MAT 170 Homework 1 Project Report

Wanzhu Zheng
Student ID: 918166670

July 25, 2024

1 Problem 1A

1.1 Model

Let x, y be the number of basic and deluxe tables respectively. Our goal is to maximize total profit through the following linear programming model:

$$
\begin{aligned}
\max\ & 200x + 350y \\
\text{subject to}\quad & x \le 50 \\
& y \le 35 \\
& 5x + 5y \le 300 \\
& 0.6x + 1.5y \le 63 \\
& x, y \ge 0 \\
& x, y \in Z
\end{aligned}
$$

1.2 Code in CVXPY

```
# Import packages
import cvxpy as cp
import numpy as np

# Define the optimization problem for Table production problem
# Define the variables
X = cp.Variable() # number of basic tables
Y = cp.Variable() # number of deluxe tables

obj = 200*X + 350*Y
```

```
constraints = [X <= 50,\
               Y <= 35,\
               5*X + 5*Y <= 300,\
               0.6*X + 1.5*Y <= 63,\
               X >= 0, Y >= 0]
prob = cp.Problem(cp.Maximize(obj), constraints)

prob.solve()
# prob.solve(verbose=True)

# Print result.
print("\nThe maximum profit is", prob.value)
print("We produce {} basic tables and {} deluxe tables.".format(X.value,Y.value))
```

1.3 Results and visualizations

The optimal solution is 30 basic tables and 30 deluxe tables with a maximum profit of $16,500.

```
The maximum profit is 16499.999999350166
We produce 29.99999999591339 basic tables and 30.00000000047854 deluxe tables.
```

Figure 1: CVXPY Results

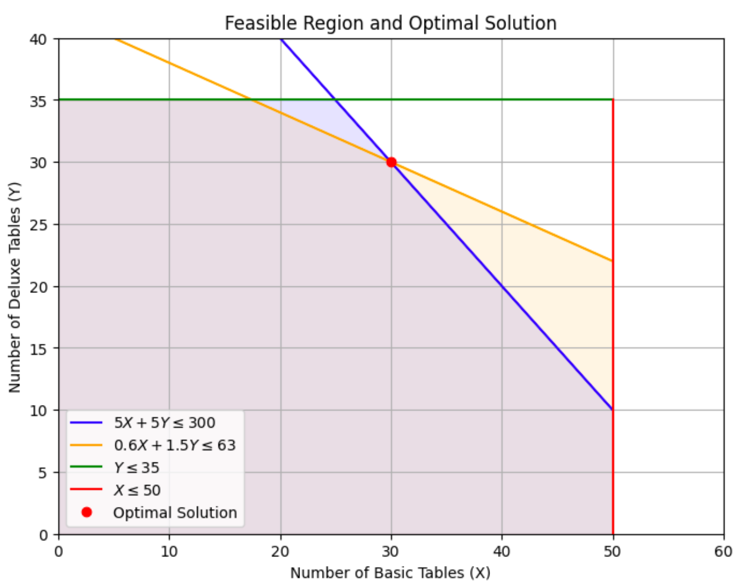

Figure 2: Feasible Region

2 Problem 1B

2.1 Model

Let x_{DI}, x_{DII} be the amount of Drug I and Drug II produced (in 1000 packs) respectively. Let x_{RI}, x_{RII} be the amount of raw materials (in kg) to be purchased. Our goal is to maximize profit with the following model:

$$
\begin{aligned}
&\min f_{costs}(x) - f_{income}(x) \\
\text{subject to } & 0.01x_{RI} + 0.02x_{RII} - 0.5x_{DI} - 0.6x_{DII} \geq 0 \\
& x_{RI} + x_{RII} \leq 1000 \\
& 90x_{DI} + 100x_{DII} \leq 2000 \\
& 40x_{DI} + 50x_{DII} \leq 800 \\
& 100x_{RI} + 199.90x_{RII} + 700x_{DI} + 800x_{DII} \leq 100,000 \\
& x_{RI} \geq 0, x_{RII} \geq 0, x_{DI} \geq 0, x_{DII} \geq 0
\end{aligned}
$$

where
$f_{revenue}(x) = 6500x_{DI} + 7100x_{DII}$ and,
$f_{costs}(x) = 100x_{RI} + 199.90x_{RII} + 700x_{DI} + 800x_{DII}$

