Operations Research, Spring 2024 (112-2)

Homework 3

B11705018 Shu-Ting Hsu

1. A company produces three kinds of products in a factory. Each unit of product 1 requires 2 units of raw materials and 30 minutes of machine processing, each unit of product 2 requires 3 units of raw materials and 20 minutes of machine processing, and each unit of product 3 requires 4 units of raw materials and 40 minutes of machine processing. In each day, there are at most 500 units of raw materials and 80 machine hours (i.e., 4800 minutes) that can be used. Currently, the daily demand (which is the maximum number of units that can be sold) for products 1, 2, and 3 are 100, 80, and 50, respectively. The unit prices of products 1, 2, and 3 are \$10, \$12, and \$20, respectively.

   (a) Let x_i be the production quantity of product i in a day, $i = 1, 2, 3$. Formulate an LP that helps the company finds a production plan to maximize its daily total sales revenue. As always, please allow the production quantities to be fractional. Let's call this LP the primal LP.

   solution:

   $$\begin{aligned} \max \quad & 10x_1 + 12x_2 + 20x_3 \\ \text{.s.t.} \quad & 2x_1 + 3x_2 + 4x_3 \leq 500 \\ & 30x_1 + 20x_2 + 40x_3 \leq 4800 \\ & x_1 \leq 100 \\ & x_2 \leq 80 \\ & x_3 \leq 50 \\ & x_i \geqslant 0 \quad \forall i = 1, 2, 3 \end{aligned}$$

   (b) Solve the primal LP with any way you like. Write down the optimal plan and its objective value. Indicate the bottleneck of this plan, i.e., the reasons that prevent you from producing more products and earning more money. Do not submit your solution process (e.g., do not write down your simplex iterations, do not copy and paste your Python program).

   solution:

   The optimal plan is to produce 48 product 1, 68 product 2, and 50 product 3. Its objective value is 2296. The bottleneck is

   (c) Continue from Part (a) to find the dual LP of the primal LP. Then continue from Part (b) to find the dual optimal solution without solving the dual LP. Prove that your dual solution is indeed optimal.
   Note. "Solving" an LP means to find its optimal solution from scratch (e.g., using the graphical approach or simplex method). Proving one solution is optimal is not "solving" an LP and is thus allowed.

   solution:

$$
\begin{aligned}
\min \quad & 500y_1 + 4800y_2 + 100y_3 + 80y_4 + 50y_5 \\
\text{s.t.} \quad & 2y_1 + 30y_2 + y_3 \geq 10 \\
& 3y_1 + 20y_2 + y_4 \geq 12 \\
& 4y_1 + 40y_2 + y_5 \geq 20 \\
& y_i \geqslant 0 \quad \forall i = 1, ..., 5
\end{aligned}
$$

Let s_i ($\forall i = 1, ..., 5$) be the slack variables of the relaxation of the primal LP and v_i ($\forall j = 1, 2, 3$) be the slack variables of the relaxation of the dual LP. From Part (a), we get $(x_1, x_2, x_3, s_1, s_2, s_3, s_4, s_5) = (48, 68, 50, 0, 0, 52, 12, 0)$. Hence, we know that $v_1, v_2, v_3, y_3, y_4 = 0$. Then we have

$$
\begin{aligned}
& 2y_1 + 30y_2 = 10 \\
& 3y_1 + 20y_2 = 2 \\
& 4y_1 + 40y_2 + y_5 = 20
\end{aligned}
$$

By solving these three equations, the dual optimal solution is $(y_1, y_2, y_3, y_4, y_5) = (3.2, 0.12, 0, 0, 2.4)$. Put it into the dual objective function, we get the objective value 2296 equals to the objective value of the primal LP. Hence, the dual solution is indeed optimal.

