MATH 340, 2024/25, Term 2, Assignment 2

Mercury Mcindoe 85594505

January 28, 2025

1. Consider the problem [Vanderbei 5th edition, Exercise 1.1].

   (a) Write this as a Linear Programming problem in the standard inequality form). You must explain your notation and variables.

   Solution:

   Let's denote the variables for Bands and Coils as x_b, x_c respectively. Then our goal is to maximize $25x_b + 30x_c$ while satisfying the constraints

   $$x_b \leq 6000, x_c \leq 4000, x_b \geq 0, x_c \geq 0, \frac{x_b}{200} + \frac{x_c}{140} \leq 40$$

   Thus, we get the LP Problem in the standard inequality form,

   $$\begin{aligned} \text{maximize} \quad & 25x_b + 30x_c \\ \text{subject to} \quad & \frac{x_b}{200} + \frac{x_c}{140} \leq 40, \\ & x_b \leq 6000, \\ & x_c \leq 4000, \\ & x_b, x_c \geq 0. \end{aligned}$$

   (b) Solve the LP by writing down a code in the Python language using the Jupyter notebook; login to UBC syzygy website and the Jupyter notebook. Attach the pdf file that include both the code and the results; note that you can save the Jupyter notebook as a pdf file.

   Solution:

In [2]:
```
from pulp import *
```

Problem 1

In [3]:
```
Lp_prob = LpProblem('Question1b', LpMaximize)
x_b = LpVariable('x_b') # Bands
x_c = LpVariable('x_c') # Coils
```

In [4]:
```
# Objective function
Lp_prob += 25 * x_b + 30 * x_c

# Constraints
Lp_prob += x_b <= 6000
Lp_prob += x_c <= 4000
Lp_prob += (x_b  * (1 / 200)) + (x_c * (1 / 140)) <= 40
Lp_prob += x_b >= 0
Lp_prob += x_c >= 0
```

In [5]:
```
print(Lp_prob)
```

```
Question1b:
MAXIMIZE
25*x_b + 30*x_c + 0
SUBJECT TO
_C1: x_b <= 6000

_C2: x_c <= 4000

_C3: 0.005 x_b + 0.00714285714286 x_c <= 40

_C4: x_b >= 0

_C5: x_c >= 0

VARIABLES
x_b free Continuous
x_c free Continuous
```

In [6]:
```
Lp_prob.solve()
LpStatus[Lp_prob.status]
```

```
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - /Users/mercurymcindoe/Documents/Mercury/UBC/CPEN 4-2/MATH 34
0/Assignments/.venv/lib/python3.13/site-packages/pulp/solverdir/cbc/osx/64/c
bc /var/folders/py/b14h3jpn1036ckyvg60q2fp40000gn/T/da9925a0843d4266b9c7c2a9
b4bc01c8-pulp.mps -max -timeMode elapsed -branch -printingOptions all -solut
ion /var/folders/py/b14h3jpn1036ckyvg60q2fp40000gn/T/da9925a0843d4266b9c7c2a
9b4bc01c8-pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 10 COLUMNS
At line 19 RHS
At line 25 BOUNDS
At line 28 ENDATA
Problem MODEL has 5 rows, 2 columns and 6 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Presolve 1 (-4) rows, 2 (0) columns and 2 (-4) elements
0  Obj -0 Dual inf 65.714284 (2)
1  Obj 192000
Optimal - objective value 192000
After Postsolve, objective 192000, infeasibilities - dual 0 (0), primal 0 (
0)
Optimal objective 192000 - 1 iterations time 0.002, Presolve 0.00
Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.00   (Wallclock seconds):       0.01
```

Out[6]: 'Optimal'

In [7]:
```
print("x = ", value(x_b), ", y = ", value(x_c))
print("Optimal Solution: ", value(Lp_prob.objective), " dollars")
```

```
x =  6000.0 , y =  1400.0
Optimal Solution:  192000.0  dollars
```

