IEOR E4004 HW3 Mark Ma km4054

Question 1

(a) Sets:

1. W: set of workers, indexed by w, where $w = 1, 2 \cdots 100$.
2. D: set of departments, indexed by d (battery, body, assembly, paint and quality control).
3. S: set of shifts, indexed by s (morning, afternoon, night).
4. T: set of days. indexed by t (Mon - Sun)

Parameters

1. $a_{w,d,s,t}$: availability of worker w for department d on shift s and day t (1 if available, 0 otherwise).
2. $p_{w,d,s,t}$: preferenee score of worker w for department d on shift s. and day t. (1 to 10)
3. $e_{w,d,s,t}$: effective score of worker w for department d on shift s. and day t. (1 to 10)
4. $min\_staff_{d,s,t}$: Minimum staffing requirement for department d on shift s and day t.
5. $max\_staff_{d,s,t}$: Maximum staffing capacity for department d on shift s and day t.

Decision Variables

$x_{w,d,s,t}$: Binary variable indicating whether worker w is assigned to department d for shift s on day t. (1 if assigned, 0 otherwise)

Objectives

$$\max \sum_{w \in W} \sum_{d \in D} \sum_{s \in S} \sum_{t \in T} p_{w,d,s,t} \times e_{w,d,s,t} \times x_{w,d,s,t}$$

Constraints

1. Single shift per day: $\sum_{d \in D} \sum_{s \in S} x_{w,d,s,t} \leq 1$, $\forall w \in W$, $\forall t \in T$
2. Weekly work limit: $\sum_{d \in D} \sum_{s \in S} \sum_{t \in T} x_{w,d,s,t} \leq 5$, $\forall w \in W$
3. Availability constraint: $x_{w,d,s,t} \leq a_{w,d,s,t}$, where $a_{w,d,s,t} \in \{0, 1\}$

$\forall w \in W$, $\forall d \in D$, $\forall s \in S$, $\forall t \in T$.

4. Staffing requirements:

$$\sum_{w \in W} x_{w,d,s,t} \geq min\_staff_{d,s,t} \quad \forall d \in D,\ \forall s \in S,\ \forall t \in T$$

$$\sum_{w \in W} x_{w,d,s,t} \leq max\_staff_{d,s,t} \quad \forall d \in D,\ \forall s \in S,\ \forall t \in T$$

5. Binary decision variable: $x_{w,d,s,t} \in \{0,1\}$

$$\forall w \in W,\ \forall d \in D,\ \forall s \in S,\ \forall t \in T.$$

(b) From above, the problem can be formulated as:

$$\max \sum_{w \in W} \sum_{d \in D} \sum_{s \in S} \sum_{t \in T} P_{w,d,s,t} \times e_{w,d,s,t} \times x_{w,d,s,t}$$

$$s.t. \sum_{d \in D} \sum_{s \in S} x_{w,d,s,t} \leq 1,\ \forall w \in W,\ \forall t \in T$$

$$\sum_{d \in D} \sum_{s \in S} \sum_{t \in T} x_{w,d,s,t} \leq 5,\ \forall w \in W$$

$$x_{w,d,s,t} \leq a_{w,d,s,t},\ \forall w \in W,\ \forall d \in D,\ \forall s \in S,\ \forall t \in T.$$

$$\sum_{w \in W} x_{w,d,s,t} \geq min\_staff_{d,s,t},\ \forall d \in D,\ \forall s \in S,\ \forall t \in T$$

$$\sum_{w \in W} x_{w,d,s,t} \leq max\_staff_{d,s,t},\ \forall d \in D,\ \forall s \in S,\ \forall t \in T$$

$$x_{w,d,s,t} \in \{0,1\},\ \forall w \in W,\ \forall d \in D,\ \forall s \in S,\ \forall t \in T.$$

(C) The result is shown below, with codes attached in appendix.

```
Solution count 3: 30953 27166 12303

Optimal solution found (tolerance 1.00e-04)
Best objective 3.095300000000e+04, best bound 3.095300000000e+04, gap 0.0000%
```

	Worker_ID	Department	Shift	Day
0	1	Body	Afternoon	Sun
1	1	Assembly	Afternoon	Mon
2	1	Assembly	Afternoon	Tue
3	1	Paint	Afternoon	Wed
4	1	Quality	Morning	Thur
...	...	...	...	...
495	100	Assembly	Afternoon	Fri
496	100	Paint	Afternoon	Thur
497	100	Paint	Afternoon	Sat
498	100	Quality	Night	Wed
499	100	Quality	Night	Sun

500 rows × 4 columns

From the result, the maximum total preference-adjusted effectiveness should be 30953, and a snapshot of the optimal solution is shown on the left.