(d) Find the shadow prices for the primal constraints without solving the primal or dual LP.

solution:

Phase-I standard form LP:

$$
\begin{aligned}
\min \quad & x_6 + x_7 \\
\text{s.t.} \quad & x_1 + x_2 - x_3 + x_6 = 4 \\
& x_1 + 2x_2 + x_4 = 9 \\
& x_2 - x_5 + x_7 = 3 \\
& x_i \geqslant 0 \quad \forall i = 1, ..., 7
\end{aligned}
$$

0	0	0	0	0	-1	-1	0
1	1	-1	0	0	1	0	$x_6 = 4$
1	2	0	1	0	0	0	$x_4 = 9$
0	1	0	0	-1	0	1	$x_7 = 3$

adjust $\longrightarrow$

1	2	-1	0	-1	0	0	7
[1]	1	-1	0	0	1	0	$x_6 = 4$
1	2	0	1	0	0	0	$x_4 = 9$
0	1	0	0	-1	0	1	$x_7 = 3$

$\longrightarrow$

0	1	0	0	-1	0	3
1	1	-1	0	0	0	$x_1 = 4$
0	1	1	1	0	0	$x_4 = 5$
0	[1]	0	0	-1	1	$x_7 = 3$

$\longrightarrow$

0	0	0	0	0	0
1	0	-1	0	1	$x_1 = 1$
0	0	1	1	1	$x_4 = 2$
0	1	0	0	-1	$x_2 = 3$

Phase-II iteration:

-1	-3	0	0	0	0
1	0	-1	0	1	$x_1 = 1$
0	0	1	1	1	$x_4 = 2$
0	1	0	0	-1	$x_2 = 3$

adjust $\longrightarrow$

0	0	-1	0	-2	10
1	0	-1	0	1	$x_1 = 1$
0	0	1	1	1	$x_4 = 2$
0	1	0	0	-1	$x_2 = 3$

From iteration, we have optimal solution $(x_1, x_2) = (1, 3)$ with objective value = 10. There is no iteration that has no improvement.

(e) The company is considering investing something to expand either the production facility or the market. They may purchase some more raw materials at the cost of \$2 per unit, and they may purchase some more machine hours (by outsourcing the production task to

some other manufacturers) at the cost of $10 per hour. Moreover, they may advertise their products and expand the daily demands of products 1, 2, and 3 by one unit at the cost of $5, $10, and $15 per unit. Without doing any calculation, use your answers from Parts (a) to (d) to determine whether the company should consider any of the five above options (two regarding the production facility and three regarding the market). If so, list all options that should be considered. Justify your answers in your own words.

2. Continue from Problem 1 but ignore the expansion possibilities in Problem 1e. Do the following two subproblems independently. The two notes that come after the two subproblems apply to both subproblems.

   (a) The company just invented a new kind of product, product 4. Producing one unit of product 4 requires 5 units of raw materials and 60 minutes of machine processing. Each unit of product 4 may be sold at $30, and there is no limit on the number of product 4 that can be sold. Let x_4 be the daily production quantity of product 4. Formulate an LP that helps the company find the production plan that maximizes its daily total sales revenue. Then utilize your answer in Problem 1b to determine whether this company should consider producing product 4. If no, prove that producing product 4 decreases the company's revenue; if yes, do one simplex iteration with the smallest index rule to search for a new optimal solution and write down a better solution you find after one iteration.

      solution:

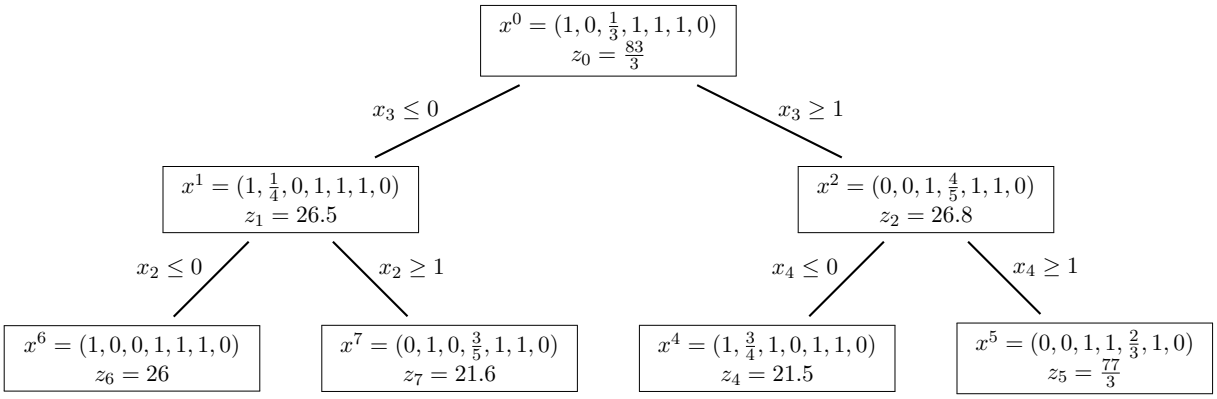
      Linear relaxation:

      $$\begin{aligned} \max \quad & 2x_1 + 2x_2 + 5x_3 + 11x_4 + 10x_5 + 3x_6 - 6x_7 \\ \text{s.t.} \quad & x_1 + 4x_2 + 3x_3 + 5x_4 + 3x_5 - 4x_6 + 2x_7 \leq 6 \\ & x_i \in [0,1] \quad \forall i = 1, ..., 7 \end{aligned}$$

      From x_1 to x_7, the value-weight ratios are 2, $\frac{1}{2}$, $\frac{5}{3}$, $\frac{11}{5}$, $\frac{10}{3}$, $\frac{-3}{4}$, -3. We can see that both the value-weight ratio of x_6 and x_7 are negative. However, from the integer program, it shows that the value of x_6 is positive while its weight is negative. Since it will increase the capacity, we should choose it first. But for x_7, the value is negative while the weight is positive. Hence, we shouldn't choose it in any circumstance. Now, we know the order of choosing variables is $x_6 \to x_5 \to x_4 \to x_1 \to x_3 \to x_2$. Hence, by the order, we get the solution $x^{ALG} = (1, 0, \frac{1}{3}, 1, 1, 1, 0)$ with objective value $= \frac{83}{3}$.

   (b) In the past, there is pretty much unlimited supply of labors, but this is no more true today. The company now faces a new constraint regarding labor hours. In total there are 200 labor hours (i.e., 12000 minutes). Producing one unit of products 1, 2, and 3 requires 90, 80, and 200 minutes of labor hours. The company is wondering whether the original optimal production plan needs to be adjusted.
   Formulate an LP that helps the company find the production plan that maximizes its daily total sales revenue. Then utilize your answer in Problem 1b to determine whether this company should change its production plan. If no, prove it; if yes, do one simplex iteration with the smallest index rule to search for a new optimal solution and write down a better solution you find after one iteration.
   Note 1. To utilize the optimal solution you found in Problem 1b, please note that you do not need the optimal tableau. Instead, all you need is the optimal basis and the matrix way. This is also true if you need to do one more simplex iteration.
   Note 2. It is possible that the process of reaching a new optimal solution requires more than one iteration. To save you some time, please do just one iteration even if that does not give you an optimal solution. Nevertheless, you will get full points only if the way you do one simplex iteration is correct.

      solution:



From the branch-and-bound tree above, we can find the optimal solution for the original IP is $x^* = (1, 0, 0, 1, 1, 1, 0)$ with objective value $z^* = 26$.

3. For each of the following program, write down the gradient and Hessian of the objective function. Then determine whether it is a convex program. Do not attempt to reformulate or solve the program.