2. Consider the problem [Vanderbei. 5th edition, Exercise 1.2].

   (a) Write this as a Linear Programming problem (in the standard inequality form). You must explain your notation and variables.
   Solution:
   We denote the variables as the following:

$$
\begin{aligned}
x_{\text{INY}} &: \text{ Ithaca–Newark-Y} \\
x_{\text{INB}} &: \text{ Ithaca–Newark-B} \\
x_{\text{INM}} &: \text{ Ithaca–Newark-M} \\
x_{\text{NBY}} &: \text{ Newark–Boston-Y} \\
x_{\text{NBB}} &: \text{ Newark–Boston-B} \\
x_{\text{NBM}} &: \text{ Newark–Boston-M} \\
x_{\text{IBY}} &: \text{ Ithaca–Boston-Y} \\
x_{\text{IBB}} &: \text{ Ithaca–Boston-B} \\
x_{\text{IBM}} &: \text{ Ithaca–Boston-M}
\end{aligned}
$$

   The LP Problem is,

$$
\begin{aligned}
\text{maximize} \quad & 300x_{\text{INY}} + 220x_{\text{INB}} + 100x_{\text{INM}} \\
& + 160x_{\text{NBY}} + 130x_{\text{NBB}} + 80x_{\text{NBM}} \\
& + 360x_{\text{IBY}} + 280x_{\text{IBB}} + 140x_{\text{IBM}} \\
\text{subject to} \quad & x_{\text{INY}} \le 4 \\
& x_{\text{INB}} \le 8 \\
& x_{\text{INM}} \le 22 \\
& x_{\text{NBY}} \le 8 \\
& x_{\text{NBB}} \le 13 \\
& x_{\text{NBM}} \le 20 \\
& x_{\text{IBY}} \le 3 \\
& x_{\text{IBB}} \le 10 \\
& x_{\text{IBM}} \le 18 \\
& x_{\text{INY}} + x_{\text{INB}} + x_{\text{INM}} + x_{\text{IBY}} + x_{\text{IBB}} + x_{\text{IBM}} \le 30 \\
& x_{\text{NBY}} + x_{\text{NBB}} + x_{\text{NBM}} + x_{\text{IBY}} + x_{\text{IBB}} + x_{\text{IBM}} \le 30 \\
& x_{\text{INY}}, x_{\text{INB}}, x_{\text{INM}}, x_{\text{NBY}}, x_{\text{NBB}}, x_{\text{NBM}}, x_{\text{IBY}}, x_{\text{IBB}}, x_{\text{IBM}} \ge 0
\end{aligned}
$$

   (b) Solve the LP by writing down a code in the Python language using the Jupyter notebook; login to UBC syzygy website and the Jupyter notebook. Attach the pdf file that include both the code and the results; note that you can save the Jupyter notebook as a pdf file.
   Hint1: You will need integer variables, those taking only integer values. For that you can add the command cat='Integer' like in the following example:
   ticketvars = LpVariable.dicts("ticket", ticket, lowBound=0, cat='Integer')
   Here the last part cat='Integer' restrict the variables to be integer variables.]
   Hint2: It is not necessary, but, to practice with 'for-loops' and 'dictionary', you can try to use dictionary variables as done in the blending (cat food) example.]
   Solution:

Assignment_2_q2

January 28, 2025

0.0.1 Problem 2

[14]:
```
from pulp import *
```

[15]:
```
# Dictionary Setup
Passengers = ['INY','INB','INM','NBY','NBB','NBM','IBY','IBB','IBM']
revenues = {
    'INY' : 300,
    'INB' : 220,
    'INM' : 100,
    'NBY' : 160,
    'NBB' : 130,
    'NBM' : 80,
    'IBY' : 360,
    'IBB' : 280,
    'IBM' : 140
}

forecast = {
    'INY' : 4,
    'INB' : 8,
    'INM' : 22,
    'NBY' : 8,
    'NBB' : 13,
    'NBM' : 20,
    'IBY' : 3,
    'IBB' : 10,
    'IBM' : 18
}
```

