Homework - Optimization Models and Methods

https://www.columbia.edu/~ks20/4106-24-Fall/4106-24-Fall.html

IEOR E4004: Optimization Models and Methods
Instructor: Dr. Yaren Bilge Kaya
Due: 11/3/2024, 11:59 pm

Assignment 3

October 17, 2024

Question 1. *(40 points) Tosla needs to schedule its workforce for the upcoming week in order to maintain efficient operation across all its departments, including battery, body, assembly, paint, and quality control. Each department has specific minimum and maximum staffing needs for three shifts: morning, afternoon, and night for each day of the week (Mon-Sun). Tosla's workforce is flexible, but workers have specific availabilities, preferences for working certain shifts, and different effectiveness scores that measure their productivity in certain roles (on a scale of 1-10, with 10 being the most preferred or effective).*

The goal is to maximize the total preference-adjusted effectiveness of the workers scheduled for each shift, subject to the following constraints:

- *Each worker can only be assigned to work a maximum of one shift per day.*
- *Each worker can only work a maximum of 5 days a week.*
- *Each worker can only be assigned to shifts that they are available for.*
- *The total number of workers assigned to a shift in a department must meet the department's specific minimum and maximum staffing requirements.*

Your task is to formulate this scheduling problem as a linear/integer programming problem and solve it using an appropriate optimization software (preferably Gurobi).

Use the randomly generated data given below to solve the model where we assume there are 100 skilled workers waiting to be assigned to a shift and department. Keep in mind that, since this is a randomly generated dataset, some of you might be working with infeasible problems.

(a) *(10 points) Define the parameters, sets, decision variables, constraints, and the objective you are using to model.*

(b) *(20 points) Formulate the problem algebraically.*

(c) *(10 points) Solve the problem using Python & Gurobi.*

```
# 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))
})
```

Assignment 3

Question 1

(a) Definition of Parameters, Sets, Decision Variables, Constraints, and Objective

---

Sets:

- **Workers (W)**: Set of workers

$$W = [1, 2, 3, \ldots, 100]$$

- **Departments (D)**: List of departments

$$D = [\text{Battery},\ \text{Body},\ \text{Assembly},\ \text{Paint},\ \text{Quality}]$$

- **Shifts (S)**: List of shifts

$$S = [\text{Morning},\ \text{Afternoon},\ \text{Night}]$$

- **Days (T)**: List of days

$$T = [\text{Mon},\ \text{Tue},\ \text{Wed},\ \text{Thur},\ \text{Fri},\ \text{Sat},\ \text{Sun}]$$

Parameters:

- **Availability ($A_{w,d,s,t}$)**: Binary parameter indicating whether worker w is available to work in department d, shift s, on day t.

$$A_{w,d,s,t} = \begin{cases} 1 & \text{if worker } w \text{ is available} \\ 0 & \text{otherwise} \end{cases}$$

- **Preference Score ($P_{w,d,s,t}$)**: Integer score from 1 to 10 indicating worker w 's preference for working in department d, shift s, on day t.

- **Effectiveness Score ($E_{w,d,s,t}$)**: Integer score from 1 to 10 indicating worker w's effectiveness in department d, shift s, on day t.

- **Minimum Workers Required ($\text{MinWorkers}_{d,s,t}$)**: Minimum number of workers required in department d, shift s, on day t.

- **Maximum Workers Allowed ($\text{MaxWorkers}_{d,s,t}$)**: Maximum number of workers allowed in department d, shift s, on day t.

Decision Variables:

- **Assignment Variable ($x_{w,d,s,t}$)**:

$$x_{w,d,s,t} = \begin{cases} 1 & \text{if worker } w \text{ is assigned to department } d, \text{ shift } s, \text{ on day } t \\ 0 & \text{otherwise} \end{cases}$$

Objective Function:

Maximize the total preference-adjusted effectiveness:

$$\max \quad Z = \sum_{w \in W} \sum_{d \in D} \sum_{s \in S} \sum_{t \in T} P_{w,d,s,t} \times E_{w,d,s,t} \times x_{w,d,s,t}$$

Constraints:

1. **Availability Constraint**:

   Workers can only be assigned to shifts they are available for.

