

Introduction

oo

Installation

oooooooooooo

Gurobi CLI

oooooooooooo  
ooooooo

Gurobi dynamic libraries

oooooooooooooooooooooooo

Solving the TSP

oooo  
oooooooooooo

Conclusions

ooo

# Mathematics for Decisions

## Integer Linear Programming: Gurobi

Romeo Rizzi, Alice Raffaele

University of Verona

*romeo.rizzi@univr.it, alice.raffaele@unitn.it*

May 2020

Introduction

oo

Installation

oooooooooooo

Gurobi CLI

oooooooooooo  
ooooooo

Gurobi dynamic libraries

oooooooooooooooooooooooo

Solving the TSP

oooo  
oooooooooooo

Conclusions

ooo

# Overview

Introduction

Installation

Gurobi CLI

Gurobi lightweight command-line

Gurobi Interactive Shell

Gurobi dynamic libraries

Python

Java

Solving the TSP

Gurobi Implementation Notes

Our TSP example with Gurobi

Conclusions

## Introduction to



- The Gurobi Optimizer is a state-of-the-art solver for mathematical programming.
- It has been developed by Zonghao **Gu**, Edward **Rothberg** (they led the CPLEX development team for a decade) and Robert **Bixby**, which is instead the founder of CPLEX.
- It includes solvers for the following types of problems:
  - Linear Programming problems (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

- Advanced implementation of the **latest algorithms**:
  - ✓ 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 modeling languages (e.g., **AMPL**, MPL).
- Extremely robust code and parallelism
- **Presolve**: before solving a problem, it performs some reductions working on the constraints (e.g., aggregation, bound strengthening), in order to make the resolution faster.

## Gurobi versions

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

- Commercial users can have 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 instead can obtain a **free academic license** for a year at the following link, after registration:  
<https://user.gurobi.com/download/licenses/free-academic>  
The license will appear immediately on your account.  
**Note:** register with your academic e-mail address!

# Installation

- Download the latest version of Gurobi for your operative system at this link:  
<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           | <a href="#">README</a> | <a href="#">Gurobi-8.1.0-win64.msi</a> |              | <a href="#">gurobi8.1.0_linux64.tar.gz</a> | <a href="#">gurobi8.1.0_mac64.pkg</a> | <a href="#">gurobi8.1.0_power64.tar.gz</a> |

## Old versions

|       |                        |  |  |  |                                       |  |
|-------|------------------------|--|--|--|---------------------------------------|--|
| 8.0.1 | <a href="#">README</a> | <a href="#">Gurobi-8.0.1-win64.msi</a> |  | <a href="#">gurobi8.0.1_linux64.tar.gz</a> | <a href="#">gurobi8.0.1_mac64.pkg</a> | <a href="#">gurobi8.0.1_power64.tar.gz</a> |
| 7.5.2 | <a href="#">README</a> | <a href="#">Gurobi-7.5.2-win64.msi</a> | <a href="#">Gurobi-7.5.2-win32.msi</a> | <a href="#">gurobi7.5.2_linux64.tar.gz</a> | <a href="#">gurobi7.5.2_mac64.pkg</a> | <a href="#">gurobi7.5.2_power64.tar.gz</a> |
| 7.0.2 | <a href="#">README</a> | <a href="#">Gurobi-7.0.2-win64.msi</a> | <a href="#">Gurobi-7.0.2-win32.msi</a> | <a href="#">gurobi7.0.2_linux64.tar.gz</a> | <a href="#">gurobi7.0.2_mac64.pkg</a> | <a href="#">gurobi7.0.2_power64.tar.gz</a> |

# 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 the 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 this link:  
<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.

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

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

```
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 instructions at the following link:

[http://abelsiqueira.github.io/blog/  
installing-gurobi-7-on-linux/](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 the Gurobi 8.1.0 files in directory **C:\gurobi810\win64** (or **C:\gurobi810\win32** for 32-bit Windows installs).

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

#### License Detail

License ID 282200

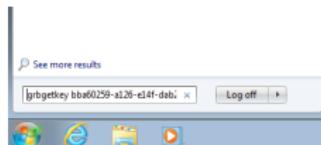
Information and installation instructions

| License ID        | 282200                       |
|-------------------|------------------------------|
| Data 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 665aaaaa-e980-11e8-a7cf-02e454ff9c50
```

- 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:

```
C:\gurobi700\win64\bin\grbgetkey.exe
Gurobi license key client (version 7.0.0)
Copyright (c) 2016, Gurobi Optimization, Inc.

-----
Contacting Gurobi key server...
-----

Key for license ID 146539 was successfully retrieved.

-----
Saving license key...
-----

In which folder would you like to store the Gurobi license key file?
[hit Enter to store it in c:\gurobi]:
--> License key saved to file 'c:\gurobi\gurobi.lic'.

Press [Enter] to exit
```

8. Press Enter to complete the activation.

## Gurobi lightweight command-line

- Firstly 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 take back the formulation of the Diet problem:
  - A variable for every portion 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:  $\min 3x_1 + 2x_2 + 3x_3 + 19x_4 + 15x_5$
  - Daily minimum 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 int declarations in the following order:
  - Objective function
  - Constraints
  - Declarations
- Here follows the **gurobi-Lpformat-diet.lp** file:

```

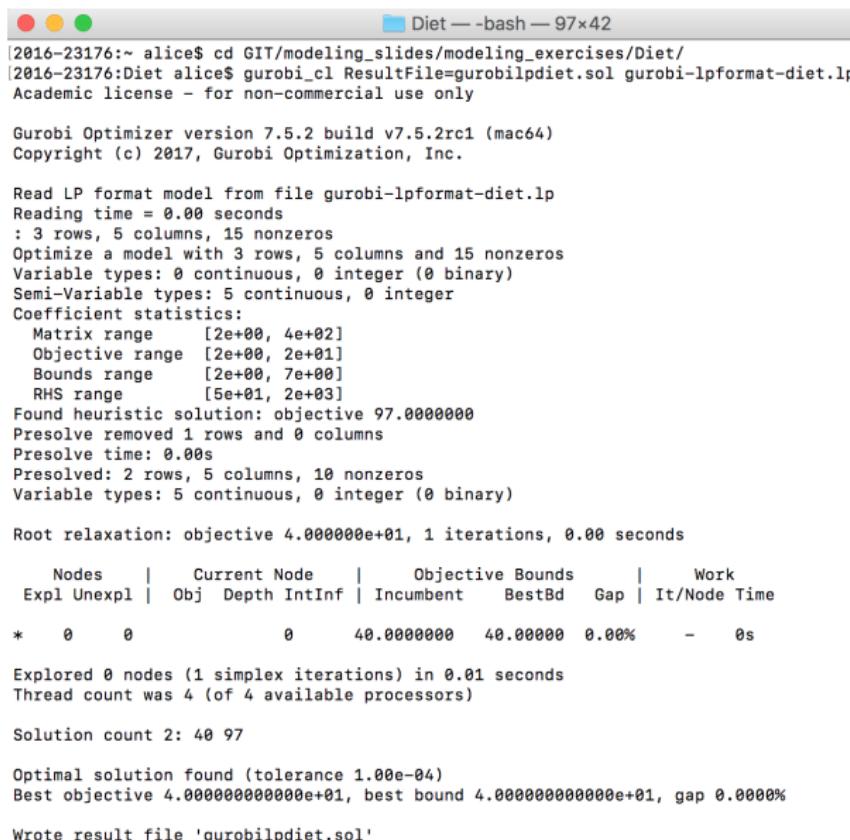
Minimize
    3 x1 + 2 x2 + 3 x3 + 19 x4 + 15 x5
Subject To
Calories: 150 x1 + 120 x2 + 160 x3 + 230 x4 + 450 x5 >= 2000
Proteins: 4 x1 + 8 x2 + 15 x3 + 14 x4 + 4 x5 >= 50
Calcium: 2 x1 + 285 x2 + 54 x3 + 80 x4 + 22 x5 >= 700
Bounds
x1 <= 4
x2 <= 7
x3 <= 2
x4 <= 3
x5 <= 2
Semi-continuous
x1 x2 x3 x4 x5
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 log and the value of the solution found, saving results in the .sol file:

