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

April 20, 2018

Marco Chiarandini

DM559/DM545 – Linear and Integer Programming

Computer Lab 2, Spring 2018

Solution:

Contains Solutions!
The problem was one of Gurobi Examples:
`http://www.gurobi.com/resources/examples/factory-planning-I`
There is also a video: `https://youtu.be/vnLc_3VnVcw?t=32m51s`
You find the solutions also in this document.

Factory Planning and Machine Maintenance

A firm makes seven products $1, \ldots, 7$ on the following machines: 4 grinders, 2 vertical drills, 3 horizontal drills, 1 borer, and 1 planer.
Each product yields a certain contribution to the profit (defined as selling price minus cost of raw materials expressed in Euro/unit). These quantities (in Euro/unit) together with the production times (hours/unit) required on each process are given below.

product	1	2	3	4	5	6	7
profit	10	6	8	4	11	9	3
grinding	0.5	0.7	0	0	0.3	0.2	0.5
vdrill	0.1	0.2	0	0.3	0	0.6	0
hdrill	0.2	0	0.8	0	0	0	0.6
boring	0.05	0.03	0	0.07	0.1	0	0.08
planning	0	0	0.01	0	0.05	0	0.05

In the first month (January) and the five subsequent months certain machines will be down for maintenance. These machines will be:

January	1 grinder
February	2 hdrill
March	1 borer
April	1 vdrill
May	1 grinder
May	1 vdrill
June	1 planer
June	1 hdrill

There are marketing limitations on each product in each month. That is, in each month the amount sold for each product cannot exceed these values:

product	1	2	3	4	5	6	7
January	500	1000	300	300	800	200	100
February	600	500	200	0	400	300	150
March	300	600	0	0	500	400	100
April	200	300	400	500	200	0	100
May	0	100	500	100	1000	300	0
June	500	500	100	300	1100	500	60

It is possible to store products in a warehouse. The capacity of the storage is 100 units per product type per month. The cost is 0.5 Euro per unit of product per months. There are no stocks in the first month but it is desired to have a stock of 50 of each product type at the end of June.

The factory works 6 days a week with two shifts of 8 hours each day. (It can be assumed that each month consists of 24 working days.)
The factory wants to determine a production plan, that is, the quantity to produce, sell and store in each month for each product, that maximizes the total profit.

Task 1

Model the factory planning problem for the month of January as an LP problem.

Solution:

The assignment is taken from the book [Wi].
The objective is to find the optimum "product mix" subject to the production capacity and the marketing limitations. If storage of single products is not allowed, the model for January can be formulated as follows. Let the real variables x_i represent the quantities of product i to be made. Let GR, VD, HD, BR and PL stand for, respectively, grinding, vertical drilling, horizontal drilling, boring and planing. Let the total working hours for each machine be $8 * 2 * 24 = 384$.

$$
\begin{array}{rrcrcrcrcrcrcrcrl}
\max & 10x_1 & + & 6x_2 & + & 8x_2 & + & 4x_4 & + & 11x_5 & + & 9x_6 & + & 3x_7 & \\
GR: & 0.5x_1 & + & 0.7x_2 & + & & + & & + & 0.3x_5 & + & 0.2x_6 & + & 0.5x_7 & \leq 1152 \\
VD: & 0.1x_1 & + & 0.2x_2 & & & + & 0.3x_4 & & & + & 0.6x_6 & & & \leq 768 \\
HD: & 0.2x_1 & & & + & 0.8x_3 & & & & & & & + & 0.6x_7 & \leq 1152 \\
BR: & 0.05x_1 & + & 0.03x_2 & & & + & 0.07x_4 & + & 0.1x_5 & & & + & 0.08x_7 & \leq 384 \\
PL: & & & & & 0.01x_3 & & & + & 0.05x_5 & & & + & 0.05x_7 & \leq 384 \\
\multicolumn{15}{r}{x_1 \leq 500,\ x_2 \leq 1000,\ x_3 \leq 300,\ x_4 \leq 300,\ x_5 \leq 800,\ x_6 \leq 200,\ x_7 \leq 100}
\end{array}
$$

The single-period problems for the other months would be similar apart from different market bounds, and different capacity figures for the different types of machine.
The matrix has no special structure, the coefficients are not just $\{-1, 1, 0\}$ as in a TUM matrix and non zeros can appear everywhere. The matrix is not necessarily sparse.

