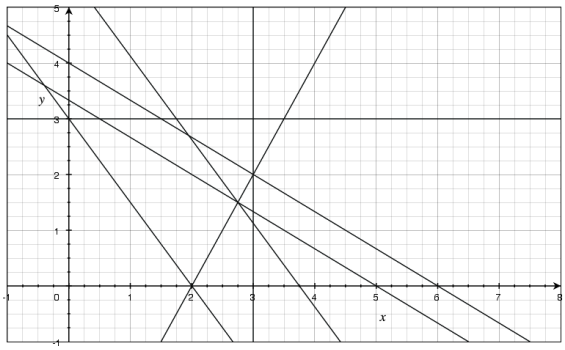
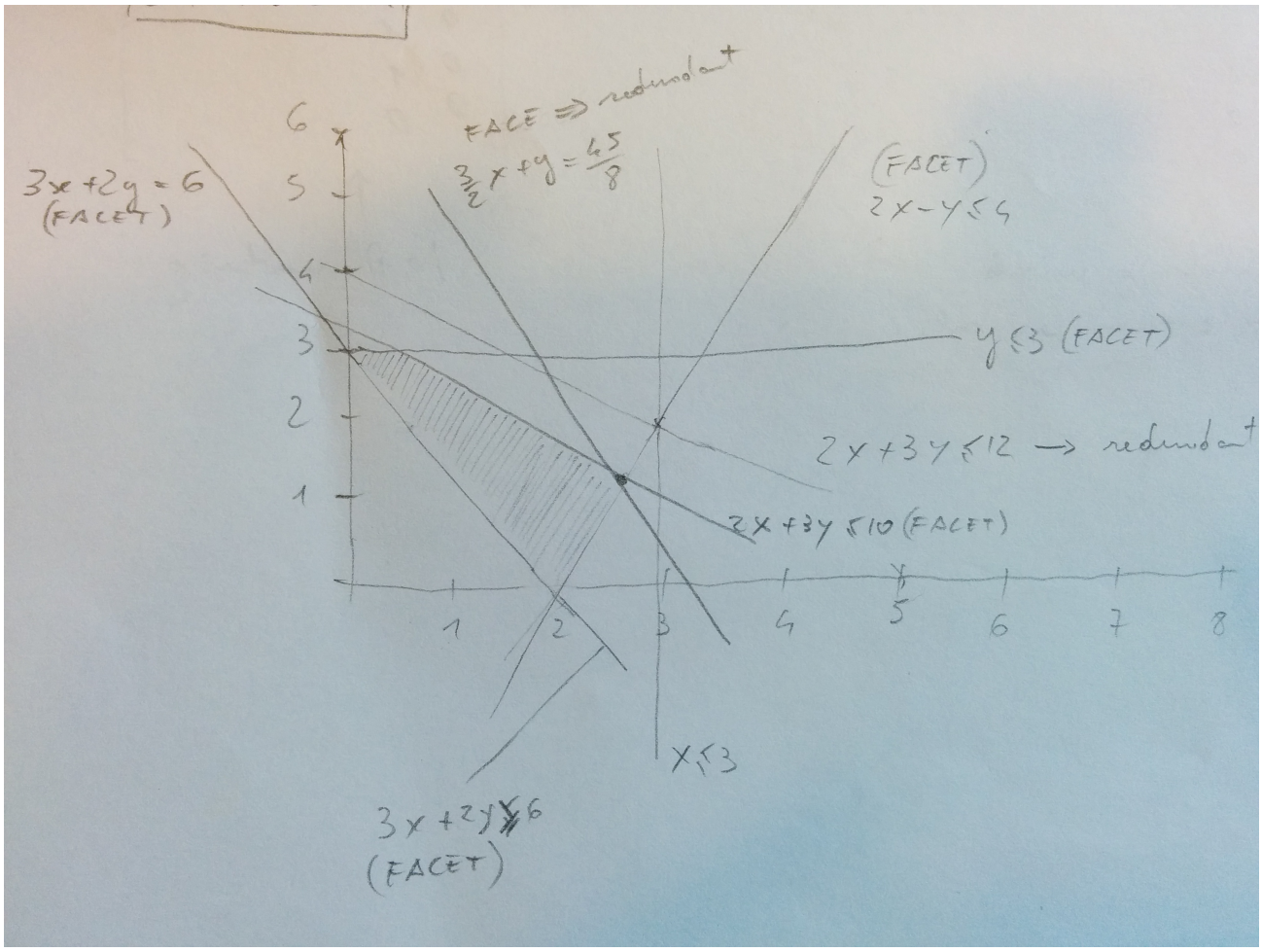