```
# Objective value = 40
x1 4
x2 7
x3 2
x4 0
x5 5.333333333333333e-01
```

### 3. Gurobi logs:



```
Diet -- bash -- 97x42
[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 97.000000
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    40.000000  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

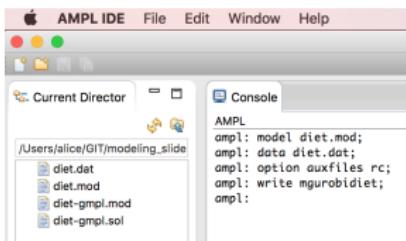
Optimal solution found (tolerance 1.00e-04)
Best objective 4.000000000000e+01, best bound 4.000000000000e+01, gap 0.0000%
```

Wrote result file 'gurobilpdiet.sol'



## Another way exploiting AMPL

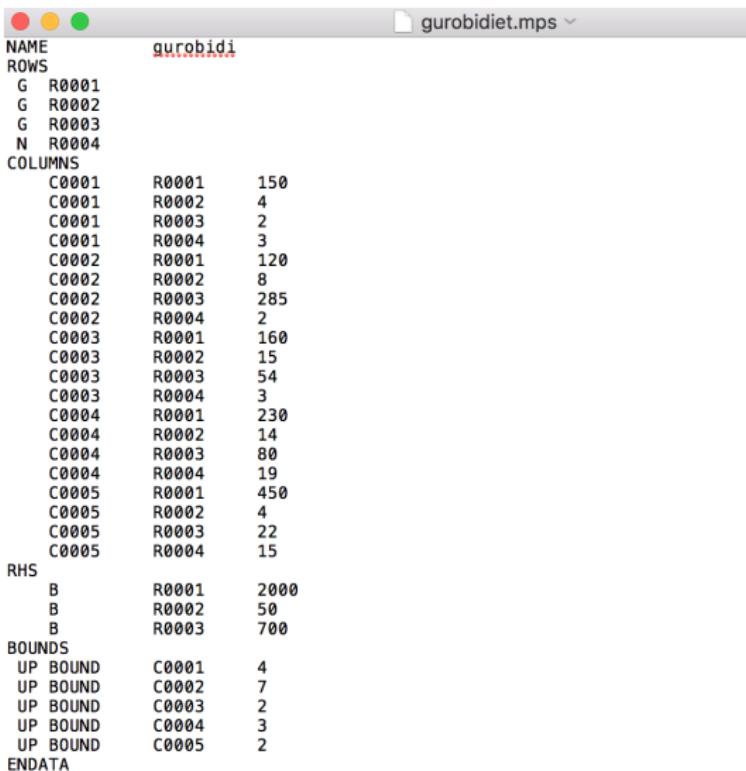
1. Launch AMPL and go to the folder where you have the .mod and .dat files of our Diet problem; in the console, write the following commands:



AMPL allows you to generate a **.mps model** with the command **write mmodelname**, saving it in the same directory. 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 will be generated: *gurobidiet.row* and *gurobidiet.col*.

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



The screenshot shows a Microsoft Word document window with the title bar "gurobidiet.mps". The document contains the following MPS file data:

```
NAME      gurobidiet
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 where you have your diet files.
4. Type **gurobi\_cl gurobidiet.mps** to launch the software and display the log in the terminal.
5. If you want to save results in an appropriate file to review the solution, instead type **gurobi\_cl ResultFile=gurobidiet.sol gurobidiet.mps**

```

Diet — -bash — 86x29
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    0.000000e+00    1.968750e+02    0.000000e+00    0s
    1    4.000000e+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:

The image shows three separate windows side-by-side, each displaying text from a Gurobi solver run.

- gurobidiet.sol**: This window contains the solution details:

```
# Solution for model gurobidiet
# Objective value = 40
C0001 4
C0002 7
C0003 2
C0004 0
C0005 5.333333333333333e-01
```
- gurobidiet.col**: This window lists the food items and their corresponding variables:

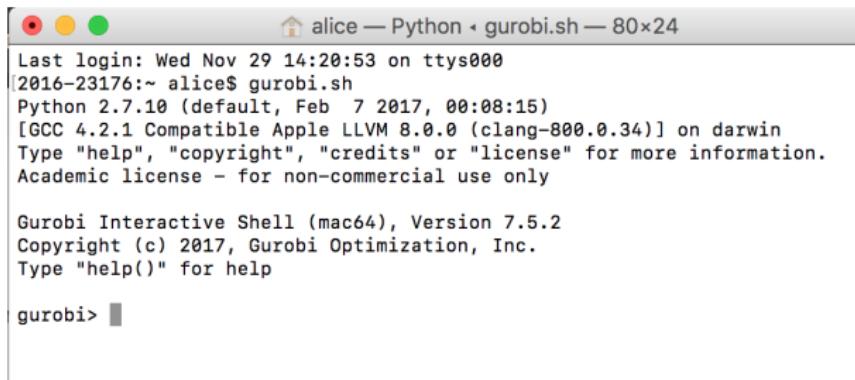
```
x['BREAD']
x['MILK']
x['EGGS']
x['MEAT']
x['SWEETS']
```
- gurobidiet.row**: This window lists the constraints and the total cost variable:

```
daily_necessities['CALORIES']
daily_necessities['PROTEINS']
daily_necessities['CALCIUM']
total_cost
```

- ## 7. As you can see, we obtained the same optimal result (the total cost is 40) and the advices about which foods and how many portions to eat.

# Using Gurobi Interactive Shell

- The Gurobi package includes a shell based on Python's one.
- 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 the models, re-optimize, and so on.
- You can type in a terminal **gurobi.sh** to open it:



The screenshot shows a terminal window titled "alice — Python · gurobi.sh — 80x24". The window contains the following text:

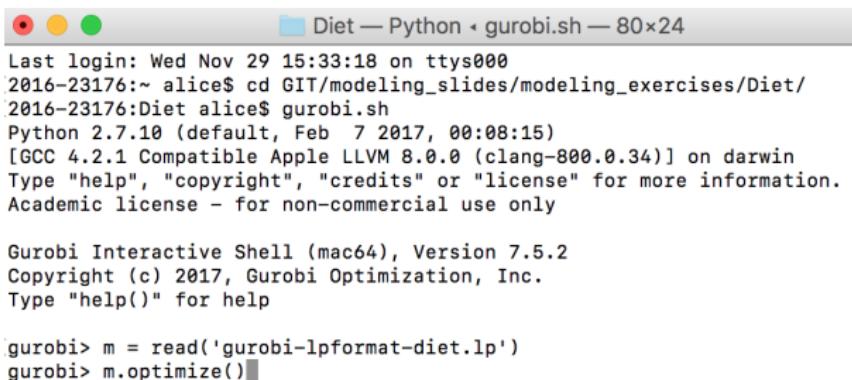
```
Last login: Wed Nov 29 14:20:53 on ttys000
[2016-23176:~ alice$ gurobi.sh
Python 2.7.10 (default, Feb  7 2017, 00:08:15)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
Academic license - for non-commercial 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 where you have saved your diet lp file that we used before.
2. Type gurobi.sh and then you can use the command **m = read('filename')** to load a model from a file and save it into the variable **m**.
3. Once read and loaded the model, run the command **m.optimize()** to solve the instance.

