OR HW2

B09303041 經濟三歐陽萱

April 15, 2023

Problems 1

(a) Find the standard form of this LP.

Standard form:

$$\begin{aligned} \text{min} \quad & x_1 + 2x_2 + 3x_3 \\ \text{s.t.} \quad & x_1 - x_2 + 2x3 + x_4 = 2 \\ & 2x_1 - x_2 + 2x_3 = 6 \\ & 3x_1 + 2x_3 - x_5 = 7 \\ & x_i \geq 0 \quad \forall i = 1, ..., 5 \end{aligned}$$

(b) List all the basic solutions and basic feasible solutions for the standard form of this LP.

$m = 3, n = 5, \binom{5}{3} = 10$: 最多有 10 個 basic solutions

B	x_1	x_2	x_3	x_4	x_5
1. (x_1, x_2, x_3)				0	0
2. (x_1, x_2, x_3)	$\frac{7}{3}$	$\frac{-4}{3}$		$\frac{-5}{3}$	0
3. (x_1, x_2, x_3)	1	0	2	-3	0
4. (x_1, x_2, x_3)	0	1	$\frac{7}{2}$	-4	0
5. (x_1, x_2, x_3)	4	2	0	0	5
6. (x_1, x_2, x_3)	4	0	-1	0	3
7. (x_1, x_2, x_3)	0			0	
8. (x_1, x_2, x_3)	3	0	0	-1	2
9. (x_1, x_2, x_3)	0	-6	0	-4	7
10.(x_1, x_2, x_3)	0	0	3	-4	-1

basic solution: 2, 3, 4, 5, 6, 8, 9, 10
basic feasible solution: 5

(c) Use the simplex method and the smallest index rule to solve the LP.

phase 1:

$$\begin{aligned} \min \quad & x_6 + x_7 \\ \text{s.t.} \quad & x_1 - x_2 + 2x3 + x_4 = 2 \\ & 2x_1 - x_2 + 2x_3 + x_6 = 6 \\ & 3x_1 + 2x_3 - x_5 + x_7 = 7 \\ & x_i \geq 0 \quad \forall i = 1, ..., 7 \end{aligned}$$

x_1	x_2	x_3	x_4	x_5	x_6	x_7	z
0	0	0	0	0	-1	-1	0
1	-1	2	1	0	0	0	$x_4 = 2$
2	-1	2	0	0	1	0	$x_6 = 6$
3	0	2	0	-1	0	1	$x_7 = 7$

$\xrightarrow{\text{adjust}}$

x_1	x_2	x_3	x_4	x_5	x_6	x_7	z
5	-1	4	0	-1	0	0	13
1	-1	2	1	0	0	0	$x_4 = 2$
2	-1	2	0	0	1	0	$x_6 = 6$
3	0	2	0	-1	0	1	$x_7 = 7$

$\xrightarrow[x_1 \ enter]{x_4 \ leave}$

x_1	x_2	x_3	x_4	x_5	x_6	x_7	z
0	4	-6	-5	-1	0	0	3
1	-1	2	1	0	0	0	$x_1 = 2$
0	1	-2	-2	0	1	0	$x_6 = 2$
0	3	-4	-3	-1	0	1	$x_7 = 1$

$\xrightarrow[x_2 \ enter]{x_7 \ leave}$

x_1	x_2	x_3	x_4	x_5	x_6	x_7	z
0	4	-6	-5	-1	0	0	3
1	-1	2	1	0	0	0	$x_1 = 2$
0	1	-2	-2	0	1	0	$x_6 = 2$
0	1	$\frac{-4}{3}$	-1	$\frac{-1}{3}$	0	$\frac{1}{3}$	$x_2 = \frac{1}{3}$

$\xrightarrow[x_5 \ enter]{x_6 \ leave}$

x_1	x_2	x_3	x_4	x_5	x_6	x_7	z
0	0	$\frac{-2}{3}$	-1	$\frac{1}{3}$	0	$\frac{-4}{3}$	$\frac{5}{3}$
1	0	$\frac{2}{3}$	0	$\frac{-1}{3}$	0	$\frac{1}{3}$	$x_1 = \frac{7}{3}$
0	0	$\frac{-2}{3}$	-1	$\frac{1}{3}$	1	$\frac{-1}{3}$	$x_5 = \frac{5}{3}$
0	1	$\frac{-4}{3}$	-1	$\frac{-1}{3}$	0	$\frac{1}{3}$	$x_2 = \frac{1}{3}$