   (a) Minimize x^2 over $x \in \mathbb{R}^2$.

   solution:

   Let $J = 1, ..., n$ be the set of candidate locations and $I = 1, ..., m$ be the set of towns. Let $x_j = 1$ if a landfill is built at location $j \in J$ or 0 otherwise, and y_i be the distance between town $i \in I$ and its closest landfill. The formulation is

$$\begin{aligned} \max \quad & \sum_{i \in I} y_i h_i \\ \text{s.t.} \quad & \sum_{j \in J} x_j \geqslant p \\ & y_i \leqslant x_j d_{ij} + M_i(1 - x_j) \quad \forall i \in I, j \in J \\ & x_j \in \{0, 1\} \quad \forall j \in J \end{aligned}$$

   where M_i is a very large number. Here we set $M_i = \max_{j \in J}\{d_{ij}\}$.

   (b) Consider the two instances contained in the file "OR112-2 hw02 data.xlsx" (if you do not use Microsoft Excel, you may use Google Spreadsheet to open the file). In each sheet, which contains an instance, parameter symbols are in blue cells, indices are in orange cells, and parameter values are in green cells. For example, in instance 2, $m = 20, n = 10, p = 5, h_2 = 28, and\ d_{12,5} = 164$.

   solution:

```python
import pandas as pd
from gurobipy import *

# Instance 1
instance1 = pd.read_excel('OR112-2_hw02_data.xlsx','Problem 3 Instance 1')
towns = range(10)
locations = range(instance1.iloc[0, 1])
landfill_min = instance1.iloc[1, 1]
human = instance1.iloc[4:14, 1]
distances = instance1.iloc[4:14, 4:9]
h = human.values
d = distances.values

eg1 = Model("eg1")

x = []
```

```
for j in locations:
    x.append(eg1.addVar(lb = 0, vtype = GRB.BINARY, name = "x" + str(j+1)))

y = []
for i in towns:
    y.append(eg1.addVar(lb = 0, vtype = GRB.INTEGER, name="y" + str(i+1)))

M = [max(d[i][j] for j in range(len(locations))) for i in towns]

# setting the objective function
eg1.setObjective(quicksum(h[i]*y[i] for i in towns), GRB.MAXIMIZE)

# add constraints and name them
eg1.addConstr(quicksum(x[j] for j in locations) >= landfill_min, "
    demand_fulfillment1")
eg1.addConstrs((y[i] <= x[j]*d[i][j] + M[i]*(1-x[j]) for i in towns for j in
     locations), "min_distance")

eg1.optimize()

# Instance 2
instance2 = pd.read_excel('OR112-2_hw02_data.xlsx','Problem 3 Instance 2')
towns = range(20)
locations = range(instance2.iloc[0, 1])
landfill_min = instance2.iloc[1, 1]
human = instance2.iloc[4:24, 1]
distances = instance2.iloc[4:24, 4:14]
h = human.values
d = distances.values

eg1.optimize()
```

Instance 1:
optimal solution $(x_1, x_2, x_3, x_4, x_5) = (1, 1, 1, 0, 0)$
objective value $z^* = 39027$
Instance 2:
optimal solution $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) = (1, 0, 1, 0, 1, 0, 1, 1, 0, 0)$
objective value $z^* = 68665$

4. Continue from Problem 3.

(a) Consider the following greedy algorithm designed for the landfill location problem. The algorithm runs in $n - p$ iterations. Initially, the algorithm starts with a feasible solution that builds a landfill in each candidate locations. In each iteration, it examines all landfills to see if one and exactly one landfill can be removed, removing which one results in the maximum objective value in that iteration (and if there is a tie, it chooses the landfill with the smallest index). It keeps removing landfills until only p landfills remain.

