GUROBI
OPTIMIZATION

Python II:
Advanced Algebraic Modeling with Python and Gurobi

Background

- This is the second in our series on optimization modeling in Python
  - You *can* build models in other popular programming languages
  - This series focuses on special features for Python that help for optimization models

- Since this is an “advanced” presentation, we assume you are familiar with:
  - Python
  - Optimization modeling

- If this is new to you, check the Resources section on www.gurobi.com
  - View the recorded presentation “Python I: Introduction to modeling with Python”

Agenda

- Get started using a simple example of workforce scheduling
- Understand general principles of optimization modeling
- Learn powerful modeling features in Python and the Gurobi interface

Getting started

Workforce scheduling example

Optimization model principles: Decisions

GUROBI OPTIMIZATION

- Decisions: What to do
  - Optimization variables
- Examples
  - Workforce: do you assign a worker to a shift
  - Others
    - Select
    - Assign
    - Make
    - Go
    - Buy / Sell
- Types of decisions
  - Yes / No
  - Quantities – how much to do

Optimization model principles: Quantities

- Quantities: Numerical values used in relationships
  - Coefficients in expressions – linear, quadratic, logical, etc.

- Examples
  - Workforce: cost for a worker in a shift
  - Others
    - Cost, price
    - Distance, space
    - Time
    - Materials required or used
    - Supply, demand
    - Capacity

Optimization model principles: Constraints

- Constraints: Requirements or limits
  - Equality or inequality relationships

- Examples
  - Workforce: assign enough workers to meet the requirements for each shift
  - Others
    - Demand
    - Equipment
    - Supplies
    - Space
    - Workers
    - Money
    - Time

Optimization model principles: Objective

GUROBI OPTIMIZATION

- Objective: Goal to achieve
  - Mathematical function to optimize

- Examples
  - Workforce
    - Initial: Minimize labor cost
    - Later: Minimize shortfall in shifts
  - Others
    - Maximize: profit, utilization, balance
    - Minimize: costs, time, waste

Optimization model principles: Indexes

- Index: Specifies a key value
  - Identifies an element of an array
- Examples
  - Workforce
    - Workers
    - Shifts
    - Availability: a list that specifies a shift when a worker is available
  - Others
    - Product models
    - Workers
    - Equipment – machines, trucks, …
    - Locations
    - Customers or suppliers
    - Supplies
    - Time – hour, day, week, month, quarter, year

Indexes are key to optimization models

- Describe individual variables, numerical quantities, constraints

- Use of indexes
  - Iterate over variables or constraints – for-all statements
  - Compute sums over variables and numerical quantities

- An efficient model depends on proper handling of indexes
  - For an efficient model, setup should require much less time and memory than solving

Python lists

- A list is a built-in data type that
  - Can store items – int, float, string, tuples (pairs, …), other objects, ...
  - Is ordered
  - Efficiently supports iteration and random access
  - Can be modified – append, delete, join, …
- Lists are a good way to create optimization indexes
  - Note: Unlike a mathematical set, lists can contain repeated elements

Python list comprehension

- Set-builder notation for creating a list
  - From another iterator

```
Squares = [i**2 for i in range(6)] # [0,1,4,9,16,25]
```

  - From multiple iterators

```
Pairs = [(i,j) for j in range(4) for i in range(j)]
        # [(0,1), (0,2), (1,2), (0,3), (1,3), (2,3)]
```

  - With a filter (if condition)

```
Nonsquares = [i for i in range(10) if i not in Squares]
        # [2,3,5,6,7,8]
```

- List comprehension syntax is similar to optimization notation
  - You can use for-loops, but they are typically more wordy

```
Pairs = []
for j in range(4):
  for i in range(j):
    Pairs.append((i,j))
```

Python generator expressions

- Similar syntax as list comprehension, without the intermediate list
  - List comprehension

```
Squares = [i**2 for i in range(6)] # [0,1,4,9,16,25]
```

  - Generator expression

```
SumSquares = sum(i**2 for i in range(6)) # 55
```

- The Gurobi interface takes advantage of lists, list comprehension and generator expressions
  - More details in a moment

Gurobi tuplelist

- A Gurobi class to store lists of tuples

```
Cities = ['A','B','C','D']
Routes = tuplelist([('A','B'), ('A','C'), ('B','C'), ('B','D'), ('C','D')])
```

- What makes it special: select statement for efficient filtering

```
for c in Cities:                          # [('A','B'), ('A','C')]
  print(Routes.select(c,'*'))             # [('B','C'), ('B','D')]
                                          # [('C','D')]
                                          # []
```

- The tuplelist is indexed to make `select()` efficient

Gurobi tupledict

- A Gurobi dictionary class, where the keys are tuples
  - The list of keys is a tuplelist, so it provides select(), sum() and prod() operators
- Used for decision variables and constraints
  - Examples in a moment

Indexed variables: Model.addVars()

- Using integers

```
x = m.addVars(2, 3, name="x")
# x[0,0], x[0,1], x[0,2], x[1,0], x[1,1], x[1,2]
```

- Using lists of scalars

```
y = m.addVars(Cities, Cities, name="y")
# y[A,A], y[A,B], y[A,C], y[A,D], y[B,A], y[B,B], y[B,C], y[B,D]
# y[C,A], y[C,B], y[C,C], y[C,D], y[D,A], y[D,B], y[D,C], y[D,D]
```

- Using a tuplelist

```
z = m.addVars(Routes, name="z")
# z[A,B], z[A,C], z[B,C], z[B,D], z[C,D]
```

- Using a generator expression (!)

```
w = m.addVars((i for i in range(5) if i != 2), name="w")
# w[0], w[1], w[3], w[4]
```

More about Model.addVars()

- Automatically takes the cross-product of multiple indexes

```
x = m.addVars(2, 3, name="x")
y = m.addVars(Cities, Cities, name="y")
```

- Generates names automatically, using the subscripts
- Requires lists of scalars or lists of tuples

Indexed constraints: Model.addConstrs()

- Uses a generator expression

```
x = m.addVars(Routes, name="x")
y = m.addVars(Routes, name="y")
m.addConstrs((x[i,j]+y[i,j] <= 2 for i,j in Routes), name="capacity")
```

Operators and Python

- Linear and quadratic expressions are used in constraints and the objective
  - Basic (binary) mathematical operators (+ − × ÷)
  - Aggregate sum operator (∑)
    - Used alone as well as in products (dot product)

- Indexes are key to the aggregate sum operator
  - Gurobi Python interface has 2 versions: full and simple
  - Why two?
    - The simple syntax is easier
    - There are some things you cannot do with the simple syntax

Aggregate sum: Full

- Use generator expression inside a quicksum function

```
obj = quicksum(cost[i,j]*x[i,j] for i,j in arcs)
```

- quicksum works just like Python's sum function, but it is more efficient for optimization models

Aggregate sum: Simple

- A tupledict of variables has a sum() function, using the same syntax as tuplelist.select():

```
x = m.addVars(3, 4, vtype=GRB.BINARY, name="x")
m.addConstrs(x.sum(i,'*')<=1 for i in range(3))
```

- This generates the constraints:

```
x[0,0] + x[0,1] + x[0,2] + x[0,3] <= 1
x[1,0] + x[1,1] + x[1,2] + x[1,3] <= 1
x[2,0] + x[2,1] + x[2,2] + x[2,3] <= 1
```

Dot product

- A tupledict of variables has a prod() function to compute the dot product

- If cost is a dictionary, then the following are equivalent:

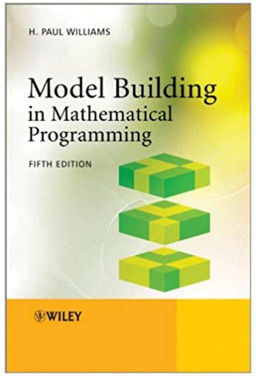
```
obj = quicksum(cost[i,j]*x[i,j] for i,j in arcs)

obj = x.prod(cost)
```

Putting it all together

Network flow example

Resources to learn more

- Code Examples - http://www.gurobi.com/resources/examples/example-models-overview
  - Functional (code) examples
  - Modeling examples
- Book:
  - Model Building in Mathematical Programming by HP Williams



- Gurobi Python documentation - http://www.gurobi.com/documentation/
- Python – www.python.org

Thank you for joining us

- If you haven't already done so, please register at www.gurobi.com, and then visit http://www.gurobi.com/get-anaconda to try Gurobi and Python for yourself.
- The next Python webinar will focus on optimization and heuristics and take place on April 19th and again on April 20th.
    - You can learn more at http://www.gurobi.com/company/events/webinars-python.
- For questions about pricing, please contact sales@gurobi.com or sales@gurobi.de.
- A recording of the webinar, including the slides, will be available in roughly one week.