2.2 Code in CVXPY

```
R1 = cp.Variable() # amount of Raw I Materials
R2 = cp.Variable() # amount of Raw II Materials
D1 = cp.Variable() # amount of Drug I
D2 = cp.Variable() # amount of Drug II

obj = 100*R1 + 199.9*R2 + 700*D1 + 800*D2 - 6500*D1 - 7100*D2
constraints = [0.01*R1 + 0.02*R2 - 0.5*D1 - 0.6*D2 >=0,\
               R1 + R2 <= 1000,\
               90*D1 + 100*D2 <= 2000,\
               40*D1 + 50*D2 <= 800,\
               100*R1 + 199.90*R2 + 700*D1 + 800*D2 <= 100000,\
               R1 >= 0, R2 >= 0, D1 >= 0, D2 >= 0]
prob = cp.Problem(cp.Minimize(obj), constraints)

prob.solve()
# prob.solve(verbose=True)

# Print result.
print("\nThe maximum profit is", prob.value)
print("We produce {} amount of Drug I and {} amount of Drug II.".format(D1.value,D2.value))
print("We use {} amount of Raw I and {} amount of Raw II.".format(R1.value,R2.value))
```

2.3 Results

The maximum profit is $14085.13

	$Profit$	$RawI$	$RawII$	$DrugI$	$DrugII$
Solution	-14085.13	0	438.789	17.552	0

Table 1: Results of the Drug Production Problem

```
The maximum profit is -14085.125051591196
We produce 17.551557696761897 amount of Drug I and 5.22042782065505e-10 amount of Drug II.
We use 5.562279621739461e-06 amount of Raw I and 438.7889396532701 amount of Raw II.
```

Figure 3: CVXPY Results

3 Problem 1C

3.1 Model

Let w_{ij} be the weights of the edges, in other words, the cost of matching i to j.

$$
\begin{aligned}
\min\ & \Sigma_{i=1}^{n}\Sigma_{j=1}^{n}\ w_{ij}x_{ij} \\
\text{subject to}\ & \Sigma_{i=1}^{n}\ x_{ij} = 1 \\
& \Sigma_{j=1}^{n}\ x_{ij} = 1 \\
& x_{ij}\ \epsilon\ \{0, 1\}
\end{aligned}
$$

3.2 Code in CVXPY

```
# Define the cost matrix
W = np.array([[5, 1, 2, 2],
              [1, 0, 5, 3],
              [2, 1, 2, 1],
              [1, 1, 2, 3]])
n = W.shape[0]

# Define the variables
x = cp.Variable((n, n))

obj = cp.sum(cp.multiply(W, x))
constraints = [
    cp.sum(x, axis=0) == np.ones(n),  # Each column sums to 1
    cp.sum(x, axis=1) == np.ones(n),  # Each row sums to 1
    x >= 0  # Non-negativity constraint
]
```

```
prob = cp.Problem(cp.Minimize(obj), constraints)
prob.solve()

# Print the optimal cost
print("\nThe optimal cost of the agent/task matching problem is", prob.value)
```

Given the cost matrix W, the optimal cost for the matching problem is 4.

```
The optimal cost of the agent/task matching problem is 4.000000004095184
```

Figure 4: CVXPY Results

3.3 Results

The optimal cost for the matching problem when the variables are set to integers is also 4. The solutions for the Linear Programming Problem and the Integer Programming Problem is summarized below in the table:

	LP	IP
Solution	4.000000004095184	4.000000065691717

Table 2: Results for Weighted Bipartite Matching Problem

4 Problem 1D

4.1 a)

4.1.1 part a- Model

$$
\begin{aligned}
\max\ & \Sigma_{i=1}^{5}\ x_i \\
\text{subject to } & x_1 + x_2 \leq 1 \\
& x_2 + x_3 \leq 1 \\
& x_3 + x_4 \leq 1 \\
& x_4 + x_5 \leq 1 \\
& x_1 + x_5 \leq 1 \\
& \qquad x_i \ \epsilon\ \{0, 1\},\ \ i = 1, 2, 3, 4, 5
\end{aligned}
$$

