Conner Reinhardt

Adv. Optimization HW #1

Problem 1

1. (10 points) Transforms the following linear programs in the standard form.

   (a)

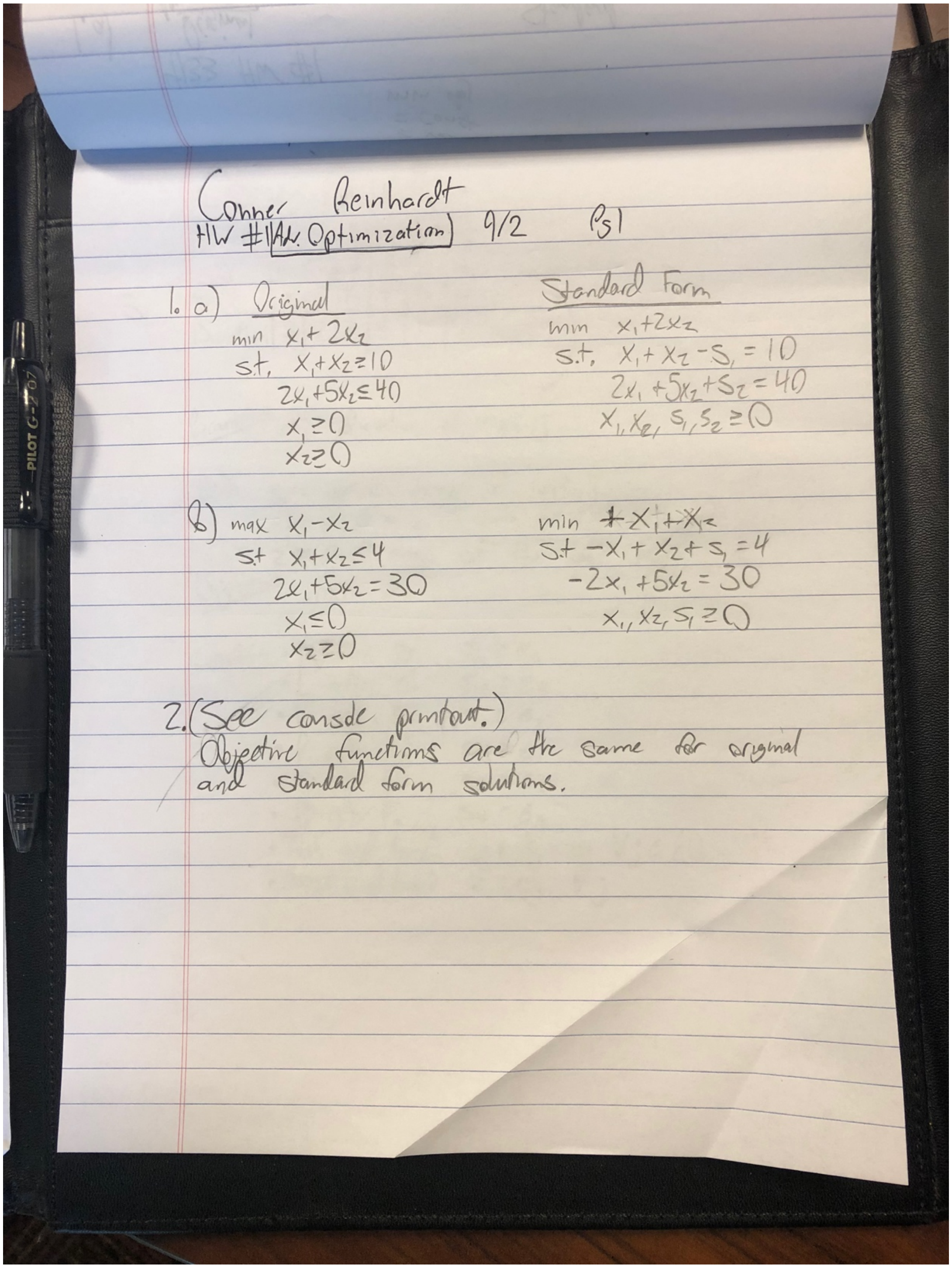
$$\begin{aligned} &\text{minimize } x_1 + 2x_2 \\ &\text{subject to} \\ &x_1 + x_2 \geq 10 \\ &2x_1 + 5x_2 \leq 40 \\ &x_1 \geq 0 \\ &x_2 \geq 0 \end{aligned}$$