solution:

```
import pandas as pd
from gurobipy import *

# Instance 1
instance1 = pd.read_excel('OR112-2_hw02_data.xlsx','Problem 3 Instance 1')
towns = range(10)
locations = range(instance1.iloc[0, 1])
landfill_min = instance1.iloc[1, 1]
human = instance1.iloc[4:14, 1]
distances = instance1.iloc[4:14, 4:9]
h = human.values
d = distances.values

x = [1] * len(locations)
```

```
15
16 max_obj = 0
17 M = [max(d[i][j] for j in range(len(locations))) for i in towns]
18 x_now = x[:]
19 for i in range(len(locations) - landfill_min):
20     temp_max = 0
21     x_new = x[:]
22     for j in range(len(locations)):  # Iterate over all landfills
23         x_temp = x_new[:]  # Make a copy of x for each iteration
24         x_temp[j] = 0  # Remove the j-th landfill temporarily
25         y = M[:]
26         for k in range(len(towns)):
27             for n in range(len(locations)):
28                 y[k] = min(y[k], x_temp[n]*d[k][n] + M[k]*(1-x_temp[n]))
29
30         sol = sum(h[m] * y[m] for m in range(len(towns)))
31
32         if sol > temp_max:
33             temp_max = sol
34             x_now = x_temp[:]
35     max_obj = max(max_obj, temp_max)
36     x = x_now[:]
37
38 for j in range(len(locations)):
39     print('x' + str(j + 1), '=', x[j])
40
41 print('objective value =', max_obj)
42
43 # Instance 2
44 instance2 = pd.read_excel('OR112-2_hw02_data.xlsx','Problem 3 Instance 2')
45 towns = range(20)
46 locations = range(instance2.iloc[0, 1])
47 landfill_min = instance2.iloc[1, 1]
48 human = instance2.iloc[4:24, 1]
49 distances = instance2.iloc[4:24, 4:14]
50 h = human.values
51 d = distances.values
52
53 x = [1] * len(locations)
54 max_obj = 0
55 M = [max(d[i][j] for j in range(len(locations))) for i in towns]
56 x_now = x[:]
57 for i in range(len(locations) - landfill_min):
58     temp_max = 0
59     x_new = x[:]
60     for j in range(len(locations)):  # Iterate over all landfills
61         x_temp = x_new[:]  # Make a copy of x for each iteration
62         x_temp[j] = 0  # Remove the j-th landfill temporarily
63         y = M[:]
64         for k in range(len(towns)):
65             for n in range(len(locations)):
66                 y[k] = min(y[k], x_temp[n]*d[k][n] + M[k]*(1-x_temp[n]))
67
68         sol = sum(h[m] * y[m] for m in range(len(towns)))
69
70         if sol > temp_max:
71             temp_max = sol
72             x_now = x_temp[:]
73     max_obj = max(max_obj, temp_max)
74     x = x_now[:]
75
76 for j in range(len(locations)):
77     print('x' + str(j + 1), '=', x[j])
78
79 print('objective value =', max_obj)
```

Instance 1:
optimal solution $(x_1, x_2, x_3, x_4, x_5) = (1, 1, 1, 0, 0)$
objective value $z^* = 39027$
Instance 2:
optimal solution $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) = (0, 1, 0, 1, 0, 1, 0, 0, 1, 1)$
objective value $z^* = 65364$

(b) Consider another heuristic algorithm designed for the landfill location problem based on linear relaxation. The idea is simple. Given an instance, first we solve its linear relaxation to obtain a probably fractional LP-optimal solution x^{LP} . We then pick the largest p values and set the corresponding x_j^{LP} to 1 (if there is a tie, pick those with smaller indices); the remaining $n - p$ variables are set to 0.