Question 2

For this question, we introduce a binary variable $y \in \{0, 1\}$

① If $y=0$, the first constraint is satisfied

② If $y=1$, the second constraint is satisfied

Then, we introduce a large positive number M.

By the big-M method, in order to satisfy at least one constraint, we have the formulation:

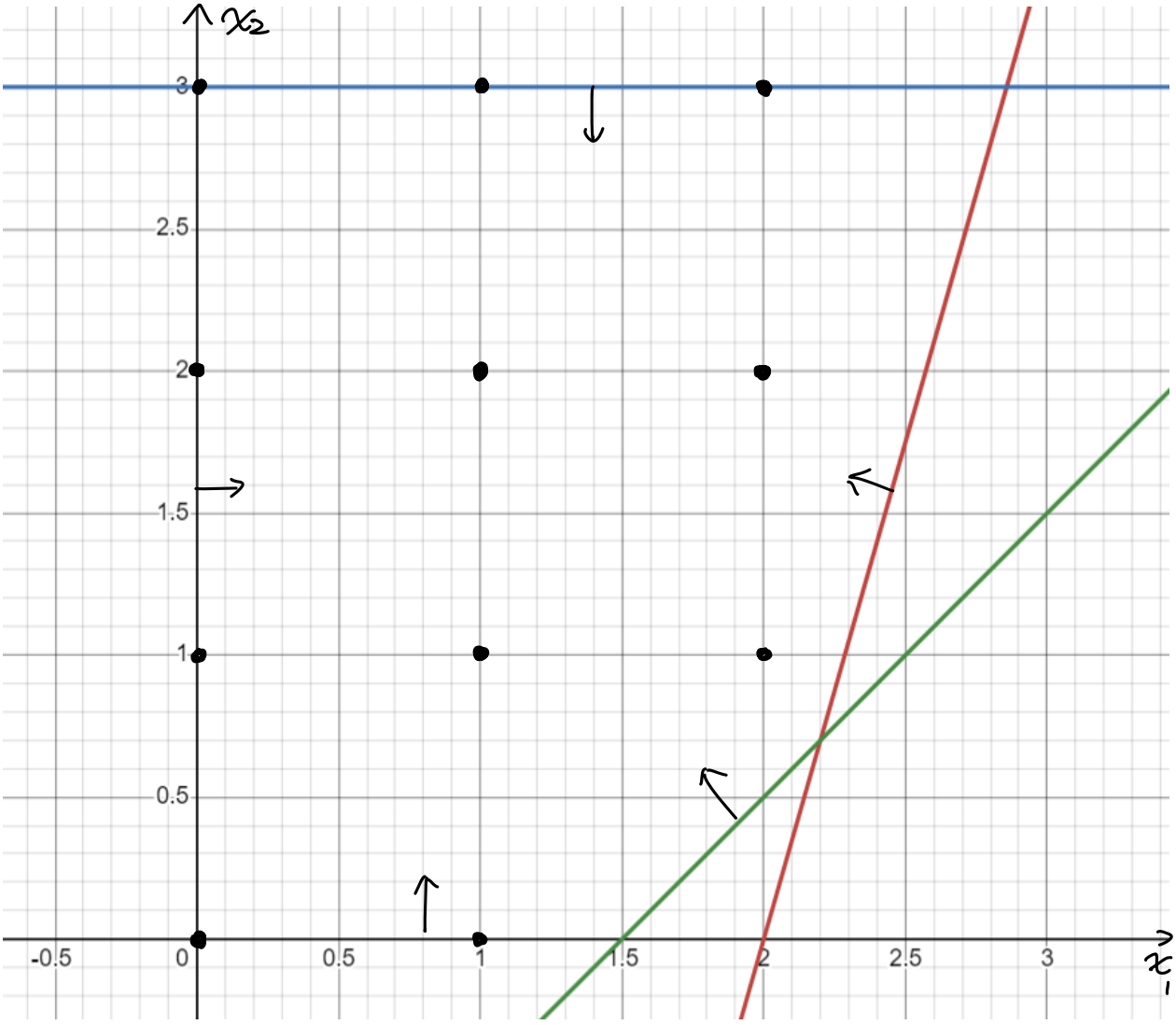
$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n \geq b_1 - M \cdot y$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n \geq b_2 - M(1-y)$$

Therefore we satisfy the requirement.