[20]:
```
prob = LpProblem("Question2b", LpMaximize)
# variables
num_passengers = LpVariable.dicts("passengers", Passengers, lowBound=0,␣
↪cat='Integer')

# objective function
prob += lpSum([revenues[i] * num_passengers[i] for i in Passengers])
```

```python
#constraints
for i in Passengers :
    prob += num_passengers[i] <= forecast[i]

prob += num_passengers["INY"] + num_passengers["INB"] + num_passengers["INM"] +␣
 ↪num_passengers["IBY"] + num_passengers["IBB"] + num_passengers["IBM"] <= 30
prob += num_passengers["NBY"] + num_passengers["NBB"] + num_passengers["NBM"] +␣
 ↪num_passengers["IBY"] + num_passengers["IBB"] + num_passengers["IBM"] <= 30

print(prob)
```

```
Question2b:
MAXIMIZE
280*passengers_IBB + 140*passengers_IBM + 360*passengers_IBY +
220*passengers_INB + 100*passengers_INM + 300*passengers_INY +
130*passengers_NBB + 80*passengers_NBM + 160*passengers_NBY + 0
SUBJECT TO
_C1: passengers_INY <= 4

_C2: passengers_INB <= 8

_C3: passengers_INM <= 22

_C4: passengers_NBY <= 8

_C5: passengers_NBB <= 13

_C6: passengers_NBM <= 20

_C7: passengers_IBY <= 3

_C8: passengers_IBB <= 10

_C9: passengers_IBM <= 18

_C10: passengers_IBB + passengers_IBM + passengers_IBY + passengers_INB
 + passengers_INM + passengers_INY <= 30

_C11: passengers_IBB + passengers_IBM + passengers_IBY + passengers_NBB
 + passengers_NBM + passengers_NBY <= 30

VARIABLES
0 <= passengers_IBB Integer
0 <= passengers_IBM Integer
0 <= passengers_IBY Integer
0 <= passengers_INB Integer
```

```
0 <= passengers_INM Integer
0 <= passengers_INY Integer
0 <= passengers_NBB Integer
0 <= passengers_NBM Integer
0 <= passengers_NBY Integer
```

[17]:
```python
prob.solve()
print("Status: ", LpStatus[prob.status])
```

```
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - /Users/mercurymcindoe/Documents/Mercury/UBC/CPEN 4-2/MATH
340/Assignments/.venv/lib/python3.13/site-packages/pulp/solverdir/cbc/osx/64/cbc
/var/folders/py/b14h3jpn1036ckyvg60q2fp40000gn/T/e59fe6404c734b139a62d5602ea4a18
8-pulp.mps -max -timeMode elapsed -branch -printingOptions all -solution /var/fo
lders/py/b14h3jpn1036ckyvg60q2fp40000gn/T/e59fe6404c734b139a62d5602ea4a188-
pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 25 COLUMNS
At line 83 RHS
At line 104 BOUNDS
At line 114 ENDATA
Problem MODEL has 20 rows, 9 columns and 30 elements
Coin0008I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Continuous objective value is 9790 - 0.00 seconds
Cgl0004I processed model has 2 rows, 9 columns (9 integer (0 of which binary))
and 12 elements
Cutoff increment increased from 1e-05 to 9.9999
Cbc0012I Integer solution of -9790 found by DiveCoefficient after 0 iterations
and 0 nodes (0.01 seconds)
Cbc0001I Search completed - best objective -9790, took 0 iterations and 0 nodes
(0.01 seconds)
Cbc0035I Maximum depth 0, 0 variables fixed on reduced cost
Cuts at root node changed objective from -9790 to -9790
Probing was tried 0 times and created 0 cuts of which 0 were active after adding
rounds of cuts (0.000 seconds)
Gomory was tried 0 times and created 0 cuts of which 0 were active after adding
rounds of cuts (0.000 seconds)
Knapsack was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)
Clique was tried 0 times and created 0 cuts of which 0 were active after adding
rounds of cuts (0.000 seconds)
MixedIntegerRounding2 was tried 0 times and created 0 cuts of which 0 were
```

```
active after adding rounds of cuts (0.000 seconds)
FlowCover was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)
TwoMirCuts was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)
ZeroHalf was tried 0 times and created 0 cuts of which 0 were active after
adding rounds of cuts (0.000 seconds)

Result - Optimal solution found

Objective value:                9790.00000000
Enumerated nodes:               0
Total iterations:               0
Time (CPU seconds):             0.00
Time (Wallclock seconds):       0.01

Option for printingOptions changed from normal to all
Total time (CPU seconds):       0.00   (Wallclock seconds):       0.02

Status:  Optimal
```

