MAT 3007 - 2023 Spring
Assignment 2 Solution

Problem 1

1. False. Consider the following counterexample:

$$\begin{array}{ll} \text{minimize} & 0 \\ \text{s.t.} & x \geq 0 \end{array}$$

then the optimal solution set is unbounded.

2. False. Consider the following counterexample: *minimize* 0. Then any feasible x is optimal no matter how many positive components it has.

3. True. If x_1 and x_2 are optimal solution, then, any convex combination of x_1, x_2 is also optimal. Specifically, $\bar{x} = \gamma x_1 + (1-\gamma)x_2$ is optimal for any $\gamma \in [0, 1]$.

Problem 2

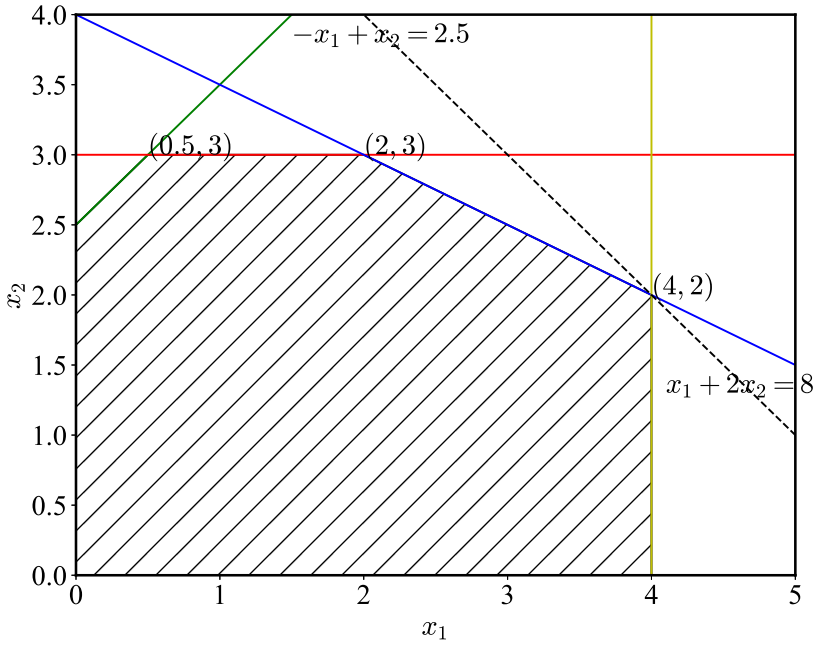

Figure 1: Graphical Illustration of Problem 2

From Figure 1, we can find all the basic feasible solutions of the feasible region:

$$(0,0),(0,2.5),(0.5,3),(2,3),(4,2),(4,0).$$

We can find that the optimal solution is $(4,2)$, and the optimal value is 6. Since the optimal solution is at $(4,2)$, the following two lines that intersect at that point are the active constraints.

$$\begin{aligned} x_1 + 2x_2 &\leq 8 \\ x_1 &\leq 4 \end{aligned}$$

Problem 3

1. Standard form is as follows.

$$\begin{aligned} \text{minimize} \quad & -x_1 - 4x_2 - x_3 \\ \text{s.t.} \quad & 2x_1 + 2x_2 + x_3 + x_4 = 4 \\ & -x_1 + x_3 + x_5 = -1 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

2. See Table 1 for the basic solutions and basic feasible solutions.

B	{1, 2}	{1, 3}	{1, 4}	{1, 5}	{2, 3}	{2, 5}	{3, 4}	{3, 5}	{4, 5}
x_B	(1, 1)	$(\frac{5}{3}, \frac{2}{3})$	(1, 2)	(2, 1)	$(\frac{5}{2}$, -1)	(2, -1)	(-1, 5)	(4, -5)	(4, -1)
Obj. Val.	-5	$-\frac{7}{3}$	-1	-2	-	-	-	-	-
BFS	Y	Y	Y	Y	N	N	N	N	N

