Mathematics for Decisions

Integer Linear Programming: Gurobi

Romeo Rizzi, Alice Raffaele, and Matteo Zavatteri

University of Verona

romeo.rizzi@univr.it, alice.raffaele@univr.it, matteo.zavatteri@univr.it

January 14, 2022

Overview

Introduction to

GUROBI OPTIMIZATION

- Gurobi Optimizer is the current state-of-the-art solver used in mathematical programming.
- It was developed by Zonghao **Gu** and Edward **Ro**thberg (which led the IBM CPLEX development team for a decade), together with Robert **Bi**xby (the founder of CPLEX).
- It solves problems of:
  - Linear Programming (LP);
  - Mixed-Integer Linear Programming (MILP);
  - Mixed-Integer Quadratic Programming (MIQP);
  - Quadratic Programming (QP);
  - Quadratically Constrained Programming (QCP);
  - Mixed-Integer Quadratically Constrained Programming (MIQCP)

Features

- Implementation of:
  - LP Solver: primal and dual simplex algorithms, parallel barrier algorithm with crossover, concurrent optimization, and sifting algorithm
  - QP Solver: simplex and parallel barrier algorithms
  - QCP Solver: parallel SOCP barrier algorithm
  - MIP Solver: deterministic, parallel branch-and-cut, non-traditional tree-of-trees search, multiple default heuristics, solution improvement, cutting planes, and symmetry detection
- Standard MIP **cutting-plane routines** and also new routines developed on purpose and not known to the public.
- **Interfaces** for object oriented languages (e.g., C++, Java, Python) and matrix-oriented languages (e.g., C, MATLAB, R).
- Links to standard modelling languages (e.g., **AMPL**, MPL).
- Extremely robust code and parallelism.
- **Presolve**: before solving a problem, Gurobi performs some reductions by working on the constraints (e.g., aggregation, bound strengthening), in order to make the resolution faster.

Versions of Gurobi

According to the type of users, Gurobi can be obtained through different licenses:

- Commercial users can get a free evaluation license to try the software for a certain period and then buy it;
- Independent Software Vendor users can make an agreement with Gurobi to exploit it in theirs products;
- Academic users can obtain a **free academic license** for a year after signing up: https://user.gurobi.com/download/licenses/free-academic
  The license will immediately appear on your account.
  Note: sign up with your academic e-mail address!

Installation

- Download the latest version of Gurobi for your operative system at http://www.gurobi.com/downloads/gurobi-optimizer

Get the software

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	32-bit Windows	64-bit Linux	64-bit macOS	64-bit AIX
8.1.0	README	Gurobi-8.1.0-win64.msi		gurobi8.1.0_linux64.tar.gz	gurobi8.1.0_mac64.pkg	gurobi8.1.0_power64.tar.gz
Old versions						
8.0.1	README	Gurobi-8.0.1-win64.msi		gurobi8.0.1_linux64.tar.gz	gurobi8.0.1_mac64.pkg	gurobi8.0.1_power64.tar.gz
7.5.2	README	Gurobi-7.5.2-win64.msi	Gurobi-7.5.2-win32.msi	gurobi7.5.2_linux64.tar.gz	gurobi7.5.2_mac64.pkg	gurobi7.5.2_power64.tar.gz
7.0.2	README	Gurobi-7.0.2-win64.msi	Gurobi-7.0.2-win32.msi	gurobi7.0.2_linux64.tar.gz	gurobi7.0.2_mac64.pkg	gurobi7.0.2_power64.tar.gz

Gurobi for Mac OS X

1. Extract the package in your Downloads folder.
2. Double-click on the appropriate installer (e.g., gurobi8.1.0_mac64.pkg for Gurobi 8.1.0) and follow the prompts. By default, the installer will place Gurobi 8.1.0 files in your system /Library/gurobi810/mac64.
3. **Check** to be connected to an academic network, because it will be needed to verify your license: Gurobi will check if the domain name is in its list of known academic domains.
4. Go to your license page on your Gurobi account at `https://user.gurobi.com/download/licenses/current`

Current Gurobi Licenses

Your installed and available licenses

Click a line to view license details.

License ID	Purpose	Type	Version	Host Name	Date Issued
282200	Trial	Free Academic	8	MacBook-Pro-di-Alice-2.local	2018-11-16

5. Click on the license to see details and copy the code below starting with *grbgetkey*:

License Detail

License ID 282200

Information and installation instructions

License ID	282200
Date Issued	2018-11-16T01:17:11-08:00
Purpose	Trial
License Type	Free Academic
Key Type	ACADEMIC
Version	8
Distributed Limit	0
Expiration Date	2019-11-16
Host Name	MacBook-Pro-di-Alice-2.local
Host ID	9b0bbce7
User Name	alice

To install this license on a computer where Gurobi Optimizer is installed, copy and paste the following command to the Start/Run menu (Windows only) or a command/terminal prompt (any system):

```
grbgetkey 665aaeaa-e980-11e8-a7cf-02e454ff9c50
```

6. Launch Gurobi and paste the code in the terminal that will be opened.

7. When the license activation is completed, by launching Gurobi you will get the following terminal window:

```
alice — Python — 80×24

Last login: Wed Dec  5 09:09:11 on ttys000
MacBook-Pro-di-Alice-2:~ alice$ exec gurobi.sh
Python 2.7.10 (default, Oct  6 2017, 22:29:07)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.31)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Academic license - for non-commercial use only

Gurobi Interactive Shell (mac64), Version 8.1.0
Copyright (c) 2018, Gurobi Optimization, LLC
Type "help()" for help

gurobi>
```

Gurobi for Linux and Ubuntu

1. Download the 32-bit or 64-bit version according to your operative system.
2. Unpack the package
3. Follow the instructions at `http://abelsiqueira.github.io/blog/installing-gurobi-7-on-linux/`

Gurobi for Windows

1. Gurobi supports Windows 7, Windows 8 and Windows 10.
2. Download the 32-bit or 64-bit version according to your operative system.
3. Double-click on the package named **Gurobi-8.1.0-winXX** to launch installation.
4. By default, the installer will place Gurobi 8.1.0 files in directory **C:****gurobi810****win64** (or **C:****gurobi810****win32** for 32-bit Windows installs).

