GUROBI OPTIMIZATION PYTHON API TUTORIAL

Version 9.5.0

This is a follow-up to the Cplex documentation. Instead of individually going over methods, this tutorial will cover the differences by providing the formulation of the previous examples given using the gurobi python API. Installation and license download instructions are also provided. For a comprehensive documentation, please refer to gurobi's website. There is also a page for a list of Gurobi examples.

Table of Contents

Download/Installation Instructions

Firstly, gurobi could be downloaded after accepting the end user agreement from here. Version 9.5.0 was just recently released (02.12.2021), hence for now obtaining 9.1.2 would suffice as well.

Gurobi Optimizer – Get the Software

Gurobi Optimizer

Gurobi Optimizer is the Gurobi optimization libraries. In addition to the software, the corresponding README file contains installation instructions. Here is the list of bug fixes for each release.

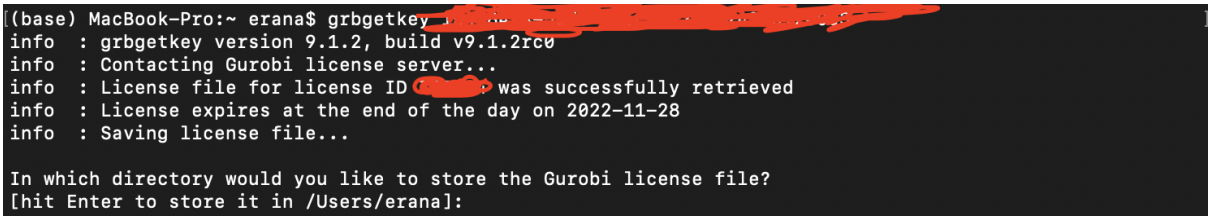
Current version		64-bit Windows	64-bit Linux	64-bit macOS Universal2	64-bit AIX
9.5.0	Read Me Release Notes	Gurobi-9.5.0-win64.msi	gurobi9.5.0_linux64.tar.gz	gurobi9.5.0_macos_universal2.pkg	gurobi9.5.0_power64.tar.gz
md5 Checksum		37f8ab8cee8fb9106920170cd3a30562	03d90b6dbe5dd0e9634299c4101bcc93	8e117b03bf0dbd88f72a24241a5a7d81	4a9f8267031463d4f1b72ad07a921add
Old versions		**64-bit Windows**	**64-bit Linux**	**64-bit macOS**	**64-bit macOS Universal2 - Experimental***
9.1.2	Read Me	Gurobi-9.1.2-win64.msi	gurobi9.1.2_linux64.tar.gz	gurobi9.1.2_mac64.pkg	gurobi9.1.2_macos_universal2.pkg
md5 Checksum		5021cc7ada7f14a0567823d24a0fc206	022f454faaa37207b3f2526a1abf928f	c1bdf37dbb1ab6da43c058cbdcd1e387	52c5047ad708d1eae4f5add4c1c0c63e
9.0.3	Read Me	Gurobi-9.0.3-win64.msi	gurobi9.0.3_linux64.tar.gz	gurobi9.0.3_mac64.pkg	
md5 Checksum		5394eff3d8f5d8c16190f9ea5bc70020	832040cce622ba7f267e26645fcd200d	758713ea51b0981928f85d9bd81e6b27	
8.1.1	Read Me	Gurobi-8.1.1-win64.msi	gurobi8.1.1_linux64.tar.gz	gurobi8.1.1_mac64.pkg	

To use gurobi, first you must register to the website using your institution issued email address and provide your credentials. After registering, you can request an academic license.

After getting a license and downloading anaconda (a conda environment is needed for Gurobi), three steps are followed:

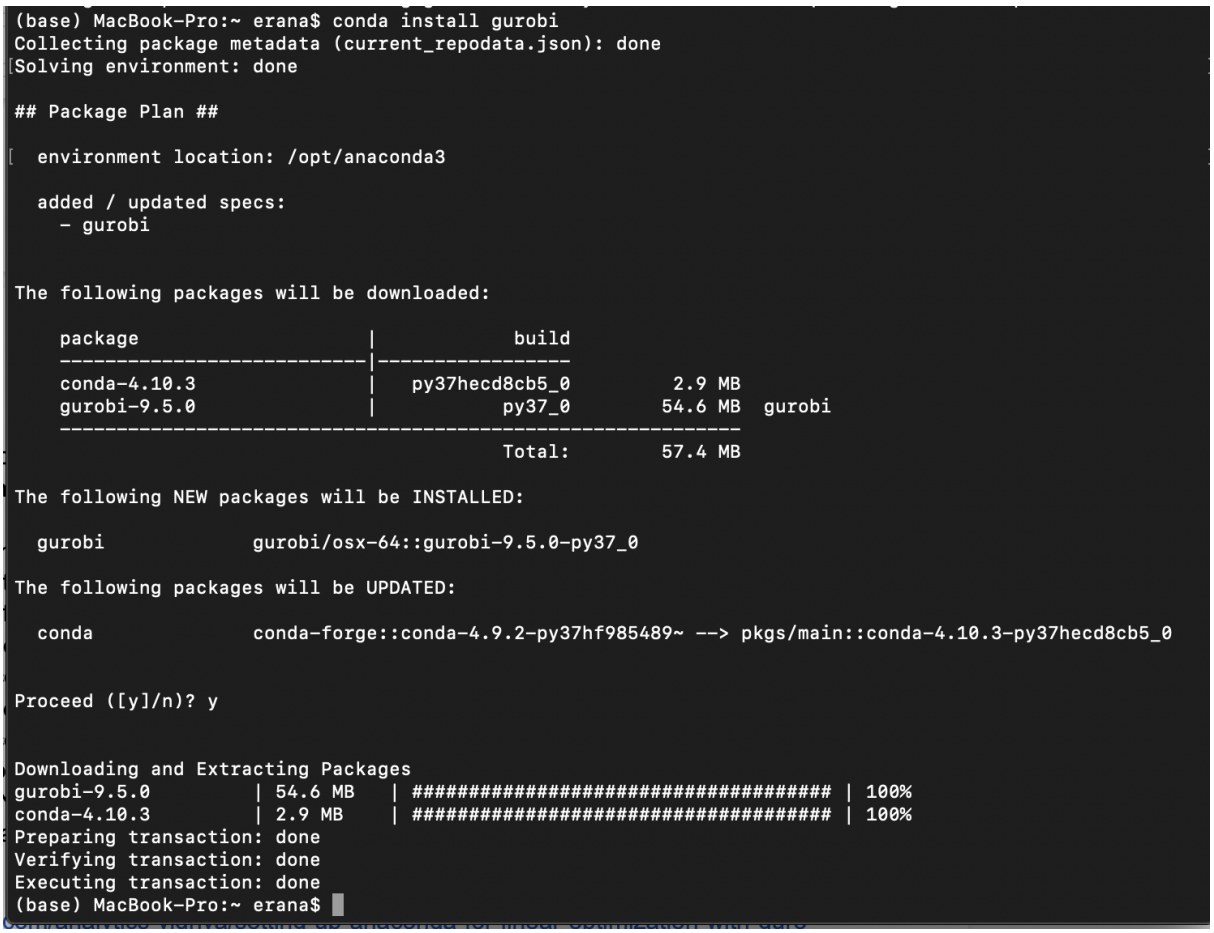
1. enter grbgetkey <Licence Key> to your computer's terminal
   an example would be→ grbgetkey ae36ac20-16e6-acd2-f242-4da6e765fa0a

You can find the key here.



During this step, you must be connected to Bilkent's VPN.
A prompt will ask you about where to store the downloaded license file. There are no specifications about where to store the license, hence it is up to the user.

2. Paste the following to the terminal directly
   conda config --add channels http://conda.anaconda.org/gurobi
3. Paste the following to the terminal directly
   conda install gurobi



Now you are done! You can access gurobi from any Conda base environment. Spyder and/or Jupyter Notebooks would be fine as well as Visual Studio if you could create a virtual Conda base to interpret the code from.