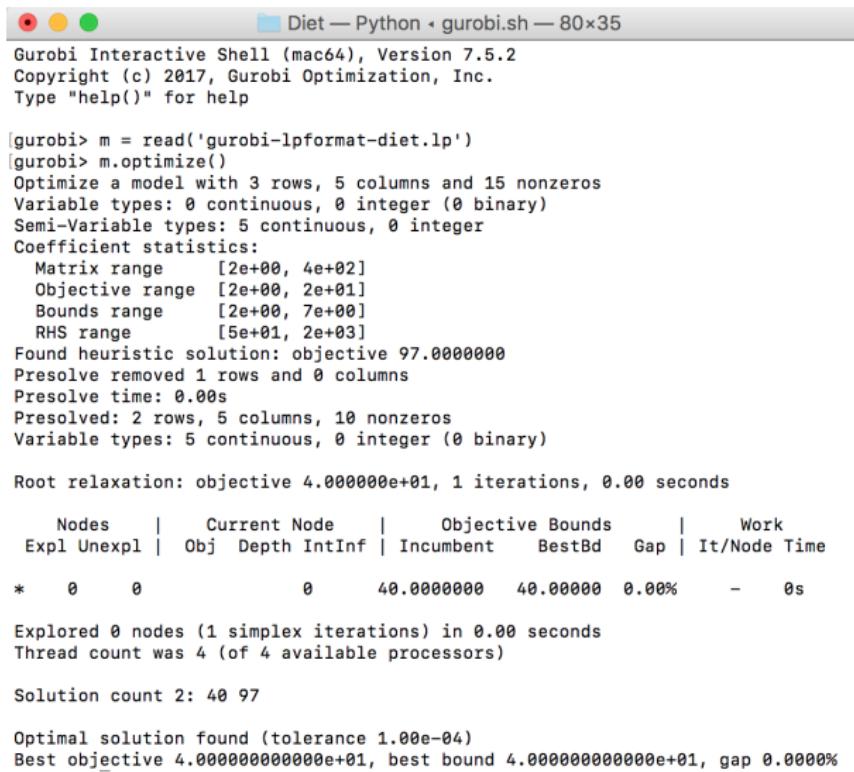


```
Last login: Wed Nov 29 15:33:18 on ttys000
2016-23176:~ alice$ cd GIT/modeling_slides/modeling_exercises/Diet/
2016-23176:Diet alice$ gurobi.sh
Python 2.7.10 (default, Feb  7 2017, 00:08:15)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
Type "help", "copyright", "credits" or "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 — 80x35
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    40.000000  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 4.00000000000e+01, best bound 4.00000000000e+01, gap 0.0000%
```

# Changing the model

- Print values of 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()**  
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 like:
  - **MIPGap** (only for MIP models):
    - the MIP solver will terminate (with an optimal result) when the gap between the lower and upper objective bound is less than MIPGap times the absolute value of the upper bound;
    - default value: 1e-4.
  - **CutOff**:
    - its value indicates that you aren't interested in solutions whose objective values are worse than the specified 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, saying that 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 the link: <http://www.gurobi.com/documentation/8.1/refman/parameters.html> .

## Gurobi and the Python interface

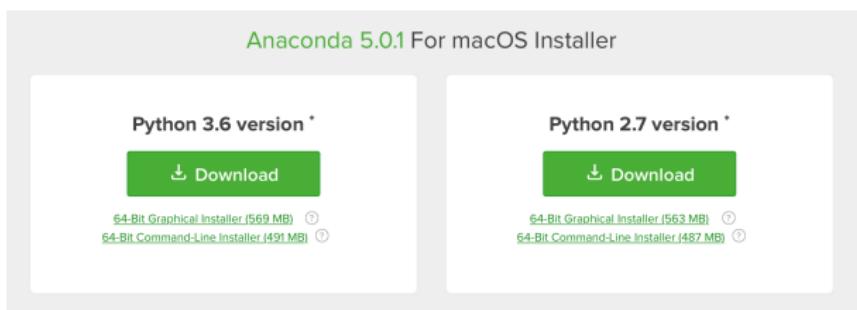
- We are going to use Python as our programming language. If you need help with Python syntax and rules, look in the References for some useful links.
- There is already a Python interpreter and a basic set of Python modules in Gurobi but, in order to increase interactivity and productivity of Python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



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

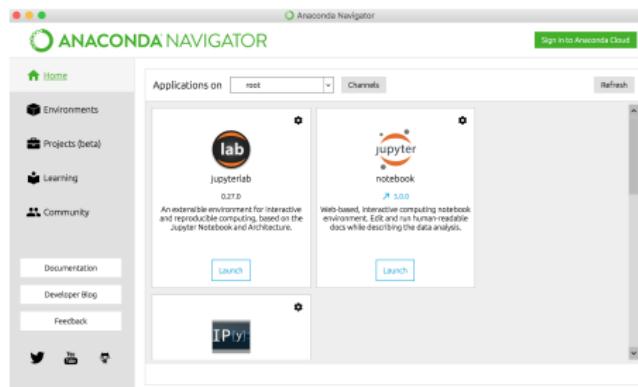
# Anaconda for Mac OS X

1. Download Anaconda 5.0.1 for macOS Installer at the following link: **<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**).

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 install is complete, verify it by opening Anaconda Navigator, a program that is included with Anaconda: from Launchpad, select Anaconda Navigator.



5. You can also open a terminal and type **python**, getting this:

```
alice - python - 80x24
Last login: Wed Nov 29 14:21:25 on ttys000
2016-23176:~ slice$ python
Python 2.7.14 |Anaconda, Inc.| (default, Oct  5 2017, 02:28:52)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/finall) on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

The image shows a terminal window with a light gray background and a dark gray header bar. The header bar contains the text 'alice - python - 80x24' and three colored window control buttons (red, yellow, green). The main terminal area displays a standard Python 2.7.14 prompt. The text in the terminal is white on a black background. At the bottom of the terminal window, there is a set of small, semi-transparent navigation icons used for navigating through the command history and the terminal window itself.

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**.