$\xrightarrow{adjust}$

x_1	x_2	x_3	x_4	x_5	x_6	x_7	z
0	0	$\frac{-2}{3}$	-1	$\frac{1}{3}$	0	$\frac{-4}{3}$	$\frac{5}{3}$
1	0	$\frac{2}{3}$	0	$\frac{-1}{3}$	0	$\frac{1}{3}$	$x_1 = \frac{7}{3}$
0	0	-2	-3	1	3	-1	$x_5 = 5$
0	1	$\frac{-4}{3}$	-1	$\frac{-1}{3}$	0	$\frac{1}{3}$	$x_2 = \frac{1}{3}$

$\xrightarrow{adjust}$

x_1	x_2	x_3	x_4	x_5	x_6	x_7	z
0	0	0	0	0	-1	-1	0
1	0	0	-1	0	1	0	$x_1 = 4$
0	0	-2	-3	1	3	-1	$x_5 = 5$
0	1	-2	-2	0	1	0	$x_2 = 2$

phase 2:

$$\begin{array}{ccccc||c}
x_1 & x_2 & x_3 & x_4 & x_5 & z \\
\hline
-1 & -2 & -3 & 0 & 0 & 0 \\
\hline
1 & 0 & 0 & -1 & 0 & x_1 = 4 \\
0 & 0 & -2 & -3 & 1 & x_5 = 5 \\
0 & 1 & -2 & -2 & 0 & x_2 = 2
\end{array}
\xrightarrow{adjust}
\begin{array}{ccccc||c}
x_1 & x_2 & x_3 & x_4 & x_5 & z \\
\hline
0 & 0 & -7 & -5 & 0 & 8 \\
\hline
1 & 0 & 0 & -1 & 0 & x_1 = 4 \\
0 & 0 & -2 & -3 & 1 & x_5 = 5 \\
0 & 1 & -2 & -2 & 0 & x_2 = 2
\end{array}$$

optimal solution: $(x_1, x_2 x_3, x_4, x_5) = (4, 2, 0, 0, 5)$
$z^* = 8$

Problems 2

(a) Solve the linear relaxation of this integer program.

$$\begin{aligned}
\max \quad & x_1 + x_2 \\
\text{s.t.} \quad & 2x_1 + 4x_2 \le 15 \\
& 6x_1 - 2x_2 \le 7 \\
& x_i \ge 0 \quad \forall i = 1, 2
\end{aligned}$$

$\Rightarrow x^{LR} = (\frac{29}{14}, \frac{19}{7})$; $\quad z^{LR} = \frac{67}{14}$

(b) Let (x_1^{LR}, x_2^{LR}) be the optimal solution obtained in Part (a). Show that rounding up and down x_1^{LR} and x_2^{LR} does not result in any feasible solution.

Four grid points around x^{LR}: $x^\alpha = (2, 2)$, $x^\beta = (2, 3)$, $x^\gamma = (3, 2)$, $x^\delta = (3, 3)$.
$x^\alpha = (2, 2)$ violates the constraint: $6x_1 - 2x_2 \le 7$
$x^\beta = (2, 3)$ violates the constraint: $2x_1 + 4x_2 \le 15$
$x^\gamma = (3, 2))$ violates the constraint: $6x_1 - 2x_2 \le 7$
$x^\delta = (3, 3)$ violates the constraint: $2x_1 + 4x_2 \le 15$
$\Rightarrow$ None of the four grid points around x^{LR} is feasible.

(c) Use the branch-and-bound algorithm to solve the original integer program.

See Figure 1.

Problems 3

In solving an integer program of a maximization problem using the branch-and-bound algorithm , why is it reasonable to branch on the node with the highest objective value?

In solving an integer program using the branch-and-bound algorithm, the goal is to find an optimal integer solution . At each node of the branch-and-bound tree, a linear programming (LP) relaxation of the integer program is solved, which provides a ~~lower bound~~ on the optimal objective value.

- Something wrong 1: upper bound.

