Gurobi Seminar 2
The Inventory Problem

Demetrios Papazaharias

UB INFORMS Student Chapter
Department of Industrial & Systems Engineering
University at Buffalo, SUNY

September 25, 2019

Welcome!

- Welcome to the second installment of our Gurobi series!
- Materials for this workshop can be found here:

  `github.com/Dpapazaharias1/UB-INFORMS-Gurobi-Seminar`

- In this workshop we will cover the classic inventory problem.
- We will first begin where we left off from the first workshop.

Contents

1. Mathematical Modeling

2. Basic Syntax

3. Inventory Problem

Introduction to Mathematical Modeling

- Mathematical models are idealized representations of a system using mathematical concepts and language
- Every mathematical model has four major components
    1. Decision Variables
    2. Constraints
    3. Objective Function
    4. Parameters
- General form of model

$$\begin{aligned} \max \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g(\mathbf{x}) \leq b \\ & \mathbf{x} \in X \end{aligned}$$

Introduction to Mathematical Modeling

- The **decision variables** are values which we control in a mathematical model , *e.g. how much of product X to produce, build facility at location A or not, etc.*
- The set of **constraints** represent restrictions of our decisions variables, *e.g. spending budget, resource scarcity, etc.*
- The **objective function** is a measure of performance based off these decisions, *e.g. profit, total path length, distance traveled*
- The **parameters** are constants in the objective function and constraints related to the decision variables, *e.g. profit per unit, cost of building facility at location A, distance between cities, etc.*

Contents

1. Mathematical Modeling

2. Basic Syntax

3. Inventory Problem

Overview

To use Gurobi with Python:

```
from gurobipy import *
```

Objects and methods to remember:

- Model
  - Model.addVar(lb, ub, obj, vtype, name)
  - Model.addConstr(constraint, name)
  - Model.update()
  - Model.modelSense or Model.SetObjective()
  - optimize()
- Var
  - Var.X $\leftarrow$ Get value of variable in current solution
  - Var.RC $\leftarrow$ Get the reduced cost of variable
- Constr
  - Constr.Pi $\leftarrow$ Dual value
- LinExpr
  - LinExpr.getValue()
- quicksum, multidict, tuplelist

Building a model

$$\boxed{\begin{array}{ll} \min & 0 \\ \text{s.t.} & \end{array}}$$

```
# Create a new model
m = Model("mip1")
```

Building a model

$$\begin{array}{ll} \text{min} & 0 \\ \text{s.t.} & x, y, z \geq 0 \end{array}$$

```
# Create a new model
m = Model("mip1")


# Create variables
x = m.addVar(vtype=GRB.CONTINUOUS, name="x")
y = m.addVar(vtype=GRB.CONTINUOUS, name="y")
z = m.addVar(vtype=GRB.CONTINUOUS, name="z")

# Integrate new variables
m.update()
```

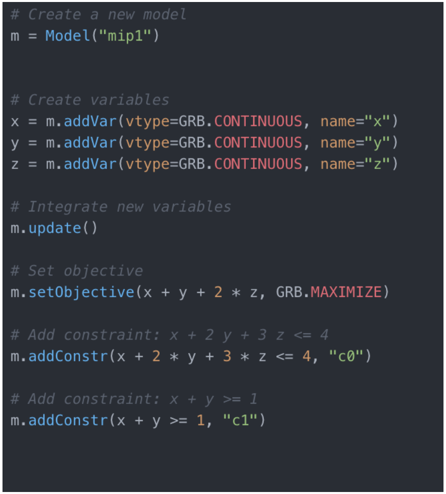
Building a model

$$\begin{aligned} \text{max} \quad & x + y + 2z \\ \text{s.t.} \quad & x, y, z \geq 0 \end{aligned}$$

```
# Create a new model
m = Model("mip1")


# Create variables
x = m.addVar(vtype=GRB.CONTINUOUS, name="x")
y = m.addVar(vtype=GRB.CONTINUOUS, name="y")
z = m.addVar(vtype=GRB.CONTINUOUS, name="z")

# Integrate new variables
m.update()

# Set objective
m.setObjective(x + y + 2 * z, GRB.MAXIMIZE)
```