4.1.2 part a- Code

```
# Define the variables
x = cp.Variable(5, integer=True)
```

```
obj = cp.sum(x)
constraints = [
    x[0] + x[1] <= 1,
    x[1] + x[2] <= 1,
    x[2] + x[3] <= 1,
    x[3] + x[4] <= 1,
    x[0] + x[4] <= 1
]
prob = cp.Problem(cp.Maximize(obj), constraints)
prob.solve(solver='ECOS_BB')

# Print the results
print("\nThe optimal solution for the maximum-cardinality stable set problem is", prob.value
```

4.1.3 part a- Results

The optimal solution for the maximum-cardinality stable set problem on a cycle with 5 vertices is 2 when we solve it with integer programming.

```
The optimal solution for the maximum-cardinality stable set problem is 2.0000000000222005
```

Figure 5: CVXPY Results for IP

4.2 b)

If we use real variables instead of integer variables, the problem becomes a linear programming (LP) problem because we relax the integer constraint. Then, our model will become:

$$
\begin{aligned}
\max \Sigma_{i=1}^{5}\ x_i & \\
\text{subject to } x_1 + x_2 &\leq 1 \\
x_2 + x_3 &\leq 1 \\
x_3 + x_4 &\leq 1 \\
x_4 + x_5 &\leq 1 \\
x_1 + x_5 &\leq 1 \\
x_i \ \epsilon\ &[0, 1]
\end{aligned}
$$

When we solve this model with CVXPY, we get the optimal solution to be $\frac{5}{2}$ so $P_{IP}^* < P_{LP}^*$.

```
The optimal solution for the maximum-cardinality stable set problem is 2.4999999995493436
```

Figure 6: CVXPY Results for LP

4.3 c)

The generalized Integer Programming Problem will be:

$$\begin{aligned} &\max \Sigma_{i=1}^{n}\ x_i \\ &\text{subject to } x_i + x_j \leq 1, \quad \{i,j\}\ \epsilon\ E \\ &\qquad\qquad x_i\ \epsilon\ \{0,1\} \end{aligned}$$

```
The optimal integer solution for the maximum-cardinality stable set problem (n=8) is 4.0
The optimal integer solution for the maximum-cardinality stable set problem (n=17) is 8.000000000068926
The optimal integer solution for the maximum-cardinality stable set problem (n=24) is 11.999999999068677
```

Figure 7: CVXPY Results for IP

We relax the integer constraint, the Linear Programming Problem will be:

$$\begin{aligned} &\max \Sigma_{i=1}^{n}\ x_i \\ &\text{subject to } x_i + x_j \leq 1, \quad \{i,j\}\ \epsilon\ E \\ &\qquad\qquad x_i\ \epsilon\ [0,1] \end{aligned}$$

```
The optimal real solution for the maximum-cardinality stable set problem (n=8) is 3.9999999995139817
The optimal real solution for the maximum-cardinality stable set problem (n=17) is 8.499999999469608
The optimal real solution for the maximum-cardinality stable set problem (n=24) is 11.999999999452477
```

Figure 8: CVXPY Results for LP

The optimal solutions for the generalized maximum-cardinality problem are recorded in the table below:

	$n = 8$	$n = 17$	$n = 24$
IP Solution	4	8	12
LP Solution	4	$\frac{17}{2}$	12

Table 3: Results for IPP and LPP of Max-Cardinality Problem

Conjecture $P^*_{IP} < P^*_{LP}$ when the number of vertices, n, in the graph is odd.
Proof
Let G be a graph such that there are $2n + 1$ vertices.