~~If the LP relaxation at a node provides an integer solution, then this solution is optimal and the branch-and-bound algorithm terminates.~~ Otherwise, the node is branched into two subproblems, one of which restricts a variable to be less than or equal to its current ~~integer value~~, while the other restricts the variable to be greater than or equal to its current ~~integer value plus one~~.

- Something wrong 2: If the LP relaxation at a node provides an integer solution, then this solution is a "candidate solution" to the original IP.
- Something wrong 3: ... one of which restricts a variable to be less than or equal to the floor of current fractional value, while the other restricts the variable to be greater than or equal to the ceiling of current fractional value's.

When branching, it is reasonable to select the variable with the highest objective value to branch on because this is the variable that has the potential to ~~yield the highest improvement in the objective value~~.

- Something wrong 4: ...lead to an IP-feasible solution with a high objective value.

To see this, suppose that the current LP relaxation has an optimal solution where variable x_i is fractional , with value v_i. The two subproblems resulting from branching on x_i correspond to the constraints $x_i \leq floor(v_i)$ and ~~$x_i \geq ceil(v_i) + 1$~~, where $floor(v_i)$ denotes the largest integer less than or equal to v_i, and $ceil(v_i)$ denotes the smallest integer greater than or equal to v_i.

- Something wrong 5: $x_i \geq ceil(v_i)$.

Let z_L denote the lower bound on the optimal objective value obtained from the LP relaxation at the current node, and let z^* denote the optimal objective value. If we solve the LP relaxation for the subproblem with $x_i \leq floor(v_i)$, we obtain a ~~lower bound~~ z_1 on z^* , and if we solve the LP relaxation for the subproblem with ~~$x_i \geq ceil(v_i) + 1$~~ , we obtain a lower bound z_2 on z^*.

- Something wrong 6: upper bound.
- Something wrong 7: $x_i \geq ceil(v_i)$.

If we branch on a variable with a lower objective value, then the resulting ~~lower bounds~~ z_1 and z_2 may not be as tight as if we had branched on a variable with a higher objective value. Therefore, branching on the node with the highest objective value is a reasonable choice to potentially yield tighter ~~lower bounds~~ and reduce the search space for finding the optimal integer solution.

- Something wrong 8: upper bound.

Problems 4

$$
\begin{aligned}
\min \quad & w && (1)\\
\text{s.t.} \quad & \sum_{j \in J} y_{ij} = 1 \quad \forall i \in I && (2)\\
& y_{ij} \leq x_j \quad \forall i \in I, j \in J && (3)\\
& \sum_{j \in J} x_j \leq p && (4)\\
& w \geq \sum_{j \in J}^{n} d_{ij} y_{ij} \quad \forall i \in I && (5)\\
& x_j \in \{0,1\} \quad \forall j \in J && (6)\\
& y_{ij} \in \{0,1\} \quad \forall i \in I, j \in J && (7)
\end{aligned}
$$

(a) Use your own words to explain the constraints in the above formulation.

(2): 每個 town 都有一個最近的 park
(3): 要確定那個 park 有蓋
(4): 最多蓋 p 個 park
(5): 每個 town 到最近的 park 的最遠距離
(6): x 是 decision variable
(7): y 是 decision variable

(b) Please comment on Michelle's suggestion. If her arguments are correct, explain why; otherwise, explain why and correct her arguments.

For a minimization IP, linear relaxation provides an lower bound. Thus it can be faster than solving the original IP. Let x' be an optimal solutions to the partial linear relaxation of an IP. If x' is feasible to the IP, it is optimal to the IP. However, the optimal solution to the partially-relaxed program is not necessarily an optimal solution to the original integer program. If x' is not feasible to original IP, we need to use branch-and-bound algorithm.

(c) Please comment on Michael's suggestion.

Yes, it's correct. Because (2) has already assure that each town has a closest park, thus $w \geq d_{ij}y_{ij}$ is sufficient for he minimization problem.

(d) Write down the objective value of the optimal solution you obtain.

objective value: $z^* = 264$

Problems 5

Write down the time complexity measurement using the big-O notations. Briefly explain how you derive the complexity measures. Show that both heuristic algorithms are polynomial-time ones.