Question 3

We can draw the feasible region as there're 2 variables below:



The feasible region is the black dots noted in the graph, since it is an integer programming problem. (In total 11 dots).

Question 4

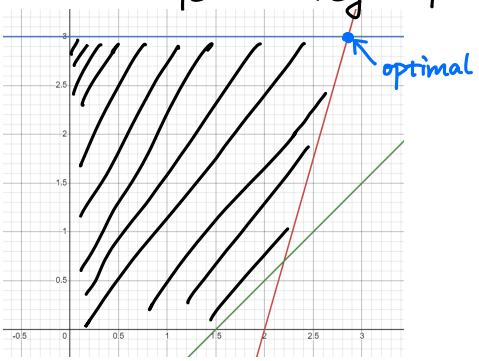
Original LP

$$\max \quad z = 4x_1 - x_2$$
$$\text{s.t.} \quad 7x_1 - x_2 \le 14$$
$$x_2 \le 3$$
$$2x_1 - 2x_2 \le 3$$
$$x_1, x_2 \ge 0$$

$\Rightarrow z = 8.429$

$(x_1, x_2) = (2.857, 3)$

feasible region for LP



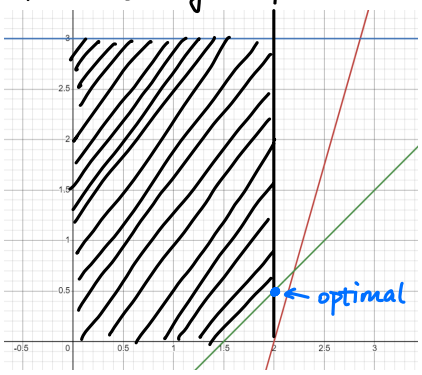
Sub-problem 1

$$\max \quad z = 4x_1 - x_2$$
$$\text{s.t.} \quad 7x_1 - x_2 \le 14$$
$$x_2 \le 3$$
$$2x_1 - 2x_2 \le 3$$
$$x_1, x_2 \ge 0$$
$$x_1 \le 2$$

$\Rightarrow z = 7.5$

$(x_1, x_2) = (2, 0.5)$

Feasible Region for sub 1



Sub-problem 2

$$\max \quad z = 4x_1 - x_2$$
$$\text{s.t.} \quad 7x_1 - x_2 \le 14$$
$$x_2 \le 3$$
$$2x_1 - 2x_2 \le 3$$
$$x_1, x_2 \ge 0$$
$$x_1 \ge 3$$

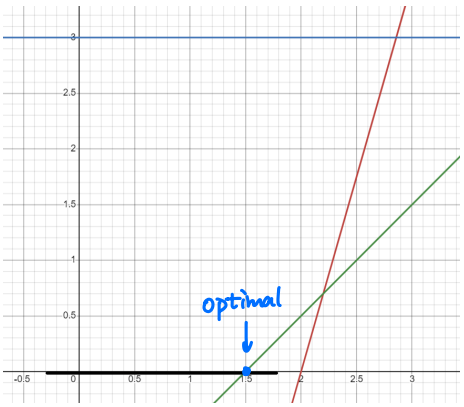
Infeasible

Sub-problem 4

$$\max \quad z = 4x_1 - x_2$$
$$\text{s.t.} \quad 7x_1 - x_2 \le 14$$
$$x_2 \le 3$$
$$2x_1 - 2x_2 \le 3$$
$$x_1, x_2 \ge 0$$
$$x_1 \le 2, \quad x_2 \le 0$$

$\Rightarrow z = 6$

$(x_1, x_2) = (1.5, 0)$



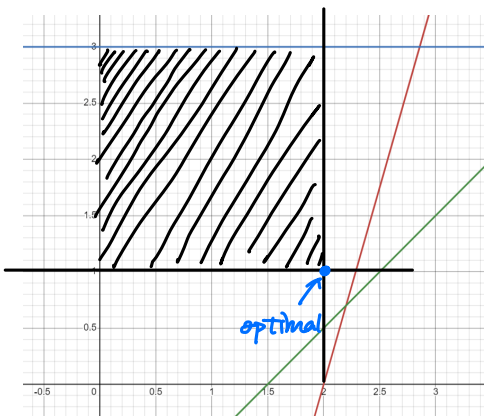
The black line is a feasible region.