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

2. **One Shift per Day Constraint**:

   Each worker can be assigned to at most one shift per day.

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

3. **Maximum Working Days per Week Constraint**:

   Each worker can work at most 5 days per week.

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

4. **Department Staffing Requirements**:

   The number of workers assigned must meet the department's minimum and maximum staffing requirements.

$$\text{MinWorkers}_{d,s,t} \leq \sum_{w \in W} x_{w,d,s,t} \leq \text{MaxWorkers}_{d,s,t} \quad \forall\, d \in D,\ s \in S,\ t \in$$

5. **Binary Decision Variables**:

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

(b) Algebraic Formulation

Objective Function:

$$\max \quad Z = \sum_{w=1}^{100} \sum_{d \in D} \sum_{s \in S} \sum_{t \in T} P_{w,d,s,t} \cdot E_{w,d,s,t} \cdot x_{w,d,s,t}$$

Subject to:

1. **Availability Constraints**:

$$x_{w,d,s,t} \leq A_{w,d,s,t} \quad \forall\, w = 1, \ldots, 100;\ d \in D;\ s \in S;\ t \in T$$

2. **One Shift per Day Constraints**:

$$\sum_{d \in D} \sum_{s \in S} x_{w,d,s,t} \leq 1 \quad \forall\, w = 1, \ldots, 100;\ t \in T$$

3. **Maximum Working Days per Week Constraints**:

$$\sum_{t \in T} \sum_{d \in D} \sum_{s \in S} x_{w,d,s,t} \leq 5 \quad \forall\, w = 1, \ldots, 100$$

4. **Department Staffing Constraints**:

$$\text{MinWorkers}_{d,s,t} \leq \sum_{w=1}^{100} x_{w,d,s,t} \leq \text{MaxWorkers}_{d,s,t} \quad \forall\, d \in D;\ s \in S;\ t \in T$$

5. **Binary Variables**:

$$x_{w,d,s,t} \in \{0, 1\} \quad \forall\, w = 1, \ldots, 100;\ d \in D;\ s \in S;\ t \in T$$

Explanation:

- **Objective Function**: Maximizes the total sum of preference-adjusted effectiveness scores for all worker assignments.

- **Constraints**:

  1. **Availability**: Ensures workers are only assigned to shifts they are available for.

  2. **One Shift per Day**: Prevents workers from being assigned to more than one shift in a single day.

3. **Maximum Working Days**: Limits each worker to a maximum of 5 working days per week.

4. **Staffing Requirements**: Ensures each department's shift meets its minimum and maximum staffing needs.

5. **Binary Variables**: Enforces that assignments are either made (1) or not made (0).

c)

In [13]:

base code given

```
import pandas as pd
import numpy as np

# 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)),ler
    'Shift': np.tile(np.repeat(shifts , len(days)), len(workers)*len(departm
    'Day': np.tile(days, len(workers)*len(departments)*len(shifts)),
    'Availability': np.random.choice([0, 1], len(workers)*len(departments)*l
    'Preference_Score': np.random.randint(1, 10, len(workers)*len(department
    'Effectiveness_Score': np.random.randint(1, 10, len(workers)*len(departm
})

# 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(
    'Max_Workers': np.random.randint(5, 10, len(departments)*len(shifts)*ler
})
```

In [15]:

```
# Initialize the model
m = Model("Workforce_Scheduling")

# Decision variables
# Create variables only where Availability is 1
x = {}
for w in workers:
    for d in departments:
        for s in shifts:
            for t in days:
                avail = workers_df[
                    (workers_df['Worker_ID'] == w) &
```

```python
                    (workers_df['Department'] == d) &
                    (workers_df['Shift'] == s) &
                    (workers_df['Day'] == t)
                ]['Availability'].values
                if avail.size > 0 and avail[0] == 1:
                    x[w, d, s, t] = m.addVar(vtype=GRB.BINARY, name=f"x_{w}_

m.update()

# Objective function:
m.setObjective(quicksum(
    workers_df[
        (workers_df['Worker_ID'] == w) &
        (workers_df['Department'] == d) &
        (workers_df['Shift'] == s) &
        (workers_df['Day'] == t)
    ]['Preference_Score'].values[0] *
    workers_df[
        (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, d, s, t in x
), GRB.MAXIMIZE)
```

