Bilkent University

Department of Computer Engineering

IE 400 Term Project

Group 36

Project Report

Project Group Members:

Agil Aliyev - 21701367

Emre Derman - 21703508

Zübeyir Bodur - 21702382

TABLE OF CONTENTS

1. Introduction

In this project, we are offering a solution for a personalized medicine optimization problem.

In this problem, Doctor Who, who works at the XYZ Medical Center, decided to find a personalized way to improve the treatment of breast cancer patients. Chemotherapy, where one or more anti-cancer drugs are used as part of a standardized regimen, is known to be one of the most effective cancer treatments. Chemotherapy may be given with a curative intent, or it may aim to prolong life and improve quality of life by reducing symptoms. Quality of Life (*Q*-score) is an important health metric that is used in the medical literature to summarize important information pertaining to many different aspects of a patient's health outcomes. A higher *Q*-score indicates better health conditions for a patient.

Doctor Who currently has more than 60 patients, and we are assigned the 36th patient, whose features are given in Section 2.1. Our goal is to find the following:

- Find the maximum Q-score the drugs in the base regimen can provide, while dosages are subject to change (part A)
- Find recipient with minimum deviation cost from the base regimen while achieving our patient's Q-threshold (part B)
- Find the recipient with minimum deviation while complying a set of constraints given by Doctor Who (part C)

In this report, we have defined our parameters, functions and indexes that are commonly used in all models. Then we have introduced our decision variables for each part. Then, in each part, we have precisely described our mathematical models that would represent the problem.

After formulating the models, we have solved our models using Gurobi Optimizer API, and also provided the source that would solve those models in Section 4.

2. Mathematical Models

Below are the mathematical models for each problem. For convenience, since each model shares common parameters, such as patient data, minimum/maximum allowed dosages, etc, those can be introduced before describing the models.

2.1. Common Constants & Parameters

p_j: j^{th} feature of patient 36

$\mathbf{p}$: Row vector of features for patient 36

$$\mathbf{p} = [p_1\ p_2 \ldots p_8\ p_9] = [0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1]$$

Q_{36}: Quality of life threshold for patient 36

$$Q_{36} = 20$$

$(min)_i$: Minimum allowed dose for the drug i, in cl

$$(min)_1 = 20$$

$$(min)_2 = 10$$

$$(min)_3 = 20$$

$$(min)_4 = 10$$

$$(min)_5 = 10$$

$$(min)_6 = 20$$

$$(min)_7 = 20$$

max_i: Maximum allowed dose for the drug i, in cl

$$(max)_1 = 80$$

$$(max)_2 = 50$$

$$(max)_3 = 100$$

$$(max)_4 = 100$$

$$(max)_5 = 70$$

$$(max)_6 = 90$$

$$(max)_7 = 50$$

$(yb)_i$: Whether drug i is included in the base regimen

$$(yb)_1 = 1$$

$$(yb)_2 = 0$$

$$(yb)_3 = 1$$

$$(yb)_4 = 1$$

$$(yb)_5 = 0$$

$$(yb)_6 = 0$$

$$(yb)_7 = 1$$

$(xb)_i$: Amount of dosage for drug i in the base regimen, in cl

$$(xb)_1 = 20$$

$$(xb)_2 = 0$$

$$(xb)_3 = 30$$

$$(xb)_4 = 15$$

$$(xb)_5 = 0$$

$$(xb)_6 = 0$$

$$(xb)_7 = 35$$

$(fb)_i$: Fixed cost of adding/removing drug i from the base regimen

$$(fb)_1 = 25$$

$$(fb)_2 = 50$$

$$(fb)_3 = 10$$

$$(fb)_4 = 25$$

$$(fb)_5 = 20$$

$$(fb)_6 = 30$$

$$(fb)_7 = 40$$

$(ub)_i$: Cost of dosage change for drug i per cl.

$$(ub)_1 = 1$$

$$(ub)_2 = 2$$

$$(ub)_3 = 1$$

$$(ub)_4 = 3$$

$$(ub)_5 = 2$$

$$(ub)_6 = 1$$

$$(ub)_7 = 1$$

2.1.1. Common Functions

$$\begin{aligned} \mathrm{Q}(\mathbf{p}, \mathbf{y}, \mathbf{x}) = & -5p_1 - 0.5p_2 - 12p_3 - 8p_4 - 5p_5 - 5p_6 - 1p_7 - 3p_8 - 2p_9 \\ & - 5y_1 - 6y_2 - 4y_3 - 4y_4 - 8y_5 - 6y_6 - 7y_7 \\ & + 0.28x_1 + 0.3x_2 + 0.25x_3 + 0.17x_4 + 0.31\,x_5 + 0.246x_6 + 0.4x_7 \end{aligned}$$

The input parameters are row vectors, where x_i represents the ith number in vector **x**, y_i represents the ith number in vector **y** and p_j represents the jth number in vector **p.**

2.1.2. Common Indexes

$i \in \{1, 2, 3, 4, 5, 6, 7\}$

$j \in \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$

2.2. Part A

In this part, it was asked to maximize the value of the quality of life value, while using the base regimen for the treatment while only changing the dosage of the drugs for patient 36.

2.2.1. Decision Variables

x_i: Amount of dosage given to patient 36 from drug i, in cl

x : Vector of drug dosages for patient 36

y_i: Whether drug i is given to patient 36

y : Vector of binary variables for patient 36, representing which drugs will be given to them

2.2.2. Objective

Z : Quality of life score of patient 36

2.2.3. Model

For the 36th patient, we will have the following maximization problem:

$$max\, z = Q(\mathbf{p}, \mathbf{y}, \mathbf{x})$$

$$s.t.$$

$$\mathbf{x} = [x_1\, x_2 \ldots\ x_6\, x_7]$$

$$\boldsymbol{y} = [y_1\, y_2 \ldots\ y_6\, y_7]$$

$$y_i = (yb)_i \qquad \forall i$$

$$x_i \geq (min)_i \qquad i = 1, 3, 4, 7$$

$$x_i \leq (max)_i \qquad i = 1, 3, 4, 7$$

$$x_i = 0 \qquad i = 2, 5, 6$$

$$x_i \geq 0 \qquad \forall i$$

$$y_i \in \{0, 1\} \qquad \forall i$$

2.3. Part B

In this part, it was asked to minimize the cost of changing the base regimen while achieving the quality of life threshold for patient 36.

We assumed that the fixed cost of adding/ removing a drug occurs only when the drug is removed or added from the base regimen.

For example, drug 1 is included in the base regimen. If the drug 1 will have a dosage of 0 and therefore is removed, then the fixed cost of removing a drug will apply. However, changing the dosage of drug 1 within limits will not incur this fixed cost.

Similarly, drug 2 is not included in the base regimen. If drug 2 will have a dosage different than zero, and its dosage is within the limits and therefore is added, then the fixed cost of adding a drug will apply.

2.3.1. Decision Variables

a_i: Whether the drug i is added or removed

$(inc)_i$: Whether the dosage of drug i is increased

$(dec)_i$: Whether the dosage of drug i is decreased

c_i: Amount of change in dosage of drug i in the new regimen, in cl

x_i: Amount of dosage given to patient 36 from drug i, in cl

x : Vector of drug dosages for patient 36

y_i: Whether drug i is given to patient 36

y : Vector of binary variables for patient 36, representing which drugs will be given to them

2.3.2. Objective

z : Cost of deviating from the base regimen

2.3.3. Model

For the 36th patient, we will have the following minimization problem:

$$min\, z = \sum_{i=1}^{7} (fb)_i a_i + (ub)_i c_i$$

$s.t.$

$$\mathbf{x} = [x_1\, x_2 \ldots\, x_6\, x_7]$$

$$\boldsymbol{y} = [y_1\, y_2 \ldots y_6\, y_7]$$

$$Q(\mathbf{p}, \mathbf{y}, \mathbf{x}) \geq Q_{36}$$

$$y_i = a_i \qquad i = 2, 5, 6$$

$$y_i = 1 - a_i \qquad i = 1, 3, 4, 7$$

$$(inc)_i + (dec)_i \leq 1 \qquad \forall i$$

$$c_i((inc)_i - (dec)_i) = x_i - (xb)_i \qquad \forall i$$

$$x_i \geq (min)_i y_i \qquad \forall i$$

$$x_i \leq (max)_i y_i \qquad \forall i$$

$$a_i, y_i, (inc)_i, (dec)_i \in \{0,1\} \qquad \forall i$$

$$c_i, x_i \geq 0 \qquad \forall i$$

2.4. Part C

In this part, in addition to model in part b, there were additional constraints given for the drugs, which were the following:

- Melphalan and Oxaliplatin, when combined in wrong amounts, decrease the effect of the therapy. Hence, when combined, the total amount of these two chemicals should be less than 70 cl and greater than 50cl in any regimen.

- Either Epirubicin should be included in the regimen or the dosage of Decitabine should be less than 25cl.
- If both Pentostatin and Lomoustine are included in the regimen, then at least one of the Thiotepa and Epirubicin should also be chosen.

2.4.1. Decision Variables

We first introduce the following parameter, M = 1000 (large enough integer).

Then we introduce the following decision variable:

W: Whether Pentostatin (4) and Lomoustine (6) are both included in the regimen

2.4.2. Objective

The objective function is the same:

Z : Cost of deviating from the base regimen

2.4.3. Model

Those constraints will be added to the model in part b:

$x_1 + x_2 \geq 50y_1y_2$ (First constraint - I)

$x_1 + x_2 \leq 70(y_1y_2) + M(1 - y_1y_2)$ (First constraint - II)

$(1 - y_5)x_3 \leq 25$ (Second constraint)

$w = y_4y_6$ (Third constraint - I)

$w(y_5 + y_7 - 1) \geq 0$ (Third constraint - II)

3. Implementation Constraints

To test and solve the models written in section 1, the Gurobi Optimizer API was used and solutions were implemented in the Python programming language. Anaconda was used as the interpreter for Python 3.9.

4. Source Code for Solvers

Below is the Python code for each part, that solves the models given in section 2.

4.1. Part A

```
"""
IE 400 Project - Part A
```

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



def q(p_in, y_in, x_in):
   p_sum = -5 * p_in[0] - 0.5 * p_in[1] - 12 * p_in[2] - 8 * p_in[3] - 5 * p_in[4] - 5 * p_in[5] - p_in[6] - 3 * p_in[7] - 2 * p_in[8]
   y_sum = -5 * y_in[0] - 6 * y_in[1] - 4 * y_in[2] - 4 * y_in[3] - 8 * y_in[4] - 6 * y_in[5] - 7 * y_in[6]
   x_sum = 0.28 * x_in[0] + 0.3 * x_in[1] + 0.25 * x_in[2] + 0.17 * x_in[3] + 0.31 * x_in[4] + 0.246 * x_in[5] + 0.4 * x_in[6]
   return p_sum + y_sum + x_sum



# Read patient data from .xlsx file
patient_data = pd.read_excel('patient_data.xlsx', sheet_name="Sheet1")

# Read the group number 36
patient_36 = patient_data.iloc[36, :]

# vector of features
p = list(patient_36[1:10])

# Q threshold
Q_36 = float(patient_36[10])

print("Group Number: " + str(patient_36[0]))
print("Patient vector: " + str(p))
```

```
print("Q threshold: " + str(Q_36))
# Maximum/Minimum allowed dosages
min_i = [20, 10, 20, 10, 10, 20, 20]
max_i = [80, 50, 100, 100, 70, 90, 50]

# Base regimen specs
yb_i = [1, 0, 1, 1, 0, 0, 1]
xb_i = [20, 0, 30, 15, 0, 0, 35]

# Cost of deviating from the base regimen
fb_i = [25, 50, 10, 25, 20, 30, 40]
ub_i = [1, 2, 1, 3, 2, 1, 1]

# Create a Model instance
model = gp.Model()

# Decision variables - from i = 1 to 7 inclusive
x = []
for i in range(1, 8):
   # Add each indexed decision variable one by one
   x.append(model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="x" + str(i)))

# Set the objective
model.setObjective(q(p, yb_i, x), GRB.MAXIMIZE)



# Add the remaining constraints
for i in range(0, 7):
```

```
    """
    Dosage bounds, if y = 0 the interval is [0, 0] = 0
    """
    if i == 0 or i == 2 or i== 3 or i ==6:
        model.addConstr(x[i] >= min_i[i], name="min" + str(i+1))
        model.addConstr(x[i] <= max_i[i], name="max" + str(i + 1))
    else :
        model.addConstr(x[i], GRB.EQUAL, 0, name="set" +str(i+1))
    model.update()

# Solve the model
model.write("parta.lp")
model.optimize()
model.printAttr('X')
model.update()
print("Decision Variables, x and y: ")
for i in range(1, 8):
      print("x" + str(i) + " = " + str(model.getVarByName("x" +
str(i)).getAttr('X')))
    print("y" + str(i) + " = " + str(yb_i[i-1]) + "\n")
print("Quality of Life, a.k.a. objective = " +
str(model.getObjective().getValue()))
```

4.2. Part B

```
"""
IE 400 Project - Part B
"""
import gurobipy as gp
from gurobipy import *
import pandas as pd
```

```python
def checkQ(p_in, my_model):
    y_ = []
    x_ = []
    for i in range(0, 7):
        x_.append(my_model.getVarByName("x" + str(i + 1)).getAttr('X'))
        y_.append(my_model.getVarByName("y" + str(i + 1)).getAttr('X'))
    return q(p_in, y_, x_)


def q(p_in, y_in, x_in):
    p_sum = -5*p_in[0] - 0.5*p_in[1] - 12*p_in[2] - 8*p_in[3] - 5*p_in[4] - 5*p_in[5] - p_in[6] - 3*p_in[7] - 2*p_in[8]
    y_sum = -5*y_in[0] - 6*y_in[1] - 4*y_in[2] - 4*y_in[3] - 8*y_in[4] - 6*y_in[5] - 7*y_in[6]
    x_sum = 0.28*x_in[0] + 0.3*x_in[1] + 0.25*x_in[2] + 0.17*x_in[3] + 0.31*x_in[4] + 0.246*x_in[5] + 0.4*x_in[6]
    return p_sum + y_sum + x_sum


# Read patient data from .xlsx file
patient_data = pd.read_excel('patient_data.xlsx', sheet_name="Sheet1")

# Read the group number 36
patient_36 = patient_data.iloc[36, :]

# vector of features
p = list(patient_36[1:10])
```

```python
# Q threshold
Q_36 = patient_36[10]

print("Group Number: " + str(patient_36[0]))
print("Patient vector: " + str(p))
print("Q threshold: " + str(Q_36))
# Maximum/Minimum allowed dosages
min_i = [20, 10, 20, 10, 10, 20, 20]
max_i = [80, 50, 100, 100, 70, 90, 50]

# Base regimen specs
yb_i = [1, 0, 1, 1, 0, 0, 1]
xb_i = [20, 0, 30, 15, 0, 0, 35]

# Cost of deviating from the base regimen
fb_i = [25, 50, 10, 25, 20, 30, 40]
ub_i = [1, 2, 1, 3, 2, 1, 1]

# Create a Model instance
model = gp.Model()

# Decision variables - from i = 1 to 7 inclusive
a = []
c = []
x = []
y = []
inc = []
dec = []
for i in range(1, 8):
```

```
    # Add each indexed decision variable one by one
    a.append(model.addVar(vtype=GRB.BINARY, name="a" + str(i)))
    inc.append(model.addVar(vtype=GRB.BINARY, name="inc" + str(i)))
    dec.append(model.addVar(vtype=GRB.BINARY, name="dec" + str(i)))
    c.append(model.addVar(vtype=GRB.CONTINIOUS, lb=0, name="c" + str(i)))
    x.append(model.addVar(vtype=GRB.CONTINIOUS, lb=0, name="x" + str(i)))
    y.append(model.addVar(vtype=GRB.BINARY, name="y" + str(i)))

# Set the objective
model.setObjective(quicksum(fb_i[i]*a[i] + ub_i[i]*c[i] for i in
range(0, 7)), GRB.MINIMIZE)

# Add the quality of life constraint
model.addConstr(q(p, y, x) >= Q_36, name="quality_of_life")

# Add the remaining constraints
for i in range(0, 7):
    if yb_i[i] == 0:
        model.addConstr(a[i] == y[i], name="added_drug_" + str(i+1))
    if yb_i[i] == 1:
        model.addConstr(a[i] == 1 - y[i], name="removed_drug_" +
str(i+1))

    """
        inc[i] NAND dec[i] = 1
    """
    model.addConstr(inc[i] + dec[i] <= 1, name="if_nand_dec_" + str(i+1))

    """
    x[i] = base_regimen + change
```

```
    if y[i] = 0, then
        x[i] = 0 (next constraint)
        inc[i] = 0
        c[i] = xb_i[i]
        if xb_i != 0, then
            dec[i] = 1
        if xb_i = 0, then
            dec[i] = 0

    if y[i] = 1, then
        min[i]           <= x[i]                     <= max[i] (next
constraint)
        min[i] - xb_i[i] <= c[i] * (inc[i] - dec[i]) <= max[i] - xb_i[i]
        if xb_i = 0, then
            (inc[i], dec[i]) = (1, 0)
        if xb_i != 0, then
            (inc[i], dec[i]) = (0, 0), (0, 1) or (1, 0)
    """
    model.addConstr(x[i] - xb_i[i] == (inc[i] - dec[i])*c[i],
name="set_x" + str(i+1))

    """
    Dosage bounds, if y = 0 the interval is [0, 0] = 0
    """
    model.addConstr(x[i] >= min_i[i] * y[i], name="min_x" + str(i+1))
    model.addConstr(x[i] <= max_i[i] * y[i], name="max_x" + str(i+1))
    model.update()

# Solve the model
```

```
model.write("partb.lp")

model.optimize()

model.printAttr('X')

model.update()

print("Decision Variables: ")

for i in range(1, 8):

   print("x" + str(i) + " = " + str(model.getVarByName("x" +
str(i)).getAttr('X')))

   print("y" + str(i) + " = " + str(model.getVarByName("y" +
str(i)).getAttr('X')) + "\n")

print("Quality Of Life = " + str(checkQ(p, model)))

print("Deviation Cost, a.k.a. objective  = " +
str(model.getObjective().getValue()))
```

4.3. Part C

```
"""

IE 400 Project - Part C

"""

import gurobipy as gp

from gurobipy import *

import pandas as pd



def checkQ(p_in, my_model):

   y_ = []

   x_ = []

   for i in range(0, 7):
```

```
        x_.append((my_model.getVarByName("x" + str(i + 1)).getAttr('X')))
        y_.append((my_model.getVarByName("y" + str(i + 1)).getAttr('X')))
    return q(p_in, y_, x_)



def q(p_in, y_in, x_in):
    p_sum = -5*p_in[0] - 0.5*p_in[1] - 12*p_in[2] - 8*p_in[3] - 5*p_in[4]
- 5*p_in[5] - p_in[6] - 3*p_in[7] - 2*p_in[8]
    y_sum = -5*y_in[0] - 6*y_in[1] - 4*y_in[2] - 4*y_in[3] - 8*y_in[4] -
6*y_in[5] - 7*y_in[6]
    x_sum = 0.28*x_in[0] + 0.3*x_in[1] + 0.25*x_in[2] + 0.17*x_in[3] +
0.31*x_in[4] + 0.246*x_in[5] + 0.4*x_in[6]
    return p_sum + y_sum + x_sum



# Read patient data from .xlsx file
patient_data = pd.read_excel('patient_data.xlsx', sheet_name="Sheet1")

# Read the group number 36
patient_36 = patient_data.iloc[36, :]

# vector of features
p = list(patient_36[1:10])

# Q threshold
Q_36 = float(patient_36[10])

print("Group Number: " + str(patient_36[0]))
print("Patient vector: " + str(p))
print("Q threshold: " + str(Q_36))
```

```python
# Maximum/Minimum allowed dosages
min_i = [20, 10, 20, 10, 10, 20, 20]
max_i = [80, 50, 100, 100, 70, 90, 50]

