**MAT3007 2023 Spring**
**Assignment 1 Solution**

---

**1.(a)** Let $x_1$ be the number of type 1 product, and $x_2$ be the number of type 2 product.

$$\begin{array}{ll} \text{maximize} & (9-1.2)x_1 + (8-0.9)x_2 \\ \text{s.t.} & \frac{1}{4}x_1 + \frac{1}{3}x_2 \le 90 \\ & \frac{1}{8}x_1 + \frac{1}{3}x_2 \le 80 \\ & x_1, x_2 \ge 0 \end{array}$$

**(b)** The standard form is as follows.

$$\begin{array}{ll} \text{minimize} & -(9-1.2)x_1 - (8-0.9)x_2 \\ \text{s.t.} & \frac{1}{4}x_1 + \frac{1}{3}x_2 + s_1 = 90 \\ & \frac{1}{8}x_1 + \frac{1}{3}x_2 + s_2 = 80 \\ & x_1, x_2, s_1, s_2 \ge 0 \end{array}$$

**(c)** Let $x_3$ be the number of overtime hours. We can form the following LP.

$$\begin{array}{ll} \text{maximize} & (9-1.2)x_1 + (8-0.9)x_2 - 7x_3 \\ \text{s.t.} & \frac{1}{4}x_1 + \frac{1}{3}x_2 \le 90 + x_3 \\ & \frac{1}{8}x_1 + \frac{1}{3}x_2 \le 80 \\ & x_3 \le 50 \\ & x_1, x_2, x_3 \ge 0 \end{array}$$

**(d)** The optimal objective value is 2808. The optimal solutions are $x_1 = 360, x_2 = 0$.

## Matlab

```
cvx_begin
variables x1 x2

minimize -(9-1.2)*x1 - (8-0.9)*x2
subject to
        1/4*x1 + 1/3*x2 <= 90;
        1/8*x1 + 1/3*x2 <= 80;
        x1 >= 0;
        x2 >= 0;
cvx_end
```

## Python

```
import numpy as np
import cvxpy as cp
```

```
x = cp.Variable(2)

A = np.array([[1/4, 1/3], [1/8, 1/3]])
c = np.array([-(9-1.2), -(8-0.9)])
b = np.array([90, 80])

objective = cp.Minimize(c@x)
constraints = [A@x <= b, x >= 0]

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

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

---

**2**. Let $y_1 = |x_1 - x_3|$, $y_2 = |x_1 + 2|$, $y_3 = |x_2|$.

$$
\begin{array}{ll}
\text{minimize} & 2x_2 + y_1 \\
\text{s.t.} & y_1 \geq x_1 - x_3 \\
& y_1 \geq -x_1 + x_3 \\
& y_2 + y_3 \leq 5 \\
& y_2 \geq x_1 + 2 \\
& y_2 \geq -x_1 - 2 \\
& y_3 \geq x_2 \\
& y_3 \geq -x_2 \\
& x_3 \leq 1 \\
& x_3 \geq -1
\end{array}
$$

---

**3**. Let $x_{ijg}$ be the number of students from neighborhood $i$ going to school $j$ in grade $g$.

$$
\begin{array}{lll}
\text{minimize} & \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{g=1}^{G} x_{ijg} d_{ij} & \\
\text{s.t.} & \sum_{j=1}^{J} x_{ijg} = S_{ig} & \forall i, g \\
& \sum_{i=1}^{I} x_{ijg} \leq C_{jg} & \forall j, g \\
& x_{ijg} \geq 0 & \forall i, j, g
\end{array}
$$

---

**4**. Let $x_{ij}$ denote the number of cars moving from place $i$ to $j$.

$$
\begin{array}{ll}
\text{minimize} & \sum_{i=1}^{5}\sum_{j=1}^{5} c_{ij}x_{ij} \\
\text{s.t.} & \sum_{i\neq 1} x_{i1} - \sum_{j\neq 1} x_{1j} \geq 200 - 115 \\
& \sum_{i\neq 2} x_{i2} - \sum_{j\neq 2} x_{2j} \geq 500 - 385 \\
& \sum_{i\neq 3} x_{i3} - \sum_{j\neq 3} x_{3j} \geq 800 - 410 \\
& \sum_{i\neq 4} x_{i4} - \sum_{j\neq 4} x_{4j} \geq 200 - 480 \\
& \sum_{i\neq 5} x_{i5} - \sum_{j\neq 5} x_{5j} \geq 300 - 610 \\
& x_{ij} \geq 0 \qquad \forall i, j
\end{array}
$$

The optimal solutions are $x_{42} = 115, x_{43} = 165, x_{51} = 85, x_{53} = 225$, with the rest $x_{ij} = 0$. The optimal objective value is 11370.

## Matlab