Task 2

Model the multi-period (from January to June) factory planning problem as an LP problem. Use mathematical notation and indicate in general terms how many variables and how many constraints your model has.

Solution:

It is necessary to distinguish for each month the quantities of each product manufactured from the quantities sold and held over in storage. These quantities must be represented by different variables. Let the quantities of product i manufactured, sold, and held over in successive months t be represented by variables x_{it}, s_{it}, h_{it}, $t = 1, \ldots, 6$.
A convenient way to represent the link between these variables is shown in Figure 1. Hence, the mass balance constraints to be imposed are:

$$h_{i,t-1} + x_{it} = s_{it} + h_{it}$$

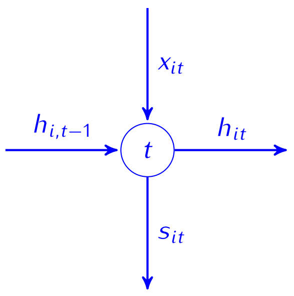

Figure 1: Mass balance constraint at each time period.

Initially (month 0), there is nothing held in stock but finally (month 6) there are (at least) 50 of each product held. This relation involving product 1 gives rise to the following constraints:

$$\begin{array}{rrrrl} & x_{11} & - s_{11} & -h_{11} & = 0 \\ h_{11} & +x_{12} & - s_{12} & -h_{12} & = 0 \\ h_{12} & +x_{13} & - s_{13} & -h_{13} & = 0 \\ h_{13} & +x_{14} & - s_{14} & -h_{14} & = 0 \\ h_{14} & +x_{15} & - s_{15} & -h_{15} & = 0 \\ h_{15} & +x_{16} & - s_{16} & & = 50 \end{array}$$

Similar constraints must be specified for the other six products. It may be more convenient to define also variables h_{16}, h_{26}, etc, and fix them at the value 50.
The general model is:

$$\max \sum_{i=1}^{7}\sum_{t=1}^{6} p_i s_{it} - \sum_{i=1}^{7}\sum_{t=1}^{6} f h_{it} \qquad (1)$$

$$\sum_i a_{ij} x_{it} \leq 384(c_j - m_{j,t}) \qquad j \in \{GR, VD, HD, BR, PL\}, t = 1\ldots,6 \qquad (2)$$

$$h_{i,t-1} + x_{it} - s_{it} - h_{it} = 0 \qquad i = 1,\ldots,7; t = 1,\ldots,6 \qquad (3)$$

$$s_{it} \leq u_{it} \qquad i = 1,\ldots,7; t = 1,\ldots,6 \qquad (4)$$

$$h_{it} \leq 100 \qquad i = 1,\ldots,7; t = 1,\ldots,6 \qquad (5)$$

$$s_{it}, x_{it}, h_{it} \geq 0 \qquad i = 1,\ldots,7; t = 1,\ldots,6 \qquad (6)$$

$$h_{i0} = 0, h_{i6} = 50 \qquad i = 1,\ldots,7 \qquad (7)$$

In the objective function (1) the "selling" variables are given the appropriate "unit profit" p_i and the "holding" variables the coefficients of $f = 0.5$. Constraints (2) are the resource constraints where c_m is the capacity for each resource m. Constraints (3) are the mass balance constraints described above and constraints (4) are the marketing limitations where u_{it} are product upper bounds.
The resulting model has the following dimensions:

$6 \times 7 = 42$	manufacturing variables
$6 \times 7 = 42$	selling variables
$6 \times 7 = 42$	holding variables
Total 126	variables
$6 \times 5 = 30$	capacity constraints
$6 \times 7 = 42$	monthly linking constraints
$6 \times 7 = 42$	marketing limitations
$6 \times 7 = 42$	holding quantity constraints
Total 156	constraints

We typically do not count positivity constraints, as those are standard.