```
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

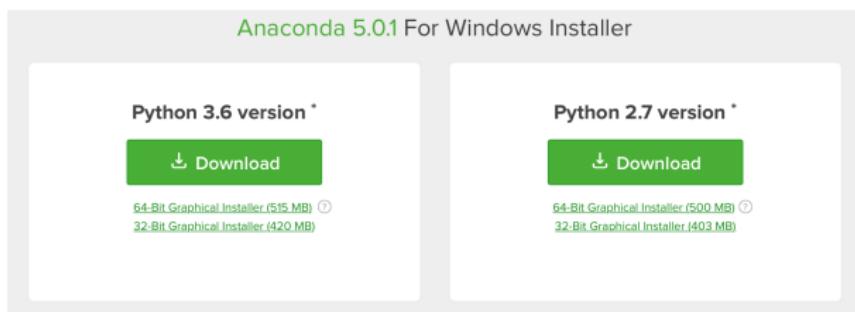
Proceed ([y]/n)? y
gurobi-7.5.2-p 100% |#####| Time: 0:00:48 324.58 kB/s
2016-23176:~ alice$
```

## Anaconda for Windows

1. Download Anaconda 5.0.1 for Windows Installer at the following link:

**<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**).

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 it finishes, verify the install by opening Anaconda Navigator, a program that is included with Anaconda: from your Windows Start menu, select the shortcut Anaconda Navigator.

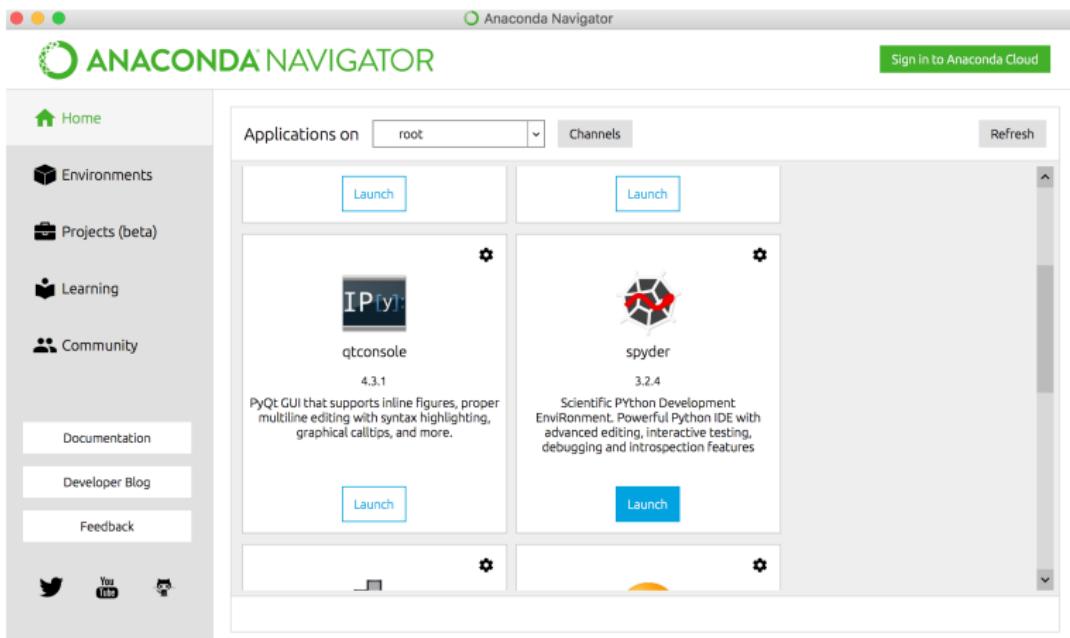
7. To install the Gurobi package into Anaconda, from your Windows Start menu open the Anaconda command prompt and follow these steps:
  - 7.1 Add the Gurobi channel to your Anaconda channels: **conda config –add channels <http://conda.anaconda.org/gurobi>**
  - 7.2 Install the package: **conda install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

## Note about Python environment

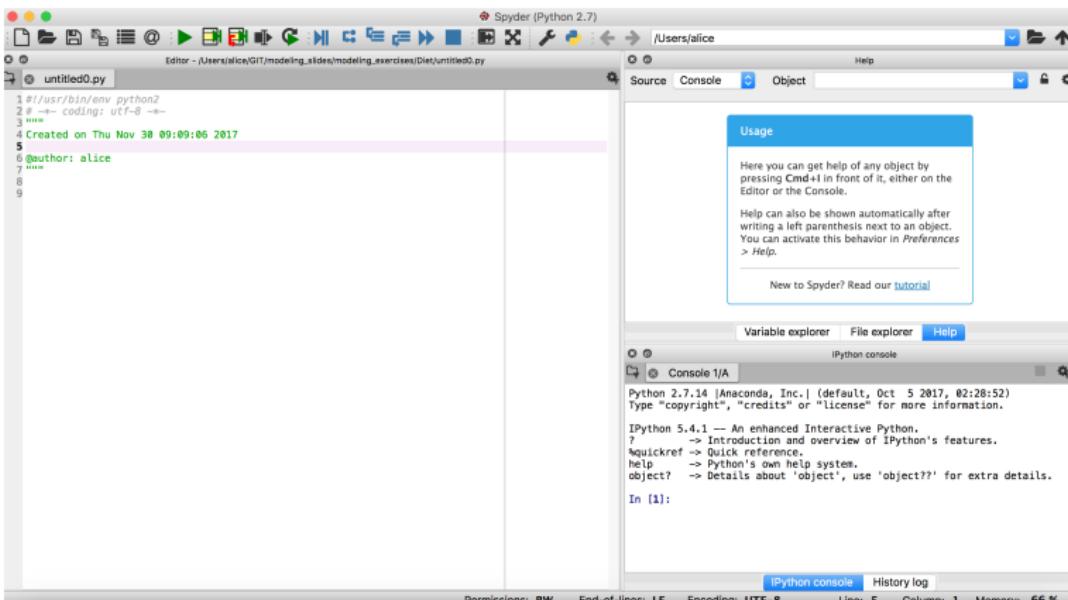
- Only if you prefer to use another environment different from Anaconda, that you had already installed before, you must add the package **gurobipy** to be able to use Gurobi.
- In Linux and Mac OS X systems, open a terminal, go to **Library/gurobi810/mac64/** and type **python setup.py install** to link Gurobi to your environment.
- In Windows systems, open command prompt, go to **C:\gurobi810\winXX** and type **python setup.py install**.
- After the installation of gurobipy package, you can type **import gurobipy** or **from gurobipy import \*** from your Python shell and access all of the 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 Gurobi module with libraries: **from gurobipy import \*** or **import gurobipy** (not required 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 with Gurobi module: **m = Model("modelname")**.

```
# Solve the diet problem

from gurobipy import *

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

6. Then we need to define 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 you have a huge number of variables).

8. We define the objective function of the model, 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 first argument is the expression, whereas the second is the purpose, that could be GRB.MINIMIZE or GRB.MAXIMIZE.

9. To add constraints about the minimum daily necessities of calories, proteins and calcium, we can use the method **m.addConstr(...)** or, if we know that there is a range to respect, with a minimum and maximum value, we can use **m.addRange(...)**, like 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 one are respectively the lower and the upper bound, the last one is the name given to the constraint.

10. Now we add everything we need to solve the model; 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 model status and prints the solution only if it is optimal.