5. Click on the license to see details and copy the code below starting with *grbgetkey*:

License Detail

License ID 282200

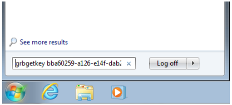
Information and installation instructions

License ID	282200
Date Issued	2018-11-16T01:17:11-08:00
Purpose	Trial
License Type	Free Academic
Key Type	ACADEMIC
Version	8
Distributed Limit	0
Expiration Date	2019-11-16
Host Name	MacBook-Pro-di-Alice-2.local
Host ID	9b0bbce7
User Name	alice

To install this license on a computer where Gurobi Optimizer is installed, copy and paste the following command to the Start/Run menu (Windows only) or a command/terminal prompt (any system):

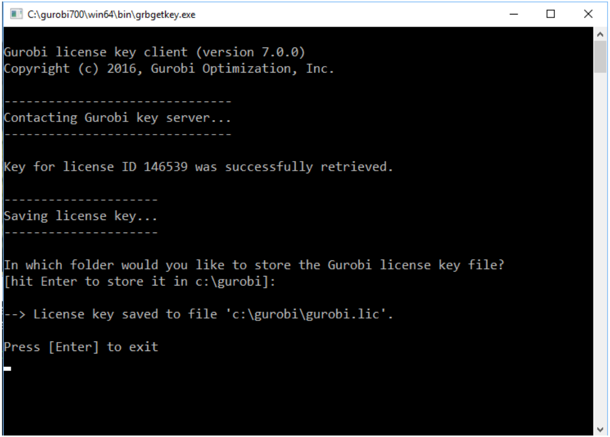
```
grbgetkey 665aaeaa-e980-11e8-a7cf-02e454ff9c50
```

6. Paste it directly into the Windows Search box and then hit Enter:



Note: you must be connected to the University wireless network, in order to activate the license.

7. You should get a command prompt window like the following:



8. Press Enter to complete the activation.

Gurobi lightweight command-line

- We can try to solve our first problem without writing code but just exploiting features of Gurobi.
- We are going to use a LP format to describe our model. Let's go back to the formulation of the Diet problem:
  - A variable to represent the number of portions of each food: x_1, x_2, x_3, x_4, x_5
  - The objective function corresponds to the sum of the costs of every food: $min\ 3x_1 + 2x_2 + 3x_3 + 19x_4 + 15x_5$
  - Baseline daily necessities:
    - Calories: $150x_1 + 120x_2 + 160x_3 + 230x_4 + 450x_5 \geq 2000$
    - Proteins: $4x_1 + 8x_2 + 15x_3 + 14x_4 + 4x_5 \geq 50$
    - Calcium: $2x_1 + 285x_2 + 54x_3 + 80x_4 + 22x_5 \geq 700$
  - Maximum number of portions:
    - Bread: $0 \leq x_1 \leq 4$
    - Milk: $0 \leq x_2 \leq 7$
    - Eggs: $0 \leq x_3 \leq 2$
    - Meat: $0 \leq x_4 \leq 3$
    - Sweets: $0 \leq x_5 \leq 2$

The LP-format

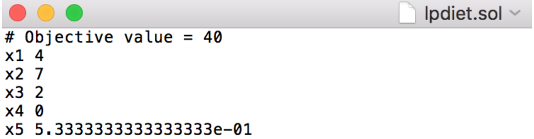
- The input syntax is a set of algebraic expressions and declarations in the following order:
  - Objective function
  - Constraints
  - Declarations
- Here follows the **gurobi-lpformat-diet.lp** file:

gurobi-lpformat-diet

```
Minimize
        3 x1 + 2 x2 + 3 x3 + 19 x4 + 15 x5
Subject To
Calories: 150 x1 + 120 x2 + 160 x3 + 230 x4 + 450 x5 >= 2000
Protein : 4 x1 + 8 x2 + 15 x3 + 14 x4 + 4 x5 >= 50
Calcium: 2 x1 + 285 x2 + 54 x3 + 80 x4 + 22 x5 >= 700
Bounds
x1 <= 4
x2 <= 7
x3 <= 2
x4 <= 3
x5 <= 2
Semi-continuous
x1 x2 x3 x4 x5
End
```

Solving the problem

1. Open a new terminal or a command prompt, go to the directory where you have saved the .lp file and type **gurobi_cl ResultFile=gurobilpdiet.sol gurobi-lpformat-diet.lp**
2. Gurobi will show you its logs and the value of the solution found, saving results in a corresponding .sol file:



lpdiet.sol

```
# Objective value = 40
x1 4
x2 7
x3 2
x4 0
x5 5.3333333333333333e-01
```

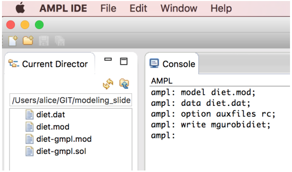
3. Gurobi logs:

Diet — -bash — 97×42

```
2016-23176:~ alice$ cd GIT/modeling_slides/modeling_exercises/Diet/
2016-23176:Diet alice$ gurobi_cl ResultFile=gurobilpdiet.sol gurobi-lpformat-diet.lp
Academic license - for non-commercial use only

Gurobi Optimizer version 7.5.2 build v7.5.2rc1 (mac64)
Copyright (c) 2017, Gurobi Optimization, Inc.

Read LP format model from file gurobi-lpformat-diet.lp
Reading time = 0.00 seconds
: 3 rows, 5 columns, 15 nonzeros
Optimize a model with 3 rows, 5 columns and 15 nonzeros
Variable types: 0 continuous, 0 integer (0 binary)
Semi-Variable types: 5 continuous, 0 integer
Coefficient statistics:
  Matrix range     [2e+00, 4e+02]
  Objective range  [2e+00, 2e+01]
  Bounds range     [2e+00, 7e+00]
  RHS range        [5e+01, 2e+03]
Found heuristic solution: objective 97.0000000
Presolve removed 1 rows and 0 columns
Presolve time: 0.00s
Presolved: 2 rows, 5 columns, 10 nonzeros
Variable types: 5 continuous, 0 integer (0 binary)

Root relaxation: objective 4.000000e+01, 1 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0               0      40.0000000   40.00000  0.00%     -    0s

Explored 0 nodes (1 simplex iterations) in 0.01 seconds
Thread count was 4 (of 4 available processors)

Solution count 2: 40 97

Optimal solution found (tolerance 1.00e-04)
Best objective 4.000000000000e+01, best bound 4.000000000000e+01, gap 0.0000%

Wrote result file 'gurobilpdiet.sol'
```

Another way by exploiting AMPL

1. Launch AMPL and go to the folder containing the .mod and .dat files of our Diet problem; in the console, write the following commands:



AMPL generates an **.mps model** by typing the command **write m***modelname* and saving the model in the same directory of the .mod file.
Note: the command **option auxfiles rc** is useful to keep the original names of variables and constraints: by default AMPL changes them because the MPS format cannot handle long variables and constraints names. Together with gurobidiet.mps, two additional files are generated: *gurobidiet.row* and *gurobidiet.col*.

