IEOR 4004

Assignment #5

Zihui Zhu

Date

1. a. infeasible proved by Gurobi as shown in the code page

b. Let x_{ij} denote the capacity of edge (i,j). $i \in \{1,2,3,4,5\}$

$j \in \{6,7,8,9\}$

$C(i,j)$: cost of edge (i,j) (given by 1a)

$\Delta(i,j)$: Increase of capacity on edge (i,j)

$$\min \sum_i \sum_j C(i,j)\Delta(i,j) = 541\Delta(1,6) + 386\Delta(1,8) + 1512\Delta(1,9) + 234\Delta(2,6) + 899\Delta(2,7) + 103\Delta(2,8) + 1256\Delta(2,9) + 543\Delta(3,6) + 257\Delta(3,7) + 1653\Delta(3,8) + 1085\Delta(3,9) + 1785(4,6) + 227\Delta(4,7) + 1670\Delta(4,8) + 823\Delta(4,9) + 490\Delta(5,6) + 1233\Delta(5,7) + 1242\Delta(5,8) + 1841\Delta(5,9)$$

s.t

$x_{16} + x_{17} + x_{18} + x_{19} = 208$

$x_{26} + x_{27} + x_{28} + x_{29} = 193$

$x_{36} + x_{37} + x_{38} + x_{39} = 195$

$x_{46} + x_{47} + x_{48} + x_{49} = 209$

$x_{56} + x_{57} + x_{58} + x_{59} = 4031$

$-x_{16} - x_{26} - x_{36} - x_{46} - x_{56} = -1530$

$-x_{17} - x_{27} - x_{37} - x_{47} - x_{57} = -1583$

$-x_{18} - x_{28} - x_{38} - x_{48} - x_{58} = -1362$

$-x_{19} - x_{29} - x_{39} - x_{49} - x_{59} = -161$

$x_{16} \le 7407 + \Delta(1,6)$

$x_{17} \le 3586 + \Delta(1,7)$

$x_{18} \le 5072 + \Delta(1,8)$

$x_{19} \le 1932 + \Delta(1,9)$

$x_{26} \le 81 + \Delta(2,6)$

$x_{27} \le 90 + \Delta(2,7)$

$x_{28} \le 29 + \Delta(2,8)$

$x_{29} \le 902 + \Delta(2,9)$

$x_{36} \le 13 + \Delta(3,6)$

$x_{37} \le 8413 + \Delta(3,7)$

$x_{38} \le 8719 + \Delta(3,8)$

$x_{39} \le 7939 + \Delta(3,9)$

$x_{46} \le 5047 + \Delta(4,6)$

$x_{47} \le 83 + \Delta(4,7)$

$x_{48} \le 58 + \Delta(4,8)$

$x_{49} \le 76 + \Delta(4,9)$

$x_{56} \le 83 + \Delta(5,6)$

$x_{57} \le 7904 + \Delta(5,7)$

$x_{58} \le 73 + \Delta(5,8)$

$x_{59} \le 65 + \Delta(5,9)$

$\Delta(i,j) \ge 0$

$x_{ij} \ge 0 \quad \forall i,j$

⇒ solving by Gurobi:

Object value = 1749022

$\Delta(2,8) = 144$, $\Delta(5,6) = 1572$, $\Delta(5,8) = 855$

$x_{18} = 208$, $x_{28} = 173$, $x_{29} = 20$

$x_{38} = 195$, $x_{46} = 75$, $x_{48} = 58$, $x_{49} = 76$

$x_{56} = 1455$ $x_{57} = 1583$, $x_{58} = 928$, $x_{59} = 65$

other variables = 0

c variables remain the same definition as denoted in 1(b)

objective: $\min \sum_i \sum_j C(i,j)\, x_{ij} = 541x_{16} + 386x_{17} + 25x_{18} + 1512x_{19}$
$+ 734x_{26} + 899x_{27} + 103x_{28} + 1256x_{29}$
$+ 543x_{36} + 257x_{37} + 1653x_{38} + 1085x_{39}$
$+ 1785x_{46} + 227x_{47} + 1670x_{48} + 823x_{49}$
$+ 490x_{56} + 1233x_{57} + 1242x_{58} + 1841x_{59}$

constraints: same as 1(b). but & remove all $\Delta(i,j)$ from the inequalities

Using Gurobi to solve it:

obj value = 4600787

$x_{18} = 208$

$x_{28} = 173$

$x_{29} = 20$

$x_{38} = 195$

$x_{46} = 75$

$x_{48} = 58$

$x_{49} = 76$

$x_{56} = 1455$

$x_{57} = 1383$

$x_{58} = 928$

$x_{59} = 65$

other variables = 0

Date

d. decomposing the flow:

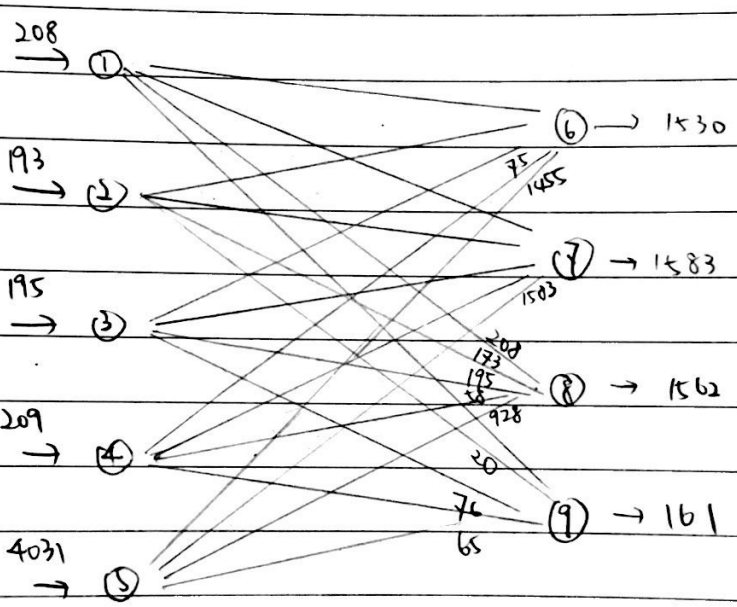
① → ⑧ * 208 +

② → ⑧ * 173 + ② → ⑨ * 20 +

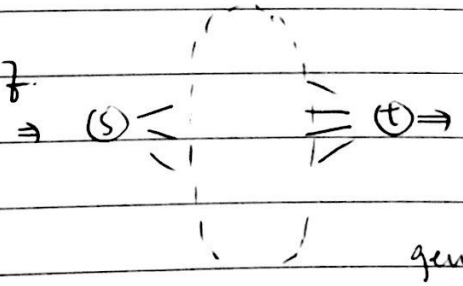
③ → ⑧ * 195 +

④ → ⑥ * 75 + ④ → ⑧ * 58 + ④ → ⑨ * 76 +

⑤ → ⑥ * 1455 + ⑤ → ⑦ * 1583 + ⑤ → ⑧ * 928 + ⑤ → ⑨ * 65



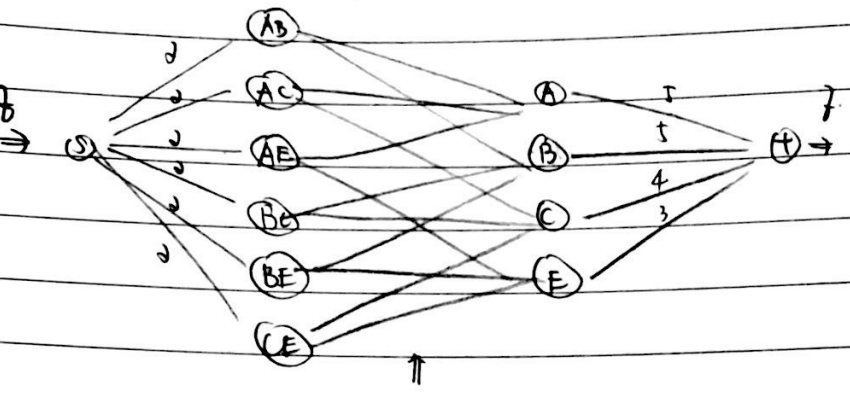
2



In the generic network, for each nodes except s,t, we duplicates them. Without loss of generality, we say for node v_i, we create a new node v_i' and connect them by an arcs $v_i v_i'$ with capacity equals ten. Then, connect v_i with all inflow arcs and v_i' with all outflow arcs. Solving this new generic network will give us the desired result.

(max flow problem)

3 Q. Decision: at most 8 games: all arrow to the right



formulate it as a max-flow problem: $X_{m,n}$ represent # of winnings of team m

max 7

Subject to: $X_{S,AB} + X_{S,AC} + X_{S,AE} + X_{S,BC} + X_{S,BE} + X_{S,CE} = 7$

$X_{S,AB} = X_{AB,A} + X_{AB,B}$

$X_{S,AC} = X_{AC,A} + X_{AC,C}$

$X_{S,AE} = X_{AE,A} + X_{AE,E}$

$X_{S,BC} = X_{BC,B} + X_{BC,C}$

$X_{S,BE} = X_{BE,B} + X_{BE,E}$

$X_{S,CE} = X_{CE,C} + X_{CE,E}$

$X_{AB,A} + X_{AC,A} + X_{AE,A} = X_{A,t}$

$X_{AB,B} + X_{BC,B} + X_{BE,B} = X_{B,t}$

$X_{AC,C} + X_{BC,C} + X_{CE,C} = X_{C,t}$

$X_{AE,E} + X_{BE,E} + X_{CE,E} = X_{E,t}$

$X_{A,t} + X_{B,t} + X_{C,t} + X_{E,t} = 7$

constraints: $0 \le X_{S,AB}, X_{S,AC}, X_{S,AE}, X_{S,BC}, X_{S,BE}, X_{S,CE} \le 2$

$0 \le X_{i,j} \le 2$ for all $i \in \{AB, AC, AE, BC, BE, CE\}$, $j \in \{A, B, C, E\}$

$0 \le X_{A,t}, X_{B,t} \le 5$

$0 \le X_{C,t} \le 4$

$0 \le X_{E,t} \le 3$

can X from $\mathbb{Z}$

Solving it using Gurobi gives:

$X_{S,AB} = X_{S,AC} = X_{S,AE} = X_{S,BC} = X_{S,BE} = X_{S,CE} = 2$

$X_{AB,B} = 2$, $X_{AC,A} = 2$, $X_{AE,A} = 1$, $X_{AE,E} = 1$, $X_{BC,C} = 2$, $X_{BE,E} = 2$, $X_{CE,C} = 2$

$X_{A,t} = 3$, $X_{B,t} = 2$, $X_{C,t} = 4$, $X_{E,t} = 3$

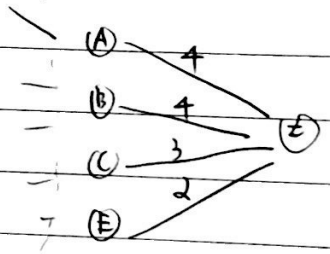
Therefore, in this case, $7+8 = 15 \geq \max(3+10, 2+10, 4+11, 2+3) = 15$

$\Rightarrow$ team D can win the tournament

b. similar to a. keep the objective and other constraints the same as a. only change the * part to:

$$\begin{cases} 0 \leq X_{A,t}, X_{B,t} \leq 4 \\ 0 \leq X_{C,t} \leq 3 \\ 0 \leq X_{E,t} \leq 2 \end{cases}$$

for the max-flow graph:



Then, solve the LP using Gurobi gives:

$X_{S,AB} = X_{S,AC} = X_{S,AE} = X_{S,BC} = X_{S,BE} = X_{S,CE} = 2$

$X_{AB,B} = 2$, $X_{AC,A} = 2$, $X_{AE,A} = 1$, $X_{AE,E} = 1$, $X_{BC,C} = 2$, $X_{BE,B} = 2$, $X_{CE,C} = X_{CE,E} = 1$

$X_{A,t} = 3$, $X_{B,t} = 4$, $X_{C,t} = 3$, $X_{E,t} = 2$

since team D wins $7+8 = 15 > \max(3+10, 4+10, 3+11, 2+12) = 14$

D is the only team wins the tournament

4. a Use dynamic programming. Let $f_i(j)$ to represent the number of ways there are if the board is of size $i \times j$ (i rows, j columns)

Then, our goal is to find $f_4(7)$. Since the robot only moves one cells to the right / one all down, then for cell $X_{m,n}$, it only be from left / up. last step could

$$f_i(j) = \begin{cases} 1 & \text{if either } i \text{ or } j \text{ is } 1 \\ f_{i-1}(j) + f_i(j-1) & \text{for both } i, j \geq 2 \end{cases}$$

$i \in \{1, 2, 3, 4\}$, $j \in \{1, 2, 3, 4, 5, 6, 7\}$

i\j	1	2	3	4	5	6	7
1	1	1	1	1	1	1	1
2	1	2	3	4	5	6	7
3	1	3	6	10	15	21	28
4	1	4	10	20	35	56	84

⇑

Therefore, $f_4(7) = f_3(7) + f_4(6) = 28 + 56 = 84 = \frac{9!}{3!6!} = \binom{9}{3}$

b. Let $k_{m,n}$ denotes the existence of coin in the cell in m-row n-th column. Then $k_{m,n} = \begin{cases} 1 & \text{if there is a coin in the cell} \\ 0 & \text{if there isn't a coin} \end{cases}$

Let $f_i^*(j)$ denotes the maximum number of coins collected from S to the destination on $i \times j$ board $\Rightarrow$ $f_i^*(j) = \begin{cases} \max(f_{i-1}^*(j), f_i^*(j-1)) + k_{i,j} & \text{otw} \\ f_{i-1}^*(j) + k_{i,j} & \text{if } j = 1, i \geq 2 \\ f_i^*(j-1) + k_{i,j} & \text{if } i = 1, j \geq 2 \\ 0 & \text{if } i = j = 1 \end{cases}$

i\j	1	2	3	4	5	6	7
1	0	0	0	0	0	1	0
2	0	0	1	2	2	2	3
3	0	1	1	2	3	3	3
4	0	1	2	2	3	4	4

← the desired path that collected the total of 4 coins.