solution:

```python
import pandas as pd
from gurobipy import *

# Instance 1
instance1 = pd.read_excel('OR112-2_hw02_data.xlsx','Problem 3 Instance 1')
towns = range(10)
locations = range(instance1.iloc[0, 1])
landfill_min = instance1.iloc[1, 1]
human = instance1.iloc[4:14, 1]
distances = instance1.iloc[4:14, 4:9]
h = human.values
d = distances.values

eg1 = Model("eg1")

x = []
for j in locations:
    x.append(eg1.addVar(lb = 0, vtype = GRB.CONTINUOUS, name = "x" + str(j+1)))

y = []
for i in towns:
    y.append(eg1.addVar(lb = 0, vtype = GRB.CONTINUOUS, name="y" + str(i+1)))

M = [max(d[i][j] for j in range(len(locations))) for i in towns]

# setting the objective function
eg1.setObjective(quicksum(h[i]*y[i] for i in towns), GRB.MAXIMIZE)

# add constraints and name them
eg1.addConstr(quicksum(x[j] for j in locations) >= landfill_min, "demand_fulfillment1")
eg1.addConstrs((y[i] <= x[j]*d[i][j] + M[i]*(1-x[j]) for i in towns for j in locations), "min_distance")
eg1.addConstrs((x[j] >= 0 for j in locations), "x_value1")
eg1.addConstrs((x[j] <= 1 for j in locations), "x_value2")

# Solve the linear relaxation
eg1.optimize()

# Get the solution
if eg1.status == GRB.OPTIMAL:
    xLP = [var.x for var in x]
    # Determine the number of variables to set to 1
    p = landfill_min  # determine the value of p
    # Select the top p variables with the largest values and set them to 1
    top_indices = sorted(range(len(xLP)), key=lambda i: xLP[i], reverse=True)[:p]
```

```
    for j in range(len(x)):
        if j in top_indices:
            x[j].lb = 1.0
            x[j].ub = 1.0
        else:
            x[j].lb = 0.0
            x[j].ub = 0.0
    # Update the model
    eg1.update()
    # Resolve the model
    eg1.optimize()
    # Print or use the solution as needed
    if eg1.status == GRB.OPTIMAL:
        # Print or use the solution
        pass  # Placeholder, you need to add code here
else:
    print("No solution found for the linear relaxation.")

# Instance 2
instance2 = pd.read_excel('OR112-2_hw02_data.xlsx','Problem 3 Instance 2')
towns = range(20)
locations = range(instance2.iloc[0, 1])
landfill_min = instance2.iloc[1, 1]
human = instance2.iloc[4:24, 1]
distances = instance2.iloc[4:24, 4:14]
h = human.values
d = distances.values

# Solve the linear relaxation
eg1.optimize()

# Get the solution
if eg1.status == GRB.OPTIMAL:
    xLP = [var.x for var in x]
    # Determine the number of variables to set to 1
    p = landfill_min  # determine the value of p
    # Select the top p variables with the largest values and set them to 1
    top_indices = sorted(range(len(xLP)), key=lambda i: xLP[i], reverse=True
    )[:p]
    for j in range(len(x)):
        if j in top_indices:
            x[j].lb = 1.0
            x[j].ub = 1.0
        else:
            x[j].lb = 0.0
            x[j].ub = 0.0
    # Update the model
    eg1.update()
    # Resolve the model
    eg1.optimize()
    # Print or use the solution as needed
    if eg1.status == GRB.OPTIMAL:
        # Print or use the solution
        pass  # Placeholder, you need to add code here
else:
    print("No solution found for the linear relaxation.")
```

Instance 1:
optimal solution $(x_1, x_2, x_3, x_4, x_5) = (1, 1, 1, 0, 0)$
objective value $z^* = 39027$
Instance 2:
optimal solution $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) = (1, 0, 0, 0, 1, 1, 0, 0, 1, 1)$
objective value $z^* = 63044$

(c) Let z_k^G be the objective value of the solution found by the greedy algorithm in Part (a)

for instance $k \in \{1, 2\}$. Similarly, let z_k^R be that by the heuristic algorithm in Part (b) for instance $k \in \{1, 2\}$. For each of the two instances, report $z_k^{IP}, z_k^{LP}, z_k^G, z_k^R$, and the four percentage optimality gaps (each algorithm has two optimality gaps, one uses z_k^{IP} and one uses z_k^{LP}). In average, which algorithm performs better in these two instances?

