Math 482 Workshop

Gurobi: Solving Linear Programs Introduction

Instructions. Write code in C++/Python to solve each of the following problems. For questions that require written explanation, use print statements. Upload an image of your terminal output to Canvas by the due date.

I. Consider the primal linear program shown below.

$$\begin{aligned} \text{maximize} \quad & z = 22x_1 + 31x_2 + 29x_3 \\ \text{subject to} \quad & x_1 + 4x_2 + 6x_3 \leq 73, \\ & 5x_1 - 2x_2 + 3x_3 \leq 68, \\ & x_j \geq 0, \ \forall j \in \{1, 2, 3\} \end{aligned}$$

We use Gurobi to solve this model. The Python code is shown in Listing 1 and the C++ code is shown in Listing 2.

(a) Write and run the Python or C++ code for Model I.

(b) You can identify the dual multipliers for each constraint using the Gurobi constraint variables. Add lines to print out the dual multipliers corresponding to each constraint. What are the active constraints corresponding to the optimal primal solution?

(c) Use the dual multipliears to identify the corresponding dual objective value.

(d) Show that the primal and dual values are the same.

(e) You can identify the slack values for each each constraint using the Gurobi constraint variables. Add lines to print out the slack corresponding to each constraint. Which constraints are tight? Which constraints have positive slack?

(f) In the final simplex tableau, the values in the objective row corresponding to primal variables indicate the dual slack values. These values are referenced as reduced costs. You can identify the reduced cost values using the Gurobi variables. Add lines to print out these values.

(g) Use the primal values, dual multipliers, primal slack values , and reduced cost values to show that the complementary slackness conditions hold.

II. Consider the primal linear program shown below.

$$\begin{aligned} \text{maximize} \quad & z = 5x_1 + 4x_2 + 3x_3 + 2x_4 + 1x_5 + 0.5x_6 \\ \text{subject to} \quad & -x_1 + x_2 + x_3 \leq -2, \\ & x_1 + x_2 + 2x_4 \leq 8, \\ & 2x_1 - x_2 + x_5 \leq 4, \\ & x_3 + x_4 + x_5 + x_6 \leq 10, \\ & x_1 + x_2 + x_3 + x_4 \leq 12, \\ & 3x_1 + 3x_2 + 6x_4 \leq 24, \\ & x_j \geq 0, \ \forall j \in \{1, 2, 3, 4, 5, 6\} \end{aligned}$$

(a) Write and run Python or C++ code to solve Model II.

(b) Record changes to the number of iterations, objective value, primal infeasibility, dual infeasibility, and time to solve when changing Gurobi Model parameters Presolve from 0 to 1 and Method from 0 to 1.

III. Consider the linear program shon below.

$$\begin{aligned} \text{maximize} \quad & z = \sum_{i \neq j} a_{i,j} x_{i,j} \\ \text{subject to} \quad & x_{i,j} + x_{j,i} = 1, \ \forall i < j, \\ & x_{i,j} + x_{j,k} + x_{k,i} \leq 2, \ \forall i < j, \ i < k, \ j \neq k, \\ & x_{i,j} \in \{0, 1\}, \ \forall i \neq j \end{aligned}$$

(a) Write and run Python or C++ code to solve Method III with the given matrix below

(b) Record changes to the number of iterations, objective value, primal infeasibility, dual infeasibility, and time to solve when changing Gurobi Model parameters Presolve from 0 to 1 and Method from 0 to 1.

$$A = \begin{bmatrix} 0 & 5 & 4.5 & 3.5 & 2 & 3.5 & 1.5 & 2.5 & 3.5 & 0.5 \\ 0 & 0 & 3 & 3 & 1.5 & 2 & 1 & 0 & 2 & 0 \\ 0.5 & 2 & 0 & 2.5 & 1 & 2.5 & 1 & 0.5 & 1.5 & 0 \\ 1.5 & 2 & 2.5 & 0 & 1 & 2 & 1 & 1 & 2 & 0 \\ 3 & 3.5 & 4 & 4 & 0 & 4 & 2 & 2.5 & 3.5 & 1 \\ 1.5 & 3 & 2.5 & 3 & 1 & 0 & 1 & 1.5 & 3 & 0 \\ 3.5 & 4 & 4 & 4 & 3 & 4 & 0 & 3.5 & 3.5 & 2 \\ 2.5 & 5 & 4.5 & 4 & 2.5 & 3.5 & 1.5 & 0 & 4.5 & 0 \\ 1.5 & 3 & 3.5 & 3 & 1.5 & 2 & 1.5 & 0.5 & 0 & 0 \\ 4.5 & 5 & 5 & 5 & 4 & 5 & 3 & 5 & 5 & 0 \end{bmatrix}.$$

```python
from gurobipy import Model, GRB

# 1) Create a model
m = Model("ModelI")

# (Optional) Make the run more "textbook simplex-like"
m.Params.Presolve = 0
m.Params.ScaleFlag = 0
m.Params.Method = 0 # primal simplex (1=dual simplex also fine)

# 2) Add decision variables (x >= 0)
x1 = m.addVar(lb=0, ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name="x1")
x2 = m.addVar(lb=0, ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name="x2")
x3 = m.addVar(lb=0, ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name="x3")

# 3) Objective: max c^T x
m.setObjective(22*x1 + 31*x2 + 29*x3, GRB.MAXIMIZE)

# 4) Constraints: A x <= b
c1 = m.addConstr(x1 + 4*x2 + 6*x3 <= 73, name="c1")
c2 = m.addConstr(5*x1 - 2*x2 + 3*x3 <= 68, name="c2")
```

```python

# 5) Solve
m.optimize()

# 6) Primal solution
print("x1 =", x1.X)
print("x2 =", x2.X)
print("x3 =", x3.X)
print("z =", m.ObjVal)
```

Listing 1: Model I in Python

```cpp
#include "gurobi_c++.h"
#include <iostream>
using namespace std;
/*
  Gurobi I Model I
    compile instructions: g++ -std=c++17 -O3 -march=native -mtune=native -o
        gurobi1_model1 model1.cpp -I/Library/gurobi1103/macos_universal2/include/
        -L/Library/gurobi1103/macos_universal2/lib/ -lgurobi_c++ -lgurobi110
*/
int main(){
    // 1) Environment: manages licensing and global settings.
    // The 'true' argument prevents starting immediately (lets us set
    // params first if needed).
    GRBEnv* env = new GRBEnv(true);
    env->start(); // activates environment; checks license, etc.

    // 2) Model: container for variables, constraints, objective.
    GRBModel* model = new GRBModel(*env);
    model->set(GRB_IntParam_Presolve, 0);
    model->set(GRB_IntParam_ScaleFlag, 0);
    model->set(GRB_IntParam_Method, 0);
    // 3) Variables
    // addVar(lb, ub, objCoeff, varType, name)
    // We set objCoeff=0 because we will set the objective separately.
    GRBVar x1 = model->addVar(0.0, GRB_INFINITY, 0.0, GRB_CONTINUOUS, "x1");
    GRBVar x2 = model->addVar(0.0, GRB_INFINITY, 0.0, GRB_CONTINUOUS, "x2");
    GRBVar x3 = model->addVar(0.0, GRB_INFINITY, 0.0, GRB_CONTINUOUS, "x3");

    // 4) Objective
    model->setObjective(22.0*x1 + 31.0*x2 + 29.0*x3, GRB_MAXIMIZE);

    // 5) Constraints
    GRBConstr c1 = model->addConstr(x1 + 4.0*x2 + 6.0*x3, GRB_LESS_EQUAL, 73, "c1");
    GRBConstr c2 = model->addConstr(5.0*x1 - 2.0*x2 + 3.0*x3, GRB_LESS_EQUAL, 68.0,
        "c2");

    // 6) Optimize
    model->optimize();

    // 7) Read results
    cout << "x1 = " << x1.get(GRB_DoubleAttr_X) << std::endl;
    cout << "x2 = " << x2.get(GRB_DoubleAttr_X) << std::endl;
```

```
    cout << "z = " << model->get(GRB_DoubleAttr_ObjVal) << std::endl;

    return 0;
}
```

Listing 2: Model I in C++