Date

Problem#1

December 2, 2019

[1]:
```
# !python3 singlecomm.py t2.dat prob1-a.lp
```

[2]:
```
# Part a
from gurobipy import *

# create a model
m = Model()

# create variables
x16 = m.addVar(vtype=GRB.CONTINUOUS, name="x16", lb=0)
x17 = m.addVar(vtype=GRB.CONTINUOUS, name="x17", lb=0)
x18 = m.addVar(vtype=GRB.CONTINUOUS, name="x18", lb=0)
x19 = m.addVar(vtype=GRB.CONTINUOUS, name="x19", lb=0)
x26 = m.addVar(vtype=GRB.CONTINUOUS, name="x26", lb=0)
x27 = m.addVar(vtype=GRB.CONTINUOUS, name="x27", lb=0)
x28 = m.addVar(vtype=GRB.CONTINUOUS, name="x28", lb=0)
x29 = m.addVar(vtype=GRB.CONTINUOUS, name="x29", lb=0)
x36 = m.addVar(vtype=GRB.CONTINUOUS, name="x36", lb=0)
x37 = m.addVar(vtype=GRB.CONTINUOUS, name="x37", lb=0)
x38 = m.addVar(vtype=GRB.CONTINUOUS, name="x38", lb=0)
x39 = m.addVar(vtype=GRB.CONTINUOUS, name="x39", lb=0)
x46 = m.addVar(vtype=GRB.CONTINUOUS, name="x46", lb=0)
x47 = m.addVar(vtype=GRB.CONTINUOUS, name="x47", lb=0)
x48 = m.addVar(vtype=GRB.CONTINUOUS, name="x48", lb=0)
x49 = m.addVar(vtype=GRB.CONTINUOUS, name="x49", lb=0)
x56 = m.addVar(vtype=GRB.CONTINUOUS, name="x56", lb=0)
x57 = m.addVar(vtype=GRB.CONTINUOUS, name="x57", lb=0)
x58 = m.addVar(vtype=GRB.CONTINUOUS, name="x58", lb=0)
x59 = m.addVar(vtype=GRB.CONTINUOUS, name="x59", lb=0)


# integrate new variables
m.update()

# set objective
m.setObjective(
    541.0*x16 + 386.0*x17 + 25.0*x18 + 1512.0*x19 + 234.0*x26 + 899.0*x27 +
```

```
    103.0*x28 + 1256.0*x29 + 543.0*x36 + 257.0*x37 + 1653.0*x38 + 1085.0*x39 +
    1785.0*x46 + 227.0*x47 + 1670.0*x48 + 823.0*x49 + 490.0*x56 + 1233.0*x57 +
    1242.0*x58 + 1841.0*x59, GRB.MINIMIZE
)

# add constraints
m.addConstr(x16 + x17 + x18 + x19 == 208.0)
m.addConstr(x26 + x27 + x28 + x29 == 193.0)
m.addConstr(x36 + x37 + x38 + x39 == 195.0)
m.addConstr(x46 + x47 + x48 + x49 == 209.0)
m.addConstr(x56 + x57 + x58 + x59 == 4031.0)
m.addConstr(-1*(x16 + x26 + x36 + x46 + x56) == -1530.0)
m.addConstr(-1*(x17 + x27 + x37 + x47 + x57) == -1583.0)
m.addConstr(-1*(x18 + x28 + x38 + x48 + x58) == -1562.0)
m.addConstr(-1*(x19 + x29 + x39 + x49 + x59) == -161.0)
m.addConstr(x16 <= 7407.0)
m.addConstr(x17 <= 3546.0)
m.addConstr(x18 <= 5072.0)
m.addConstr(x19 <= 1932.0)
m.addConstr(x26 <= 81.0)
m.addConstr(x27 <= 90.0)
m.addConstr(x28 <= 29.0)
m.addConstr(x29 <= 902.0)
m.addConstr(x36 <= 13.0)
m.addConstr(x37 <= 8413.0)
m.addConstr(x38 <= 8719.0)
m.addConstr(x39 <= 7439.0)
m.addConstr(x46 <= 5047.0)
m.addConstr(x47 <= 83.0)
m.addConstr(x48 <= 58.0)
m.addConstr(x49 <= 76.0)
m.addConstr(x56 <= 83.0)
m.addConstr(x57 <= 7904.0)
m.addConstr(x58 <= 73.0)
m.addConstr(x59 <= 65.0)

# optimize
m.optimize()
print("Model status: ", m.status)

# print out decision variables
for v in m.getVars():
    print(v.varName, v.x, "\n")

print("-"*15)
print("Obj Value: ", m.objVal)
```

```
Academic license - for non-commercial use only
Optimize a model with 29 rows, 20 columns and 60 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [2e+01, 2e+03]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+01, 9e+03]
Presolve removed 20 rows and 0 columns
Presolve time: 0.02s

Solved in 0 iterations and 0.03 seconds
Infeasible model
Model status:  3

        ---------------------------------------------------------------------------

        AttributeError                            Traceback (most recent call last)

        <ipython-input-2-1321418960ab> in <module>
         76 # print out decision variables
         77 for v in m.getVars():
    ---> 78     print(v.varName, v.x, "\n")
         79
         80 print("-"*15)


        var.pxi in gurobipy.Var.__getattr__()


        var.pxi in gurobipy.Var.getAttr()


        AttributeError: b"Unable to retrieve attribute 'x'"
```