$$\max \quad Z = \sum_{w=1}^{100} \sum_{d \in D} \sum_{s \in S} \sum_{t \in T} P_{w,d,s,t} \cdot E_{w,d,s,t} \cdot x_{w,d,s,t}$$

```python
# Constraints

# 1. Each worker can be assigned to at most one shift per day
for w in workers:
    for t in days:
        shifts_worked = [x[w, d, s, t] for d in departments for s in shifts
        if shifts_worked:
            m.addConstr(quicksum(shifts_worked) <= 1, name=f"OneShiftPerDay_
```

One Shift per Day Constraints:

$$\sum_{d \in D} \sum_{s \in S} x_{w,d,s,t} \leq 1 \quad \forall\, w = 1, \ldots, 100;\ t \in T$$

```python
# 2. Each worker can work at most 5 days per week
for w in workers:
    days_worked = [x[w, d, s, t] for d in departments for s in shifts for t
    if days_worked:
        m.addConstr(quicksum(days_worked) <= 5, name=f"MaxDaysPerWeek_{w}")
```

Maximum Working Days per Week Constraints:

$$\sum_{t \in T} \sum_{d \in D} \sum_{s \in S} x_{w,d,s,t} \leq 5 \quad \forall\, w = 1, \ldots, 100$$

```python
# 3. Department staffing needs
for d in departments:
    for s in shifts:
        for t in days:
            staff_needed = [x[w, d, s, t] for w in workers if (w, d, s, t) i
            if staff_needed:
                min_workers = dept_df[
                    (dept_df['Department'] == d) &
                    (dept_df['Shift'] == s) &
                    (dept_df['Day'] == t)
                ]['Min_Workers'].values[0]
                max_workers = dept_df[
                    (dept_df['Department'] == d) &
                    (dept_df['Shift'] == s) &
```

Department Staffing Constraints:

$$\text{MinWorkers}_{d,s,t} \leq \sum_{w=1}^{100} x_{w,d,s,t} \leq \text{MaxWorkers}_{d,s,t} \quad \forall\, d \in D;\ s \in S;\ t \in T$$

```
                (dept_df['Day'] == t)
            ]['Max_Workers'].values[0]
            m.addConstr(quicksum(staff_needed) >= min_workers, name=f"Mi
            m.addConstr(quicksum(staff_needed) <= max_workers, name=f"Ma
```

```
# 4. Department staffing needs
for d in departments:
    for s in shifts:
        for t in days:
            min_workers = dept_df[
                (dept_df['Department'] == d) &
                (dept_df['Shift'] == s) &
                (dept_df['Day'] == t)
            ]['Min_Workers'].values[0]
            max_workers = dept_df[
                (dept_df['Department'] == d) &
                (dept_df['Shift'] == s) &
                (dept_df['Day'] == t)
            ]['Max_Workers'].values[0]
            m.addConstr(
                quicksum(
                    x[w, d, s, t] for w in workers if (w, d, s, t) in x
                ) >= min_workers,
                name=f"MinStaff_{d}_{s}_{t}"
            )
            m.addConstr(
                quicksum(
                    x[w, d, s, t] for w in workers if (w, d, s, t) in x
                ) <= max_workers,
                name=f"MaxStaff_{d}_{s}_{t}"
            )
```

Availability Constraint:

Workers can only be assigned to shifts they are available for.