   (b)

$$\begin{aligned} &\text{maximize } x_1 - x_2 \\ &\text{subject to} \\ &x_1 + x_2 \leq 4 \\ &2x_1 + 5x_2 = 30 \\ &x_1 \leq 0 \\ &x_2 \geq 0 \end{aligned}$$

Conner Reinhardt
Adv. Optimization HW #1

Problem 1



Conner Reinhardt
HW #1 $\boxed{\text{Adv. Optimization}}$ 9/2 PS1

1. a) Original

$$\begin{aligned} \min\ & x_1 + 2x_2 \\ \text{s.t.}\ & x_1 + x_2 \geq 10 \\ & 2x_1 + 5x_2 \leq 40 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$

Standard Form

$$\begin{aligned} \min\ & x_1 + 2x_2 \\ \text{s.t.}\ & x_1 + x_2 - s_1 = 10 \\ & 2x_1 + 5x_2 + s_2 = 40 \\ & x_1, x_2, s_1, s_2 \geq 0 \end{aligned}$$

b)

$$\begin{aligned} \max\ & x_1 - x_2 \\ \text{s.t.}\ & x_1 + x_2 \leq 4 \\ & 2x_1 + 5x_2 = 30 \\ & x_1 \leq 0 \\ & x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \min\ & \cancel{+}x_1 + x_2 \\ \text{s.t.}\ & -x_1 + x_2 + s_1 = 4 \\ & -2x_1 + 5x_2 = 30 \\ & x_1, x_2, s_1 \geq 0 \end{aligned}$$

2. (See console printout.)
Objective functions are the same for original and standard form solutions.

Conner Reinhardt
Adv. Optimization HW #1

Problem 1 Code

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Aug 26 17:16:49 2019

@author: connerreinhardt
"""
from gurobipy import *
############################################################################
print("\n-----------------------HW1_1a_Original-----------------------\n")
############################################################################
m = Model("1a_Original")

x1 = m.addVar(vtype=GRB.CONTINUOUS, name="x1")
x2 = m.addVar(vtype=GRB.CONTINUOUS, name="x2")

m.setObjective(x1+2*x2, GRB.MINIMIZE)

m.addConstr(x1 + x2 >= 10)
m.addConstr(2*x1 + 5*x2 <= 40)

m.addConstr(x1 >= 0)
m.addConstr(x2 >= 0)

m.optimize()

print("\n***************Results***************\n")

for v in m.getVars():
    print(v.varName + ":", v.X)
print("Objective value: " + str(m.objVal))
############################################################################
print("\n-----------------------HW1_1a_Standard-----------------------\n")
############################################################################
m = Model("1a_Standard")

x1 = m.addVar(vtype=GRB.CONTINUOUS, name="x1")
x2 = m.addVar(vtype=GRB.CONTINUOUS, name="x2")
s1 = m.addVar(vtype=GRB.CONTINUOUS, name="s1")
s2 = m.addVar(vtype=GRB.CONTINUOUS, name="s2")

m.setObjective(x1+2*x2, GRB.MINIMIZE)

m.addConstr(x1 + x2 -s1 == 10)
m.addConstr(2*x1 + 5*x2 +s2 == 40)
```

```
m.addConstr(x1 >= 0)
m.addConstr(x2 >= 0)
m.addConstr(s1 >= 0)
m.addConstr(s2 >= 0)

m.optimize()

print("\n**************Results**************\n")

for v in m.getVars():
    print(v.varName + ":", v.X)
print("Objective value: " + str(m.objVal))

#############################################################################
print("\n-----------------------HW1_1b_Original-----------------------\n")
#############################################################################
m = Model("1b_Original")

x1 = m.addVar(vtype=GRB.CONTINUOUS, lb=-GRB.INFINITY, name="x1") #can be allowed to go
below 0
x2 = m.addVar(vtype=GRB.CONTINUOUS, name="x2")

m.setObjective(x1-x2, GRB.MAXIMIZE)

m.addConstr(x1 + x2 <= 4)
m.addConstr(2*x1 + 5*x2 == 30)

m.addConstr(x1 <= 0)
m.addConstr(x2 >= 0)

m.optimize()

print("\n**************Results**************\n")

for v in m.getVars():
    print(v.varName + ":", v.X)
print("Objective value: " + str(m.objVal))

#############################################################################
print("\n-----------------------HW1_1b_Standard-----------------------\n")
#############################################################################
m = Model("1b_Standard")

x1 = m.addVar(vtype=GRB.CONTINUOUS, name="x1")
x2 = m.addVar(vtype=GRB.CONTINUOUS, name="x2")
s1 = m.addVar(vtype=GRB.CONTINUOUS, name="s1")
```