2. The *gurobidiet.mps* file will look like this:

gurobidiet.mps

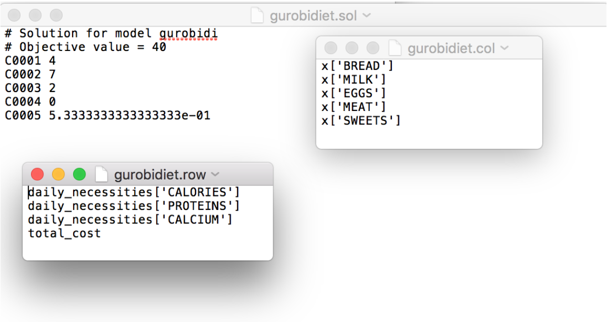
```
NAME          gurobidi
ROWS
 G   R0001
 G   R0002
 G   R0003
 N   R0004
COLUMNS
    C0001     R0001     150
    C0001     R0002     4
    C0001     R0003     2
    C0001     R0004     3
    C0002     R0001     120
    C0002     R0002     8
    C0002     R0003     285
    C0002     R0004     2
    C0003     R0001     160
    C0003     R0002     15
    C0003     R0003     54
    C0003     R0004     3
    C0004     R0001     230
    C0004     R0002     14
    C0004     R0003     80
    C0004     R0004     19
    C0005     R0001     450
    C0005     R0002     4
    C0005     R0003     22
    C0005     R0004     15
RHS
    B         R0001     2000
    B         R0002     50
    B         R0003     700
BOUNDS
 UP BOUND     C0001     4
 UP BOUND     C0002     7
 UP BOUND     C0003     2
 UP BOUND     C0004     3
 UP BOUND     C0005     2
ENDATA
```

3. To solve the problem with Gurobi command-line, open a new terminal and go to the folder containing the diet files.
4. Type **gurobi_cl gurobidiet.mps** to launch the software and display the logs.
5. If you want to save the results in an appropriate file to review the solution, type instead **gurobi_cl ResultFile=gurobidiet.sol gurobidiet.mps**

Diet — -bash — 86×29

```
Last login: Sat Nov 18 10:59:48 on ttys000
MacBook-Pro-di-Alice:~ alice$ cd GIT/modeling_slides/modeling_exercises/Diet/
MacBook-Pro-di-Alice:Diet alice$ gurobi_cl ResultFile=gurobidiet.sol gurobidiet.mps
Academic license - for non-commercial use only

Gurobi Optimizer version 7.5.1 build v7.5.1rc0 (mac64)
Copyright (c) 2017, Gurobi Optimization, Inc.

Read MPS format model from file gurobidiet.mps
Reading time = 0.00 seconds
gurobidi: 3 rows, 5 columns, 15 nonzeros
Optimize a model with 3 rows, 5 columns and 15 nonzeros
Coefficient statistics:
  Matrix range     [2e+00, 4e+02]
  Objective range  [2e+00, 2e+01]
  Bounds range     [2e+00, 7e+00]
  RHS range        [5e+01, 2e+03]
Presolve time: 0.00s
Presolved: 3 rows, 5 columns, 15 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   1.968750e+02   0.000000e+00      0s
       1    4.0000000e+01   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.01 seconds
Optimal objective  4.000000000e+01

Wrote result file 'gurobidiet.sol'
```

6. The *gurobidiet.sol*, *gurobidiet.row* and *gurobidiet.col* files will be something like the following:



7. As you can see, we obtained the known optimal result (the total cost is 40) and suggestions about which foods and how many portions to eat, in order to minimize the total cost of the diet.

Using Gurobi Interactive Shell

- The Gurobi package includes an interactive shell based on Python.
- It allows us to perform hands-on interaction and experimentation with optimization models.
- In fact, it is possible to read models from files, perform complete or partial optimization runs on them, change parameters, modify models, re-optimize, and so on.
- To open the shell, type **gurobi.sh** inside a terminal:

alice — Python ◂ gurobi.sh — 80×24

```
Last login: Wed Nov 29 14:20:53 on ttys000
2016-23176:~ alice$ gurobi.sh
Python 2.7.10 (default, Feb  7 2017, 00:08:15)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Academic license - for non-commercial use only

Gurobi Interactive Shell (mac64), Version 7.5.2
Copyright (c) 2017, Gurobi Optimization, Inc.
Type "help()" for help

gurobi>
```

The Diet Problem with the Interactive Shell

1. Open a terminal and go to the directory containing the lp file we used before.
2. First type **gurobi.sh** and then the command **m = read('filename')** to load a model from the file and save it into the variable **m**.
3. Once read and loaded the model, run the command **m.optimize()** to solve the instance.

