Problem #1

(a) $-u_1(1-y) \le f_1(x)$

$-u_2 y \le f_2(x)$

$y \in \{0, 1\}$

(b) $f_1(x) \le u_1 x \quad (1-y)$

$-u_2 y \le f_2(x)$

$y \in \{0, 1\}$

(c) $(u_1+u_2)(1-y) \ge -f_1(x) \qquad -u_1 y \le f_1(x)$

$(u_1+u_2) y \ge -f_2(x) \qquad -u_2(1-y) \le f_2(x)$

$y \in \{0, 1\}$

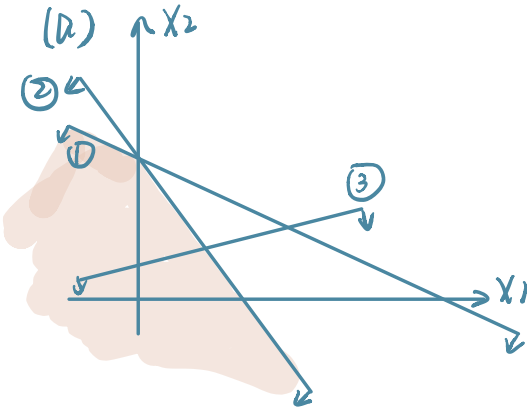
Problem # 2

$x_1 + 2x_2 \leq 12$ ①

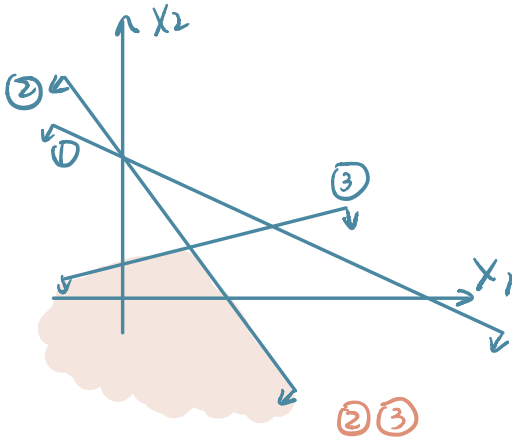
$3x_1 + 2x_2 \leq 12$ ②

$-x_1 + 3x_2 \leq 3$ ③

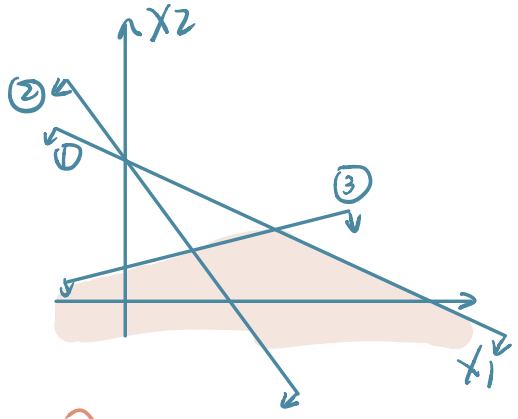
(a)



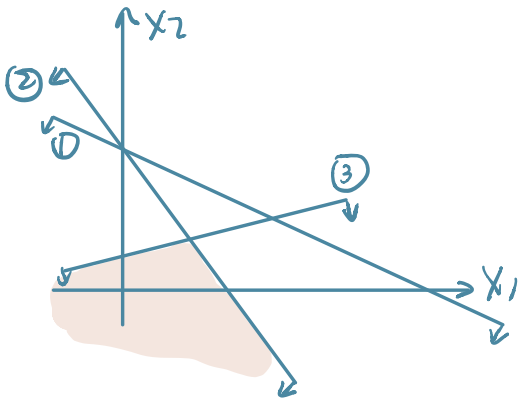
①②



②③



①③



①②③

(b)

$x_1 + 2x_2 \leq 12 + M(1 - y_1)$

$3x_1 + 2x_2 \leq 12 + M(1 - y_2)$

$-x_1 + 3x_2 \leq 3 + M(1 - y_3)$

$y_1 + y_2 + y_3 \geq 2$

y_1, y_2, y_3 are binary numbers.