Figure 1: The polyhedron of exercise 4.

$$
\begin{array}{rcrcrl}
2x & - & y & \leq & 4 & \text{Facet} \\
2x & + & 3y & \leq & 12 & \text{Redundant} \\
 & & y & \leq & 3 & \text{Facet} \\
3x & + & 2y & \geq & 6 & \text{Facet} \\
x & & & \leq & 3 & \text{Redundant} \\
3/2x & + & y & \leq & 45/8 & \text{Face, Redundant} \\
2x & + & 3y & \leq & 10 & \text{Facet}
\end{array}
$$



Exercise 5*

Given the following LP problem:

$$\begin{aligned} \min \quad & 2x_1 + x_2 + 6x_3 - x_4 \\ & 3x_1 + x_2 - x_5 = 2 \\ & x_2 + 4x_3 = 4 \\ & x_3 + 4x_4 + x_5 = 5 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

consider the set of columns $[\mathbf{a}_2, \mathbf{a}_4, \mathbf{a}_5]$. Does it determine a basis? If so is it a feasible basis? Is it optimal?

Solution:

This is an application of the revised simplex method. The problem is already in equational standard form. Let's write the matrix A of the problem.

$$A = \begin{bmatrix} 3 & 1 & 0 & 0 & -1 \\ 0 & 1 & 4 & 0 & 0 \\ 0 & 0 & 1 & 4 & 1 \end{bmatrix}$$

Hence:

$$A_B = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & 4 & 1 \end{bmatrix} \qquad A_N = \begin{bmatrix} 3 & 0 \\ 0 & 4 \\ 0 & 1 \end{bmatrix}$$

We calculate $\mathbf{x}_B = A_B^{-1}\mathbf{b}$ and $\bar{c}_N^T = c_N^T - \mathbf{c}_B^T A_B^{-1} A_N$.
In Python:

```
import numpy as np
A=np.array([[3,1,0,0,-1],[0,1,4,0,0],[0,0,1,4,1]])
b=np.array([2,4,5])
c=np.array([2,1,6,-1,0])

basis = np.array([1,3,4]) # indices start at zero
nonbasis = np.array([0,2])

A_B = A[:,basis]
A_N = A[:,nonbasis]

A_B_i = np.linalg.inv(A_B)
x_B = np.dot(A_B,b)

x_B
# array([-3,  2, 21])
```

Hence $\mathbf{x}_B \not\geq \mathbf{0}$ and the basic solution is not feasible. To find an initial feasible basis we could solve an auxiliary problem with added slack variables to minimize to zero. Alternatively one can devise a revised dual simplex method and proceed with that.
In fact the model is infeasible:

```
from gurobipy import *

m=gr.Model('inf')
m.setParam(GRB.param.Method, 0)
x1, x2, x3, x4, x5 = m.addVars(5)
#m.addVars(lb=0.0, ub=GRB.INFINITY, obj=1.0, vtype=GRB.CONTINUOUS, names=['x1','x2','x3','
    x4','x5'])

# The objective is to maximize (this is redundant now, but it will overwrite Var
    declaration
m.setObjective(2.0*x1 + 2.0*x2 + 6.0*x3 - 1.0*x4, GRB.MINIMIZE)

# Add constraints to the model
```

```
m.addConstr(1*x1 + 1*x2-x5 == 2, "c1")
m.addConstr(1*x2 + 4*x3 == 4, "c2")
m.addConstr(1*x3 + 4*x4 + 1*x5 == 5, "c3")

m.optimize()

# Solved in 0 iterations and 0.03 seconds
# Infeasible model
```

Exercise 6*

In class, we proved that the (mininum) vertex covering problem and the (maximum) matching problem are a weak dual pair. Prove that for bipartite graphs they, actually, are a strong dual pair.