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, and then **printSolution()**:

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

12. To run the code, press the green arrow in the top menu or press 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    0.000000e+00    1.968750e+02    0.000000e+00    0s
        1    4.000000e+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 solve the problem 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:

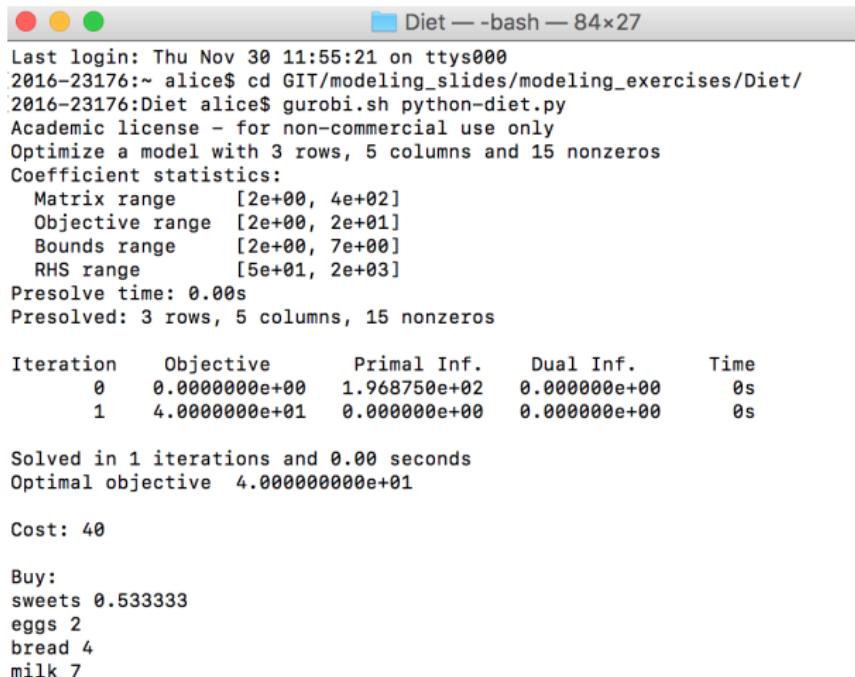
```
Adding constraint: at most 6 servings of milk and eggs
Optimize a model with 4 rows, 5 columns and 17 nonzeros
Coefficient statistics:
    Matrix range      [1e+00, 4e+02]
    Objective range   [2e+00, 2e+01]
    Bounds range     [2e+00, 7e+00]
    RHS range        [6e+00, 2e+03]
    Iteration          0      Objective       Primal Inf.    Dual Inf.      Time
                      0  4.0000000e+01  4.8000000e+01  0.0000000e+00   0s
                      1  4.6000000e+01  0.0000000e+00  0.0000000e+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 like and solve the model using the Gurobi Interactive Shell:



```
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

The screenshot shows a Jupyter Notebook window titled "ipython Notebook Example". It contains two code cells and one output cell.

```
In [45]: from gurobipy import *
m = Model()
v0 = m.addVar(r)
v1 = m.addVar(r)
m.update()
m.addConstr(v0 - v1 == 4) # Constraint 1
m.addConstr(v0 + v1 == 4) # Constraint 2
m.addConstr(v0 + v1 == 11) # Constraint 3
m.setObjective(v1, GRB.MAXIMIZE) # objective: maximize v2
m.params.outputFlag = 0
m.optimize()
```

Plot the optimal solution...

```
In [46]: import matplotlib.pyplot as plt
plt.plot([0,4], [0,4]) # constraint 1
plt.plot([0,4], [11,4]) # constraint 2
plt.plot([0,4], [1,2]) # constraint 3
plt.plot([v0.x], [v1.x], 'ro') # plot the optimal vertex
plt.show()
```

The plot shows a 2D coordinate system with axes ranging from 0 to 4. Three lines are plotted: a red line from (0,0) to (4,4), a green line from (0,11) to (4,4), and a blue line from (0,0) to (1,2). The intersection of these lines at (2.5, 1.5) is highlighted with a red dot, representing the optimal vertex found by the solver.

- Notebook-style interface to mix executable code, text and graphics, to create a self-documenting stream of results;
- We are not going to use it in the course  
(but it could be part of your project...)

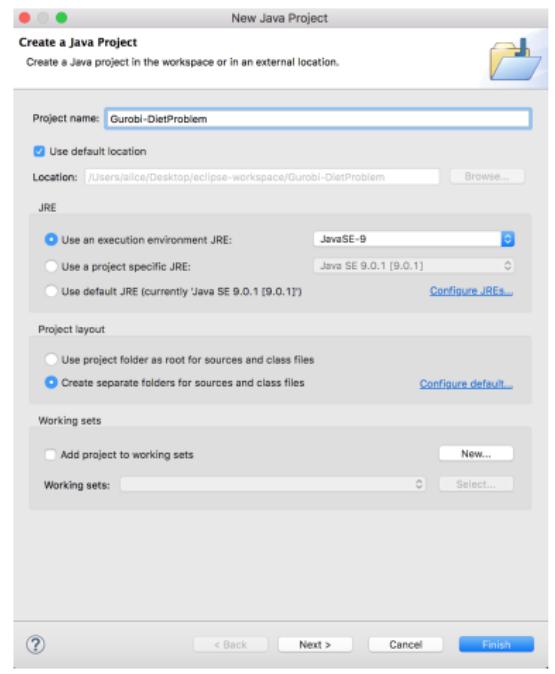
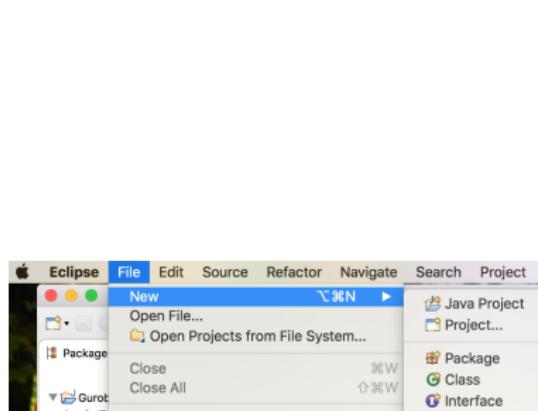
## Gurobi and the Java interface



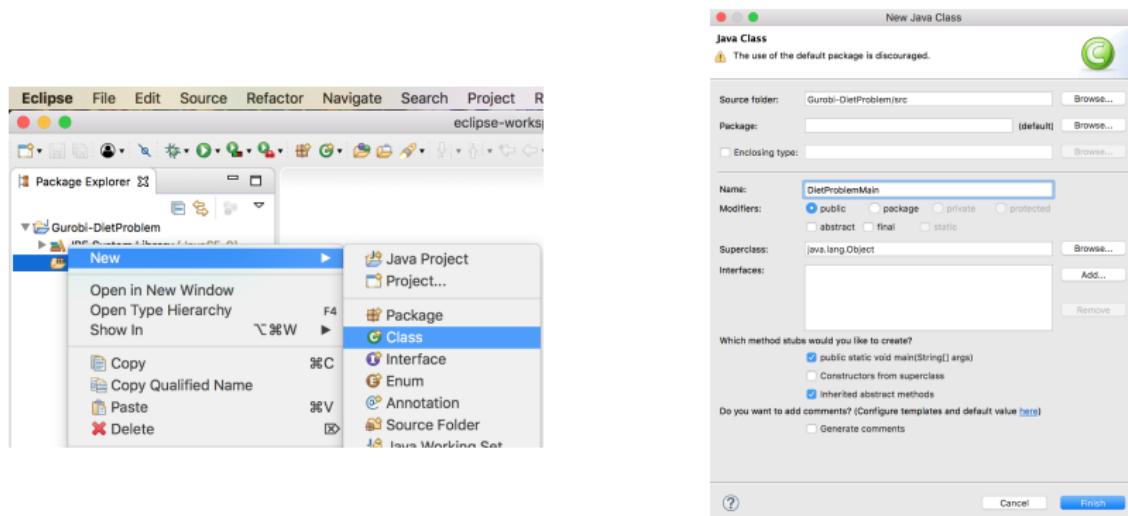
- In this optional part, we are going to use **Eclipse**, one of the most common integrated development environment for Java, after importing the Gurobi library into the Java project.
- You can download and then install Eclipse at the following link: <http://www.eclipse.org/downloads/>.