M = constant

Problem #3.

mixed integer programming

maximize $z = f_1(x_1) + f_2(x_2)$

S.t.

$x_1 + 3x_2 \le 12 + M(1-y_1)$

$2x_1 + x_2 \le 16 + M(1-y_2)$

$x_1 + x_2 \le 9 + M(1-y_3)$

$\max z = f_1(x_1) + f_2(x_2)$

$y_1 + y_2 + y_3 \ge 2$

$x_1 = x_{11} + x_{12}$

$x_2 = x_{21} + x_{22}$

$f_1(x_1) = 10 + 2x_{11} + x_{12}$

$f_2(x_2) = 8 + x_{21} + 3x_{22}$

$5z_1 \le x_{11} \le 5$

$0 \le x_{12} \le 8$

$0 \le x_{12} \le M$

$2z_2 \le x_{21} \le 2$

$0 \le x_{22} \le 8$

$0 \le x_{22} \le M$

$x_1, x_2 \ge 0$

y_1, y_2, y_3, z_1, z_2 are binary #s.

Problem #4.

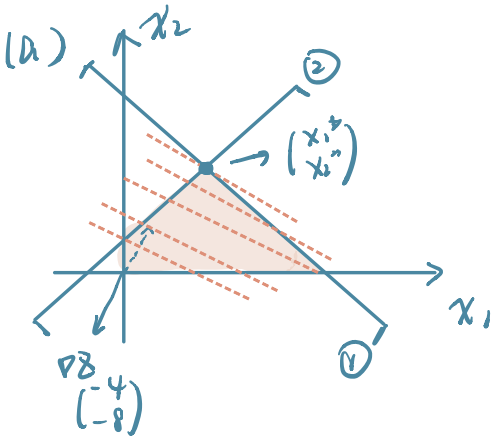
$$\min \quad z = -4x_1 - 8x_2$$

$$\text{s.t.} \quad 2x_1 + 2x_2 \le 19$$

$$-2x_1 + 2x_2 \le 3$$

$$x_1, x_2 \in \mathbb{I}^+$$

(a)



$x_1^* = 4$

$x_2^* = 5.$

$z^* = -4 \times 4 - 8 \times 5 = -16 - 40 = \boxed{-56}$

(b) cvxpy

```
@author: fionafei
"""

import cvxpy as cp



x = cp.Variable(2, nonneg  = True) # vector variable


#obj_func=170*x[0]+220*x[1]+250*x[2]+55*x[3]
obj_func_neg=-4*x[0]-8*x[1]

constraints = []
constraints.append(x[0]>=0)
constraints.append(x[1]>=0)
constraints.append(2*x[0]+2*x[1]<=19)
constraints.append(-2*x[0]+2*x[1]<=3)



#problem = cp.Problem(cp.Maximize(obj_func), constraints)
problem = cp.Problem(cp.Minimize(obj_func_neg), constraints)

#problem.solve(solver=cp.CVXOPT,verbose = True)
#problem.solve(verbose = True)
problem.solve(solver=cp.GUROBI,verbose = True)

print("obj_func =")
#print(obj_func.value)
print(obj_func_neg.value)
print("x =")
print(x.value)
```

```
    Value: 1  Min: 0  Max: 1  Default: 1
Changed value of parameter QCPDual to 1
   Prev: 0  Min: 0  Max: 1  Default: 0
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (mac64)
Optimize a model with 6 rows, 2 columns and 8 nonzeros
Model fingerprint: 0xe66ed663
Coefficient statistics:
  Matrix range     [1e+00, 2e+00]
  Objective range  [4e+00, 8e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [3e+00, 2e+01]
Presolve removed 6 rows and 2 columns
Presolve time: 0.00s
Presolve: All rows and columns removed
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0   -6.0000000e+01   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.01 seconds
Optimal objective -6.000000000e+01
obj_func =
-60.0
x =
[4.  5.5]
```

Problem #5

7 full, 7 half, 7 empty.