Case 1: Integer Programming. We are maximizing $x_1 + x_2 + \cdots + x_{2n+1}$ subject to $2n + 1$ inequality constraints, and the integer constraint.

$$\begin{aligned} \max\, & x_1 + x_2 + \cdots + x_{2n+1} \\ \text{subject to } & x_1 + \quad x_2 \leq 1 \\ & x_2 + \quad x_3 \leq 1 \\ & \vdots \\ & x_{2n} + x_{2n_1} \leq 1 \\ & x_1 + x_{2n_1} \leq 1 \\ & x_i \,\epsilon\, \{0, 1\} \end{aligned}$$

Since $x_i + x_j \leq 1$, that means only x_i or x_j can be 1 and the other must be 0. WLOG, let's set $x_1 = 1$ and consequently, $x_2 = 0$. Then our objective function becomes

$$\begin{aligned} & \max 1 + 0 + 1 + 0 + +0 + 1 \\ & = \lfloor \frac{2n+1}{2} \rfloor \end{aligned}$$

Case 2: Linear Programming. Again, we are maximizing $x_1 + x_2 + \cdots + x_{2n+1}$ subject to $2n + 1$ inequality constraints, but we relax the integer constraint.

$$\begin{aligned} \max\, & x_1 + x_2 + \cdots + x_{2n+1} \\ \text{subject to } & x_1 + \quad x_2 \leq 1 \\ & x_2 + \quad x_3 \leq 1 \\ & \vdots \\ & x_{2n} + x_{2n_1} \leq 1 \\ & x_1 + x_{2n_1} \leq 1 \\ & x_i \,\epsilon\, [0, 1] \end{aligned}$$

Since $x_i + x_j \leq 1$, that means only x_i or x_j can be 1 and the other must be 0. WLOG, let's set $x_1 = 1$ and consequently, $x_2 = 0$. Then our objective function becomes

$$\begin{aligned} & \max 1 + 0 + 1 + 0 + +0 + 1 \\ & = \frac{2n+1}{2} \end{aligned}$$

Note that we don't floor our objective sum because we do not need to uphold integer qualities.

Now, let V be a graph such that there are $2n$ vertices. Using the same logic as above, our integer programming model will result in $\lfloor \frac{2n}{2} \rfloor = n$ and out linear programming model will result in $\frac{2n}{2} = n$.

We have shown that for an odd number of vertices, $\lfloor \frac{2n+1}{2} \rfloor < \frac{2n+1}{2}$ so $P^*_{IP} < P^*_{LP}$, and for an even number of vertices, $\lfloor \frac{2n}{2} \rfloor = \frac{2n}{2} = n$ so $P^*_{IP} = P^*_{LP}$. Therefore, our conjecture holds.

5 Problem 1E

a) Model
Let m, n be the number of students and seminars respectively. Our goal is to minimize the ranking w_{ij} each student i has to seminar j. Note that each seminar has a capacity of b.

$$
\begin{aligned}
\min \; & \Sigma_{i=1}^{m} \Sigma_{j=1}^{n} \; w_{ij} x_{ij} \\
\text{subject to } & \Sigma_{i=1}^{m} \; x_{ij} \leq b, \quad \forall \; j = 1, 2, \ldots, n \\
& \Sigma_{j=1}^{n} \; x_{ij} = 1, \quad \forall \; i = 1, 2, \ldots, m \\
& x_i \; \epsilon \; [0, 1]
\end{aligned}
$$

b) Code

```
# Read data
df = pd.read_csv('/Users/wanzhu_zheng/Downloads/student assignment.csv')

# Define cost matrix
W = (df.values[:, 1:6]) # 26x5

# Define the variables
x = cp.Variable((26, 5), nonneg=True)

obj = cp.sum(cp.multiply(W, x))
constraints = [
    cp.sum(x, axis=0) <= 6,
    cp.sum(x, axis=1) == np.ones(26),
    x >= 0,
    x <= 1
]
prob = cp.Problem(cp.Minimize(obj), constraints)
prob.solve(solver='SCIPY')

avg_ranking = np.sum(np.multiply(W, x.value)) / 26
worst_ranking = np.max(np.multiply(W, x.value))
```

```
# Print the results
print("The optimal cost of the student/seminar matching problem is", prob.value)
print("The average assigned student ranking is", avg_ranking)
print("The worst assigned student ranking is", worst_ranking)
print("The optimal assignmet schedule is:\n", x.value)
```

c) Solution to part a)
The optimal objective cost for the student/seminar assignment is 35. The average assigned student ranking is 1.35. The worst assigned student ranking is 3.