Diet — Python ◂ gurobi.sh — 80×24

```
Last login: Wed Nov 29 15:33:18 on ttys000
2016-23176:~ alice$ cd GIT/modeling_slides/modeling_exercises/Diet/
2016-23176:Diet alice$ gurobi.sh
Python 2.7.10 (default, Feb  7 2017, 00:08:15)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Academic license - for non-commercial use only

Gurobi Interactive Shell (mac64), Version 7.5.2
Copyright (c) 2017, Gurobi Optimization, Inc.
Type "help()" for help

gurobi> m = read('gurobi-lpformat-diet.lp')
gurobi> m.optimize()
```

4. Here follow Gurobi resolution logs:

Diet — Python ‹ gurobi.sh — 80×35

```
Gurobi Interactive Shell (mac64), Version 7.5.2
Copyright (c) 2017, Gurobi Optimization, Inc.
Type "help()" for help

gurobi> m = read('gurobi-lpformat-diet.lp')
gurobi> m.optimize()
Optimize a model with 3 rows, 5 columns and 15 nonzeros
Variable types: 0 continuous, 0 integer (0 binary)
Semi-Variable types: 5 continuous, 0 integer
Coefficient statistics:
  Matrix range     [2e+00, 4e+02]
  Objective range  [2e+00, 2e+01]
  Bounds range     [2e+00, 7e+00]
  RHS range        [5e+01, 2e+03]
Found heuristic solution: objective 97.0000000
Presolve removed 1 rows and 0 columns
Presolve time: 0.00s
Presolved: 2 rows, 5 columns, 10 nonzeros
Variable types: 5 continuous, 0 integer (0 binary)

Root relaxation: objective 4.000000e+01, 1 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0               0      40.0000000   40.00000  0.00%     -    0s

Explored 0 nodes (1 simplex iterations) in 0.00 seconds
Thread count was 4 (of 4 available processors)

Solution count 2: 40 97

Optimal solution found (tolerance 1.00e-04)
Best objective 4.000000000000e+01, best bound 4.000000000000e+01, gap 0.0000%
```

Changing the model

- To print the values of the variables:
  - **m.printAttr('X')** will display all nonzero variables and their values in a formatted way;
  - **m.getVars()** instead will print all variables as an array.
- You can do some modifications to the model, for example changing lower and upper bounds:
  gurobi> v = m.getVars()
  gurobi> v[1].ub = 2
  m.optimize()
  (with this new UB, you will obtain a different solution, with value 50).
- A list of all methods on Model objects can be obtained typing **help(Model)** or **help(m)**.

Changing Gurobi parameters

- Gurobi solving operations are controlled by parameters such as:
  - **MIPGap** (only for MIP models):
    - the MIP solver will terminate (with an optimal result) when the gap between the lower and upper objective bounds is less than MIPGap times the absolute value of the upper bound;
    - default value: 1e-4.
  - **CutOff**:
    - you are not interested in solutions whose objective values are worse than the specified cutoff value;
    - a solution with value better than the specified cutoff will be considered optimal.
  - **MIPFocus**:
    - it allows you to modify your high-level solution strategy, depending on your goals;
    - by default, its value is 0 (no strategy);
    - if its value is 1, the goal is to obtain a feasible solution; else if you want a feasible and optimal solution, its value is 2; finally, if its value is 3, Gurobi will focus on the objective bound.

- Let's try to change the CutOff in our diet problem, by assuming we are satisfied with a total cost of 50;
- Type **m.setParam('CutOff', 50)** and then again **m.optimize()**, to obtain a different solution, with value exactly 50.
- To reset any changes we introduced, we can always type **m.resetParams()**.
- To reset the model to an unsolved state, discarding any previously computed solution information, type **m.reset()**.
- There is a simple automated tool to try different set of parameters: **m.tune()**.
- You can find the **complete list** of parameters at https://www.gurobi.com/documentation/9.5/refman/parameters.html.

Gurobi and the Python interface

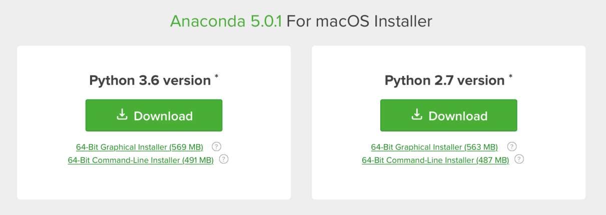
- We are going to use Python as our programming language. If you need help with Python syntax and rules, look in the References for some useful links.
- Actually, there are already a Python interpreter and a basic set of Python modules in Gurobi but, in order to increase interactivity and productivity of Python experience on model building, it is better to install a widely-used Python platform.

ANACONDA.

- We can install the Anaconda Python distribution, which includes:
  - Spyder, a graphical development environment;
  - Jupyter, a notebook-style interface.

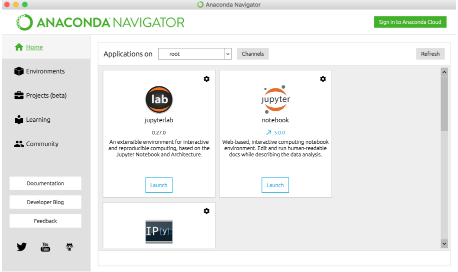
Anaconda for Mac OS X

1. Download the latest version of Anaconda for macOS Installer at `https://www.anaconda.com/download/#macos`
   You can choose between two versions:



   (If you want, you can check before which Python version you already have in your system, opening a terminal and typing **python –version**).

2. Extract the package and launch the installation
3. Answer the prompts on the Introduction, Read Me and License screens, then choose "Install for me only" and go ahead.
4. After your installation is complete, verify it by opening Anaconda Navigator, a program included in Anaconda: from Launchpad, select Anaconda Navigator.



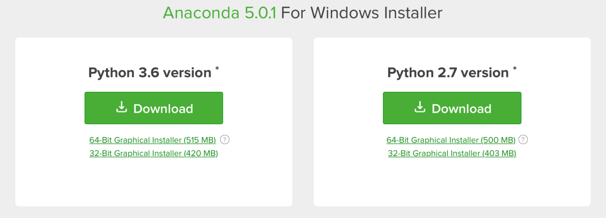
6. To install the Gurobi package into Anaconda, open a terminal and follow these steps:
   6.1 Add the Gurobi channel to your Anaconda channels: **conda config –add channels http://conda.anaconda.org/gurobi**
   6.2 Install the package: **conda install gurobi** and type **y** to confirm.
   6.3 If you want to remove it: **conda remove gurobi**.

alice — -bash — 94×25

```
Last login: Mon Nov 27 18:29:07 on ttys000
2016-23176:~ alice$ conda config --add channels http://conda.anaconda.org/gurobi
2016-23176:~ alice$ conda install gurobi
Fetching package metadata .............
Solving package specifications: .

Package plan for installation in environment /Users/alice/anaconda2:

The following NEW packages will be INSTALLED:

    gurobi: 7.5.2-py27_0 gurobi

Proceed ([y]/n)? y

gurobi-7.5.2-p 100% |################################| Time: 0:00:48 324.58 kB/s
2016-23176:~ alice$
```

Anaconda for Windows

1. Download the latest version of Anaconda for Windows Installer at https://www.anaconda.com/download/#windows
   You can choose between two versions:



   (If you want, you can check before which Python version you already have in your system, opening a command prompt and typing **python –version**).