[ ]:
```python
# Part b
from gurobipy import *

# create a model
m = Model()

# create variables
x16 = m.addVar(vtype=GRB.CONTINUOUS, name="x16", lb=0)
x17 = m.addVar(vtype=GRB.CONTINUOUS, name="x17", lb=0)
x18 = m.addVar(vtype=GRB.CONTINUOUS, name="x18", lb=0)
x19 = m.addVar(vtype=GRB.CONTINUOUS, name="x19", lb=0)
x26 = m.addVar(vtype=GRB.CONTINUOUS, name="x26", lb=0)
```

```
x27 = m.addVar(vtype=GRB.CONTINUOUS, name="x27", lb=0)
x28 = m.addVar(vtype=GRB.CONTINUOUS, name="x28", lb=0)
x29 = m.addVar(vtype=GRB.CONTINUOUS, name="x29", lb=0)
x36 = m.addVar(vtype=GRB.CONTINUOUS, name="x36", lb=0)
x37 = m.addVar(vtype=GRB.CONTINUOUS, name="x37", lb=0)
x38 = m.addVar(vtype=GRB.CONTINUOUS, name="x38", lb=0)
x39 = m.addVar(vtype=GRB.CONTINUOUS, name="x39", lb=0)
x46 = m.addVar(vtype=GRB.CONTINUOUS, name="x46", lb=0)
x47 = m.addVar(vtype=GRB.CONTINUOUS, name="x47", lb=0)
x48 = m.addVar(vtype=GRB.CONTINUOUS, name="x48", lb=0)
x49 = m.addVar(vtype=GRB.CONTINUOUS, name="x49", lb=0)
x56 = m.addVar(vtype=GRB.CONTINUOUS, name="x56", lb=0)
x57 = m.addVar(vtype=GRB.CONTINUOUS, name="x57", lb=0)
x58 = m.addVar(vtype=GRB.CONTINUOUS, name="x58", lb=0)
x59 = m.addVar(vtype=GRB.CONTINUOUS, name="x59", lb=0)
delta_x16 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x16", lb=0)
delta_x17 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x17", lb=0)
delta_x18 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x18", lb=0)
delta_x19 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x19", lb=0)
delta_x26 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x26", lb=0)
delta_x27 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x27", lb=0)
delta_x28 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x28", lb=0)
delta_x29 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x29", lb=0)
delta_x36 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x36", lb=0)
delta_x37 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x37", lb=0)
delta_x38 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x38", lb=0)
delta_x39 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x39", lb=0)
delta_x46 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x46", lb=0)
delta_x47 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x47", lb=0)
delta_x48 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x48", lb=0)
delta_x49 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x49", lb=0)
delta_x56 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x56", lb=0)
delta_x57 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x57", lb=0)
delta_x58 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x58", lb=0)
delta_x59 = m.addVar(vtype=GRB.CONTINUOUS, name="delta_x59", lb=0)


# integrate new variables
m.update()

# set objective
m.setObjective(
    541.0*delta_x16 + 386.0*delta_x17 + 25.0*delta_x18 + 1512.0*delta_x19 + 234.
 ↪0*delta_x26 +
    899.0*delta_x27 + 103.0*delta_x28 + 1256.0*delta_x29 + 543.0*delta_x36 + 257.
 ↪0*delta_x37 +
```

```
    1653.0*delta_x38 + 1085.0*delta_x39 + 1785.0*delta_x46 + 227.0*delta_x47 +␣
 ↪1670.0*delta_x48 +
    823.0*delta_x49 + 490.0*delta_x56 + 1233.0*delta_x57 + 1242.0*delta_x58 +␣
 ↪1841.0*delta_x59,
    GRB.MINIMIZE
)

# add constraints
m.addConstr(x16 + x17 + x18 + x19 == 208)
m.addConstr(x26 + x27 + x28 + x29 == 193)
m.addConstr(x36 + x37 + x38 + x39 == 195)
m.addConstr(x46 + x47 + x48 + x49 == 209)
m.addConstr(x56 + x57 + x58 + x59 == 4031)
m.addConstr(- x16 - x26 - x36 - x46 - x56 == -1530.0)
m.addConstr(- x17 - x27 - x37 - x47 - x57 == -1583.0)
m.addConstr(- x18 - x28 - x38 - x48 - x58 == -1562.0)
m.addConstr(- x19 - x29 - x39 - x49 - x59 == -161.0)
m.addConstr(x16 <= 7407.0 + delta_x16)
m.addConstr(x17 <= 3546.0 + delta_x17)
m.addConstr(x18 <= 5072.0 + delta_x18)
m.addConstr(x19 <= 1932.0 + delta_x19)
m.addConstr(x26 <= 81.0 + delta_x26)
m.addConstr(x27 <= 90.0 + delta_x27)
m.addConstr(x28 <= 29.0 + delta_x28)
m.addConstr(x29 <= 902.0 + delta_x29)
m.addConstr(x36 <= 13.0 + delta_x36)
m.addConstr(x37 <= 8413.0 + delta_x37)
m.addConstr(x38 <= 8719.0 + delta_x38)
m.addConstr(x39 <= 7439.0 + delta_x39)
m.addConstr(x46 <= 5047.0 + delta_x46)
m.addConstr(x47 <= 83.0 + delta_x47)
m.addConstr(x48 <= 58.0 + delta_x48)
m.addConstr(x49 <= 76.0 + delta_x49)
m.addConstr(x56 <= 83.0 + delta_x56)
m.addConstr(x57 <= 7904.0 + delta_x57)
m.addConstr(x58 <= 73.0 + delta_x58)
m.addConstr(x59 <= 65.0 + delta_x59)


# optimize
m.optimize()
print("Model status: ", m.status)

# print out decision variables
for v in m.getVars():
    print(v.varName, v.x, "\n")
```

```
print("-"*15)
print("Obj Value: ", m.objVal)
```