Sub-problem 3

$$\max \quad z = 4x_1 - x_2$$
$$\text{s.t.} \quad 7x_1 - x_2 \le 14$$
$$x_2 \le 3$$
$$2x_1 - 2x_2 \le 3$$
$$x_1, x_2 \ge 0$$
$$x_1 \le 2, \quad x_2 \ge 1$$

$\Rightarrow z = 7$

$(x_1, x_2) = (2, 1)$



We can start the branch and bound method by relaxing the original IP to LP first.

1. We solve the LP, and get $z = 8.429$, $(x_1, x_2) = (2.857, 3)$.

we further use x_1 to branch.

2. Sub-problem 1 (add $x_1 \leq 2$)
   $z = 7.5$, $(x_1, x_2) = (2, 0.5)$.

3. Sub-problem 2 (add $x_1 \geq 3$) Infeasible.

4. Sub-problem 3, branch from sub-problem 1 + $(x_2 \geq 1)$
   $z = 7$, $(x_1, x_2) = (2, 1)$. Optimal.

5. Sub-problem 4, branch from sub-problem 1 + $(x_2 \leq 0)$.
   $z = 6$, $(x_1, x_2) = (1.5, 0)$.

From the tree, since $z_3 = 7 > 6 = z_4$, we have no need to branch sub-problem 4 since it has provided an upper bound for this branch. So the optimal value should be 7, with $(x_1, x_2) = (2, 1)$.

Appendices

Mark Ma km4054

Nov 4, 2024

1 Question 1

[2]:
```
import pandas as pd
import numpy as np
from gurobipy import Model, GRB, quicksum
```

[3]:
```
# Define Workers , Departments , Shifts , and Days
workers = [i for i in range(1, 101)]
departments = ['Battery', 'Body', 'Assembly', 'Paint', 'Quality']
shifts = ['Morning', 'Afternoon', 'Night']
days = ['Mon', 'Tue', 'Wed', 'Thur', 'Fri', 'Sat', 'Sun']

# Create Workers DataFrame
workers_df = pd.DataFrame({
'Worker_ID': np.repeat(workers , len(departments)*len(shifts)*len(days)),
'Department': np.tile(np.repeat(departments ,␣
 ↪len(shifts)*len(days)),len(workers)),
'Shift': np.tile(np.repeat(shifts , len(days)), len(workers)*len(departments)),
'Day': np.tile(days, len(workers)*len(departments)*len(shifts)),
'Availability': np.random.choice([0, 1],␣
 ↪len(workers)*len(departments)*len(shifts)*len(days)),
'Preference_Score': np.random.randint(1, 10,␣
 ↪len(workers)*len(departments)*len(shifts)*len(days)),
'Effectiveness_Score': np.random.randint(1, 10,␣
 ↪len(workers)*len(departments)*len(shifts)*len(days))
})

# Create Department DataFrame
dept_df = pd.DataFrame({
'Department': np.repeat(departments , len(shifts)*len(days)),
'Shift': np.tile(np.repeat(shifts , len(days)), len(departments)),
'Day': np.tile(days, len(departments)*len(shifts)),
'Min_Workers': np.random.randint(1, 5, len(departments)*len(shifts)*len(days)),
'Max_Workers': np.random.randint(5, 10, len(departments)*len(shifts)*len(days))
})
```