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

```
m.setParam('OutputFlag', 0)  # cleaner output
m.optimize()

# Output results
if m.status == GRB.OPTIMAL:
    print("\nOptimal solution found:\n")
    for w, d, s, t in x:
        if round(x[w, d, s, t].X) == 1:
            print(f"Worker {w} assigned to department {d}, shift {s}, on {t}
else:
    print("No optimal solution found.")
```

```
Optimal solution found:

Worker 1 assigned to department Body, shift Morning, on Wed
Worker 1 assigned to department Body, shift Night, on Mon
Worker 1 assigned to department Paint, shift Morning, on Sun
Worker 1 assigned to department Quality, shift Night, on Tue
Worker 1 assigned to department Quality, shift Night, on Thur
Worker 2 assigned to department Battery, shift Morning, on Tue
Worker 2 assigned to department Battery, shift Morning, on Thur
Worker 2 assigned to department Body, shift Night, on Sat
Worker 2 assigned to department Paint, shift Morning, on Wed
Worker 2 assigned to department Paint, shift Afternoon, on Sun
Worker 3 assigned to department Body, shift Morning, on Wed
Worker 3 assigned to department Body, shift Afternoon, on Fri
Worker 3 assigned to department Assembly, shift Morning, on Sun
Worker 3 assigned to department Paint, shift Morning, on Thur
Worker 3 assigned to department Quality, shift Afternoon, on Mon
Worker 4 assigned to department Body, shift Afternoon, on Wed
Worker 4 assigned to department Paint, shift Night, on Thur
Worker 4 assigned to department Paint, shift Night, on Fri
Worker 4 assigned to department Quality, shift Afternoon, on Tue
Worker 4 assigned to department Quality, shift Afternoon, on Sat
Worker 5 assigned to department Battery, shift Morning, on Wed
Worker 5 assigned to department Battery, shift Morning, on Sun
Worker 5 assigned to department Body, shift Night, on Thur
Worker 5 assigned to department Assembly, shift Afternoon, on Sat
Worker 5 assigned to department Paint, shift Morning, on Fri
Worker 6 assigned to department Battery, shift Morning, on Wed
Worker 6 assigned to department Assembly, shift Morning, on Mon
Worker 6 assigned to department Assembly, shift Morning, on Sat
Worker 6 assigned to department Assembly, shift Afternoon, on Fri
Worker 6 assigned to department Paint, shift Night, on Sun
Worker 7 assigned to department Battery, shift Afternoon, on Fri
Worker 7 assigned to department Body, shift Night, on Thur
Worker 7 assigned to department Paint, shift Morning, on Mon
Worker 7 assigned to department Quality, shift Morning, on Sat
Worker 7 assigned to department Quality, shift Night, on Sun
Worker 8 assigned to department Battery, shift Morning, on Mon
Worker 8 assigned to department Assembly, shift Morning, on Sun
Worker 8 assigned to department Assembly, shift Afternoon, on Thur
Worker 8 assigned to department Assembly, shift Night, on Sat
Worker 8 assigned to department Quality, shift Morning, on Fri
Worker 9 assigned to department Battery, shift Afternoon, on Mon
Worker 9 assigned to department Battery, shift Afternoon, on Fri
Worker 9 assigned to department Assembly, shift Morning, on Thur
Worker 9 assigned to department Paint, shift Night, on Wed
Worker 9 assigned to department Quality, shift Morning, on Sun
Worker 10 assigned to department Battery, shift Afternoon, on Thur
Worker 10 assigned to department Body, shift Morning, on Fri
Worker 10 assigned to department Assembly, shift Morning, on Sun
Worker 10 assigned to department Quality, shift Afternoon, on Sat
Worker 10 assigned to department Quality, shift Night, on Tue
Worker 11 assigned to department Battery, shift Night, on Mon
Worker 11 assigned to department Body, shift Afternoon, on Tue
Worker 11 assigned to department Assembly, shift Night, on Sat
Worker 11 assigned to department Paint, shift Afternoon, on Wed
```

continues up to worker 100

IEOR E4004: Optimization Models and Methods
Instructor: Dr. Yaren Bilge Kaya
Due: 11/3/2024, 11:59 pm

Assignment 3

October 17, 2024

Question 2. *(10 points) Suppose that $x_1, \dots, x_n$ are unrestricted but bounded continuous variables, and the following two constraints are given:*

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n \geq b_1$$
$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \dots + a_{2n}x_n \geq b_2$$

How can the satisfaction of at least one of these constraints be modeled?