Solution:

The formulation of the matching problem is:

$$\begin{aligned} \max \quad & \sum_{e \in E} w_e x_e & & \\ & \sum_{e \in E : v \in e} x_e \leq 1 & & \forall v \in V \\ & x_e \in \{0,1\} & & \forall e \in E \end{aligned}$$

If we take the linear relaxation and make the dual of it we obtain:

$$\begin{aligned} \min \quad & \sum_{v \in V} y_v & & \\ & y_v + y_u \geq 1 & & \forall u, v \in V, uv \in E \\ & y_v \geq 0 & & \forall v \in V \end{aligned}$$

This latter is the linear relaxation of the vertex cover problem.
Hence the two problems make a weak dual pair. It is not strong, indeed a triangle has optimal vertex cover 2 and optimal matching 1.
In bipartite graphs instead the pair is strong dual. Indeed, the solution of the linear relaxation of the matching problem on bipartite graphs is always integer. The same must old for the dual and hence the gap is closed.
A more formal proof is on page 24-25 of [Wo].

Exercise 7*

Generalized Assignment Problem. Suppose there are n types of tracks available to delivery products to m clients. The cost of track of type i serving client j is c_{ij}. The capacity of track type i is C_i and the demand of each client is d_j. There are a_i tracks for each type. Formulate an IP model to decide how many tracks of each type are needed to satisfy all clients so that the total cost of doing the deliveries is minimized. If all the input data will be integer, will the solution to the linear programming relaxation be integer?

Solution:

It is good trying to model the problem as a min cost flow problem. However, one can soon realize that this is not possible since we are asked for the number of tracks but we need to take into account a demand and a capacity of products.
We can however write an ILP model:

$$\min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} \tag{1}$$

$$\sum_{j \in J} x_{ij} = 1 \qquad \forall i \in I \tag{2}$$

$$\sum_{i \in I} a_{ij} x_{ij} \leq d_j \qquad \forall j \in J \tag{3}$$

$$x \geq 0 \text{ and integer} \qquad \forall (i,j) \in A \tag{4}$$

The solutions of the linear relaxation are not necessarily integer, becasue this is not a min cost flow model and the matrix is not trivially TUM.

Exercise 8*

Consider the following three matrices:

$$\begin{bmatrix} 1 & 1 & -1 & 0 & 1 \\ 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & -1 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & -1 \\ 0 & -1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 1 \end{bmatrix}$$

For each of them say if it is totally unimodular and justify your answer.

Solution:

We look for the satisfaction of the conditions of the theorem saw in class. Accordingly, it is *sufficient* for a matrix to be TUM to find a partition of the rows such that the ones with same sign are in different partitions and those with different sign in the same partition.
The first matrix is TUM. The partition is $I_1 = \{1, 4\}$ and $I_2 = \{2, 3\}$.
The second matrix is TUM. The partition is $I_1 = \{1, 2, 3\}$ and $I_2 = \{4\}$.
The third matrix is not TUM. Here the theorem does not apply and there is a submatrix with deteminant -2, hence we cannot be sure that the solutions associated with the matrix will be integer.

Exercise 9*

In class we stated that for the uncapaciteted facility location problem there are two formulations:

$$X = \{(x, y) \in \mathbb{R}^m_+ \times \mathbb{B}^1 : \sum_{i=1}^{m} x_i \leq my, x_i \leq 1 \text{ for } i = 1, \ldots, m\}$$

$$P = \{(x, y) \in \mathbb{R}^n_+ \times \mathbb{R}^1 : x_i \leq y \text{ for } i = 1, \ldots, m, y \leq 1\}$$

Proove that the polyhedron P describes conv(X). [Hint: use the TUM theory.]

Exercise 10

1. Prove that the polyhedron $P = \{(x_1, \ldots, x_m, y) \in \mathbb{R}^{m+1} : y \leq 1, x_i \leq y \text{ for } i = 1, \ldots, m\}$ has integer vertices. [Hint: start by writing the constraint matrix.]