```python
m.setObjective(x1+x2, GRB.MINIMIZE)

m.addConstr(-x1 + x2 + s1 == 4)
m.addConstr(-2*x1 + 5*x2 == 30)

m.addConstr(x1 >= 0)
m.addConstr(x2 >= 0)
m.addConstr(s1 >= 0)

m.optimize()

print("\n***************Results***************\n")

for v in m.getVars():
    print(v.varName + ":", v.X)
print("Objective value: " + str(m.objVal))
```

Conner Reinhardt

Adv. Optimization HW #1

Problem 1 Results

----------------------HW1_1a_Original----------------------

```
Academic license - for non-commercial use only
Optimize a model with 4 rows, 2 columns and 6 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 5e+00]
  Objective range  [1e+00, 2e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+01, 4e+01]
Presolve removed 2 rows and 0 columns
Presolve time: 0.00s
Presolved: 2 rows, 2 columns, 4 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   2.500000e+00   0.000000e+00      0s
       1    1.0000000e+01   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.00 seconds
Optimal objective  1.000000000e+01

**************Results**************

x1: 10.0
x2: 0.0
Objective value: 10.0
```

----------------------HW1_1a_Standard----------------------

```
Optimize a model with 6 rows, 4 columns and 10 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 5e+00]
  Objective range  [1e+00, 2e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+01, 4e+01]
Presolve removed 4 rows and 2 columns
Presolve time: 0.00s
Presolved: 2 rows, 2 columns, 4 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   2.500000e+00   0.000000e+00      0s
       1    1.0000000e+01   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.00 seconds
Optimal objective  1.000000000e+01

**************Results**************

x1: 10.0
x2: 0.0
s1: 0.0
s2: 20.0
Objective value: 10.0
```

Both solutions are the same objective value, and the slack variables pick up the changes to require equivalence for standard form constraints

----------------------HW1_1b_Original----------------------

Conner Reinhardt
Adv. Optimization HW #1

```
Optimize a model with 4 rows, 2 columns and 6 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 5e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [4e+00, 3e+01]
Presolve removed 4 rows and 2 columns
Presolve time: 0.00s
Presolve: All rows and columns removed
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0   -1.0666667e+01   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.00 seconds
Optimal objective -1.066666667e+01

***************Results***************

x1: -3.3333333333333335
x2: 7.333333333333334
Objective value: -10.666666666666668

----------------------HW1_1b_Standard----------------------

Optimize a model with 5 rows, 3 columns and 8 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 5e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [4e+00, 3e+01]
Presolve removed 5 rows and 3 columns
Presolve time: 0.00s
Presolve: All rows and columns removed
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    1.0666667e+01   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.00 seconds
Optimal objective  1.066666667e+01