# The Diet Problem with Eclipse

1. Launch Eclipse and create a new Java project named **Gurobi-DietProblem**:



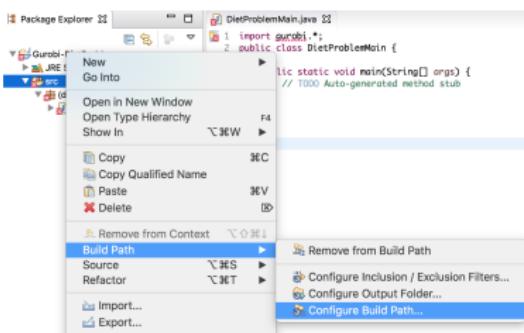
## 2. Add a new class named DietProblemMain:



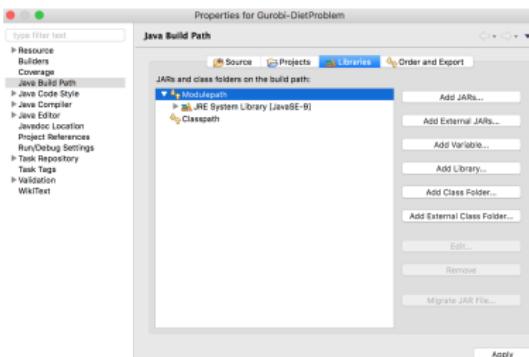
## 3. At the beginning add the instruction **import gurobi.\*** that will show an error because the library which refers is missing:

```
*DietProblemMain.java
1 import gurobi.*;
2 public class DietProblemMain {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6
7     }
8
9 }
```

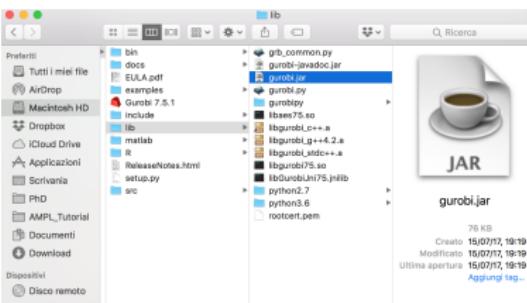
4. To import Gurobi library in Eclipse, double-click on the project and select **Build Path** and then **Configure Build Path...** :



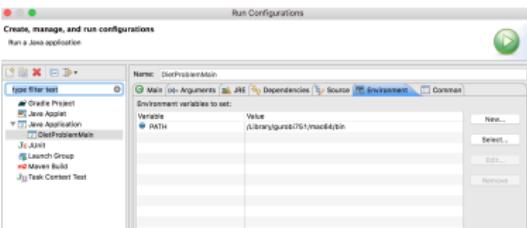
5. In the Libraries tab, click on Modulepath and then click on **Add External JARs...** :



- Go to the Gurobi folder (e.g., on Mac OS X, /Library/gurobi751/mac64/lib) and select **gurobi.jar** to be added.



- Once added, go to **Run → Run Configurations...**; on the left menu, select Java Application and then DietProblemMain; go to the Environment tab and add the variable PATH with the directory of Gurobi bin folder as value.



8. The error on the instruction should disappear, because now the project is linked to the Gurobi library.
9. Now we can write our code for the Diet Problem, starting with the section including sets and parameters, using arrays and matrixes as main data structures:

```
public static void main(String[] args) {  
    try {  
  
        // Set of substances  
        String Substances[] = new String[] { "CALORIES", "PROTEINS", "CALCIUM" };  
        int nSubstances = Substances.length;  
        double minNutrition[] = new double[] { 2000, 50, 700 };  
  
        // Set of foods  
        String Foods[] = new String[] { "BREAD", "MILK", "EGGS", "MEAT", "SWEETS" };  
        int nFoods = Foods.length;  
        double cost[] = new double[] { 3, 2, 3, 19, 15 };  
  
        // Maximum number of portions for each food  
        int maxNbPortions[] = new int[] { 4, 7, 2, 3, 2 };  
  
        // Nutrition values for one portion of each food  
        double amountPortion[][] = new double[][] {  
            { 150, 120, 160, 230, 450 }, // CALORIES  
            { 4, 8, 15, 14, 4 }, // PROTEINS  
            { 2, 285, 54, 80, 22 }, // CALCIUM  
        };
```

10. We need to define the **environment** and create an instance of the class **GRBModel**, in order to build one:

```
// Model
GRBEnv env = new GRBEnv();
GRBModel model = new GRBModel(env);
model.set(GRB.StringAttr.ModelName, "diet");
```

11. Optimization models live within an environment: as a good practice, every programme should create (and in the end destroy/dispose) just one environment.
12. The model instance is created passing as parameter the environment, therefore is strictly linked to it. Set the model String attribute **ModelName** with an appropriate name.

**13.** Create appropriate decision variables and add constraints to the model through methods **addVar** and **addConstr**:

```
// Model
GRBEnv env = new GRBEnv();
GRBModel model = new GRBModel(env);
model.set(GRB.StringAttr.ModelName, "diet");

// Create decision variables for the nutrition information,
// which we limit via bounds
GRBVar[] nutrition = new GRBVar[nSubstances];
for (int i = 0; i < nSubstances; ++i) {
    nutrition[i] =
        model.addVar(minNutrition[i], GRB.INFINITY, 0, GRB.CONTINUOUS,
                    Substances[i]);
}

// Create decision variables for the foods to buy
GRBVar[] buy = new GRBVar[nFoods];
for (int j = 0; j < nFoods; ++j) {
    buy[j] =
        model.addVar(0, maxNbPortions[j], cost[j], GRB.CONTINUOUS, Foods[j]);
}

// Nutrition constraints
for (int i = 0; i < nSubstances; ++i) {
    GRBLinExpr ntot = new GRBLinExpr();
    for (int j = 0; j < nFoods; ++j) {
        ntot.addTerm(amountPortion[i][j], buy[j]);
    }
    model.addConstr(ntot, GRB.EQUAL, nutrition[i], Substances[i]);
}

// The objective is to minimize the total costs
model.set(GRB.IntAttr.ModelSense, GRB.MINIMIZE);
```