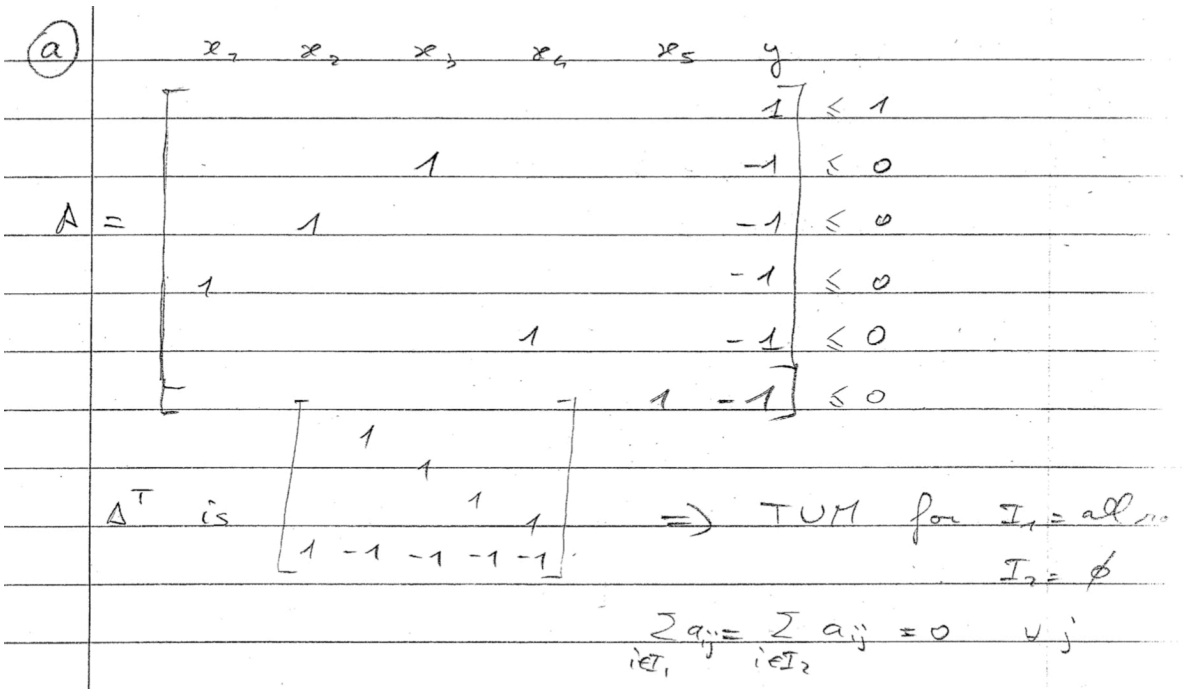
   Solution:

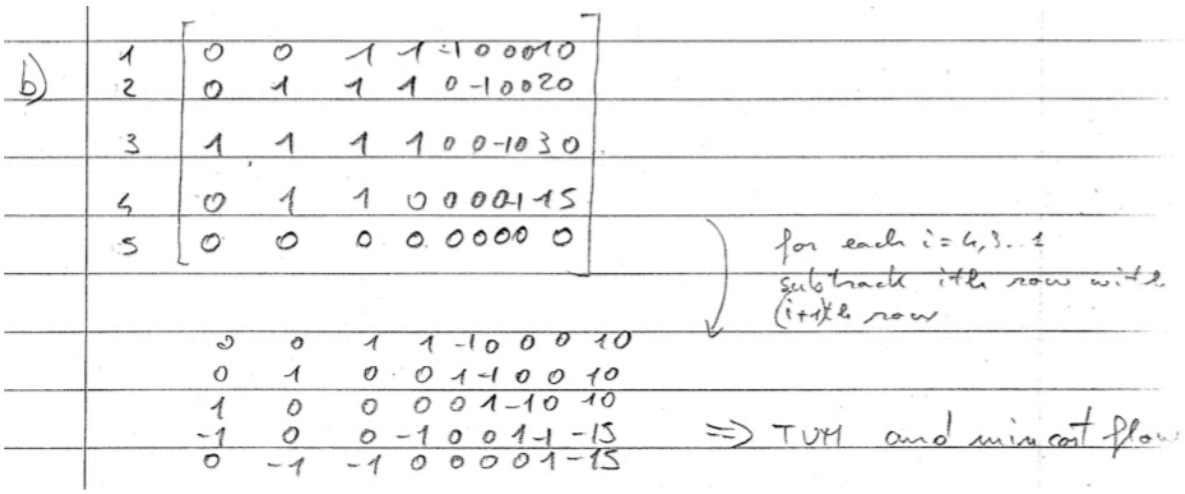
2. Consider the following (integer) linear programming problem:

