Solution in TPK4191 - Production optimization and control

Jørn Vatn
Email: jorn.vatn@ntnu.no

September 8, 2024

Problem 2.6

Decision variables

Introduce:

- x_1 to x_6 are order size for each month (October = 1, November = 2, etc).
- x_7 to x_{12} are inventory level at end of month (October = 7, November = 8, etc)
- y_1 to y_6 are binary variables, 1 = Order, 0 = Do not order

Objective function

The cost vector for the x-variables is defined by the following Python statement:

```
c = [100 , 100 , 110 , 120 , 100 , 100 , 5, 5, 5, 5, 5, 5]
```

Note that since `linprog` cannot handle binary variables, the y-variables are not included in the objective function. However, we use the y-variables as part of the constraint definition.

Constraint modelling

The following constraints ensures that we do not order more than 50 units, if we order:

$$1 \cdot x_i \leq 50 \cdot y_i, i = 1, \ldots, 6$$

which translates to:

$$1 \cdot x_i - 50 \cdot y_i \leq 0, i = 1, \ldots, 6$$

if we included the y-variables as part of the LP-model. However since we will not include the y-variables as part of our LP-model, we stick to $1 \cdot x_i \leq 50 \cdot y_i, i = 1, \ldots, 6$ and the Python code when we consider a given value of the y-vector reads:

```
    for i in range(nMonths):
        A_i = np.zeros([nVar])
        A_i[i] = 1
        LP.upperBound(maxOrder[i]*y[i],A_i)
```

To fulfil the demand, i.e., d[i] is the demand in month i we have the following equality constraint:

$$1 \cdot x_i + 1 \cdot x_{i+6-1} - 1 \cdot x_{i+6} = d_i, i = 1, \ldots, 6$$

where "6-1" is the shift to address the inventory level at the beginning of each month, and "6-1" is the shift to address the inventory level at the end of each month. Special attention should be paid to the first month were there is "no inventory level". The Python code reads:

```
    for i in range(nMonths):
        A_i = np.zeros([nVar])
        A_i[i] = 1
        A_i[i+nMonths] = -1
        if i > 0:
            A_i[i+nMonths-1] = 1
        LP.equalityConstraint(d[i],A_i)
```

Finally, inventory level at end of each month cannot exceed 30, which in Python reads:

```
    for i in range(nMonths):
        A_i = np.zeros([nVar])
        A_i[i + nMonths]= 1
        LP.upperBound(30,A_i)
```

To optimize the idea is test all combinations of the y-vector. For each value of y we run the `linprog` to find the corresponding x-vector, and the value of the objective function. Since the `linprog` model does not include the fixed cost, c_F , we add the fixed cost in the comparison. The essential Python code reads:

```
best=1e30
yComb = LP.all_binary_combinations(6)
for y in yComb:
    #
    # Here come the specification of the cost coefficients of the objective
    # function and all the constraints as specified above
    #
    res=LP.lpSolve(True)
    if res.success:
```

```
        test = res.fun + sum(y)*c_F
        if test < best:
            best = test
            best_y = y
            best_x = x
```

A simplified approach and Excel failure

In the first version of the solution, I made the assumption that it was only required to test the situation where we only in one month did not made an order. That is, we could test the following y-vectors: `[0,1,1,1,1,1]`, `[1,0,1,1,1,1]`, ..., `[1,1,1,1,1,0]`. This resulted in that the best option was not to order the last month.

However this solution was not optimal. The optimal solution is to in October, November, January and February as found by the solution indicated above. Therefore the updated solution where all possible combination of the y-vector was investigated is the only guarantee that we obtain the optimal solution.

Note also that Excel *fails* to find the optimal solution. This is why I was "fooled" in the first place, since the Excel solution showed that it was only one month where we should not place an order, I assumed that it was only required to test month by month.... The first version of the solution is available in the solution folder: `Problem2_6testOneByOne.py`.