If we present the problem in a diagrammatic form we obtain the illustration on the left of Figure 2.
The matrix is not apparently TUM. It has however a *block angular structure*. A *block angular structure* is made by common rows and blocks in diagonal representing submodels. In our case the common rows are the linking equality constraints of mass balance while the submodels are the per period production planning as the one seen in Task 1. Clearly, a matrix with *block angular structure* without common constraints could be decomposed and each submodel solved separately. Nevertheless advanced techniques exist to handle efficiently problems with block angular structure. A typical problem with this structure often used in examples is the multi-commodity flow problem (we will see this in one of the next classes).
Another type of structure which may arise in multi-period models is the *staircase* structure which is illustrated in Figure 2, right. In fact a staircase structure such as this could be converted into a block angular structure. If alternate "steps" such as (A_0, B_1), (A_2, B_3) were treated as subproblem constraints and the intermidiate "steps" as common rows we would have a block angular structure.

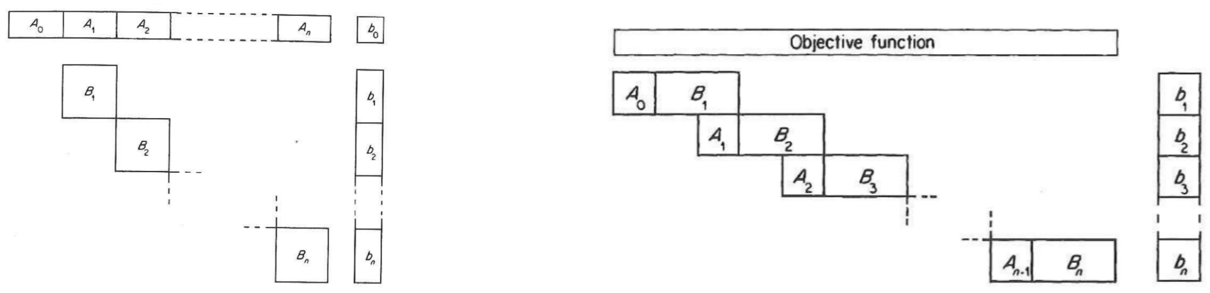

Figure 2: On the left a block angular structure and on the right a staircase structure

Task 3

Implement the multi-period model in Python and Gurobi and solve the problem on the data given. A starting script containing the numerical data is available at multiperiod.py.

- Report and comment relevant information from the run of Gurobi on the data.
- Report the production plan, that is, how much of each product should factory produce in each month.
- 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 (3) 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

(This task needs Integer Programming. The task is optional in the sense that a failure to solve the task does not influence the decision passed/not passed for the whole assignment, but a solution to the task must be attempted and reported!)
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 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 point B 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 the place down times?

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:

$$\begin{aligned}
&(1), (3) - (7) && && (8)\\
&\sum_i a_{ij} x_{it} \leq 384(c_j - y_{j,t}) && j \in \{GR, VD, HD, BR, PL\}, t = 1\dots, 6 && (9)\\
&\sum_{t=1}^{6} y_{j,t} = c_j && j \in \{VD, HD, BR, PL\} && (10)\\
&\sum_{t=1}^{6} y_{GR,t} = 2 && j \in \{VD, HD, BR, PL\} && (11)\\
&y_{j,t} \in \mathbb{Z}_0^+ && j \in \{GR, VD, HD, BR, PL\}, t = 1\dots, 6 && (12)\\
& && && (13)
\end{aligned}$$

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
```

```
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
```

```
  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!