***************Results***************

x1: 3.3333333333333335
x2: 7.333333333333334
s1: 0.0
Objective value: 10.666666666666668
```

Both solutions are the same absolute value, except the sign is switched as it was rewritten to be a minimization instead of maximization problem in standard form

Conner Reinhardt

Adv. Optimization HW #1

Problem 3

(10 points) A builder needs to build a house and has divided the process into a number of tasks, namely

B. excavation and building the foundation.

F. raising the wooden frame.

E. electrical wiring.

P. Plumbing.

D. dry walls and flooring.

L. Landscaping.

The estimated time for each process (in weeks) is as follows.

Task	B	F	E	P	D	L
Duration	3	2	3	4	1	2

Some of the tasks can only be started when some other tasks are completed. For instance, you can online build the frame once the foundation has been completed, i.e., F can only start after B is completed. All these precedence constraints are summarized as follows.

- F can start only after B is completed.
- L can start only after B is completed.
- E can start only after F is completed.
- P can start only after F is completed.
- D can start only after E is completed.
- D can start only after P is completed.

The goal is to schedule the starting time of each task such that the entire project is completed as soon as possible.

As an example here is a feasible schedule with a completion time of ten weeks.

Task	B	F	E	P	D	L
Starting Time	0	3	6	5	9	6
Ending Time	3	5	9	9	10	8

Formulate a linear program that solves the problem. Explain your formulation. Note, there is no limit on the number of tasks that can be done in parallel. Solve the linear program using Gurobi and report the optimal solution.

Conner Reinhardt

Adv. Optimization HW #1

Problem 3 Setup

Conner Reinhardt

HW#1 [Adv. Optimization]

3.

j	0	1	2	3	4	5
Task	B	F	E	P	D	L
Duration	3	2	3	4	1	2

Parameters: d_j = duration of task j

Vars: s_j = start time of task j

e_j = end time of task j

~~Objective~~ z = End time of the last task

Objective: $\min z$

Constr:

- F after B: $s_1 \geq e_0$
- L after B: $s_5 \geq e_0$
- E after F: $s_2 \geq e_1$
- P after F: $s_3 \geq e_1$
- D after E: $s_4 \geq e_2$
- D after P: $s_4 \geq e_3$
- Task end time: $s_j + d_j = e_j \quad \forall j \in \{0, \ldots, 5\}$
- Project end time: $z \geq e_j \quad \forall j$

Conner Reinhardt
Adv. Optimization HW #1

Problem 3 Code

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Aug 26 17:16:49 2019

@author: connerreinhardt
"""
from gurobipy import *
############################################################################
print("\n-----------------------HW1_3-----------------------\n")
############################################################################
m = Model("Pr3")
#--Params
d = [3,2,3,4,1,2] #durations of tasks
taskname = ["B","F","E","P","D","L"]
#--Vars
s = m.addVars(len(d),vtype=GRB.INTEGER, name="s") #start time of task j
e = m.addVars(len(d),vtype=GRB.INTEGER, name="e") #end time of task j
z = m.addVar(vtype=GRB.INTEGER) #end time of project
#--Objective
m.setObjective(z, GRB.MINIMIZE)
#--Constr
for i in range(len(d)):
    m.addConstr(e[i]==s[i]+d[i]) #task end time
    m.addConstr(z>=e[i]) #project end time

m.addConstr(s[1]>=e[0]) #Prereq: F after B
m.addConstr(s[5]>=e[0]) #Prereq: L after B
m.addConstr(s[2]>=e[1]) #Prereq: E after F
m.addConstr(s[3]>=e[1]) #Prereq: P after F
m.addConstr(s[4]>=e[2]) #Prereq: D after E
m.addConstr(s[4]>=e[3]) #Prereq: D after P

m.optimize()

print("\n***************Results***************\n")

for v in m.getVars():
    print(v.varName + ":", v.X)
print("\nObjective value: " + str(m.objVal) + "\n")

for i in range(len(d)):
    print(taskname[i] + " Started: " + str(s[i].x))
    print(taskname[i] + " Ended: " + str(e[i].x))
    print("\n")
```

Conner Reinhardt

Adv. Optimization HW #1

Problem 3 Output

```
-----------------------HW1_3-----------------------
Optimize a model with 18 rows, 13 columns and 36 nonzeros
Variable types: 0 continuous, 13 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 4e+00]
Found heuristic solution: objective 10.0000000
Presolve removed 18 rows and 13 columns
Presolve time: 0.01s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.02 seconds
Thread count was 1 (of 4 available processors)

Solution count 1: 10

Optimal solution found (tolerance 1.00e-04)
Best objective 1.000000000000e+01, best bound 1.000000000000e+01, gap 0.0000%
***************Results***************
s[0]: 0.0
s[1]: 3.0
s[2]: 5.0
s[3]: 5.0
s[4]: 9.0
s[5]: 3.0
e[0]: 3.0
e[1]: 5.0
e[2]: 8.0
e[3]: 9.0
e[4]: 10.0
e[5]: 5.0
C12: 10.0

Objective value: 10.0

B Started: 0.0
B Ended: 3.0
F Started: 3.0
F Ended: 5.0
E Started: 5.0
E Ended: 8.0
P Started: 5.0
P Ended: 9.0
D Started: 9.0
D Ended: 10.0
L Started: 3.0
L Ended: 5.0

(base) lawn-128-61-55-191:~ connerreinhardt$
```

Conner Reinhardt

Adv. Optimization HW #1

Problem 4

(10 points) For $m \in \{10, 20, 50, 100, 500, 1000, 10000\}$ and $n \in \{10, 20, 50, 100, 1000, 10000\}$ generate matrices $A \in \mathbb{R}^{m\times n}$ whose entries are uniformly random between $[0, 1]$. Similarly generate $b \in \mathbb{R}^m$ randomly with entries randomly between $[0, 1000]$. Also generate a cost function $c \in \mathbb{R}^n$ with entries randomly between $[0, 1000]$.