2. Extract the package and launch the installation.
3. Click Next, read the licensing terms and click "I agree", then choose "Just me" and click Next.
4. Select a destination folder and click Next.
5. Choose whether to add Anaconda to your PATH environment variable (on the website they recommend not adding Anaconda to the PATH environment variable, since this can interfere with other software)
6. Click on the Install button and, after the installation is complete, verify it by opening Anaconda Navigator, a program that is included with Anaconda: from your Windows Start menu, select the shortcut Anaconda Navigator.

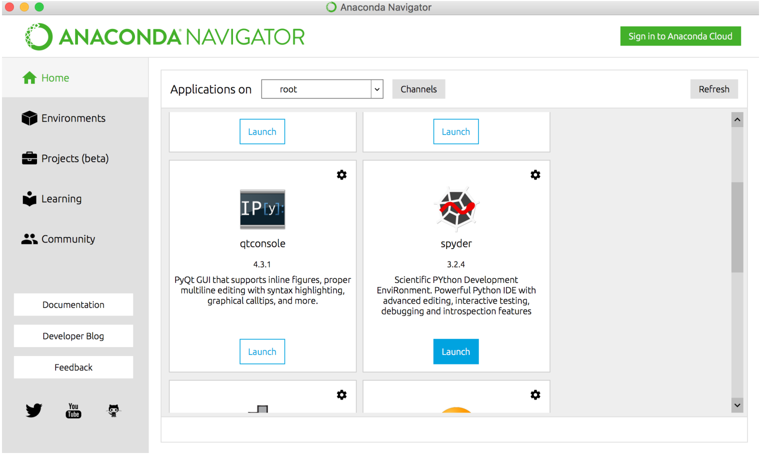
7. To install the Gurobi package into Anaconda, from your Windows Start menu open the Anaconda command prompt and follow these steps:
   7.1 Add the Gurobi channel to your Anaconda channels: **conda config –add channels http://conda.anaconda.org/gurobi**
   7.2 Install the package: **conda install gurobi** and type **y** to confirm.
   7.3 If you want to remove it: **conda remove gurobi**.

Note about Python environment

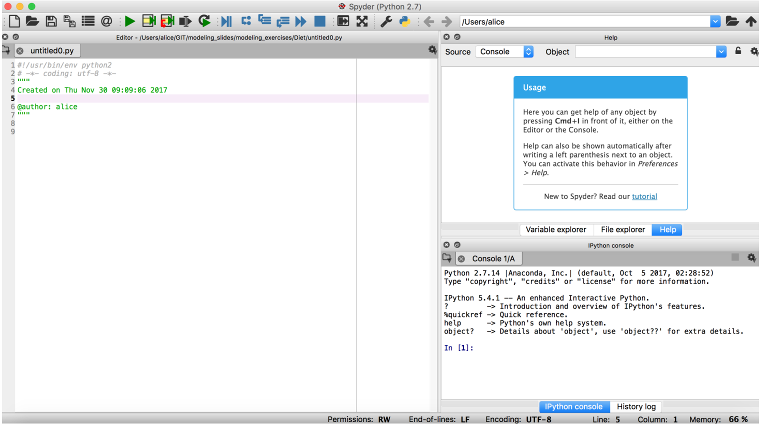
- **Only** if you prefer to use another environment different from Anaconda, you must add the package **gurobipy** in order to use Gurobi.
- In Linux and Mac OS X systems, open the terminal, go to **Library/gurobi810/mac64/** and type **python setup.py install** to link Gurobi to your environment.
- In Windows systems, open the command prompt, go to **C:\gurobi810\winXX** and type **python setup.py install**.
- After the installation of the gurobipy package, you can type **import gurobipy** or **from gurobipy import *** from your Python shell and access all Gurobi classes and methods.

The Diet Problem with Anaconda

1. Launch Anaconda Navigator and open Spyder:



2. Spyder will look like this:



Gurobi Interactive Shell commands can be typed directly in the Spyder console, but the purpose now is to write instructions in Python in order to build the model.

3. In the File menu, click "New file..."
4. **Note:** the first thing to do, in every Gurobi project developed in a Python IDE like Spyder, is to import the Gurobi module: **from gurobipy import *** or **import gurobipy** (not required when launching gurobi.sh).
5. To define a model, it is enough to declare a new object *m* and instantiate it with the appropriate method imported: **m = Model("modelname")**.

```
# Solve the diet problem

from gurobipy import *

# Model
m = Model("diet")
```

6. Then, we need to define the sets and parameters of our instances: we can exploit Python data structures, such as lists, tuples and dictionaries, which allow to map arbitrary key values to pieces of data; the function to use is **multidict**:

```
# Sets and parameters
categories, minNutrition, maxNutrition = multidict({
  'calories': [2000, GRB.INFINITY],
  'protein':  [50, GRB.INFINITY],
  'calcium':  [700, GRB.INFINITY]})

foods, cost, maxPortions = multidict({
  'bread':  [3, 4],
  'milk':   [2, 7],
  'eggs':   [3, 2],
  'meat':   [19, 3],
  'sweets': [15, 2]})

# Nutrition values for the foods
nutritionValues = {
  ('bread', 'calories'): 150,
  ('bread', 'protein'): 4,
  ('bread', 'calcium'): 2,
  ('milk', 'calories'): 120,
  ('milk', 'protein'): 8,
  ('milk', 'calcium'): 285,
  ('eggs', 'calories'): 160,
  ('eggs', 'protein'): 15,
  ('eggs', 'calcium'): 54,
  ('meat', 'calories'): 230,
  ('meat', 'protein'): 14,
  ('meat', 'calcium'): 4,
  ('sweets', 'calories'): 450,
  ('sweets', 'protein'): 4,
  ('sweets', 'calcium'): 22}
```

7. Now we can introduce our variables:

```
# Using Python looping constructs and m.addVar() to create decision variables:
buy = {}
for f in foods:
    buy[f] = m.addVar(0.0, maxPortions[f], name=f)
```

The buy object is initially defined as an empty dictionary; then, for every food, a variable is added using **m.addVar(...)**. The first argument is the lower bound of the variable, followed by the upper bound and finally by the name given to the variable (always recommended, especially when working with a huge number of variables).