↓

7. same.

Total wine $= 7 + \frac{7}{2} = \frac{14+7}{2} = \frac{21}{2}$.

Amount for each person:

$\frac{21}{2} \cdot \frac{1}{3} = \frac{7}{2}$

x_{1i} = # of full bottles to person 1, 2, 3

x_{2i} = # .. half-full .....

x_{3i} = # .. empty .. .. ..

$i = 1, 2, 3$.

person 1

$x_{11} + x_{12} + x_{13} = 7$

$x_{21} + x_{22} + x_{23} = 7$

$x_{31} + x_{32} + x_{33} = 7$

$x_{11} + x_{21} + x_{31} = 7$

$x_{12} + x_{22} + x_{32} = 7$

$x_{13} + x_{23} + x_{33} = 7$

$x_{11} + \frac{x_{12}}{2} = \frac{7}{2}$

$$x_{21} + \frac{x_{22}}{2} = \frac{7}{2}$$

$$x_{31} + \frac{x_{32}}{2} = \frac{7}{2}$$

$$x_{ij} \in \mathbb{I}^{+}, \quad i = 1, 2, 3, \quad j = 1, 2, 3$$

objective fun:

$$\min z = \sum_{i,j=1}^{3} 0 \cdot x_{ij}$$

```python
import cvxpy as cp



x = cp.Variable((3,3), nonneg  = True) # vector variable


#obj_func=170*x[0]+220*x[1]+250*x[2]+55*x[3]
obj_func_neg=0*x[0,0]+0*x[0,1]+0*x[0,2]+0*x[1,0]+0*x[1,1]+0*x[1,2]+0*x[2,0]+0*x[2,1]+0*x[2,2]

constraints = []
constraints.append(x[0,0]>=0)
constraints.append(x[0,1]>=0)
constraints.append(x[0,2]>=0)
constraints.append(x[1,0]>=0)
constraints.append(x[1,1]>=0)
constraints.append(x[1,2]>=0)
constraints.append(x[2,0]>=0)
constraints.append(x[2,1]>=0)
constraints.append(x[2,2]>=0)

constraints.append(x[0,0]+x[1,0]+x[2,0]==7)
constraints.append(x[0,1]+x[1,1]+x[2,1]==7)
constraints.append(x[0,2]+x[1,2]+x[2,2]==7)

constraints.append(x[0,0]+x[0,1]+x[0,2]==7)
constraints.append(x[1,0]+x[1,1]+x[1,2]==7)
constraints.append(x[2,0]+x[2,1]+x[2,2]==7)

constraints.append(x[0,0]+x[0,1]/2==3.5)
constraints.append(x[1,0]+x[1,1]/2==3.5)
constraints.append(x[2,0]+x[2,1]/2==3.5)




#problem = cp.Problem(cp.Maximize(obj_func), constraints)
problem = cp.Problem(cp.Minimize(obj_func_neg), constraints)

#problem.solve(solver=cp.CVXOPT,verbose = True)
#problem.solve(verbose = True)
problem.solve(solver=cp.GUROBI,verbose = True)

print("obj_func =")
#print(obj_func.value)
print(obj_func_neg.value)
print("x =")
print(x.value)
```

```
    Prev: 0  Min: 0  Max: 1  Default: 0
Gurobi Optimizer version 9.0.3 build v9.0.3rc0 (mac64)
Optimize a model with 27 rows, 9 columns and 42 nonzeros
Model fingerprint: 0xb402285e
Coefficient statistics:
  Matrix range     [5e-01, 1e+00]
  Objective range  [0e+00, 0e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [4e+00, 7e+00]
Presolve removed 27 rows and 9 columns
Presolve time: 0.01s
Presolve: All rows and columns removed
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.01 seconds
Optimal objective  0.000000000e+00
obj_func =
0.0
x =
[[0.  7.  0. ]
 [3.5 0.  3.5]
 [3.5 0.  3.5]]
```