(a) Formulate the linear program $\{\min c^T x : Ax \geq b, x \geq 0\}$ for the above random data. Solve 10 instances of the program for each pair of values of m and n with a time out of 2 minutes. Note the time taken and objective value for each run and average over the 10 runs for each pair of (m, n).

(b) Update the formulation to insist the variables are integers. Repeat the experiment. Note the time taken and objective value for each run.

(c) Plot the time and objective value as the y-axis and size $(m + n)$ as the x-axis.

Conner Reinhardt
Adv. Optimization HW #1

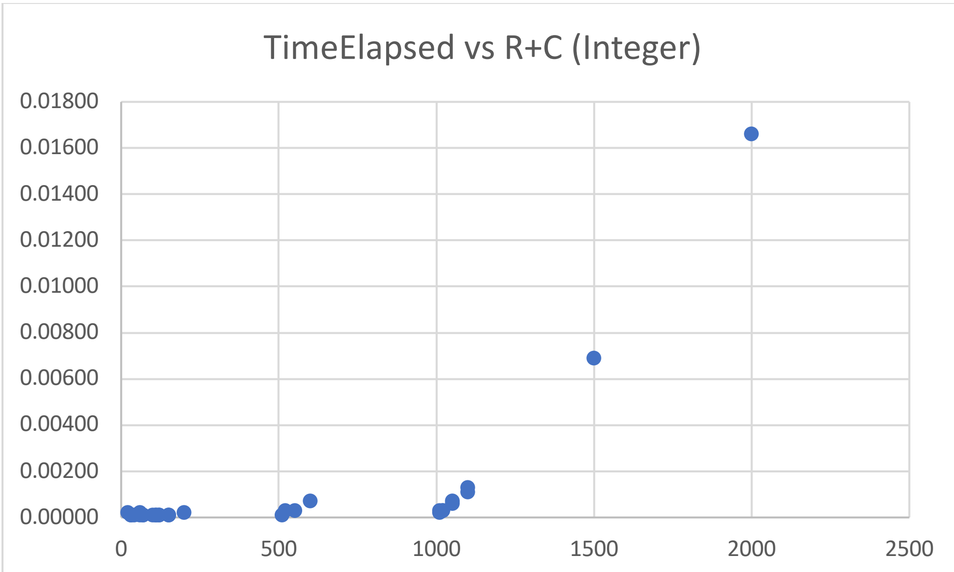
Problem 4 Code

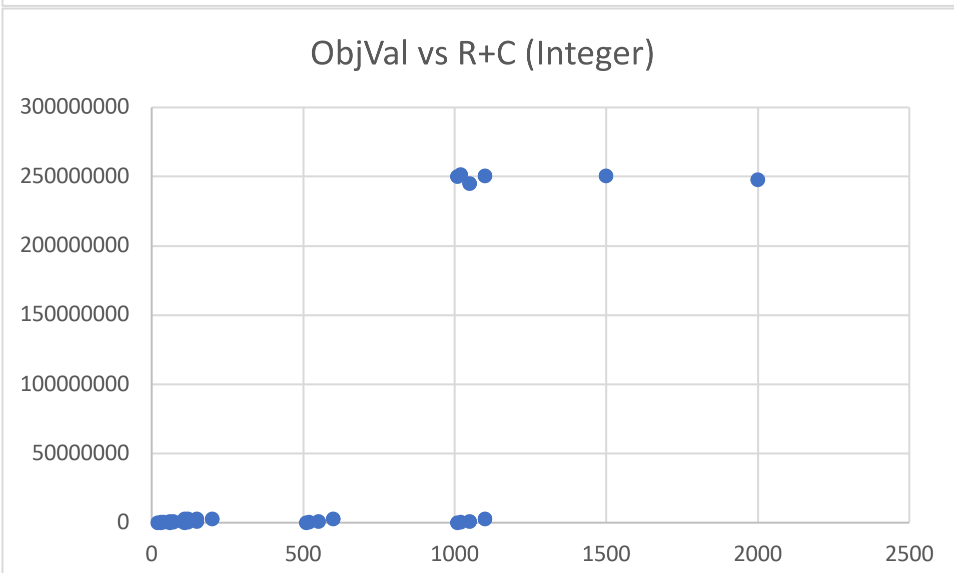
```
from gurobipy import *
import numpy as np

#X:  0   1   2    3    4      5    6
M = [10, 20, 50, 100, 500, 1000] #removed entries for simplicity
#Y:  0   1   2    3    4      5
N = [10, 20, 50, 100, 1000] #removed entries for simplicity

total_time_elapsed = 0.0
times_elapsed = []

objVals = []

matrix_array = [np.random.rand(M[x],N[y]) for x in range(0,len(M)) for y in
range(0,len(N))]

times_elapsed_avg_dict = {}
times_elapsed_dict = {}
objVals_avg_dict = {}
objVals_dict = {}

for r in range(0,len(M)*len(N)):

    for c in range(0,5): #changed from 10-> 5 to run fast

        matrix = matrix_array[r]
        X = list(range(len(matrix[0]))) #full MxN matrix
        B = np.random.rand(len(matrix),1)*1000
        C = np.random.rand(len(matrix[0]),1)*1000 #cost matrix

        itnM = len(matrix)
        itnN = len(matrix[0])
        print("")