8. We define the objective function of the model by using **m.setObjective(...)**:

```
# The objective is to minimize the costs, using looping constructs:
m.setObjective(sum(buy[f]*cost[f] for f in foods), GRB.MINIMIZE)
```

The former argument is the expression, whereas the latter is the purpose, that could be GRB.MINIMIZE or GRB.MAXIMIZE.

9. To add the constraints about the baseline daily amounts of calories, proteins and calcium, we can use the method **m.addConstr(...)** or, if we know that there is a range to respect (i.e., minimum and maximum values), we can use **m.addRange(...)**, as here:

```
# Nutrition constraints to respect minimum daily necessities:
for c in categories:
    m.addRange(
            sum(nutritionValues[f,c] * buy[f] for f in foods), minNutrition[c], maxNutrition[c], c)
```

The first argument is the expression, the second and the third ones are the lower and upper bounds respectively, the last one is the name assigned to the constraint.

10. Now we have everything we need to solve the model; anyway, before doing it, we can define a method to display results nicely:

```
def printSolution():
    if m.status == GRB.Status.OPTIMAL:
        print('\nCost: %g' % m.objVal)
        print('\nBuy:')
        buyx = m.getAttr('x', buy)
        for f in foods:
            if buy[f].x > 0.0001:
                print('%s %g' % (f, buyx[f]))
    else:
        print('No solution')
```

This method is based on the value of *m.status* and prints the solution only if this represents the optimum.
The objective function value can be obtained with **m.objVal**, while variables with **m.getAttr('x', objectName)**.

11. To solve the problem, the instruction is again **m.optimize()**, as in the command-line or in the Interactive Shell; then type **printSolution()**:

```
# Solve
m.optimize()
printSolution()
```

12. To run the code, press the green arrow in the top menu or F5; then, look at the console:

```
In [3]: runfile('/Users/alice/GIT/modeling_slides/modeling_exercises/Diet/python-
diet.py', wdir='/Users/alice/GIT/modeling_slides/modeling_exercises/Diet')
Optimize a model with 3 rows, 5 columns and 15 nonzeros
Coefficient statistics:
  Matrix range     [2e+00, 4e+02]
  Objective range  [2e+00, 2e+01]
  Bounds range     [2e+00, 7e+00]
  RHS range        [5e+01, 2e+03]
Presolve time: 0.02s
Presolved: 3 rows, 5 columns, 15 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   1.968750e+02   0.000000e+00      0s
       1    4.0000000e+01   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.03 seconds
Optimal objective  4.000000000e+01

Cost: 40

Buy:
sweets 0.533333
eggs 2
bread 4
milk 7
```

13. If we want to add another constraint, for example that the sum of portions of milk and eggs cannot be more than 6, we can type another instruction at the end of our code (or directly in the console) and optimize the model again:

```
print('\nAdding constraint: at most 6 servings of milk and eggs')
m.addConstr(buy['milk'] + buy['eggs'] <= 6, "limit_milk_eggs")

# Solve
m.optimize()
printSolution()
```

14. Run the file again to obtain the new solution:

```
Adding constraint: at most 6 servings of milk and eggs
Optimize a model with 4 rows, 5 columns and 17 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 4e+02]
  Objective range  [2e+00, 2e+01]
  Bounds range     [2e+00, 7e+00]
  RHS range        [6e+00, 2e+03]
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    4.0000000e+01   4.800000e+01   0.000000e+00      0s
       1    4.6000000e+01   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.04 seconds
Optimal objective  4.600000000e+01

Cost: 46

Buy:
sweets 1.33333
eggs 2
bread 4
milk 4
```

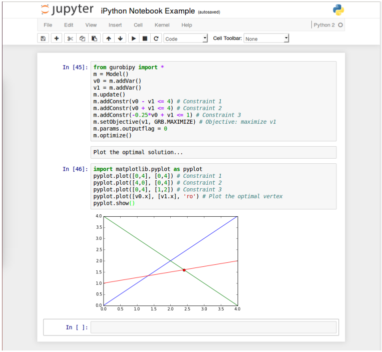
15. You can also write your Python code in any text editor you prefer and solve the model by using the Gurobi Interactive Shell:

Diet — -bash — 84×27

```
Last login: Thu Nov 30 11:55:21 on ttys000
2016-23176:~ alice$ cd GIT/modeling_slides/modeling_exercises/Diet/
2016-23176:Diet alice$ gurobi.sh python-diet.py
Academic license - for non-commercial use only
Optimize a model with 3 rows, 5 columns and 15 nonzeros
Coefficient statistics:
  Matrix range     [2e+00, 4e+02]
  Objective range  [2e+00, 2e+01]
  Bounds range     [2e+00, 7e+00]
  RHS range        [5e+01, 2e+03]
Presolve time: 0.00s
Presolved: 3 rows, 5 columns, 15 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   1.968750e+02   0.000000e+00      0s
       1    4.0000000e+01   0.000000e+00   0.000000e+00      0s

Solved in 1 iterations and 0.00 seconds
Optimal objective  4.000000000e+01

Cost: 40

Buy:
sweets 0.533333
eggs 2
bread 4
milk 7
```

16. As before, in the code you can set Gurobi params as you like.

Just a note about Jupyter



Notebook-style interface to mix executable code, text and graphics, in order to create a self-documenting stream of results;

Solving TSP with Gurobi

- Let's use Gurobi to implement the Branch-and-Cut framework we saw last time.
- We let Gurobi solve the relaxed problem and, when it finds a new MIP solution, we look for **connected components**:
  - We compute the shortest (sub)tour of vertices visited by the solution;
  - If the number of vertices in the tour is equal to the total number of vertices in V, the solution is ok; otherwise, it means that there are subtours.
- To do this, we need to use **Callbacks** and **Lazy Constraints**.