[18]:
```python
for i in Passengers :
    print(f"Ticket ({i}): ",num_passengers[i].varValue)
```

```
Ticket (INY):  4.0
Ticket (INB):  8.0
Ticket (INM):  5.0
Ticket (NBY):  8.0
Ticket (NBB):  9.0
Ticket (NBM):  0.0
Ticket (IBY):  3.0
Ticket (IBB):  10.0
Ticket (IBM):  0.0
```

[19]:
```python
print("Max Revenue: ", value(prob.objective))
```

```
Max Revenue:  9790.0
```

3. For a nonempty $S \subset \mathbf{R}^n$ and a positive real number $r \in \mathbb{R}$ (and $r > 0$), define the set rS as follows:

$$rS := \{z \in \mathbf{R}^n \mid z = rx, x \in S\}$$

Here rx is the multiplication of the vector $x \in \mathbf{R}^n$ by the scalar $r \in \mathbf{R}$; in your more familiar notation, $r\vec{x}$. The set rS is the set of all points that are obtained by multiplying r with the vectors $x \in S$.

For a given nonempty $S \subset \mathbf{R}^n$ and a given positive number $r > 0$, prove that if S is a convex set then rS is a convex set as well.

Solution:

Since S is convex, for all $x, y \in S$ and $t \in [0, 1]$, we have

$$(1 - t)x + ty \in S.$$

Let $w = (1 - t)x + ty \in S$. For some $r \in \mathbf{R}^+$, we consider:

$$rw = r(1 - t)x + rty = (1 - t)(rx) + t(ry).$$

By the definition of rS, if $x, y, w \in S$, then $rx, ry, rw \in rS$.

Therefore, rS is also a convex set.

4. For given two nonempty sets $S_1, S_2 \subset \mathbf{R}^n$, define the operation $S_1 + S_2$ as follows:

$$S_1 + S_2 := \{z \in \mathbf{R}^n \mid \text{ there exist some } x \in S_1 \text{ and some } y \in S_2 \text{ such that } z = x + y\}$$

that is, each point $z \in S_1 + S_2$ is the one that can be expressed as the sum $x + y$ for some $x \in S_1$, and $y \in S_2$; here the sum $x + y$ is the vector sum between the two vectors. One does this for all $x \in S_1$ and $y \in S_2$ and get the set $S_1 + S_2$.

(a) Consider $S_1 = \{(x_1, x_2) \in \mathbf{R}^2 \mid |x_1 - 1| \leq 1 \& |x_2 - 2| \leq 1\}$ and $S_2 = \{(x_1, x_2) \in \mathbf{R}^2 \mid |x_1| \leq 2 \& |x_2| \leq 1\}$. Sketch the set $S_1 + S_2$. You do not need to explain your solution for this question. But, your sketch should be neat and very clear, indicating all the relevant coordinate values.

Solution:

(b) Is it true that $S_1 + S_2$ must be convex for **any** nonempty convex sets S_1 and S_2 in $\mathbf{R}^n$? Justify your answer carefully. [**This problem is independent of part (a). The sets S_1, S_2 are arbitrary convex sets in this question, not the particular example given in part (a). If you do this problem only for the sets of part (a) or a particular example, you will get zero mark.**]

Solution:

Let $z_1, z_2 \in S_1 + S_2$ such that $z_1 = x_1 + y_1, z_2 = x_2 + y_2$ and $x_1, x_2 \in S_1$, $y_1, y_2 \in S_2$. First, let

$$\begin{aligned} w &= w_1 + w_2 \\ &= (1 - t)z_1 + tz_2 \\ &= (1 - t)(x_1 + y_1) + t(x_2 + y_2) \\ &= \{(1 - t)x_1 + tx_2\} + \{(1 - t)y_1 + ty_2\}. \end{aligned}$$

for some $t \in [0, 1]$.
Since S_1 and S_2 are non-empty convex sets, we have:

$$w_1 = (1-t)x_1 + tx_2 \in S_1 \quad \text{and} \quad w_2 = (1-t)y_1 + ty_2 \in S_2.$$

By the definition of $S_1 + S_2$, we can write:

$$w = w_1 + w_2 = (1-t)z_1 + tz_2 \in S_1 + S_2, \quad t \in [0, 1],$$

proving that $S_1 + S_2$ is also a convex set.