        print("M : " + str(itnM))
        print("N : " + str(itnN))
        print("")
        print(str(len(matrix)) + "x" + str(len(matrix[0])) + " iteration " + str(c+1))

        m = Model("Pr4")
        x = m.addVars(X, vtype=GRB.CONTINUOUS, name = "x")

        m.addConstrs(quicksum(matrix[r][c]*x[c] for c in range(len(matrix[0]))) >=
B[r] for r in range(len(matrix)))
```

```
        m.setObjective(quicksum(X[c] * C[c] for c in range(len(matrix[0]))),
GRB.MINIMIZE)

        m.optimize()

        states =
{1:'LOADED',2:'OPTIMAL',3:'INFEASIBLE',4:'INF_OR_UNBD',5:'UNBOUNDED',6:'CUTOFF',7:'ITE
RATION_LIMIT',8:'NODE_LIMIT',9:'TIME_LIMIT',10:'SOLUTION_LIMIT',11:'INTERRUPTED',12:'N
UMERIC',13:'SUBOPTIMAL',14:'INPROGRESS',15:'USER_OBJ_LIMIT'}
        state = m.status

        total_time_elapsed += m.RunTime
        times_elapsed.append(m.RunTime)
        objVals.append(m.objVal)

        if c == 9: #after 10 iterations
            times_elapsed_avg_dict["Avg Time Elapsed " + str(itnM) + "x" + str(itnN)]
= (str(round((total_time_elapsed/20),4)) + " sec")
            times_elapsed_dict["Time Elapsed " + str(itnM) + "x" + str(itnN)] =
times_elapsed
            objVals_avg_dict["Avg Obj Vals " + str(itnM) + "x" + str(itnN)] =
np.mean(objVals)
            objVals_dict["Obj Vals " + str(itnM) + "x" + str(itnN)] = objVals

            total_time_elapsed = 0.0
            times_elapsed = []

            objVals = []

print(times_elapsed_avg_dict)
print(objVals_avg_dict)
print("\n\n\n")
print(times_elapsed_dict)
print(objVals_dict)
```