[4]:
```
# Create model
model = Model('q1')

x = model.addVars(workers, departments, shifts, days, vtype=GRB.BINARY,␣
 ↪name='x')

model.setObjective(
    quicksum(
        workers_df.loc[
            (workers_df['Worker_ID'] == w) &
            (workers_df['Department'] == d) &
            (workers_df['Shift'] == s) &
            (workers_df['Day'] == t),
            'Preference_Score'
        ].values[0] * workers_df.loc[
            (workers_df['Worker_ID'] == w) &
            (workers_df['Department'] == d) &
            (workers_df['Shift'] == s) &
            (workers_df['Day'] == t),
            'Effectiveness_Score'
        ].values[0] * x[w, d, s, t]
        for w in workers for d in departments for s in shifts for t in days
    ),
    GRB.MAXIMIZE
)

# 1. Each worker can only work one shift per day
for w in workers:
    for t in days:
        model.addConstr(
            quicksum(x[w, d, s, t] for d in departments for s in shifts) <= 1,
            name=f"one_shift_per_day_{w}_{t}"
        )

# 2. Each worker can work a maximum of 5 days per week
for w in workers:
    model.addConstr(
        quicksum(x[w, d, s, t] for d in departments for s in shifts for t in␣
 ↪days) <= 5,
        name=f"max_5_days_{w}"
    )

# 3. Workers can only be assigned to shifts they are available for
for w in workers:
    for d in departments:
        for s in shifts:
            for t in days:
```

```
                availability = workers_df.loc[
                    (workers_df['Worker_ID'] == w) &
                    (workers_df['Department'] == d) &
                    (workers_df['Shift'] == s) &
                    (workers_df['Day'] == t),
                    'Availability'
                ].values[0]
                model.addConstr(
                    x[w, d, s, t] <= availability,
                    name=f"availability_{w}_{d}_{s}_{t}"
                )

# 4. Staffing requirements for each department shift
for d in departments:
    for s in shifts:
        for t in days:
            min_workers = dept_df.loc[
                (dept_df['Department'] == d) &
                (dept_df['Shift'] == s) &
                (dept_df['Day'] == t),
                'Min_Workers'
            ].values[0]
            max_workers = dept_df.loc[
                (dept_df['Department'] == d) &
                (dept_df['Shift'] == s) &
                (dept_df['Day'] == t),
                'Max_Workers'
            ].values[0]
            model.addConstr(
                quicksum(x[w, d, s, t] for w in workers) >= min_workers,
                name=f"min_staff_{d}_{s}_{t}"
            )
            model.addConstr(
                quicksum(x[w, d, s, t] for w in workers) <= max_workers,
                name=f"max_staff_{d}_{s}_{t}"
            )

# Solve model
model.optimize()
```

```
Set parameter Username
Academic license - for non-commercial use only - expires 2025-09-09
Gurobi Optimizer version 11.0.3 build v11.0.3rc0 (win64 - Windows 10.0
(19045.2))

CPU model: Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz, instruction set
[SSE2|AVX|AVX2]
Thread count: 8 physical cores, 16 logical processors, using up to 16 threads
```

```
Optimize a model with 11510 rows, 10500 columns and 52500 nonzeros
Model fingerprint: 0x3d100e94
Variable types: 0 continuous, 10500 integer (10500 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 8e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 9e+00]
Found heuristic solution: objective 12303.000000
Presolve removed 10500 rows and 5213 columns
Presolve time: 0.03s
Presolved: 1010 rows, 5287 columns, 21148 nonzeros
Variable types: 0 continuous, 5287 integer (5287 binary)
Found heuristic solution: objective 27166.000000

Root relaxation: objective 3.095300e+04, 329 iterations, 0.00 seconds (0.00 work
units)

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

*    0     0               0    30953.000000 30953.0000  0.00%     -    0s

Explored 1 nodes (329 simplex iterations) in 0.05 seconds (0.04 work units)
Thread count was 16 (of 16 available processors)

Solution count 3: 30953 27166 12303

Optimal solution found (tolerance 1.00e-04)
Best objective 3.095300000000e+04, best bound 3.095300000000e+04, gap 0.0000%
```

[5]:
```python
# Output results
if model.status == GRB.OPTIMAL:
    print("Optimal Solution Found")
    solution = []
    for w in workers:
        for d in departments:
            for s in shifts:
                for t in days:
                    if x[w, d, s, t].x > 0.5:  # Worker is assigned
                        solution.append((w, d, s, t))
    # Convert to DataFrame for easy visualization
    solution_df = pd.DataFrame(solution, columns=["Worker_ID", "Department",␣
 ↪"Shift", "Day"])
    print(solution_df)
else:
```

```
    print("No optimal solution found.")
```

```
Optimal Solution Found
     Worker_ID Department      Shift   Day
0            1       Body  Afternoon   Sun
1            1   Assembly  Afternoon   Mon
2            1   Assembly  Afternoon   Tue
3            1      Paint  Afternoon   Wed
4            1    Quality    Morning  Thur
..         …         …        …    …
495        100   Assembly  Afternoon   Fri
496        100      Paint  Afternoon  Thur
497        100      Paint  Afternoon   Sat
498        100    Quality      Night   Wed
499        100    Quality      Night   Sun

[500 rows x 4 columns]
```