Examples:

To get an idea of the differences between gurobi and Cplex Python APIs, the examples given in the Cplex Tutorial will be used here as well:

LP

Consider the following linear programming problem:

$$\begin{aligned} \max\ & 2x_1 + x_2 + 6x_3 - 4x_4 \\ \text{s.t.}\ & x_1 + 2x_2 + 4x_3 - x_4 \leq 6 \\ & 2x_1 + 3x_2 - x_3 + x_4 \leq 12 \\ & x_1 + x_3 + x_4 \leq 6 \\ & x_i \geq 0\ i = 1, 2, 3, 4 \end{aligned}$$

Here's an example code that can find the optimal solution and a sample output:

```
import gurobipy as gp
from gurobipy import GRB

# Create a Model instance
m = gp.Model()
# Add variables
X1 = m.addVar(vtype=GRB.CONTINUOUS, name="X1")
X2 = m.addVar(vtype=GRB.CONTINUOUS, name="X2")
X3 = m.addVar(vtype=GRB.CONTINUOUS, name="X3")
X4 = m.addVar(vtype=GRB.CONTINUOUS, name="X4")
# Providing the coefficients and the sense of the objective function
m.setObjective(2*X1 + X2 + 6*X3 - 4 * X4, GRB.MAXIMIZE)

# Adding the 3 constraints
c1 = m.addConstr(X1 + 2*X2 + 4*X3 - X4 <= 6)
c2 = m.addConstr(2*X1 + 3*X2 - 1*X3 + X4 <= 12)
c3 = m.addConstr(X1 + X3 + X4 <= 6)

# Solving the model
m.optimize()
m.printAttr('X') #This prints the non-zero solutions found
```

As expected, we have the same optimal value of 12 and the decision variables take the values $X_1 = 6, X_2 = 0, X_3 = 0, X_4 = 0$ at optimality (Only non-zero solutions are shown):

```
(base) MacBook-Pro:Desktop erana$ /opt/anaconda3/bin/python /Users/erana/Desktop/LP_Example_Gurobi.py
Set parameter Username
Academic license - for non-commercial use only - expires 2022-11-28
Gurobi Optimizer version 9.5.0 build v9.5.0rc5 (mac64[x86])
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
Optimize a model with 3 rows, 4 columns and 11 nonzeros
Model fingerprint: 0x92a93d38
Coefficient statistics:
  Matrix range     [1e+00, 4e+00]
  Objective range  [1e+00, 6e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [6e+00, 1e+01]
Presolve time: 0.00s
Presolved: 3 rows, 4 columns, 11 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    9.0000000e+30   4.750000e+30   9.000000e+00      0s
       4    1.2000000e+01   0.000000e+00   0.000000e+00      0s

Solved in 4 iterations and 0.00 seconds (0.00 work units)
Optimal objective  1.200000000e+01

    Variable            X
-------------------------
          X1            6
```

IP

Consider the following integer programming problem:

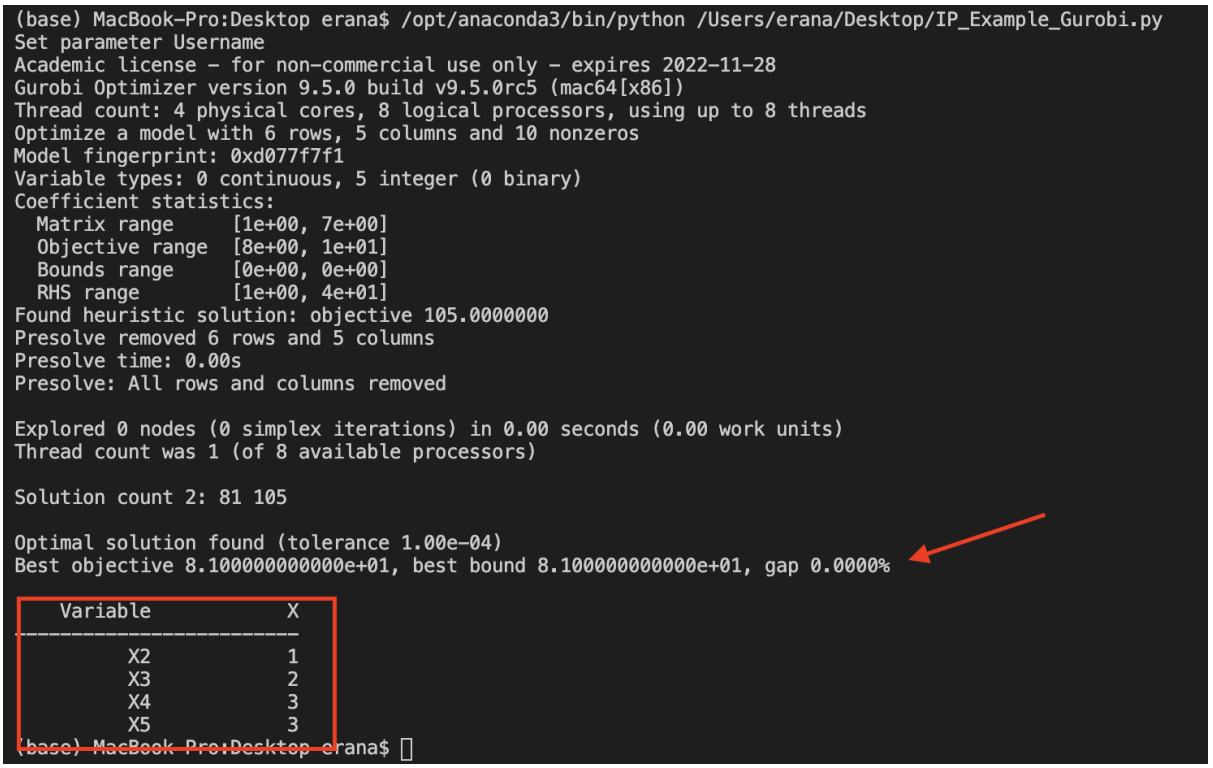
$$
\begin{aligned}
\min\ & 9x_1 + 13x_2 + 10x_3 + 8x_4 + 8x_5 \\
\text{s.t. } & 6x_1 + 3x_2 + 2x_3 + 4x_4 + 7x_5 \geq 40 \\
& x_1 \leq 1, x_2 \geq 1, x_3 \geq 2, x_4 \geq 1, x_5 \leq 3 \\
& x_1, x_2, x_3, x_4, x_5 \geq 0 \\
& x_1, x_2, x_3, x_4, x_5 \text{ integer}
\end{aligned}
$$

Here's an example code that can find the optimal solution and a sample output:

```python
import gurobipy as gp
from gurobipy import GRB

# Create a Model instance
m = gp.Model()
# Add variables
X1 = m.addVar(vtype=GRB.INTEGER, name="X1")
X2 = m.addVar(vtype=GRB.INTEGER, name="X2")
X3 = m.addVar(vtype=GRB.INTEGER, name="X3")
X4 = m.addVar(vtype=GRB.INTEGER, name="X4")
X5 = m.addVar(vtype=GRB.INTEGER, name="X5")
# Providing the coefficients and the sense of the objective function
m.setObjective(9*X1 + 13*X2 + 10*X3 + 8*X4 + 8*X5, GRB.MINIMIZE)

# Adding the 6 constraints
c1 = m.addConstr(6*X1 + 3*X2 + 2*X3 + 4*X4 + 7*X5 >= 40)
c2 = m.addConstr(X1 <= 1)
```

```
c3 = m.addConstr(X2 >= 1)
c4 = m.addConstr(X3 >= 2)
c5 = m.addConstr(X4 >= 1)
c6 = m.addConstr(X5 <= 3)

# Solving the model
m.optimize()

m.printAttr('X') #This prints the non-zero solutions found
```

Again as expected, once we run the code, we get the same solution as the Cplex method. The optimal value is 81 and the decision variables take the values $X_1 = 0, X_2 = 1, X_3 = 2,$ $X_4 = 3, X_5 = 3$ at optimality.



For tips on more complicated examples including sum functions available in python to formulate complex constraints, you could refer to here.

References:

"A List of the Gurobi Examples." Gurobi, January 17, 2019. https://www.gurobi.com/documentation/9.5/examples/a_list_of_the_grb_examples.html .

"B3 - Modeling 2 - Assets.gurobi.com." Accessed December 3, 2021. https://assets.gurobi.com/pdfs/user-events/2017-frankfurt/Modeling-2.pdf.

Lucas, Rafael. "Setting up Anaconda for Linear Optimization with Gurobi." Medium. Analytics Vidhya, April 6, 2020. https://medium.com/analytics-vidhya/setting-up-anaconda-for-linear-optimization-with-gurobi-ff22ec92e39.