Conner Reinhardt
Adv. Optimization HW #1

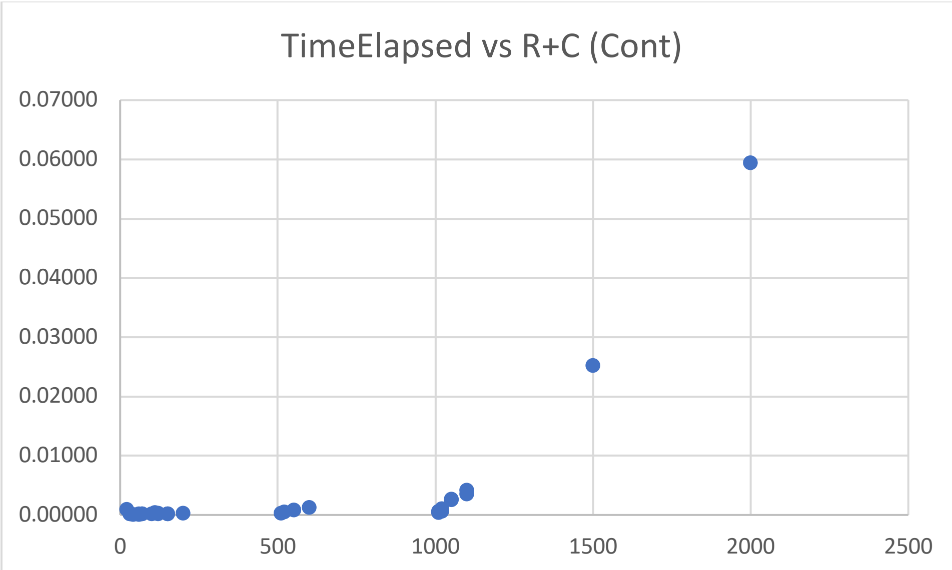
Problem 4 Results Output

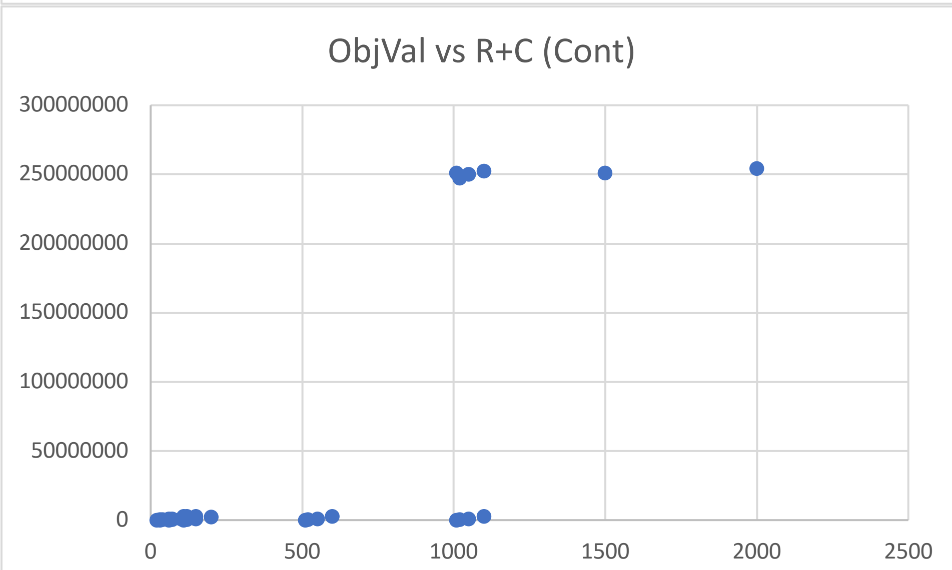




Conner Reinhardt

Adv. Optimization HW #1





Conner Reinhardt

Adv. Optimization HW #1

R+C	TimeElapsedInteger	ObjValInteger	TimeElapsedCont	ObjValCont
20	0.00020	19130.65301	0.00090	16947.5095
30	0.00010	100177.484	0.00020	97319.4993
60	0.00020	594420.8247	0.00020	609696.816
110	0.00010	2459155.626	0.00040	2447240.93
1010	0.00030	250011430.9	0.00040	250943664
30	0.00010	26070.8719	0.00020	20212.5073
40	0.00010	100984.7585	0.00010	81400.8732
70	0.00010	614841.6963	0.00020	603429.927
120	0.00010	2374668.267	0.00030	2452462.34
1020	0.00030	251480838.8	0.00060	247288484
60	0.00010	20783.21142	0.00010	20270.1772
70	0.00010	93020.0766	0.00020	104106.782
100	0.00010	633781.3294	0.00020	620541.01
150	0.00010	2485494.245	0.00020	2519222.97
1050	0.00060	245036272.7	0.00260	249768366
110	0.00010	25229.46292	0.00030	18749.6161
120	0.00010	86901.41445	0.00020	97888.8527
150	0.00010	632228.4174	0.00020	607450.129
200	0.00020	2556493.744	0.00030	2354128.92
1100	0.00110	250259564.3	0.00420	252452799
510	0.00010	25371.81362	0.00030	24422.43
520	0.00030	98470.0105	0.00050	98449.9862
550	0.00030	627874.7089	0.00080	584015.574
600	0.00070	2554302.244	0.00120	2421317.3
1500	0.00690	250611174.9	0.02520	250834764
1010	0.00020	26437.97634	0.00060	24688.2605
1020	0.00030	84974.4461	0.00100	98718.8959
1050	0.00070	624568.5946	0.00250	611778.516
1100	0.00130	2538445.521	0.00350	2476213.14
2000	0.01660	247762271.1	0.05940	253938003

Conner Reinhardt

Adv. Optimization HW #1

Problem 5

(10 points) A furniture manufacturing company makes two models of tables for libraries and other university facilities. Both models use the same table tops but model A has 4 short (18-inch) legs and model B has 4 longer ones (30 inches). It takes 0.10 labor hour to cut and shape a short leg from stock, 0.15 labor hour to do the same for a long leg and 0.50 labor hour to produce a tabletop. An additional 0.30 labor hour is needed to attach the set of leges for either model after all parts are available. Estimated profit is 30 for each model A sold and 45 for each model B sold. Plenty of top material is on hand but the company wants to decide how to use the available 5000 feet of leg stock and 800 labor hours to maximize profit assuming that everything made can be sold.