[6]:
```
solution_df
```

[6]:
```
     Worker_ID Department      Shift   Day
0            1       Body  Afternoon   Sun
1            1   Assembly  Afternoon   Mon
2            1   Assembly  Afternoon   Tue
3            1      Paint  Afternoon   Wed
4            1    Quality    Morning  Thur
..         …         …        …    …
495        100   Assembly  Afternoon   Fri
496        100      Paint  Afternoon  Thur
497        100      Paint  Afternoon   Sat
498        100    Quality      Night   Wed
499        100    Quality      Night   Sun

[500 rows x 4 columns]
```

2 Question 4

2.1 Original IP

[1]:
```
import gurobipy as gp
from gurobipy import GRB

# Initialize the model
model = gp.Model("Original IP")

# Define variables x1 and x2 as integer non-negative variables
x1 = model.addVar(vtype=GRB.INTEGER, name="x1", lb=0)
```

```python
x2 = model.addVar(vtype=GRB.INTEGER, name="x2", lb=0)

# Set objective function: Maximize z = 4*x1 - x2
model.setObjective(4 * x1 - x2, GRB.MAXIMIZE)

# Add constraints
model.addConstr(7 * x1 - 2 * x2 <= 14, "constraint_1")
model.addConstr(x2 <= 3, "constraint_2")
model.addConstr(2 * x1 - 2 * x2 <= 3, "constraint_3")

# Optimize the model
model.optimize()

# Print the results
if model.status == GRB.OPTIMAL:
    print(f"Optimal solution found:")
    print(f"x1 = {x1.X}")
    print(f"x2 = {x2.X}")
    print(f"Objective value (z) = {model.ObjVal}")
else:
    print("No optimal solution found.")
```

```
Set parameter Username
Academic license - for non-commercial use only - expires 2025-09-09
Gurobi Optimizer version 11.0.3 build v11.0.3rc0 (win64 - Windows 10.0
(19045.2))

CPU model: Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz, instruction set
[SSE2|AVX|AVX2]
Thread count: 8 physical cores, 16 logical processors, using up to 16 threads

Optimize a model with 3 rows, 2 columns and 5 nonzeros
Model fingerprint: 0xbc615aeb
Variable types: 0 continuous, 2 integer (0 binary)
Coefficient statistics:
  Matrix range     [1e+00, 7e+00]
  Objective range  [1e+00, 4e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [3e+00, 1e+01]
Found heuristic solution: objective 4.0000000
Presolve removed 3 rows and 2 columns
Presolve time: 0.01s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.02 seconds (0.00 work units)
Thread count was 1 (of 16 available processors)

Solution count 2: 7 4
```

```
Optimal solution found (tolerance 1.00e-04)
Best objective 7.000000000000e+00, best bound 7.000000000000e+00, gap 0.0000%
Optimal solution found:
x1 = 2.0
x2 = 1.0
Objective value (z) = 7.0
```

2.2 Original LP

[2]:
```python
# Initialize the model
model = gp.Model("Original LP")

# Define variables x1 and x2 as integer non-negative variables
x1 = model.addVar(vtype=GRB.CONTINUOUS, name="x1", lb=0)
x2 = model.addVar(vtype=GRB.CONTINUOUS, name="x2", lb=0)

# Set objective function: Maximize z = 4*x1 - x2
model.setObjective(4 * x1 - x2, GRB.MAXIMIZE)

# Add constraints
model.addConstr(7 * x1 - 2 * x2 <= 14, "constraint_1")
model.addConstr(x2 <= 3, "constraint_2")
model.addConstr(2 * x1 - 2 * x2 <= 3, "constraint_3")

# Optimize the model
model.optimize()

# Print the results
if model.status == GRB.OPTIMAL:
    print(f"Optimal solution found:")
    print(f"x1 = {x1.X}")
    print(f"x2 = {x2.X}")
    print(f"Objective value (z) = {model.ObjVal}")
else:
    print("No optimal solution found.")
```

```
Gurobi Optimizer version 11.0.3 build v11.0.3rc0 (win64 - Windows 10.0
(19045.2))

CPU model: Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz, instruction set
[SSE2|AVX|AVX2]
Thread count: 8 physical cores, 16 logical processors, using up to 16 threads

Optimize a model with 3 rows, 2 columns and 5 nonzeros
Model fingerprint: 0xbc5ea0ab
Coefficient statistics:
  Matrix range     [1e+00, 7e+00]
```

```
  Objective range  [1e+00, 4e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [3e+00, 1e+01]
Presolve removed 1 rows and 0 columns
Presolve time: 0.01s
Presolved: 2 rows, 2 columns, 4 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    8.4305000e+00   1.350000e-02   0.000000e+00      0s
       1    8.4285714e+00   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.02 seconds (0.00 work units)
Optimal objective  8.428571429e+00
Optimal solution found:
x1 = 2.857142857142857
x2 = 3.0
Objective value (z) = 8.428571428571429
```