solution:

Instance 1:
$z_1^{IP} = 39027,\ z_1^{LP} = 53434.9,\ z_1^G = 39027,\ z_1^R = 39027$
z_1^G optimality gap$(z_1^{IP}) = 0\%$
z_1^G optimality gap$(z_1^{LP}) = 26.96\%$
z_1^R optimality gap$(z_1^{IP}) = 0\%$
z_1^R optimality gap$(z_1^{LP}) = 26.96\%$

Instance 2:
$z_2^{IP} = 68665,\ z_2^{LP} = 117810.4,\ z_2^G = 65364,\ z_2^R = 63044$
z_2^G optimality gap$(z_2^{IP}) = 4.81\%$
z_2^G optimality gap$(z_2^{LP}) = 44.52\%$
z_2^R optimality gap$(z_2^{IP}) = 8.19\%$
z_2^R optimality gap$(z_2^{LP}) = 46.49\%$

From the result above, we can see that there is no difference between two algorithm in instance 1, but it is clear that the greedy algorithm performs better in instance 2. In average, the greedy algorithm performs better in these two instances.

(d) Comment on the time complexity (with the big-O notation) and performance of the two heuristic algorithms. If you need to solve the landfill location problem with $m = 500, n = 100,$ and $p = 20$, which algorithm do you prefer? Why?

solution:

The greedy algorithm:
Time complexity is $O(n^3 m)$, where n is the number of potential locations and m is the number of towns.
The heuristic algorithm in part(b):
Time complexity is $O((nm)^3)$, where n is the number of potential locations.
It shows that the time complexity of the greedy algorithm in part(b) is better, and from part(c), we know that the greedy algorithm also perform better. Hence, if I need to solve the landfill location problem with $m = 500, n = 100,$ and $p = 20$, I prefer the greedy algorithm.

5. Consider the following nonlinear program

$$\min \quad 3x_1^2 + 2x_2^2 + 4x_1x_2 + 6e^{x_1} + x_2.$$

Later when needed, use numerical solutions rather than analytical solutions. For example, when solving $-x = e^x$, using any calculator or software to find $x = -0.567$ as a numerical solution is good enough. There is no need to analytically solve $-x = e^x$.

(a) Start from $(x_1, x_2) = (0, 0)$ to run one iteration of the gradient descent method to solve this instance. In this iteration, move to the global minimum along the direction you choose. Write down the detailed process of the iteration.

solution:

Let $f(x) = 3x_1^2 + 2x_2^2 + 4x_1x_2 + 6e^{x_1} + x_2$

$$\nabla f(x) = \begin{bmatrix} 6x_1 + 4x_2 + 6e^{x_1} \\ 4x_1 + 4x_2 + 1 \end{bmatrix}$$

$x^0 = (x_1, x_2) = (0, 0)$
$\nabla f(x^0) = (6, 1)$
$a_0 = \text{argmin}_{a \geq 0}\ f((0,0) - a(6,1)) = 134a^2 + 6e^{-6a} - a$
$a_0 = 0.08$
$x^1 = (0,0) - 0.08(6,1) = (-0.48, -0.08)$

(b) Start from $(x_1, x_2) = (0, 0)$ to run one iteration of the Newton's method to solve this instance. Write down the detailed process of the iteration.

solution:

$$\nabla^2 f(x) = \begin{bmatrix} 6 + 6e^{x_1} & 4 \\ 4 & 4 \end{bmatrix}$$

$$\begin{aligned} x^1 &= x^0 - [\nabla^2 f(x^0)]^{-1} \nabla f(x^0) \\ &= (0,0) - (\frac{5}{8}, \frac{-3}{8}) \\ &= (\frac{-5}{8}, \frac{3}{8}) \end{aligned}$$