```
c = [0 10 12 17 34;
    10 0 18 8 46;
    12 18 0 9 27;
    17 8 9 0 20;
    34 46 27 20 0];

cvx_begin
variables x(5,5)


minimize sum(sum(c.*x))
subject to
        x(2,1)+x(3,1)+x(4,1)+x(5,1)-x(1,2)-x(1,3)-x(1,4)-x(1,5) >= 200-115 ;
        x(1,2)+x(3,2)+x(4,2)+x(5,2)-x(2,1)-x(2,3)-x(2,4)-x(2,5) >= 500-385 ;
        x(1,3)+x(2,3)+x(4,3)+x(5,3)-x(3,1)-x(3,2)-x(3,4)-x(3,5) >= 800-410 ;
        x(1,4)+x(2,4)+x(3,4)+x(5,4)-x(4,1)-x(4,2)-x(4,3)-x(4,5) >= 200-480 ;
        x(1,5)+x(2,5)+x(3,5)+x(4,5)-x(5,1)-x(5,2)-x(5,3)-x(5,4) >= 300-610 ;
        x >= zeros(5,5);
cvx_end
```

## Python

```
import numpy as np
import cvxpy as cp

X = cp.Variable((5, 5))
C = np.array([
    [0, 10, 12, 17, 34],
    [10, 0, 18, 8, 46],
    [12, 18, 0, 9, 27],
    [17, 8, 9, 0, 20],
    [34, 46, 27, 20, 0],
])
b = np.array([200 - 115, 500 - 385, 800 - 410, 200 - 480, 300 - 610])
```

```
objective = cp.Minimize(cp.sum(cp.multiply(C, X)))
constraints = [sum(X[:, i])-sum(X[i, :]) >= b[i] for i in range(5)]
constraints += [X >= 0]

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

print(X.value)
print(prob.value)
```

---

**5**. The optimal solution to the vertex cover problem without the integrality constraint is $x_a = x_b = x_c = x_d = x_e = x_f = x_g = x_h = x_i = x_j = 0.5$ and the optimal value is 5.

## Matlab

```
cvx_begin
variables xa xb xc xd xe xf xg xh xi xj
minimize xa + xb + xc + xd + xe + xf + xg + xh + xi + xj
subject to
    xa + xb >= 1
    xa + xe >= 1
    xa + xf >= 1
    xb + xc >= 1
    xb + xg >= 1
    xc + xd >= 1
    xc + xh >= 1
    xd + xe >= 1
    xd + xi >= 1
    xe + xj >= 1
    xf + xh >= 1
    xf + xi >= 1
    xg + xi >= 1
    xg + xj >= 1
    xh + xj >= 1
    0 <= xa, xb, xc, xd, xe, xf, xg, xh, xi, xj <= 1
cvx_end
```

## Python

```
import numpy as np
import cvxpy as cp

x = cp.Variable(10)

A = np.zeros((15, 10))
node_set = {"a":0, "b":1, "c":2, "d":3, "e":4, "f":5, "g":6, "h":7, "i":8, "j":9}
edge_set = [["a", "b"], ["a", "e"], ["a", "f"], ["b", "c"], ["b", "g"],
            ["c", "d"], ["c", "h"], ["d", "e"], ["d", "i"], ["e", "j"],
```

```
            ["f", "h"], ["f", "i"], ["g", "i"], ["g", "j"], ["h", "j"]]

for i in range(15):
    n1 = edge_set[i][0]
    n2 = edge_set[i][1]
    A[i][node_set[n1]] = 1
    A[i][node_set[n2]] = 1

c = np.ones(10)
b = np.ones(15)

objective = cp.Minimize(c@x)
constraints = [A@x >= b, x >= 0, x <= 1]

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

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

However, for this problem, it is easy to observe that the optimal value (minimal number of vertex to cover the graph) is 6. (One can also solve the integer version of the linear optimization problem using Gurobi or Mosek and obtain that the optimal value is 6.) Therefore, for this problem, one cannot remove the integer constraints when solving it.