**14.** Set the objective of the problem.

## 15. Solve the problem and dispose *env* and *model*:

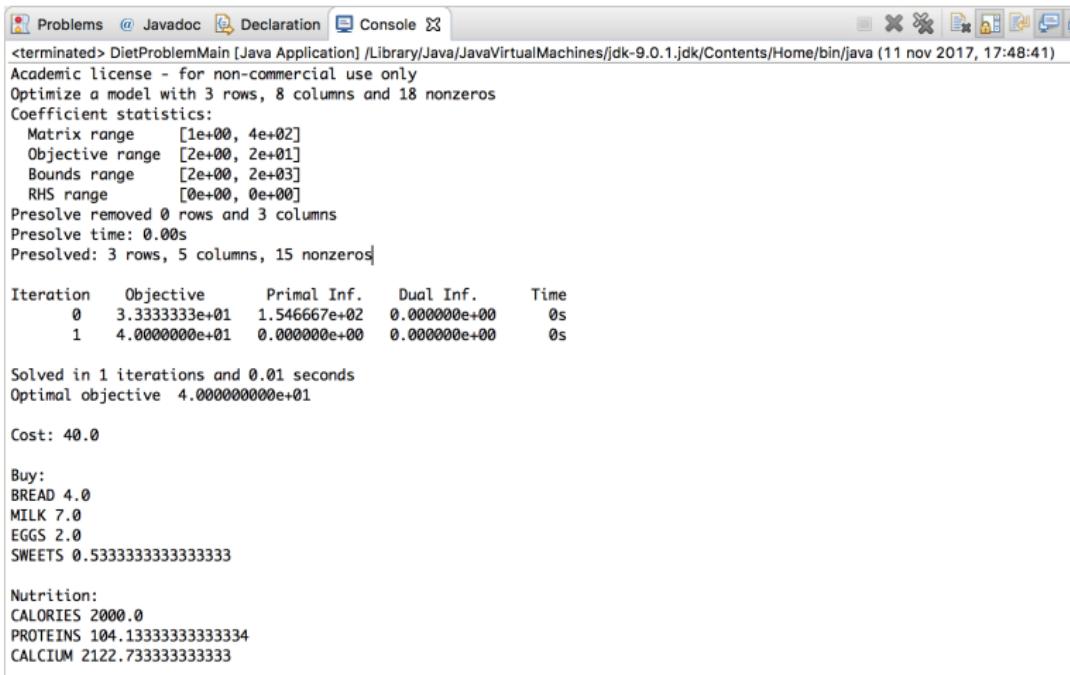
```
// Solve
model.optimize();
printSolution(model, buy, nutrition);

// Dispose of model and environment
model.dispose();
env.dispose();
```

## 16. The procedure **printSolution** was developed on purpose and it prints as output what you need to know about the resolution.

```
private static void printSolution(GRBModel model, GRBVar[] buy,
                                 GRBVar[] nutrition) throws GRBException {
    if (model.get(GRB.IntAttr.Status) == GRB.Status.OPTIMAL) {
        System.out.println("\nCost: " + model.get(GRB.DoubleAttr.ObjVal));
        System.out.println("\nBuy:");
        for (int j = 0; j < buy.length; ++j) {
            if (buy[j].get(GRB.DoubleAttr.X) > 0.0001) {
                System.out.println(buy[j].get(GRB.StringAttr.VarName) + " " +
                                   buy[j].get(GRB.DoubleAttr.X));
            }
        }
        System.out.println("\nNutrition:");
        for (int i = 0; i < nutrition.length; ++i) {
            System.out.println(nutrition[i].get(GRB.StringAttr.VarName) + " " +
                               nutrition[i].get(GRB.DoubleAttr.X));
        }
    } else {
        System.out.println("No solution");
    }
}
```

## 17. And this is the output:



The screenshot shows a Java application window titled "DietProblemMain [Java Application]". The console tab displays the following output:

```
<terminated> DietProblemMain [Java Application] [/Library/Java/JavaVirtualMachines/jdk-9.0.1.jdk/Contents/Home/bin/java (11 nov 2017, 17:48:41)]
Academic license - for non-commercial use only
Optimize a model with 3 rows, 8 columns and 18 nonzeros
Coefficient statistics:
    Matrix range      [1e+00, 4e+02]
    Objective range   [2e+00, 2e+01]
    Bounds range      [2e+00, 2e+03]
    RHS range         [0e+00, 0e+00]
Presolve removed 0 rows and 3 columns
Presolve time: 0.00s
Presolved: 3 rows, 5 columns, 15 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.    Time
      0    3.3333333e+01    1.546667e+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

Cost: 40.0

Buy:
BREAD 4.0
MILK 7.0
EGGS 2.0
SWEETS 0.5333333333333333

Nutrition:
CALORIES 2000.0
PROTEINS 104.1333333333334
CALCIUM 2122.733333333333
```

# Solving TSP with Gurobi



- We can use another easy approach in Gurobi too.
- We let Gurobi solve the problem and, when it finds a new MIP solution, we look for **connected components**:
  - We will compute the shortest (sub)tour of vertices that the solution visits;
  - If the number of vertices in the tour is equal to the total number of vertices in  $V$ , the solution is fine; otherwise it means that there are subtours.
- To do so, we need to use **Callbacks** and **Lazy Constraints**.

## Callbacks

- Callbacks are 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 each case, develop appropriate code.
- In our case, we want to check subtours everytime a new MIP solution is found: the related value of *where* is 4.

# Lazy Constraints

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

```
model.params.LazyConstraints = 1
```

# Gurobi TSP Example

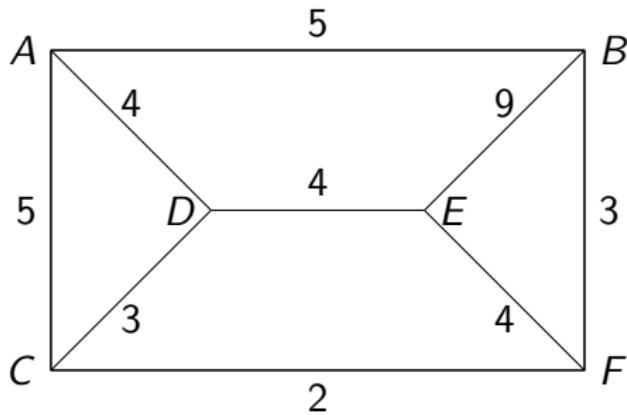
- You can find a TSP example on Gurobi website (<http://examples.gurobi.com/traveling-salesman-problem/>), that solves the problem considering  $n$  random points in the US, minimizing their Eulerian distances:

The screenshot shows the Gurobi Optimization website with the following details:

- Header:** GUROBI OPTIMIZATION, PRODUCTS, DOWNLOADS, RESOURCES, ACADEMIA, SUPPORT, ABOUT, Get Gurobi.
- Breadcrumbs:** Home > Interactive Examples > Traveling Salesman Problem
- Left Sidebar:** Intro (selected), Problem, Model, Implementation, Live Demo, Try Gurobi for Free.
- Main Content:**
  - Title:** The Traveling Salesman Problem with integer programming and Gurobi
  - Description:** In this example we'll solve the Traveling Salesman Problem.
  - Text:** We'll construct a mathematical model of the problem, implement this model in Gurobi's Python interface, and compute and visualize an optimal solution.
  - Note:** Although your own business may not involve traveling salesmen, the same basic techniques used in this example can be used for many other applications like vehicle routing, circuit design and DNA sequencing.
  - Call-to-action:** Click the screenshot to skip directly to the Live Demo!
- Image:** A map of the United States with several cities marked by black dots and connected by a purple line, illustrating a possible tour path.
- Page Navigation:** Back, Forward, Home, Search, etc.