Hint: *Consider a larger variable space, as well as a "big enough" positive number M.*

- We need to use M = very large number and a binary variable $w_i \in \{0,1\}$:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \geq b_1 - M(1-w_1) \quad \text{and} \quad 1 \leq w_1 + w_2 \leq 2 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \geq b_2 - M(1-w_2) \end{cases}$$

So that, if $w_1 = 1$, the first linear equation is "always true" and the second one is the only one active (w_2 forced to be equal to 0)
Similarly, if $w_2 = 1$, the second linear equation is "always true" and the first one is the only one active (w_1 forced to be equal to 0)

IEOR E4004: Optimization Models and Methods
Instructor: Dr. Yaren Bilge Kaya
Due: 11/3/2024, 11:59 pm

IEOR Industrial Engineering & Operations Research COLUMBIA | ENGINEERING

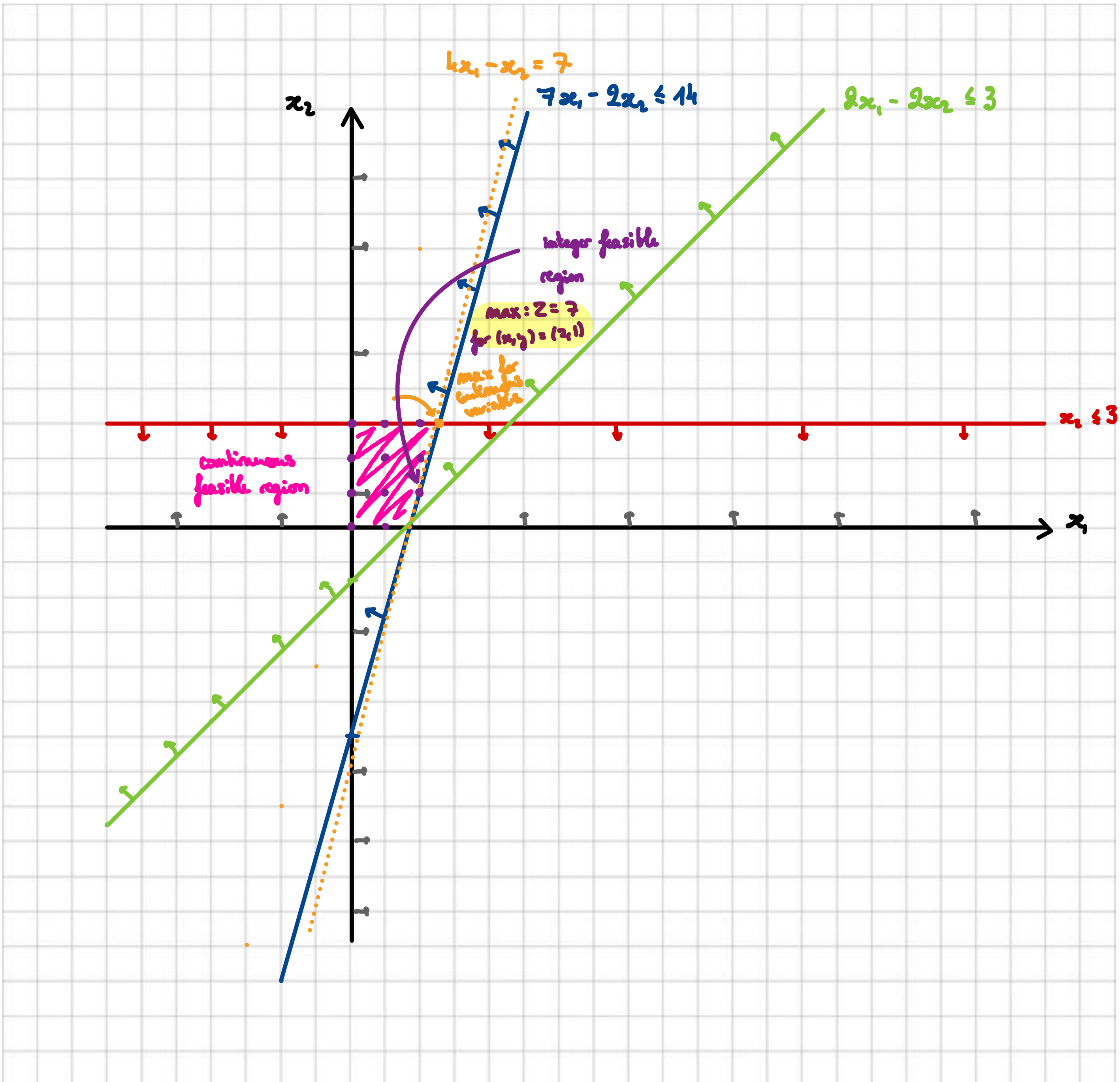
Assignment 3

October 17, 2024

Question 3. *(10 points) Consider the following optimization problem featuring non-negative general integer variables:*

$$
\begin{aligned}
\max \quad & z = 4x_1 - x_2 \\
\textit{subject to} \quad & 7x_1 - 2x_2 \leq 14 \\
& x_2 \leq 3 \\
& 2x_1 - 2x_2 \leq 3 \\
& x_j \in Z_+^2.
\end{aligned}
$$