Callbacks

- Functions that can be defined by the user to perform some custom actions automatically in particular cases, for example:
  - During presolve;
  - When a new MIP incumbent solution is found;
  - When printing a log message.
- The callback function must be specified and passed as parameter to the optimize method:

  model.optimize(callbackname)
- The callback routines use mainly the **where** argument: it indicates in which state the Gurobi optimizer is (*presolve*, *simplex*, *MIP*, etc.); for any possible value, we can develop the appropriate code.
- In our case, we want to check subtours everytime a new MIP solution is found; then, the *where* value is 4.

Lazy Constraints

- By using callbacks, during runtime execution we can add two types of custom constraints:
  - **Cutting Planes**:
    - They do not cut off any integer feasible solutions, but just strengthen the continuous relaxation to speed-up the process.
  - **Lazy Constraints**:
    - They affect only MIP models, by cutting off integer solutions that are feasible for the remaining constrained system.
    - They are necessary for the correctness of the model, but still they are dynamically added because the general form could be composed of exponential constraints.
    - To introduce them in the model, the parameter LazyConstraints must be set to 1:

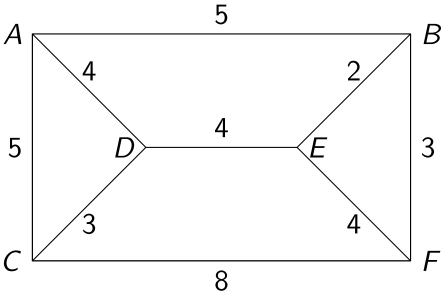
$$\mathbf{model.params.LazyConstraints = 1}$$

Gurobi TSP Example

- You can find a TSP example on Gurobi website (`http://examples.gurobi.com/traveling-salesman-problem/`) over n random points in the US; the goal is to minimize their Eulerian distances:

Our TSP Example with Gurobi

- Let's consider again our graph, with the following edges and costs:



- It is the same instance we tried to solve before with AMPL, where edges (B,E) and (CF) cost respectively 2 and 8.

Sets definition

1. Firstable, import required libraries:

```python
import math
import random
from gurobipy import *
```

2. Define the sets of vertices and edges:

```python
# ------------------------------------------------------------------
# MODEL DEFINITION

# Create vertices and edges for our example
vertices = [0,1,2,3,4,5] #vertices = ['A', 'B', 'C', 'D', 'E', 'F']
# We do not use letters just to avoid issues with indexing and syntax errors
edges = {
    (0,1): 5,
    (0,2): 5,
    (0,3): 4,
    (1,4): 2, #9
    (1,5): 3,
    (2,3): 3,
    (2,5): 8,
    (3,4): 4,
    (4,5): 4}

n = len(vertices)
```

Model creation

3. Create the model and add a variable for each edge.
4. Set the objective function.
5. Add degree constraints.

```
# Create the model
m = Model()

# Create the variables
vars = {}
for i,j in edges.keys():
    vars[i,j] = m.addVar(obj=edges[i,j], vtype=GRB.BINARY,
                         name='e[%d,%d]'%(i,j))
for i,j in vars.keys():
    vars[j,i] = vars[i,j] # Edges in both directions

# To create the model data structure only once, after variables creation
m.update()

# Add the objective function
m.setObjective(sum(vars[i,j]*edges[i,j]
                   for (i,j) in edges.keys()),GRB.MINIMIZE)

# Add degree-2 constraint
for i in vertices:
    m.addConstr(sum(vars[i,j] for j in vertices
                    if (i,j) in edges.keys() or (j,i) in edges.keys()) == 2)

```

A useful function: Subtour

6. We implement the method that, given a set of edges as parameter, looks for the shortest subtour inside them:

```python
# Given a list of edges, this method finds the shortest subtour
def subtour(edges):
    visited = [False]*n
    cycles = []
    lengths = []
    selected = [[] for i in vertices]
    for x,y in edges:
        selected[x].append(y)
        selected[y].append(x) # Edges in both directions
    while True:
        current = visited.index(False)
        thiscycle = [current]
        while True:
            visited[current] = True
            neighbors = [x for x in selected[current] if not visited[x]]
            if len(neighbors) == 0:
                break
            current = neighbors[0]
            thiscycle.append(current)
        cycles.append(thiscycle)
        lengths.append(len(thiscycle))
        if sum(lengths) == n:
            break
    return cycles[lengths.index(min(lengths))]
```

Note: this code has to be inserted at the beginning, before sets and model definition, so we can call it in the rows below.

Model optimization - Part I

7. We start by solving the problem without adding Subtour Elimination Constraints:

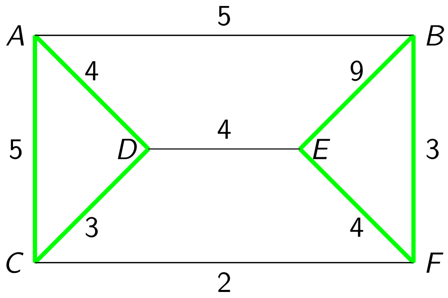
```
# ------------------------------------------------------------------
# MODEL OPTIMIZATION

# Without SECs
print '\n------------------------------------------------------------\n'
print 'Optimize model without SECs\n'
m._vars = vars
m.optimize()
# Print optimal solution
print '\n------------------------------------------------------------\n'
solution = m.getAttr('x', vars)
selected = [(i,j) for i,j in edges.keys() if solution[i,j] > 0.5]
print 'Edges in solution: ' + str(selected)
print('Optimal tour: %s' % str(subtour(selected)))
print('Optimal cost: %g' % m.objVal)
```

8. Launching the execution, the console will output this:

```
Optimize model without SECs

Optimize a model with 6 rows, 9 columns and 18 nonzeros
Variable types: 0 continuous, 9 integer (9 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [2e+00, 8e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [2e+00, 2e+00]
Presolve removed 3 rows and 3 columns
Presolve time: 0.00s
Presolved: 3 rows, 6 columns, 9 nonzeros
Variable types: 0 continuous, 6 integer (6 binary)
Found heuristic solution: objective 26.0000000

Root relaxation: objective 2.100000e+01, 3 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0               0      21.0000000   21.00000  0.00%     -    0s

Explored 0 nodes (3 simplex iterations) in 0.05 seconds
Thread count was 4 (of 4 available processors)

Solution count 2: 21 26

Optimal solution found (tolerance 1.00e-04)
Best objective 2.100000000000e+01, best bound 2.100000000000e+01, gap 0.0000%

-------------------------------------------------------------

Edges in solution: [(1, 5), (2, 3), (4, 5), (0, 3), (0, 2), (1, 4)]
Optimal tour: [0, 3, 2]
Optimal cost: 21
```

Note: the optimal subtour visits just three vertices!