## 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:

```
9 import math
10 import random
11 from gurobipy import *
```

2. Define the sets of vertices and edges:

```
66 # _____
67 # MODEL DEFINITION
68
69 # Create vertices and edges for our example
70 vertices = [0,1,2,3,4,5] #vertices = ['A', 'B', 'C', 'D', 'E', 'F']
71 # We do not use letters just to avoid issues with indexing and syntax errors
72 edges = {
73     (0,1): 5,
74     (0,2): 5,
75     (0,3): 4,
76     (1,4): 2, #9
77     (1,5): 3,
78     (2,3): 3,
79     (2,5): 8,
80     (3,4): 4,
81     (4,5): 4}
82
83 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.

```
85 # Create the model
86 m = Model()
87
88 # Create the variables
89 vars = {}
90 for i,j in edges.keys():
91     vars[i,j] = m.addVar(obj=edges[i,j], vtype=GRB.BINARY,
92                           name='e[%d,%d]'%(i,j))
93 for i,j in vars.keys():
94     vars[j,i] = vars[i,j] # Edges in both directions
95
96 # To create the model data structure only once, after variables creation
97 m.update()
98
99 # Add the objective function
100 m.setObjective(sum(vars[i,j]*edges[i,j]
101                     for (i,j) in edges.keys()),GRB.MINIMIZE)
102
103 # Add degree-2 constraint
104 for i in vertices:
105     m.addConstr(sum(vars[i,j] for j in vertices
106                      if (i,j) in edges.keys() or (j,i) in edges.keys()) == 2)
107
```

## A useful function: Subtour

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

```
41 # Given a list of edges, this method finds the shortest subtour
42 def subtour(edges):
43     visited = [False]*n
44     cycles = []
45     lengths = []
46     selected = [[] for i in vertices]
47     for x,y in edges:
48         selected[x].append(y)
49         selected[y].append(x) # Edges in both directions
50     while True:
51         current = visited.index(False)
52         thiscycle = [current]
53         while True:
54             visited[current] = True
55             neighbors = [x for x in selected[current] if not visited[x]]
56             if len(neighbors) == 0:
57                 break
58             current = neighbors[0]
59             thiscycle.append(current)
60             cycles.append(thiscycle)
61             lengths.append(len(thiscycle))
62             if sum(lengths) == n:
63                 break
64     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 solving the problem without adding Subtour Elimination Constraints:

```
108 # -----
109 # MODEL OPTIMIZATION
110
111 # Without SECs
112 print '\n-----\n'
113 print 'Optimize model without SECs\n'
114 m._vars = vars
115 m.optimize()
116 # Print optimal solution
117 print '\n-----\n'
118 solution = m.getAttr('x', vars)
119 selected = [(i,j) for i,j in edges.keys() if solution[i,j] > 0.5]
120 print 'Edges in solution: ' + str(selected)
121 print('Optimal tour: %s' % str(subtour(selected)))
122 print('Optimal cost: %g' % m.objVal)
123
```

## 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 26.000000

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     |                  | 21.000000 | 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 2.10000000000e+01, best bound 2.10000000000e+01, gap 0.0000%

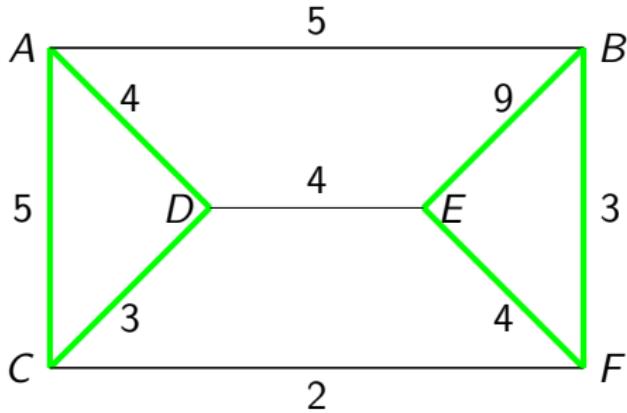
Edges in solution: [(1, 5), (2, 3), (4, 5), (0, 3), (0, 2), (1, 4)]

Optimal tour: [0, 3, 2]

Optimal cost: 21

**Note:** the optimal subtour visits just three vertices!

9. We do not like the solution we obtained:



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

## Model optimization - Part II

10. We define the function which Gurobi will call automatically when it finds a new MIP incumbent solution:

```
16 # Callback: use lazy constraints to eliminate subtours
17 def subtourelim(model, where):
18     if where == GRB.callback.MIPSOL:
19         print '\nNew solution found: checking the presence of subtours...'
20         selected = []
21         # Make a list of edges selected in the solution
22         vals = model.cbGetSolution(model._vars)
23         selected = tuplelist((i,j) for i,j in model._vars.keys()
24                               if vals[i,j] > 0.5)
25
26         # Find the shortest cycle in the selected edge list
27         tour = subtour(selected)
28         if len(tour) < n:
29             # Add a subtour elimination constraint
30             print 'One subtour found: ' + str(tour)
31             expr = 0
32             for i in range(len(tour)):
33                 for j in range(i+1, len(tour)):
34                     expr += model._vars[tour[i], tour[j]]
35             print 'Subtour Elimination Constraint added:'
36             print str(expr) + ' <= ' + str(len(tour)-1)
37             model.cbLazy(expr <= len(tour)-1)
38         else:
39             print 'No subtour found!'
```

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

```
124 # With Lazy Constraints
125 print '\n-----\n'
126 print 'Adding Lazy Constraints\n'
127 m.reset()
128 m.params.LazyConstraints = 1
129 m.optimize(subtourelim)
130
131 # Print optimal solution
132 print '\n-----\n'
133 solution = m.getAttr('x', vars)
134 selected = [(i,j) for i,j in edges.keys() if solution[i,j] > 0.5]
135 print 'Edges in solution: ' + str(selected)
136 print('Optimal tour: %s' % str(subtour(selected)))
137 print('Optimal cost: %g' % m.objVal)
138
139 # Check if the solution has only one connected component
140 assert len(subtour(selected)) == n
```

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

## 12. This is the output using 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 26.000000

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 callback and finds a subtour composed of vertices 0, 2 and 3  
→ It adds to the current formulation  $e_{0,3} + e_{0,2} + e_{2,3} \leq 2$ .

### 13. Here is the right optimal solution:

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

| Nodes | Expl | Unexpl | Current Node Obj | Depth | IntInf | Objective  | Incumbent | BestBd   | Gap   | Work | It/Node | Time |
|-------|------|--------|------------------|-------|--------|------------|-----------|----------|-------|------|---------|------|
| *     | 0    | 0      |                  | 0     |        | 24.0000000 | 24.00000  | 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 2.40000000000e+01, best bound 2.40000000000e+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 it is possible to solve ILP problems using Gurobi Optimizer, both exploiting its command-line options (lightweight version and Interactive Shell) and its libraries.
- In particular, we solved the Diet problem and the TSP, using also advanced features as lazy constraints.
- Gurobi can interface not only with Python and Java, but also with C, C++, MATLAB and R: why not trying to do these exercises with other languages?

## 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](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](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](http://www.or.deis.unibo.it/algottm/files/8_ATSP.pdf)