Draw the feasible region of the integer program (possible as there are just two variables).



IEOR E4004: Optimization Models and Methods
Instructor: Dr. Yaren Bilge Kaya
Due: 11/3/2024, 11:59 pm

IEOR Industrial Engineering & Operations Research COLUMBIA | ENGINEERING

Assignment 3

October 17, 2024

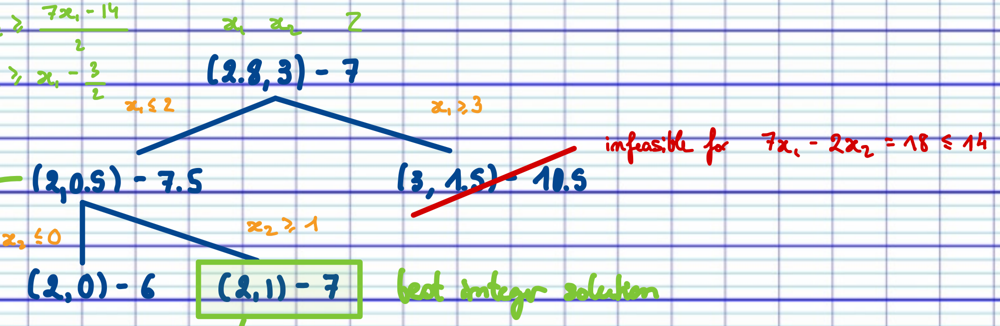
Question 4. *(40 points) Consider the IP in Question 3 and solve the problem using branch-and-bound. Draw out the tree, as well as the changes to the feasible region. Use the following criterion for node selection. For the first branch, choose the left-most branch. After this, always choose the right-most branch (if one exists), and prefer higher branches to lower branches.*

$$
\begin{aligned}
\max \quad & z = 4x_1 - x_2 \\
\textit{subject to} \quad & 7x_1 - 2x_2 \le 14 \\
& x_2 \le 3 \\
& 2x_1 - 2x_2 \le 3 \\
& x_j \in Z_+^2.
\end{aligned}
$$

- From the previous question, we get an optimal solution from the graph of: $\begin{cases} x_1 = 2.5 \\ x_2 = 3 \end{cases}$
For an objective value of $Z = 4(2.5) - 3 = 7$

$x_2 \geq \frac{7x_1 - 14}{2}$

$x_2 \geq x_1 - \frac{3}{2}$




```
m.addConstr(x1 <= 2, "branch")
x1 = 2.0
x2 = 0.5
Objective value (z) = 7.5
```

```
m.addConstr(x1 <= 2, "branch")
m.addConstr(x2 >= 1, "branch1")
```

"proceed step by step"

- For each step I used this code and simply added branching constraints on the decision variables.

```
m = Model("IP")

x1 = m.addVar(vtype=GRB.CONTINUOUS, name="x1")
x2 = m.addVar(vtype=GRB.CONTINUOUS, name="x2")

m.setObjective(4 * x1 - x2, GRB.MAXIMIZE)

m.addConstr(7 * x1 - 2 * x2 <= 14, "c1")
m.addConstr(x2 <= 3, "c2")
m.addConstr(2 * x1 - 2 * x2 <= 3, "c3")

m.addConstr(x1 <= 2, "branch")
m.addConstr(x2 >= 1, "branch1")

m.optimize()

if m.status == GRB.OPTIMAL:
    print(f"Optimal solution found:")
    print(f"x1 = {x1.x}")
    print(f"x2 = {x2.x}")
    print(f"Objective value (z) = {m.objVal}")
else:
    print("No optimal solution found.")
```

change depending on branching