Let m = number of sets; n = number of elements,

The heuristic algorithms in problem 7:
Step 1. Calculate ratios for each set and select the lowest one. $\Rightarrow O(m)$
Step 2. Updates the uncovered elements. $\Rightarrow O(n)$
Iterations: In the worst case of first iteration, there are $n-1$ elements that are not covered. Therefore, the maximum iterations is n. $\Rightarrow O(nm + n^2)$
The heuristic algorithm is polynomial-time one.

The heuristic algorithms in problem 8:
Step 1. Calculate ratios for each set and select the lowest one. $\Rightarrow O(m)$
Step 2. Updates the uncovered elements. $\Rightarrow O(n)$
Iterations: In the worst case of first iteration, there are $n-1$ elements that are not covered. Therefore, the maximum iterations is n. $\Rightarrow O(nm + n^2)$
The heuristic algorithm is polynomial-time one.

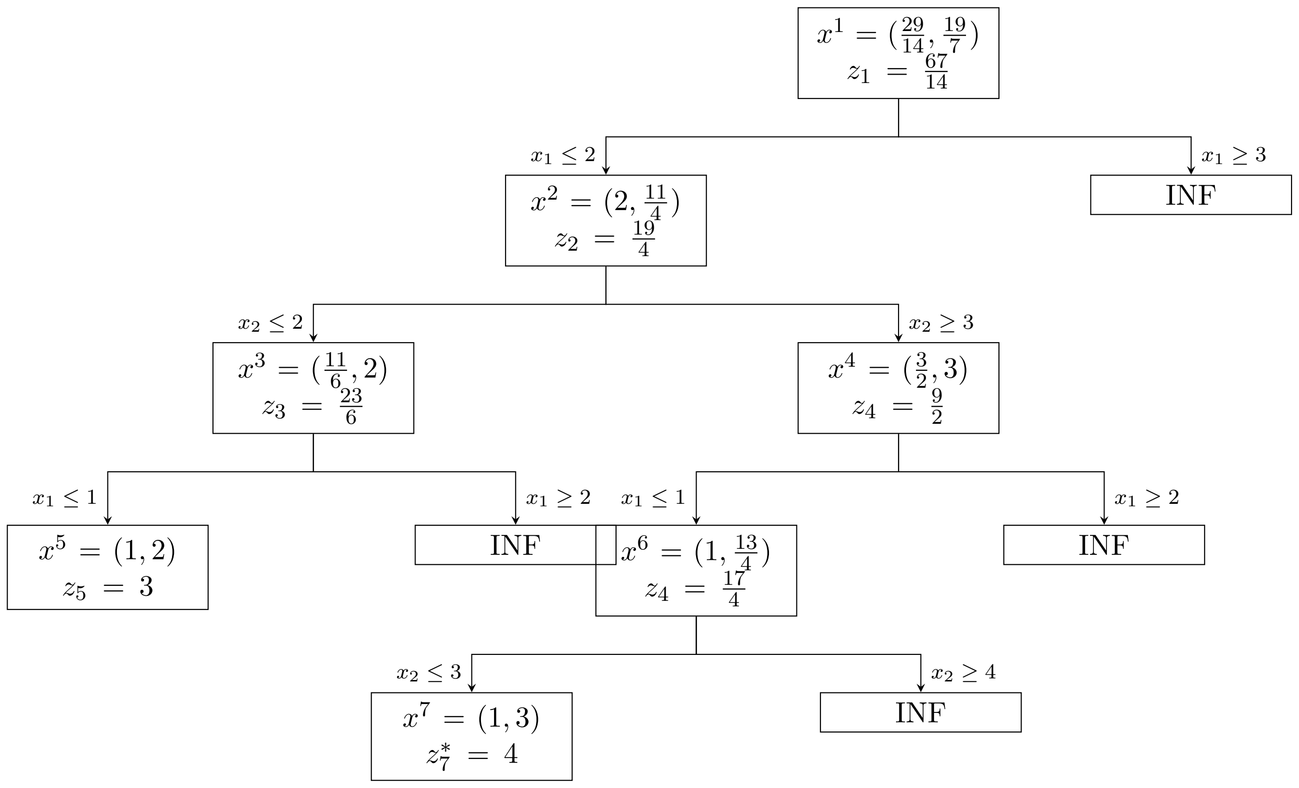

Figure 1: Branch and Bound Algorithm

OR | HW2

Status	In progress
Due date	@April 15, 2023
Type	Assignments

```
from gurobipy import *
import pandas as pd
import numpy as np
import grblogtools as glt

df = pd.read_csv('OR_hw02_data.csv', header=None)
print(df.to_string())

p = 8 # build at most p parks
n, m = df.shape
print(df.shape) # (m=60, n=20)
towns  = range(m)
potential_locations = range(n)

eg4d = Model("eg4d")

#-------- Add variables as a list ---------#
# xj = 1 if a park is built at loc j
x = []
for j in potential_locations:
    x.append(eg4d.addVar(lb=0, vtype = GRB.BINARY, name = "x" + str(j+1)))

# yij = 1 if the park in location j is the closest one for town i
y = []
for i in towns:
    y.append([])
    for j in potential_locations:
        y[i].append(eg4d.addVar(lb = 0, vtype = GRB.BINARY, name = "y" + str(i+1) + st
r(j+1)))

# dij = the distance between town i and location j
d = []
for i in towns:
    d.append([])
    for j in potential_locations:
        d[i].append(eg4d.addVar(lb = df.iloc[j, i], vtype = GRB.INTEGER, name = "d" +
 str(i+1) + str(j+1)))

# build at most p parks in potential locations.
```

```
p = eg4d.addVar(lb = 0, vtype = GRB.INTEGER, name = "p")

# w = the maximum distance for each people to move to her/his closest park.
w = eg4d.addVar(lb = 0, vtype = GRB.INTEGER, name = "max_distance")

eg4d.setObjective(w,GRB.MINIMIZE)