[ ]:
```
# Part c

from gurobipy import *

# create a model
m = Model()

# create variables
x16 = m.addVar(vtype=GRB.CONTINUOUS, name="x16", lb=0)
x17 = m.addVar(vtype=GRB.CONTINUOUS, name="x17", lb=0)
x18 = m.addVar(vtype=GRB.CONTINUOUS, name="x18", lb=0)
x19 = m.addVar(vtype=GRB.CONTINUOUS, name="x19", lb=0)
x26 = m.addVar(vtype=GRB.CONTINUOUS, name="x26", lb=0)
x27 = m.addVar(vtype=GRB.CONTINUOUS, name="x27", lb=0)
x28 = m.addVar(vtype=GRB.CONTINUOUS, name="x28", lb=0)
x29 = m.addVar(vtype=GRB.CONTINUOUS, name="x29", lb=0)
x36 = m.addVar(vtype=GRB.CONTINUOUS, name="x36", lb=0)
x37 = m.addVar(vtype=GRB.CONTINUOUS, name="x37", lb=0)
x38 = m.addVar(vtype=GRB.CONTINUOUS, name="x38", lb=0)
x39 = m.addVar(vtype=GRB.CONTINUOUS, name="x39", lb=0)
x46 = m.addVar(vtype=GRB.CONTINUOUS, name="x46", lb=0)
x47 = m.addVar(vtype=GRB.CONTINUOUS, name="x47", lb=0)
x48 = m.addVar(vtype=GRB.CONTINUOUS, name="x48", lb=0)
x49 = m.addVar(vtype=GRB.CONTINUOUS, name="x49", lb=0)
x56 = m.addVar(vtype=GRB.CONTINUOUS, name="x56", lb=0)
x57 = m.addVar(vtype=GRB.CONTINUOUS, name="x57", lb=0)
x58 = m.addVar(vtype=GRB.CONTINUOUS, name="x58", lb=0)
x59 = m.addVar(vtype=GRB.CONTINUOUS, name="x59", lb=0)


# integrate new variables
m.update()

# set objective
m.setObjective(
    541.0*x16 + 386.0*x17 + 25.0*x18 + 1512.0*x19 + 234.0*x26 + 899.0*x27 +
    103.0*x28 + 1256.0*x29 + 543.0*x36 + 257.0*x37 + 1653.0*x38 + 1085.0*x39 +
    1785.0*x46 + 227.0*x47 + 1670.0*x48 + 823.0*x49 + 490.0*x56 + 1233.0*x57 +
    1242.0*x58 + 1841.0*x59, GRB.MINIMIZE
)

# add constraints
m.addConstr(x16 + x17 + x18 + x19 == 208)
m.addConstr(x26 + x27 + x28 + x29 == 193)
```

```
m.addConstr(x36 + x37 + x38 + x39 == 195)
m.addConstr(x46 + x47 + x48 + x49 == 209)
m.addConstr(x56 + x57 + x58 + x59 == 4031)
m.addConstr(- x16 - x26 - x36 - x46 - x56 == -1530.0)
m.addConstr(- x17 - x27 - x37 - x47 - x57 == -1583.0)
m.addConstr(- x18 - x28 - x38 - x48 - x58 == -1562.0)
m.addConstr(- x19 - x29 - x39 - x49 - x59 == -161.0)
m.addConstr(x16 <= 7407.0)
m.addConstr(x17 <= 3546.0)
m.addConstr(x18 <= 5072.0)
m.addConstr(x19 <= 1932.0)
m.addConstr(x26 <= 81.0)
m.addConstr(x27 <= 90.0)
m.addConstr(x28 <= 29.0+144)
m.addConstr(x29 <= 902.0)
m.addConstr(x36 <= 13.0)
m.addConstr(x37 <= 8413.0)
m.addConstr(x38 <= 8719.0)
m.addConstr(x39 <= 7439.0)
m.addConstr(x46 <= 5047.0)
m.addConstr(x47 <= 83.0)
m.addConstr(x48 <= 58.0 )
m.addConstr(x49 <= 76.0)
m.addConstr(x56 <= 83.0 + 1372)
m.addConstr(x57 <= 7904.0)
m.addConstr(x58 <= 73.0 + 855)
m.addConstr(x59 <= 65.0 )


# optimize
m.optimize()
print("Model status: ", m.status)

# print out decision variables
for v in m.getVars():
    print(v.varName, v.x, "\n")

print("-"*15)
print("Obj Value: ", m.objVal)
```

[ ]:

Problem#3

December 2, 2019

[1]:
```
# Part a

from gurobipy import *

# create a model
m = Model()

# create variables
xs_AB = m.addVar(vtype=GRB.INTEGER, name="xs_AB", lb=0, ub=2)
xs_AC = m.addVar(vtype=GRB.INTEGER, name="xs_AC", lb=0, ub=2)
xs_AE = m.addVar(vtype=GRB.INTEGER, name="xs_AE", lb=0, ub=2)
xs_BC = m.addVar(vtype=GRB.INTEGER, name="xs_BC", lb=0, ub=2)
xs_BE = m.addVar(vtype=GRB.INTEGER, name="xs_BE", lb=0, ub=2)
xs_CE = m.addVar(vtype=GRB.INTEGER, name="xs_CE", lb=0, ub=2)
xAB_A = m.addVar(vtype=GRB.INTEGER, name="xAB_A", lb=0, ub=2)
xAB_B = m.addVar(vtype=GRB.INTEGER, name="xAB_B", lb=0, ub=2)
xAC_A = m.addVar(vtype=GRB.INTEGER, name="xAC_A", lb=0, ub=2)
xAC_C = m.addVar(vtype=GRB.INTEGER, name="xAC_C", lb=0, ub=2)
xAE_A = m.addVar(vtype=GRB.INTEGER, name="xAE_A", lb=0, ub=2)
xAE_E = m.addVar(vtype=GRB.INTEGER, name="xAE_E", lb=0, ub=2)
xBC_B = m.addVar(vtype=GRB.INTEGER, name="xBC_B", lb=0, ub=2)
xBC_C = m.addVar(vtype=GRB.INTEGER, name="xBC_C", lb=0, ub=2)
xBE_B = m.addVar(vtype=GRB.INTEGER, name="xBE_B", lb=0, ub=2)
xBE_E = m.addVar(vtype=GRB.INTEGER, name="xBE_E", lb=0, ub=2)
xCE_C = m.addVar(vtype=GRB.INTEGER, name="xCE_C", lb=0, ub=2)
xCE_E = m.addVar(vtype=GRB.INTEGER, name="xCE_E", lb=0, ub=2)
xA_t = m.addVar(vtype=GRB.INTEGER, name="xA_t", lb=0, ub=5)
xB_t = m.addVar(vtype=GRB.INTEGER, name="xB_t", lb=0, ub=5)
xC_t = m.addVar(vtype=GRB.INTEGER, name="xC_t", lb=0, ub=4)
xE_t = m.addVar(vtype=GRB.INTEGER, name="xE_t", lb=0, ub=3)
z = m.addVar(vtype=GRB.INTEGER, name="z", lb=0, ub=12)


# integrate new variables
m.update()

# set objective
```

```
m.setObjective(
    z, GRB.MAXIMIZE
)

# add constraints
m.addConstr(xs_AB + xs_AC + xs_AE + xs_BC + xs_BE + xs_CE == z)
m.addConstr(xs_AB == xAB_A + xAB_B)
m.addConstr(xs_AC == xAC_A + xAC_C)
m.addConstr(xs_AE == xAE_A + xAE_E)
m.addConstr(xs_BC == xBC_B + xBC_C)
m.addConstr(xs_BE == xBE_B + xBE_E)
m.addConstr(xs_CE == xCE_C + xCE_E)
m.addConstr(xAB_A + xAC_A + xAE_A == xA_t)
m.addConstr(xAB_B + xBC_B + xBE_B == xB_t)
m.addConstr(xAC_C + xBC_C + xCE_C == xC_t)
m.addConstr(xAE_A + xBE_E + xCE_E == xE_t)
m.addConstr(xA_t + xB_t + xC_t + xE_t == z)

# optimize
m.optimize()
print("Model status: ", m.status)

# print out decision variables
for v in m.getVars():
    print(v.varName, v.x, "\n")

print("-"*15)
print("Obj Value: ", m.objVal)
```

```
Academic license - for non-commercial use only
Optimize a model with 12 rows, 23 columns and 46 nonzeros
Variable types: 0 continuous, 23 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [2e+00, 1e+01]
  RHS range        [0e+00, 0e+00]
Found heuristic solution: objective -0.0000000
Presolve removed 1 rows and 2 columns
Presolve time: 0.00s
Presolved: 11 rows, 21 columns, 43 nonzeros
Variable types: 0 continuous, 21 integer (0 binary)

Root relaxation: objective 1.200000e+01, 7 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time
```

```
*    0     0               0      12.0000000   12.00000  0.00%     -    0s

Explored 0 nodes (7 simplex iterations) in 0.03 seconds
Thread count was 4 (of 4 available processors)

Solution count 2: 12 -0

Optimal solution found (tolerance 1.00e-04)
Best objective 1.200000000000e+01, best bound 1.200000000000e+01, gap 0.0000%
Model status:  2
xs_AB 2.0

xs_AC 2.0

xs_AE 2.0

xs_BC 2.0

xs_BE 2.0

xs_CE 2.0

xAB_A -0.0

xAB_B 2.0

xAC_A 2.0

xAC_C 0.0

xAE_A 1.0

xAE_E 1.0

xBC_B -0.0

xBC_C 2.0

xBE_B -0.0

xBE_E 2.0

xCE_C 2.0

xCE_E -0.0

xA_t 3.0
```

```
xB_t 2.0

xC_t 4.0

xE_t 3.0

z 12.0

---------------
Obj Value:  12.0
```