2.3 Sub-Optimal 1

[3]:
```python
# Initialize the model
model = gp.Model("Sub-optimal 1")

# Define variables x1 and x2 as integer non-negative variables
x1 = model.addVar(vtype=GRB.CONTINUOUS, name="x1", lb=0)
x2 = model.addVar(vtype=GRB.CONTINUOUS, name="x2", lb=0)

# Set objective function: Maximize z = 4*x1 - x2
model.setObjective(4 * x1 - x2, GRB.MAXIMIZE)

# Add constraints
model.addConstr(7 * x1 - 2 * x2 <= 14, "constraint_1")
model.addConstr(x2 <= 3, "constraint_2")
model.addConstr(2 * x1 - 2 * x2 <= 3, "constraint_3")
model.addConstr(x1 <= 2, "constraint_4")  # Additional constraint

# Optimize the model
model.optimize()

# Print the results
if model.status == GRB.OPTIMAL:
    print(f"Optimal solution found:")
    print(f"x1 = {x1.X}")
    print(f"x2 = {x2.X}")
    print(f"Objective value (z) = {model.ObjVal}")
else:
    print("No optimal solution found.")
```

```
Gurobi Optimizer version 11.0.3 build v11.0.3rc0 (win64 - Windows 10.0
(19045.2))

CPU model: Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz, instruction set
[SSE2|AVX|AVX2]
Thread count: 8 physical cores, 16 logical processors, using up to 16 threads

Optimize a model with 4 rows, 2 columns and 6 nonzeros
Model fingerprint: 0x4ecb2306
Coefficient statistics:
  Matrix range     [1e+00, 7e+00]
  Objective range  [1e+00, 4e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [2e+00, 1e+01]
Presolve removed 3 rows and 0 columns
Presolve time: 0.00s
Presolved: 1 rows, 2 columns, 2 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    7.5000000e+00   0.000000e+00   0.000000e+00      0s
       0    7.5000000e+00   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.01 seconds (0.00 work units)
Optimal objective  7.500000000e+00
Optimal solution found:
x1 = 2.0
x2 = 0.5
Objective value (z) = 7.5
```

2.4 Sub-optimal 2

[4]:
```python
# Initialize the model
model = gp.Model("Sub-optimal 2")

# Define variables x1 and x2 as integer non-negative variables
x1 = model.addVar(vtype=GRB.CONTINUOUS, name="x1", lb=0)
x2 = model.addVar(vtype=GRB.CONTINUOUS, name="x2", lb=0)

# Set objective function: Maximize z = 4*x1 - x2
model.setObjective(4 * x1 - x2, GRB.MAXIMIZE)

# Add constraints
model.addConstr(7 * x1 - 2 * x2 <= 14, "constraint_1")
model.addConstr(x2 <= 3, "constraint_2")
model.addConstr(2 * x1 - 2 * x2 <= 3, "constraint_3")
model.addConstr(x1 >= 3, "constraint_4")  # Additional constraint
```

```python
# Optimize the model
model.optimize()

# Print the results
if model.status == GRB.OPTIMAL:
    print(f"Optimal solution found:")
    print(f"x1 = {x1.X}")
    print(f"x2 = {x2.X}")
    print(f"Objective value (z) = {model.ObjVal}")
else:
    print("No optimal solution found.")
```

```
Gurobi Optimizer version 11.0.3 build v11.0.3rc0 (win64 - Windows 10.0
(19045.2))

CPU model: Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz, instruction set
[SSE2|AVX|AVX2]
Thread count: 8 physical cores, 16 logical processors, using up to 16 threads

Optimize a model with 4 rows, 2 columns and 6 nonzeros
Model fingerprint: 0x3dface64
Coefficient statistics:
  Matrix range     [1e+00, 7e+00]
  Objective range  [1e+00, 4e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [3e+00, 1e+01]
Presolve removed 2 rows and 0 columns
Presolve time: 0.01s

Solved in 0 iterations and 0.01 seconds (0.00 work units)
Infeasible or unbounded model
No optimal solution found.
```