Problem #6.

z_i = if choose company i, $i = A, B, C$

x_i = # of minutes for company i, $i = A, B, C$

y_i = amount of money paid for company i, $i = A, B, C$

$$\min \; z = y_A + y_B + y_C$$

S.t.

$$x_A = x_{A1} + x_{A2}$$

$$x_C = x_{C1} + x_{C2}$$

$$x_A + x_B + x_C = W \geq 3000$$

$$0 \leq x_A \leq z_A M$$

$$0 \leq x_B \leq z_B M$$

$$0 \leq x_C \leq z_C M$$

$$1000\, z_1 A \leq x_{A1} \leq 1000$$

$$0 \leq x_{A2} \leq z_{A1} M$$

$$1000\, z_1 C \leq x_{C1} \leq 1000$$

$$0 \leq x_{C2} \leq z_1 C M$$

$$y_A = 10 z_A + 0.05 x_{A1} + 0.04 x_{A2}$$

$$y_B = 20 z_B + 0.04 x_B$$

$$y_C = 25 z_C + 0.05 x_{C1} + 0.035 x_{C2}$$

$$z_A + z_B + z_C = 1$$

$z_A, z_B, z_C, z_1 A, z_1 C$ are binary numbers

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sat Nov 14 13:10:17 2020

@author: fionafei
"""

import cvxpy as cp

x = cp.Variable((5,1), integer = True) # vector variable
z = cp.Variable((3,1), boolean = True)
w = cp.Variable((2,1), boolean = True)


#obj_func=170*x[0]+220*x[1]+250*x[2]+55*x[3]
obj_func_neg=10*z[0,0]+0.05*x[0,0]+0.04*x[1,0]+20*z[1,0]+0.04*x[2,0]+25*z[2,0]
+0.05*x[3,0]+0.035*x[4,0]

constraints = []
constraints.append(x[0,0]+x[1,0]+x[2,0]+x[3,0]+x[4,0]>=3000)
constraints.append(x>=0)
constraints.append(x[0,0]+x[1,0]<=z[0,0]*3000)
constraints.append(x[2,0]<=z[1,0]*3000)
constraints.append(x[3,0]+x[4,0]<=z[2,0]*3000)
constraints.append(x[0,0]<=1000)
constraints.append(x[0,0]>=1000*w[0,0])
constraints.append(x[1,0]<=3000*w[0,0])
constraints.append(x[3,0]<=1000)
constraints.append(x[3,0]>=1000*w[1,0])
constraints.append(x[4,0]<=3000*w[1,0])
constraints.append(z[0,0]+z[1,0]+z[2,0]==1)

#problem = cp.Problem(cp.Maximize(obj_func), constraints)
problem = cp.Problem(cp.Minimize(obj_func_neg), constraints)

#problem.solve(solver=cp.CVXOPT,verbose = True)
#problem.solve(verbose = True)
problem.solve(solver=cp.GUROBI,verbose = True)

print("obj_func =")
#print(obj_func.value)
print(obj_func_neg.value)
print("x =")
print(x.value)
```

```
  Matrix range     [1e+00, 3e+03]
  Objective range  [4e-02, 2e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 3e+03]
Presolve removed 16 rows and 10 columns
Presolve time: 0.02s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.03 seconds
Thread count was 1 (of 4 available processors)

Solution count 1: 140

Optimal solution found (tolerance 1.00e-04)
Best objective 1.400000000000e+02, best bound 1.400000000000e+02, gap 0.0000%
obj_func =
140.0
x =
[[1000.]
 [2000.]
 [   0.]
 [   0.]
 [   0.]]
```