eg4d.addConstrs((quicksum(y[i][j] for j in potential_locations) == 1 for i in towns),
 "每個town都有一個最近的park")
eg4d.addConstrs((y[i][j] <= x[j] for i in towns for j in potential_locations), "要確定
那個park有蓋")
eg4d.addConstr((quicksum(x[j] for j in potential_locations) <= p), "最多蓋p個park")
eg4d.addConstrs((d[i][j] * y[i][j] <= w for i in towns for j in potential_locations),
 "每個town到最近的park的最遠距離")
#eg4d.addConstrs((quicksum(d[i][j] * y[i][j] for j in potential_locations) <= w for i
 in towns), "每個town到最近的park的最遠距離")

eg4d.optimize()
print("z* = ", eg4d.ObjVal)
```

```

Gurobi Optimizer version 10.0.1 build v10.0.1rc0 (win64)

CPU model: Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz, instruction set [SSE2|AVX|AVX2|A
VX512]
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads

Optimize a model with 1261 rows, 2422 columns and 3621 nonzeros
Model fingerprint: 0xfb78fc48
Model has 1200 quadratic constraints
Variable types: 0 continuous, 2422 integer (1220 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  QMatrix range    [1e+00, 1e+00]
  QLMatrix range   [1e+00, 1e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 5e+02]
  RHS range        [1e+00, 1e+00]
Presolve removed 1201 rows and 21 columns
Presolve time: 0.01s
Presolved: 3660 rows, 6001 columns, 9600 nonzeros
Presolved model has 2400 SOS constraint(s)
Variable types: 0 continuous, 6001 integer (2400 binary)
Found heuristic solution: objective 456.0000000
Found heuristic solution: objective 447.0000000
Found heuristic solution: objective 340.0000000
Found heuristic solution: objective 272.0000000

Root relaxation: objective 0.000000e+00, 2182 iterations, 0.01 seconds (0.00 work unit
s)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    0.00000    0   60  272.00000    0.00000   100%     -    0s
H    0     0                        267.0000000    0.00000   100%     -    0s
```

```
H    0     0                     265.0000000    0.00000   100%     -    0s
H    0     0                     264.0000000    0.00000   100%     -    0s

Explored 1 nodes (2182 simplex iterations) in 0.34 seconds (0.09 work units)
Thread count was 8 (of 8 available processors)

Solution count 7: 264 265 267 ... 456

Optimal solution found (tolerance 1.00e-04)
Best objective 2.640000000000e+02, best bound 2.640000000000e+02, gap 0.0000%
z* =  264.0
```