[2]:
```python
# Part b

from gurobipy import *

# create a model
m = Model()

# create variables
xs_AB = m.addVar(vtype=GRB.INTEGER, name="xs_AB", lb=0, ub=2)
xs_AC = m.addVar(vtype=GRB.INTEGER, name="xs_AC", lb=0, ub=2)
xs_AE = m.addVar(vtype=GRB.INTEGER, name="xs_AE", lb=0, ub=2)
xs_BC = m.addVar(vtype=GRB.INTEGER, name="xs_BC", lb=0, ub=2)
xs_BE = m.addVar(vtype=GRB.INTEGER, name="xs_BE", lb=0, ub=2)
xs_CE = m.addVar(vtype=GRB.INTEGER, name="xs_CE", lb=0, ub=2)
xAB_A = m.addVar(vtype=GRB.INTEGER, name="xAB_A", lb=0, ub=2)
xAB_B = m.addVar(vtype=GRB.INTEGER, name="xAB_B", lb=0, ub=2)
xAC_A = m.addVar(vtype=GRB.INTEGER, name="xAC_A", lb=0, ub=2)
xAC_C = m.addVar(vtype=GRB.INTEGER, name="xAC_C", lb=0, ub=2)
xAE_A = m.addVar(vtype=GRB.INTEGER, name="xAE_A", lb=0, ub=2)
xAE_E = m.addVar(vtype=GRB.INTEGER, name="xAE_E", lb=0, ub=2)
xBC_B = m.addVar(vtype=GRB.INTEGER, name="xBC_B", lb=0, ub=2)
xBC_C = m.addVar(vtype=GRB.INTEGER, name="xBC_C", lb=0, ub=2)
xBE_B = m.addVar(vtype=GRB.INTEGER, name="xBE_B", lb=0, ub=2)
xBE_E = m.addVar(vtype=GRB.INTEGER, name="xBE_E", lb=0, ub=2)
xCE_C = m.addVar(vtype=GRB.INTEGER, name="xCE_C", lb=0, ub=2)
xCE_E = m.addVar(vtype=GRB.INTEGER, name="xCE_E", lb=0, ub=2)
xA_t = m.addVar(vtype=GRB.INTEGER, name="xA_t", lb=0, ub=4)
xB_t = m.addVar(vtype=GRB.INTEGER, name="xB_t", lb=0, ub=4)
xC_t = m.addVar(vtype=GRB.INTEGER, name="xC_t", lb=0, ub=3)
xE_t = m.addVar(vtype=GRB.INTEGER, name="xE_t", lb=0, ub=2)
z = m.addVar(vtype=GRB.INTEGER, name="z", lb=0, ub=12)


# integrate new variables
m.update()
```

```
# set objective
m.setObjective(
    z, GRB.MAXIMIZE
)

# add constraints
m.addConstr(xs_AB + xs_AC + xs_AE + xs_BC + xs_BE + xs_CE == z)
m.addConstr(xs_AB == xAB_A + xAB_B)
m.addConstr(xs_AC == xAC_A + xAC_C)
m.addConstr(xs_AE == xAE_A + xAE_E)
m.addConstr(xs_BC == xBC_B + xBC_C)
m.addConstr(xs_BE == xBE_B + xBE_E)
m.addConstr(xs_CE == xCE_C + xCE_E)
m.addConstr(xAB_A + xAC_A + xAE_A == xA_t)
m.addConstr(xAB_B + xBC_B + xBE_B == xB_t)
m.addConstr(xAC_C + xBC_C + xCE_C == xC_t)
m.addConstr(xAE_A + xBE_E + xCE_E == xE_t)
m.addConstr(xA_t + xB_t + xC_t + xE_t == z)

# optimize
m.optimize()
print("Model status: ", m.status)

# print out decision variables
for v in m.getVars():
    print(v.varName, v.x, "\n")

print("-"*15)
print("Obj Value: ", m.objVal)
```

```
Optimize a model with 12 rows, 23 columns and 46 nonzeros
Variable types: 0 continuous, 23 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [2e+00, 1e+01]
  RHS range        [0e+00, 0e+00]
Found heuristic solution: objective -0.0000000
Presolve removed 1 rows and 2 columns
Presolve time: 0.00s
Presolved: 11 rows, 21 columns, 43 nonzeros
Variable types: 0 continuous, 21 integer (0 binary)

Root relaxation: objective 1.200000e+01, 7 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
```

```
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0               0      12.0000000   12.00000  0.00%     -    0s

Explored 0 nodes (7 simplex iterations) in 0.04 seconds
Thread count was 4 (of 4 available processors)

Solution count 2: 12 -0

Optimal solution found (tolerance 1.00e-04)
Best objective 1.200000000000e+01, best bound 1.200000000000e+01, gap 0.0000%
Model status:  2
xs_AB 2.0

xs_AC 2.0

xs_AE 2.0

xs_BC 2.0

xs_BE 2.0

xs_CE 2.0

xAB_A 0.0

xAB_B 2.0

xAC_A 2.0

xAC_C 0.0

xAE_A 1.0

xAE_E 1.0

xBC_B -0.0

xBC_C 2.0

xBE_B 2.0

xBE_E 0.0

xCE_C 1.0

xCE_E 1.0
```

```
xA_t 3.0

xB_t 4.0

xC_t 3.0

xE_t 2.0

z 12.0

---------------
Obj Value:  12.0
```

[ ]: