COMP331/557: Optimisation

Introduction to Gurobi

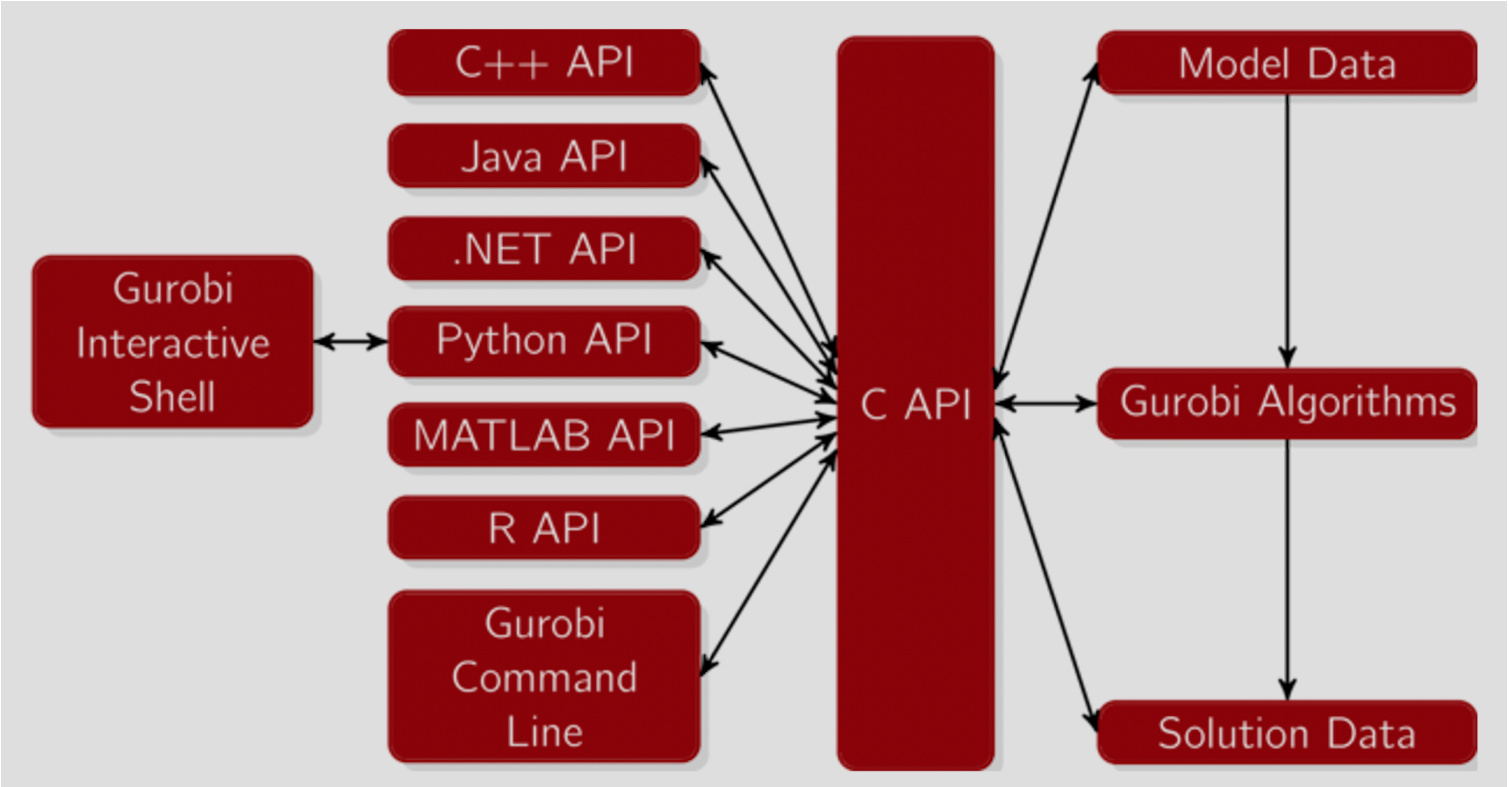
Outline

What is Gurobi?

- State-of-the-art simplex based linear programming (LP) and mixed-integer programming (MIP) solver.
- Built from the ground up to exploit modern multi-core processors.
- Developed by
  - Zonghao Gu
  - Edward Rothberg
  - Robert Bixby

  (the brains behind CPLEX)
- Performance is comparable to CPLEX.
- Many resources available from the Gurobi website (`www.gurobi.com`)
  - Videos, Code Examples, ...
  - Reference Manual (in particular Section 6)
    `http://www.gurobi.com/documentation/8.0/refman.pdf`

Gurobi Language Interfaces



Picture taken from `http://www.gurobi.com`

- We will use the Gurobi Interactive Shell,
  - i.e., a Python shell with Gurobi modules pre-imported

Python: What is it?

- Object oriented scripting language.
- Interpreted rather than being compiled.
- Easy to learn, read, use.
- Code is usually shorter and more readable.
- Open source
- `python.org`: *"Python is a programming language that lets you work quickly and integrate systems more effectively."*

- We will not need much Python.
- If you want to learn more, I recommend this:
  `http://heather.cs.ucdavis.edu/~matloff/Python/PLN/FastLanePython.pdf`

Some Python Examples: Lists

```
>>> x = [5,10,16,300]
>>> x
[5, 10, 16, 300]

>>> x.append(-3)
>>> x
[5, 10, 16, 300, -3]

>>> del x[1]
>>> x
[5, 16, 300, -3]

>>> y=x[2:]
>>> y
[300, -3]

>>> x.insert(2,49)
>>> x
[5, 16, 49, 300, -3]
```

```
>>> x.remove(300)
>>> x
[5, 16, 49, -3]

>>> x.index(16)
1

>>> y=['python','gurobi']
>>> y
['python', 'gurobi']

>>> z=x+y
>>> z
[5, 16, 49, -3, 'python', 'gurobi']

>>> z=2*x
>>> z
[5, 16, 49, -3, 5, 16, 49,-3]
```

In most other programming languages you need much more code to do this.

Solving a Model in Gurobi

lp1.lp

```
Minimize
  2 x + 4 y
Subject To
 R0: x + y >= 3
 R1: 3 x + 2 y >= 14
 R2: 3 x + 2 y <= 14
Bounds
  x <= 10
  y >= 1
End
```

- Start Gurobi by typing `gurobi` in terminal
- Read model file `lp1.lp`

  ```
  m=read('lp1.lp')
  ```

- Optimize model

  ```
  m.optimize()
  ```

- Print solution

  ```
  m.getVars()
  ```

- Write solution to file:

  ```
  m.write('lp1.sol')
  ```

- System call to output `lp1.sol` file:

  ```
  system('cat lp1.sol')
  ```

Short documentation on the supported LP-file format:

`http://www.gurobi.com/documentation/8.0/refman/lp_format.html`

Building a Model from Scratch

Brewery Example

$$
\begin{array}{lrl}
\max & 13\,A + 23\,B & \\
\text{s.t.} & 5\,A + 15\,B & \leq 480 \\
& 4\,A + \ 4\,B & \leq 160 \\
& 35\,A + 20\,B & \leq 1190 \\
& A, B & \geq 0 \\
& A, B \text{ integer} &
\end{array}
$$

- Call the model constructor

```
m=Model('brewery')
```

- Add variables

```
A=m.addVar(vtype=GRB.INTEGER, name="Ale")
B=m.addVar(vtype=GRB.INTEGER, name="Beer")
m.update()
```

- Add objective function and contraints

```
m.setObjective(13*A + 23*B, GRB.MAXIMIZE)
c1=m.addConstr(5*A + 15*B <= 480)
c2=m.addConstr(4*A + 4*B <= 160)
c3=m.addConstr(35*A + 20*B <= 1190)
```

- Optimize and Output Solution

```
m.optimize()
m.getVars()
```

Solving a Model in a Python script

brewery.py

```
# import gurobi libraries
from gurobipy import *

m=Model('brewery')

# Add Variables
A=m.addVar(vtype=GRB.INTEGER, name="Ale")
B=m.addVar(vtype=GRB.INTEGER, name="Beer")
m.update()

# Add Constraints
c1=m.addConstr(4*A + 4*B <= 160)
c2=m.addConstr(35*A + 20*B <= 1190)
c3=m.addConstr(5*A + 15*B <= 480)

# Add Objective Function
m.setObjective(13*A + 23*B, GRB.MAXIMIZE)

# Optimize Model
m.optimize()

# Output formatted solution
for v in m.getVars():
    print v.varName, v.x
print 'Obj:', m.objVal
```

- Run this by calling

  `python brewery.py`

  from the shell