9. We obtained the following solution:



Let's apply the approach discussed before, with lazy constraints...

Model optimization - Part II

10. We define the function called automatically by Gurobi when it finds a new MIP incumbent solution:

```
# Callback: use lazy constraints to eliminate subtours
def subtourelim(model, where):
    if where == GRB.callback.MIPSOL:
        print '\nNew solution found: checking the presence of subtours...'
        selected = []
        # Make a list of edges selected in the solution
        vals = model.cbGetSolution(model._vars)
        selected = tuplelist((i,j) for i,j in model._vars.keys()
                              if vals[i,j] > 0.5)

        # Find the shortest cycle in the selected edge list
        tour = subtour(selected)
        if len(tour) < n:
            # Add a subtour elimination constraint
            print 'One subtour found: ' + str(tour)
            expr = 0
            for i in range(len(tour)):
                for j in range(i+1, len(tour)):
                    expr += model._vars[tour[i], tour[j]]
            print 'Subtour Elimination Constraint added:'
            print str(expr) + ' <= ' + str(len(tour)-1)
            model.cbLazy(expr <= len(tour)-1)
        else:
            print 'No subtour found!'
```

11. We tell Gurobi to use it during the optimization, after setting the LazyConstraints parameter to 1 and bringing the model back to an unsolved state with *reset()*:

```python
# With Lazy Constraints
print '\n--------------------------------------------------------------\n'
print 'Adding Lazy Constraints\n'
m.reset()
m.params.LazyConstraints = 1
m.optimize(subtourelim)

# Print optimal solution
print '\n--------------------------------------------------------------\n'
solution = m.getAttr('x', vars)
selected = [(i,j) for i,j in edges.keys() if solution[i,j] > 0.5]
print 'Edges in solution: ' + str(selected)
print('Optimal tour: %s' % str(subtour(selected)))
print('Optimal cost: %g' % m.objVal)

# Check if the solution has only one connected component
assert len(subtour(selected)) == n
```

If you prefer, you can also define a **printSolution** function.

12. This is the output obtained by introducing Lazy Constraints:

```
Adding Lazy Constraints

Changed value of parameter LazyConstraints to 1
   Prev: 0  Min: 0  Max: 1  Default: 0
Optimize a model with 6 rows, 9 columns and 18 nonzeros
Variable types: 0 continuous, 9 integer (9 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [2e+00, 8e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [2e+00, 2e+00]
Presolve removed 3 rows and 3 columns
Presolve time: 0.00s
Presolved: 3 rows, 6 columns, 9 nonzeros
Variable types: 0 continuous, 6 integer (6 binary)

New solution found: checking the presence of subtours...
No subtour found!
Found heuristic solution: objective 26.0000000

Root relaxation: objective 2.100000e+01, 3 iterations, 0.00 seconds

New solution found: checking the presence of subtours...
One subtour found: [0, 3, 2]
Subtour Elimination Constraint added:
<gurobi.LinExpr: e[0,3] + e[0,2] + e[2,3]> <= 2
```

When Gurobi finds the solution with value 21, it uses the defined callback and finds a subtour composed of vertices 0, 2 and 3 $\rightarrow$ Thus, it adds to the current formulation the following cut: $e_{0,3} + e_{0,2} + e_{2,3} \leq 2$.

13. Here is the right optimal solution:

```
New solution found: checking the presence of subtours...
No subtour found!

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0               0      24.0000000   24.00000  0.00%     -    0s

Cutting planes:
  Lazy constraints: 1

Explored 0 nodes (4 simplex iterations) in 0.07 seconds
Thread count was 4 (of 4 available processors)

Solution count 2: 24 26

Optimal solution found (tolerance 1.00e-04)
Best objective 2.400000000000e+01, best bound 2.400000000000e+01, gap 0.0000%

--------------------------------------------------------------

Edges in solution: [(0, 1), (1, 5), (2, 3), (4, 5), (3, 4), (0, 2)]
Optimal tour: [0, 1, 5, 4, 3, 2]
Optimal cost: 24
```

Note: now the optimal tour visits all vertices.

Conclusions

- We showed how to solve ILP problems by using Gurobi Optimizer, both exploiting its command-line options (lightweight version and Interactive Shell) and its libraries (with Spyder and Anaconda).
- In particular, we solved the Diet problem and TSP, also by introducing some advanced features as lazy constraints.
- Gurobi can interface not only with Python, but also with C, C++, Java, MATLAB and R: why not trying to do these exercises with other languages, if you are more familiar?

References

Gurobi Optimization and Official Documentation
http://www.gurobi.com
http://www.gurobi.com/documentation/

Anaconda
https://www.anaconda.com

Python Official Documentation (English):
https://docs.python.org/release/2.7/tutorial/

Python Tutorial (English):
https://www.tutorialspoint.com/python/

Learn Python (English, Italian, Spanish):
https://www.learnpython.org/en/

Eclipse
http://www.eclipse.org/home/index.php

AMPL Faqs about Files and Preprocessing:
http://ampl.com/faqs/category/filespreprocessing/

Gurobi Optimization and Official Documentation
http://www.gurobi.com
http://www.gurobi.com/documentation/

Gurobi Quick Start Guide
https://www.gurobi.com/documentation/7.5/quickstart_linux.pdf

Gurobi Python Implementation of TSP with random points
http://www.gurobi.com/documentation/7.5/examples/tsp_py.html

Operations Research Group, *Asymmetric TSP*, University of Bologna
http://www.or.deis.unibo.it/algottm/files/8_ATSP.pdf