	Objective Cost	*Average*	*Worst*
Solution	35	1.35	3

Table 4: Results for Student/Seminar Problem

```
The optimal cost of the student/seminar matching problem is 35.0
The average assigned student ranking is 1.3461538461538463
The worst assigned student ranking is 3.0
The optimal assignmet schedule is:
 [[-0. -0. -0.  1. -0.]
 [-0.  1. -0. -0. -0.]
 [-0. -0. -0. -0.  1.]
 [-0.  1. -0. -0. -0.]
 [-0. -0. -0.  1. -0.]
 [-0. -0. -0. -0.  1.]
 [-0. -0. -0.  1. -0.]
 [-0. -0.  1. -0. -0.]
 [-0. -0. -0. -0.  1.]
 [-0.  1. -0. -0. -0.]
 [ 1. -0. -0. -0. -0.]
 [-0. -0. -0.  1. -0.]
 [-0.  1. -0. -0. -0.]
 [-0. -0. -0. -0.  1.]
 [-0. -0. -0.  1. -0.]
 [-0.  1. -0. -0. -0.]
 [ 1. -0. -0. -0. -0.]
 [-0.  1. -0. -0. -0.]
 [-0. -0.  1. -0. -0.]
 [-0. -0. -0.  1. -0.]
 [-0. -0. -0. -0.  1.]
 [ 1. -0. -0. -0. -0.]
 [-0. -0. -0. -0.  1.]
 [ 1. -0. -0. -0. -0.]
 [-0. -0.  1. -0. -0.]
 [ 1. -0. -0. -0. -0.]]
```

Figure 9: CVXPY Results

d) Solution to part b)
The optimal objective cost for the student/seminar assignment is 35. The average assigned student ranking is 1.35. The worst assigned student ranking is now 2.

	Objective Cost	*Average*	*Worst*
Solution	35	1.35	2

Table 5: Results for Student/Seminar Problem with added constraint

```
The optimal cost of the student/seminar matching problem is 35.0
The average assigned student ranking is 1.3461538461538463
The worst assigned student ranking is 2.0
The optimal assignmet schedule is:
 [[-0.         -0.         -0.          1.         -0.        ]
 [-0.          1.         -0.         -0.         -0.        ]
 [-0.         -0.         -0.         -0.          1.        ]
 [-0.          1.         -0.         -0.         -0.        ]
 [-0.         -0.         -0.          1.         -0.        ]
 [-0.         -0.         -0.         -0.          1.        ]
 [-0.         -0.         -0.          1.         -0.        ]
 [-0.         -0.          1.         -0.         -0.        ]
 [-0.         -0.         -0.         -0.          1.        ]
 [-0.          1.         -0.         -0.         -0.        ]
 [ 0.66666667 -0.         -0.          0.33333333 -0.        ]
 [-0.         -0.         -0.          1.         -0.        ]
 [-0.          1.         -0.         -0.         -0.        ]
 [-0.         -0.         -0.         -0.          1.        ]
 [ 0.66666667 -0.         -0.          0.33333333 -0.        ]
 [-0.          1.         -0.         -0.         -0.        ]
 [ 1.         -0.         -0.         -0.         -0.        ]
 [-0.          1.         -0.         -0.         -0.        ]
 [-0.         -0.          1.         -0.         -0.        ]
 [-0.         -0.         -0.          1.         -0.        ]
 [-0.         -0.         -0.         -0.          1.        ]
 [ 1.         -0.         -0.         -0.         -0.        ]
 [-0.         -0.         -0.         -0.          1.        ]
 [ 1.         -0.         -0.         -0.         -0.        ]
 [-0.         -0.          0.66666667  0.33333333 -0.        ]
 [ 1.         -0.         -0.         -0.         -0.        ]]
```

Figure 10: CVXPY Results

From Figure 10, we notice that because we are using a linear programming model, the decision variable contains non-integers. Now, if we set stricter constraints such that the decision variable must be an integer, the problem can't be solved to optimality.