Building a model

$$
\begin{array}{ll}
\max & x + y + 2z \\
\text{s.t.} & x + 2y + 3z \leq 4 \\
 & x + y \geq 1 \\
 & x, y, z \geq 0
\end{array}
$$

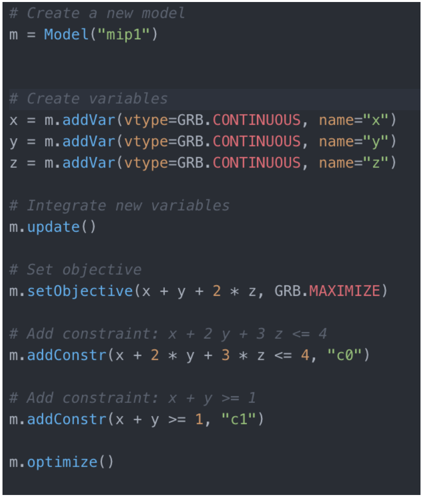


```
# Create a new model
m = Model("mip1")


# Create variables
x = m.addVar(vtype=GRB.CONTINUOUS, name="x")
y = m.addVar(vtype=GRB.CONTINUOUS, name="y")
z = m.addVar(vtype=GRB.CONTINUOUS, name="z")

# Integrate new variables
m.update()

# Set objective
m.setObjective(x + y + 2 * z, GRB.MAXIMIZE)

# Add constraint: x + 2 y + 3 z <= 4
m.addConstr(x + 2 * y + 3 * z <= 4, "c0")

# Add constraint: x + y >= 1
m.addConstr(x + y >= 1, "c1")
```

Building a model

$$\begin{array}{ll} \max & x + y + 2z \\ \text{s.t.} & x + 2y + 3z \leq 4 \\ & x + y \geq 1 \\ & x, y, z \geq 0 \end{array}$$

Solution: $(x, y, z) = (4, 0, 0)$



```
# Create a new model
m = Model("mip1")


# Create variables
x = m.addVar(vtype=GRB.CONTINUOUS, name="x")
y = m.addVar(vtype=GRB.CONTINUOUS, name="y")
z = m.addVar(vtype=GRB.CONTINUOUS, name="z")

# Integrate new variables
m.update()

# Set objective
m.setObjective(x + y + 2 * z, GRB.MAXIMIZE)

# Add constraint: x + 2 y + 3 z <= 4
m.addConstr(x + 2 * y + 3 * z <= 4, "c0")

# Add constraint: x + y >= 1
m.addConstr(x + y >= 1, "c1")

m.optimize()
```

Contents

1. Mathematical Modeling

2. Basic Syntax

3. Inventory Problem

Inventory Problems

- The inventory problem we are concerned with is a special type of planning problem
- The decision level of these problems often covers a time horizon of one or several months, broken into week long segments.
- These models provide a **macroscopic** view of the production system; ignoring the detailed constraints for workshops and machines.
- The objective is to determine the best quantities to produce each time period in order to minimize the cost incurred through production and storage.

Example: Drinking Glasses

The main activity of a company in northern France is the production of drinking glasses. It currently sells two different types of glasses (wine and martini) that are produced in batches of 10. The company wishes to plan its production for the next 4 weeks.

- The demand of wine and martini glasses is given below

Glass	Week 1	Week 2	Week 3	Week 4
Wine	1000	500	300	4000
Martini	2500	200	800	350

- The initial stock of wine and martini glasses is 250 and 100, respectively.
- The required final stock of wine and martini glasses is required to be 1000 and 600, respectively.

Example: Drinking Glasses

- The production and storage costs per batch of wine glasses are \$15 and \$8, and \$20 and \$12 per batch of martini glasses.
- The required processing time for workers and machines (in hours) to produce a batch of wine and martini glasses are 2 and 5, and 1.5 and 3, respectively.
- The number of working hours of personnel is limited to 1000 hours per week, and the machines have a weekly capacity of 1400 hours.
- Each batch of glasses, independent of type, requires 20 in^3 of storage
- The available storage space is 6000 in^3.

Example: Objective

With this information, we would like to answer the following question:

- Which quantities of the different glass types need to be produced in every period to minimize the total cost of production and storage?
- We will use Gurobi to answer that question!
- Start a new Jupyter session and open Inventory.ipynb