# Base regimen specs
yb_i = [1, 0, 1, 1, 0, 0, 1]
xb_i = [20, 0, 30, 15, 0, 0, 35]

# Cost of deviating from the base regimen
fb_i = [25, 50, 10, 25, 20, 30, 40]
ub_i = [1, 2, 1, 3, 2, 1, 1]

# Create a Model instance
model = gp.Model()

# Decision variables - from i = 1 to 7 inclusive
a = []
c = []
x = []
y = []
inc = []
dec = []
M = 1000
for i in range(1, 8):
   # Add each indexed decision variable one by one
   a.append(model.addVar(vtype=GRB.BINARY, name="a" + str(i)))
   inc.append(model.addVar(vtype=GRB.BINARY, name="inc" + str(i)))
   dec.append(model.addVar(vtype=GRB.BINARY, name="dec" + str(i)))
   c.append(model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="c" + str(i)))
```

```
    x.append(model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="x" + str(i)))
    y.append(model.addVar(vtype=GRB.BINARY, name="y" + str(i)))

w = model.addVar(vtype=GRB.BINARY, name="w")

# Set the objective
model.setObjective(quicksum(fb_i[i] * a[i] + ub_i[i] * c[i] for i in
range(0, 7)), GRB.MINIMIZE)

# Add the quality of life constraint
model.addConstr(q(p, y, x), GRB.GREATER_EQUAL, Q_36,
name="quality_of_life")

# Add the remaining constraints
for i in range(0, 7):
    if yb_i[i] == 0:
        model.addConstr(a[i] == y[i], name="added_drug_" + str(i + 1))
    if yb_i[i] == 1:
        model.addConstr(a[i] == 1 - y[i], name="removed_drug_" + str(i +
1))

    """
        inc[i] NAND dec[i] = 1
    """
    model.addConstr(inc[i] + dec[i] <= 1, name="if_nand_dec_" + str(i +
1))

    """
    x[i] = base_regimen + change
```

```
    if y[i] = 0, then
        x[i] = 0 (next constraint)
        inc[i] = 0
        c[i] = xb_i[i]
        if xb_i != 0, then
            dec[i] = 1
        if xb_i = 0, then
            dec[i] = 0

    if y[i] = 1, then
        min[i]           <= x[i]                      <= max[i] (next
constraint)
        min[i] - xb_i[i] <= c[i] * (inc[i] - dec[i]) <= max[i] - xb_i[i]
        if xb_i = 0, then
            (inc[i], dec[i]) = (1, 0)
        if xb_i != 0, then
            (inc[i], dec[i]) = (0, 0), (0, 1) or (1, 0)
    """
    model.addConstr(x[i] - xb_i[i] == (inc[i] - dec[i]) * c[i],
name="set_x" + str(i + 1))
    """
    Dosage bounds, if y = 0 the interval is [0, 0] = 0
    """
    model.addConstr(x[i] >= min_i[i] * y[i], name="min_x" + str(i + 1))
    model.addConstr(x[i] <= max_i[i] * y[i], name="max_x" + str(i + 1))
    model.update()

model.addConstr(x[0] + x[1] >= 50*y[0]*y[1], name="gt_50")
model.addConstr(x[0] + x[1] <= 70*y[0]*y[1] + M*(1-y[0]*y[1]),
name="lt_70")
```

```
model.addConstr((1 - y[4])*x[2] <= 25, name="lt_25")
model.addConstr(w, GRB.EQUAL, y[3]*y[5], name="cnt_3_1")
model.addConstr(w * (y[4] + y[6] - 1) >= 0, name="cnt_3_2")

# Solve the model
model.write("partc.lp")
model.optimize()
model.printAttr('X')
model.update()
print("Decision Variables, x and y: ")
for i in range(1, 8):
   print("x" + str(i) + " = " + str(model.getVarByName("x" +
str(i)).getAttr('X')))
   print("y" + str(i) + " = " + str(model.getVarByName("y" +
str(i)).getAttr('X')) + "\n")
print("Quality Of Life = " + str(checkQ(p, model)))
print("Deviation Cost, a.k.a. objective  = " +
str(model.getObjective().getValue()))
```