Sensitivity Analysis and Revised Simplex

nesko25

1 - Mathematical Formulation

Given the data, the factory-planning problem model will look like this:

$$\begin{aligned} \max\ & 19x_1 + 13x_2 + 12x_3 + 17x_4 \\ s.t.\ & 3x_1 + 2x_2 + x_3 + 2x_4 \leq 225 \\ & x_1 + x_2 + x_3 + x_4 \leq 117 \\ & 4x_1 + 3x_2 + 3x_3 + 4x_4 \leq 420 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

We can implement it in python and bring to the equational standard form to get the initial tableau:

```
import numpy as np
from fractions import Fraction
from util import tableau

n = 4
m = 3

# model in standard form
A = np.array([
    [3, 2, 1, 2],
    [1, 1, 1, 1],
    [4, 3, 3, 4]
])
b = np.array([225, 117, 420])
c = np.array([19, 13, 12, 17])

# equational standard form
```

```
A = np.hstack([A, np.identity(m)])
c = np.hstack([c, np.zeros(m)])

# inital tableau
T = np.column_stack([
    np.vstack([A, c]),
    np.append(np.zeros(m), [1]), # -z
    np.append(b, [0]) # b
])

# make it pretty
T = np.array(T, dtype = object)
for j in range(m + 1):
    for i in range(n + m + 2):
        T[j][i] = Fraction(T[j][i])

tableau(T)
```

x1	x2	x3	x4	x5	x6	x7	-z	b
3	2	1	2	1	0	0	0	225
1	1	1	1	0	1	0	0	117
4	3	3	4	0	0	1	0	420
19	13	12	17	0	0	0	1	0

2 - Optimal Tableau

Using Original Simplex

Choose variable to bring to the basis

```
def ratio_test():
    enter = np.argmax(T[-1][:n + m])
    if T[-1][enter] <= 0:
        return [None, None] # optimal

    ratios = [
        T[j][-1] / T[j][enter]
        if T[j][enter] > 0
        else float('inf')
        for j in range(m)
    ]

    leave = np.argmin(ratios)
    if ratios[leave] == float('inf'):
        return [None, None] # unbounded

    return [enter, leave]

enter, leave = ratio_test()
print(f"pivot: ({enter}, {leave})")
```

```
pivot: (0, 0)
```

So according to the ratio_test x_1 should enter and x_5 should leave

```
def update_tableau(enter, leave):
    T[leave] /= T[leave][enter]
    for j in range(m + 1):
        if j != leave:
            T[j] -= T[leave] * T[j][enter]

update_tableau(enter, leave)
tableau(T)
```

x1	x2	x3	x4	x5	x6	x7	-z	b
1	2/3	1/3	2/3	1/3	0	0	0	75
0	1/3	2/3	1/3	-1/3	1	0	0	42
0	1/3	5/3	4/3	-4/3	0	1	0	120
0	1/3	17/3	13/3	-19/3	0	0	1	-1425

And we repeat:

```
def pivot():
    enter, leave = ratio_test()
    update_tableau(enter, leave)
    tableau(T)

pivot()
```

x1	x2	x3	x4	x5	x6	x7	-z	b
1	1/2	0	1/2	1/2	-1/2	0	0	54
0	1/2	1	1/2	-1/2	3/2	0	0	63
0	-1/2	0	1/2	-1/2	-5/2	1	0	15
0	-5/2	0	3/2	-7/2	-17/2	0	1	-1782

```
pivot()
```

x1	x2	x3	x4	x5	x6	x7	-z	b
1	1	0	0	1	2	-1	0	39
0	1	1	0	0	4	-1	0	48
0	-1	0	1	-1	-5	2	0	30
0	-1	0	0	-2	-1	-3	1	-1827

And we've reached optimality as there are no more positive reduced costs. The solution agrees with the statement in the exercise that in the optimal solution x_1, x_3 and x_4 will be in basis.

Using Revised Simplex

```
B = [0, 2, 3] # final basis
N = [1, 4, 5, 6]

A_Binv = np.linalg.inv(A[:, B])

# A and c - basic
T[:m, B] = np.identity(m)
T[-1, B] = np.zeros(m)

# A and c - nonbasic
T[:m, N] = A_Binv @ A[:, N]
T[-1, N] = c[N] - c[B] @ T[:m, N]

# b and d
T[:m, -1] = A_Binv @ b
T[-1, -1] = - c[B] @ (T[:m, -1])

# make it pretty again
T = np.array(T, dtype = object)
for j in range(m + 1):
    for i in range(n + m + 2):
        T[j][i] = Fraction(round(T[j][i]))

tableau(T)
```

x1	x2	x3	x4	x5	x6	x7	-z	b
1	1	0	0	1	2	-1	0	39
0	1	1	0	0	4	-1	0	48
0	-1	0	1	-1	-5	2	0	30
0	-1	0	0	-2	-1	-3	1	-1827

We achieved the exact same tableau with non-positive reduced cost therefore we found the optimal solution - again.

3 - Reduced Cost

What's the increase in price that would make x_2 worth being produced?
We remember that:

$$\bar{c}_2 = c_2 - \sum_i y_i a_{i2}$$

where $\bar{c}_2$ is the reduced cost of x_2 after last iteration.
To make x_2 worth producing the reduced cost should be positive, and from the above formula we see that the increase in price will correspond to the same increase in the reduced cost. As the reduced cost of x_2 is equal to -1 after the last iteration, the increase in price that would make x_2 worth being produced is any amount > 1.

4 - Shadow Price

Shadow prices can be read directly from the last tableau:

$$2 - for\ an\ hour\ of\ work$$
$$1 - for\ a\ unit\ of\ metal$$
$$3 - for\ a\ unit\ of\ wood$$

5 - Complementary Slackness Theorem

Are all the constraints binding?
The Complementary Slackness Theorem implies for $x*$ and $y*$ - optimal solutions of the Primal and Dual problems respectively:

$$\forall_j \ (b_j - \sum_i a_{ji} x*_i) y*_j = 0$$

Which means that for each constraint it's either binding, or it's shadow price is 0. Seeing that all of the shadow prices are non-zero (previous subtask) we can conclude that all constraint of the Primal problem are binding (active).
On a side note: We can also observe that none of the slack variables are in basis in the optimal solution, which means there is no slack capacity, which is yet another way of concluding the above.

6 - Economical Interpretation

The imputed (reduced) cost of x_2 is:

$$\bar{c_2} = c_2 - \sum_i y *_i a_{i2} = -1 < 0$$

Which means that the cost of resources needed to produce a unit of product x_2 is significantly higher than the price of the product itself, making it not a viable option to produce.
In other words: there is slack in the 2^{nd} constraint of the Dual.

7 - Sensitivity analysis - increase desk net profit

TODO