Table 1: Basic solutions and basic feasible solutions

3. Comparing basic feasible solutions, we can find that the optimal solution is (1, 1, 0, 0, 0)

Problem 4

1. First, the decision variables should be $\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)^T$, which are purchase amounts of these securities. Second, we have constraints that purchase amounts are nonnegative ($\mathbf{x} \geq 0$), and each purchase amount can not exceed the corresponding share limit ($\mathbf{x} \leq \mathbf{q}$). Third, the objective is to maximize the worst-case payoff, $\min_j\{A_j^T\mathbf{x} - \pi^T\mathbf{x}\}$, where A_j is the j-th column of the matrix A. To transform this problem into

linear programming, we can split the objective into multiple constraints and have the LP formulation as follows:

$$\begin{array}{ll} \textit{maximize} & z \\ \textit{s.t.} & z \leq A_j^T \mathbf{x} - \pi^T \mathbf{x} \quad \forall\, j = 1, 2, \cdots, 5 \\ & \mathbf{x} \leq \mathbf{q} \\ & \mathbf{x} \geq 0 \end{array} \tag{1}$$

2. Using MATLAB or Python to solve (1), we get the optimal value 1 and the optimal solution $(0, 0, 5, 5, 5)$. Therefore, to maximize the worst-case revenue, we should buy 5 shares each of security 3, 4 and 5. In this case, we can earn at least 1 dollar.

Matlab Code

```
  A = [1 1 1 0 0
  0 0 0 1 1
  1 0 1 0 1
  1 1 1 1 0
  0 1 0 1 1];
pi = [0.75; 0.35; 0.40; 0.75; 0.65];
q = [10; 5; 10; 10; 5];

cvx_begin
 variables x(5) z

 maximize z
 subject to
     for i = 1:5
         z <= A(:, i)'*x - pi'*x
     end
     x <= q
     x >= 0
cvx_end
```

Python Code

```
import numpy as np
import cvxpy as cp

x = cp.Variable(5)
z = cp.Variable(1)

A = np.array([[1, 1, 1, 0, 0],
              [0, 0, 0, 1, 1],
              [1, 0, 1, 0, 1],
              [1, 1, 1, 1, 0],
              [0, 1, 0, 1, 1]])
pi = np.array([0.75, 0.35, 0.4, 0.75, 0.65])
q = np.array([10, 5, 10, 10, 5])
```

```
objective = cp.Maximize(z)
constraints = [z <= A[:, i].T@x - pi@x for i in range(5)] + [x<=q, x>=0]

prob = cp.Problem(objective, constraints)
prob.solve()

print(prob.value)
print(x.value)
```

Problem 5

The polygon is illustrated in Figure 2, and can be described using the following inequalities $\{\mathbf{a}_i^T\mathbf{x} - b_i \leq 0, i = 1, 2, \ldots, 8\}$:

$$\begin{aligned} -x_1 - x_2 &\leq -1 \\ -x_1 + x_2 &\leq 6 \\ x_1 + x_2 &\leq 18 \\ x_1 - x_2 &\leq 7 \\ x_1 &\leq 11 \\ x_2 &\leq 10 \\ -x_1 &\leq 0 \\ -x_2 &\leq 0 \end{aligned}$$

We only need a center and a radius to uniquely define a circle, and hence we have the following decision variables: $\mathbf{y} = (y_1, y_2)$ is the center of the circle, and r is the radius. Similar to Exercise 1 in Tutorial 1, this problem can be solved via a linear program.

$$\begin{array}{lll} \text{maximize} & r & \\ \text{s.t.} & \mathbf{a}_i^T\mathbf{y} \leq b_i & \forall\, i = 1, 2, \ldots, 8 \\ & r \leq \frac{|\mathbf{a}_i^T\mathbf{y} - b_i|}{||\mathbf{a}_i||} & \forall\, i = 1, 2, \ldots, 8 \\ & r \geq 0 & \end{array} \tag{2}$$

The first constraint in (2) makes sure that the center of the circle is in the feasible region. In order for the circle to be in the feasible region, the distance from the center of the circle to the line defined by the extreme points must be greater than the radius r, which gives us the second constraint in (2).