(a) Formulate a LP to choose the optimal plan. Assume that the number of tables and legs manufactured can take fractional values.

(b) Solve the linear program using Gurobi.

(c) Can you justify the assumption that the variables can take fractional values?

Conner Reinhardt

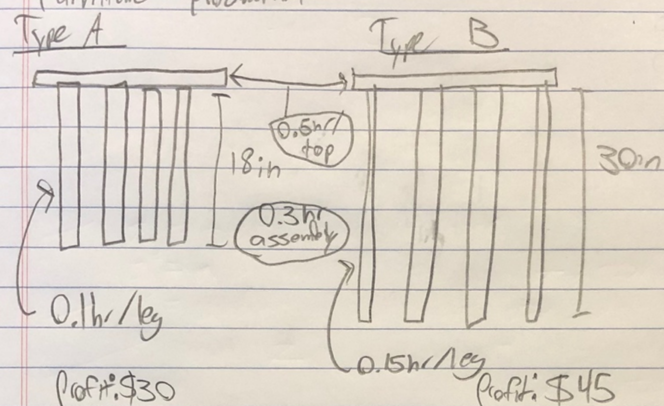
Adv. Optimization HW #1

Problem 5 Setup

Conner Reinhardt

HW#1 $\boxed{\text{Adv. Optimization}}$ 9/2

5. Furniture Production



→ 5000ft of leg stock (60,000 in) 800 labor hours

Param

table a hrs $h_A = 1.2$ table b hrs $h_B = 1.4$

profit $p_A = \$30$ $p_B = \$45$

leg stock $s_A = 72$ $s_B = 120$

Var

X_A = amount of table A to make

X_B = amount of table B to make

obj

$\max \quad 30X_a + 45X_b$

constr

$1.2x_a + 1.4x_b \le 800$ $x_a, x_b \ge 0$

$72x_A + 120x_B \le 60{,}000$

Conner Reinhardt
Adv. Optimization HW #1

Problem 5 Code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Aug 26 17:16:49 2019

@author: connerreinhardt
"""
from gurobipy import *
m = Model("Pr5")
#--Vars
xa = m.addVar(vtype=GRB.CONTINUOUS, name="xa") #amount of table A to make
xb = m.addVar(vtype=GRB.CONTINUOUS, name="xb") #amount of table B to make
#--Objective
m.setObjective(30*xa + 45*xb, GRB.MAXIMIZE)
#--Constr
m.addConstr(1.2*xa+1.4*xb<=800) #hours to create table
m.addConstr(72*xa+120*xb<=60000) #inches of legs available

m.addConstr(xa>=0)
m.addConstr(xb>=0)

m.optimize()

print("\n**************Results**************\n")

for v in m.getVars():
    print(v.varName + ":", v.X)
print("\nObjective value: " + str(m.objVal) + "\n")
```

Conner Reinhardt

Adv. Optimization HW #1

Problem 5 Output

```
lawn-128-61-55-191:~ connerreinhardt$ source /Users/connerreinhardt/miniconda3/bin/activate
(base) lawn-128-61-55-191:~ connerreinhardt$ conda activate base
(base) lawn-128-61-55-191:~ connerreinhardt$ /Users/connerreinhardt/miniconda3/bin/python
"/Users/connerreinhardt/Google Drive/Advanced Optimization/HW1/#5.py"

--------------------------------------------
Warning: your license will expire in 2 days
--------------------------------------------

Academic license - for non-commercial use only
Optimize a model with 4 rows, 2 columns and 6 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 1e+02]
  Objective range  [3e+01, 4e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [8e+02, 6e+04]
Presolve removed 2 rows and 0 columns
Presolve time: 0.00s
Presolved: 2 rows, 2 columns, 4 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    2.5714286e+04   2.048561e+02   0.000000e+00      0s
       2    2.3333333e+04   0.000000e+00   0.000000e+00      0s

Solved in 2 iterations and 0.00 seconds
Optimal objective  2.333333333e+04

***************Results***************

xa: 277.777777777778
xb: 333.3333333333332

Objective value: 23333.333333333336

(base) lawn-128-61-55-191:~ connerreinhardt$
```