$$\begin{aligned} \min \quad & c_1x_1 + c_2x_2 + c_3x_3 + c_3x_4 \\ & x_3 + x_4 \geq 10 \\ & x_2 + x_3 + x_4 \geq 20 \\ & x_1 + x_2 + x_3 + x_4 \geq 30 \\ & x_2 + x_3 \geq 15 \\ & x_1, x_2, x_3, x_4 \in \mathbb{Z}^+_0 \end{aligned} \tag{5}$$

   The constraint matrix has consecutive 1's in each column. Matrices with consecutive 1's property for each column are totally unimodular. Show that this fact holds for the specific numerical example (5). That is, show first that the constraint matrix of the problem has consecutive 1s in the columns and then that you can transform this matrix into one that you should recognize to be a TUM matrix. [Hint: rewrite the problem in standard form (that is, in equation form) and add a redundant row $\mathbf{0} \cdot \mathbf{x} = 0$ to the set of constraints. Then perform elementary row operations to bring the matrix to a known form.]

   Solution:


a)
x_1 x_2 x_3 x_4 x_5 y
$A =$
≤ 1
≤ 0
≤ 0
≤ 0
≤ 0
≤ 0
A^T is
$\Rightarrow$ TUM for I_1 = all ...
$I_2 = \emptyset$
$\sum_{i \in I_1} a_{ij} = \sum_{i \in I_2} a_{ij} = 0 \quad \forall j$


b)
for each i = 4, 3, .. 1
subtract ith row with (i+1)th row
$\Rightarrow$ TUM and min cost flow

3. Use one of the two previous results to show that the *shift scheduling problem* in Exercise 1 of this Sheet can be solved efficiently when formulated as a mathematical programming problem. (You do not need to find numerical results.)

Solution:

c)

$$\min \quad 720x_1 + 800x_2 + 740x_3 + 680x_4 + 720x_5 + 780x_6 + 640x_7$$

$$0-6: \quad x_1 + x_2 \geq 2$$

$$6-8: \quad x_2 + x_3 \geq 8$$

$$8-11: \quad x_3 + x_4 \geq 5$$

$$11-14: \quad x_3 + x_4 \geq 7$$

$$14-16: \quad x_4 + x_5 \geq 3$$

$$16-18: \quad x_5 + x_6 \geq 6$$

$$18-20: \quad x_5 + x_6 + x_7 \geq 6$$

$$20-22: \quad x_5 + x_6 + x_7 \geq 3$$

$$22-24: \quad x_6 + x_7 \geq 1$$

$x_1, x_2 \dots x_7 \geq 0$ and integer.

The matrix has consecutive 1's property on cols

hence LP relax gives integer results.

Exercise 11*

This is a continutation of the Factory Planning problem from the computer lab class Sheet 5. The setting is the multiperiod problem discussed in tasks 2 and 3.
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 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?
- 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:

$$(??), (??) - (??) \tag{6}$$

$$\sum_i a_{ij} x_{it} \leq 384(c_j - y_{j,t}) \qquad j \in \{GR, VD, HD, BR, PL\}, t = 1 \ldots, 6 \tag{7}$$

$$\sum_{t=1}^{6} y_{j,t} = c_j \qquad j \in \{VD, HD, BR, PL\} \tag{8}$$

$$\sum_{t=1}^{6} y_{GR,t} = 2 \qquad j \in \{VD, HD, BR, PL\} \tag{9}$$

$$y_{j,t} \in \mathbb{Z}_0^+ \qquad j \in \{GR, VD, HD, BR, PL\}, t = 1 \ldots, 6 \tag{10}$$

$$\tag{11}$$

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 2.
The solution 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))

    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
```

```
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
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!