2.5 Sub-optimal 3

[ ]:
```python
# Initialize the model
model = gp.Model("Sub-optimal 3")

# Define variables x1 and x2 as integer non-negative variables
x1 = model.addVar(vtype=GRB.CONTINUOUS, name="x1", lb=0)
x2 = model.addVar(vtype=GRB.CONTINUOUS, name="x2", lb=0)

# Set objective function: Maximize z = 4*x1 - x2
model.setObjective(4 * x1 - x2, GRB.MAXIMIZE)

# Add constraints
model.addConstr(7 * x1 - 2 * x2 <= 14, "constraint_1")
```

```python
model.addConstr(x2 <= 3, "constraint_2")
model.addConstr(2 * x1 - 2 * x2 <= 3, "constraint_3")
model.addConstr(x1 <= 2, "constraint_4")  # Additional constraint
model.addConstr(x2 >= 1, "constraint_5")  # Additional constraint

# Optimize the model
model.optimize()

# Print the results
if model.status == GRB.OPTIMAL:
    print(f"Optimal solution found:")
    print(f"x1 = {x1.X}")
    print(f"x2 = {x2.X}")
    print(f"Objective value (z) = {model.ObjVal}")
else:
    print("No optimal solution found.")
```

```
Gurobi Optimizer version 11.0.3 build v11.0.3rc0 (win64 - Windows 10.0
(19045.2))

CPU model: Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz, instruction set
[SSE2|AVX|AVX2]
Thread count: 8 physical cores, 16 logical processors, using up to 16 threads

Optimize a model with 5 rows, 2 columns and 7 nonzeros
Model fingerprint: 0x68e4634b
Coefficient statistics:
  Matrix range     [1e+00, 7e+00]
  Objective range  [1e+00, 4e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+01]
Presolve removed 5 rows and 2 columns
Presolve time: 0.00s
Presolve: All rows and columns removed
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    7.0000000e+00   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.01 seconds (0.00 work units)
Optimal objective  7.000000000e+00
Optimal solution found:
x1 = 2.0
x2 = 1.0
Objective value (z) = 7.0
```

2.6 Sub-optimal 4

[ ]:
```python
# Initialize the model
model = gp.Model("Sub-optimal 4")

# Define variables x1 and x2 as integer non-negative variables
x1 = model.addVar(vtype=GRB.CONTINUOUS, name="x1", lb=0)
x2 = model.addVar(vtype=GRB.CONTINUOUS, name="x2", lb=0)

# Set objective function: Maximize z = 4*x1 - x2
model.setObjective(4 * x1 - x2, GRB.MAXIMIZE)

# Add constraints
model.addConstr(7 * x1 - 2 * x2 <= 14, "constraint_1")
model.addConstr(x2 <= 3, "constraint_2")
model.addConstr(2 * x1 - 2 * x2 <= 3, "constraint_3")
model.addConstr(x1 <= 2, "constraint_4")  # Additional constraint
model.addConstr(x2 <= 0, "constraint_5")  # Additional constraint

# Optimize the model
model.optimize()

# Print the results
if model.status == GRB.OPTIMAL:
    print(f"Optimal solution found:")
    print(f"x1 = {x1.X}")
    print(f"x2 = {x2.X}")
    print(f"Objective value (z) = {model.ObjVal}")
else:
    print("No optimal solution found.")
```

```
Gurobi Optimizer version 11.0.3 build v11.0.3rc0 (win64 - Windows 10.0
(19045.2))

CPU model: Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz, instruction set
[SSE2|AVX|AVX2]
Thread count: 8 physical cores, 16 logical processors, using up to 16 threads

Optimize a model with 5 rows, 2 columns and 7 nonzeros
Model fingerprint: 0xfec00530
Coefficient statistics:
  Matrix range     [1e+00, 7e+00]
  Objective range  [1e+00, 4e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [2e+00, 1e+01]
Presolve removed 5 rows and 2 columns
Presolve time: 0.01s
Presolve: All rows and columns removed
```

```
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    6.0000000e+00   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.01 seconds (0.00 work units)
Optimal objective  6.000000000e+00
Optimal solution found:
x1 = 1.5
x2 = 0.0
Objective value (z) = 6.0
```