However, the formulation in (2) is not yet a LP, and hence we may find some way to get rid of the abs operation. In this case, we cannot use the technique mentioned in Lecture 3 to eliminate the abs operation. Observing that all points in the feasible region satisfy the constraints $\{\mathbf{a}_i^T\mathbf{x} - b_i \leq 0, i = 1, 2, \ldots, 8\}$, we can get that $|\mathbf{a}_i^T\mathbf{y} - b_i| = b_i - \mathbf{a}_i^T\mathbf{y}$, and hence

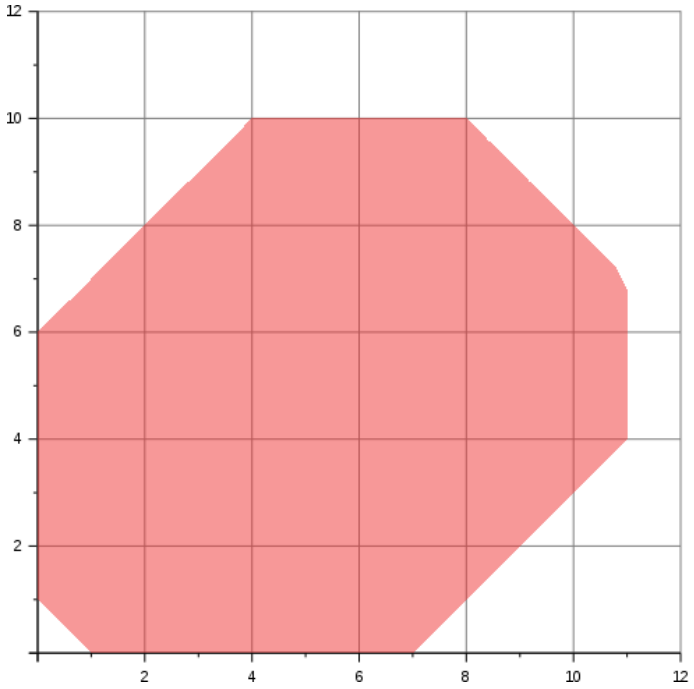

Figure 2: Graphical Illustration of Problem 4

can simplify (2) as follows:

$$\begin{array}{ll} \text{maximize} & r \\ \text{s.t.} & \mathbf{a}_i^T\mathbf{y} + ||\mathbf{a}_i||r \leq b_i \quad \forall\, i = 1, 2, \ldots, 8 \\ & r \geq 0 \end{array} \tag{3}$$

Using MATLAB or Python to solve (3), we get the optimal solution $r = 4.596$, and one possible solution of the center (y_1, y_2) is $(5.4352, 4.9352)$. Note that the center is not unique. It lies on the line segment $y_1 - y_2 = 0.5$ with interval $y_1 \in [5.096, 5.904]$.

Matlab Code

```
cvx_begin
  variables y1 y2 r

  maximize r
  subject to
      -y1 - y2 + sqrt(2)*r <= -1
      -y1 + y2 + sqrt(2)*r <= 6
      y1 + y2 + sqrt(2)*r <= 18
      y1 - y2 + sqrt(2)*r <= 7
      y1       + r        <= 11
           y2 + r         <= 10
      -y1      + r        <= 0
         - y2 + r         <= 0
```

```
cvx_end
```

Python Code

```
import numpy as np
import cvxpy as cp

y = cp.Variable(2)
r = cp.Variable(1)

A = np.array([[-1, -1], [-1, 1], [1, 1], [1, -1], [1, 0], [0, 1], [-1, 0], [0, -1]])
d = np.array([np.sqrt(2)]*4 + [1]*4)
c = np.array([0, 0])
b = np.array([-1, 6, 18, 7, 11, 10, 0, 0])

objective = cp.Maximize(c@y + r)
constraints = [A@y + d*r <= b]

prob = cp.Problem(objective, constraints)
prob.solve()

print(prob.value)
print(y.value)
```