Python + Gurobi 使用简介

Gurobi 简介

Gurobi是什么?

Gurobi是由美国 Gurobi Optimization 公司开发新一代大规模优化器。

技术优势

- 问题尺度：只受限制于计算机内存容量，不对变量数量和约束数量有限制
- 接口：支持 C++, Java, Python, .Net, Matlab 和R
- 平台：支持包括 Windows, Linux, Mac OS X

线性混合整数规划 MILP								
1 线程	**CBC**	**CPLEX**	**GUROBI**	**SCIPC**	**SCIPS**	**XPRESS**	**MATLAB**	**SAS**
速度比例	39	1.74	1	5.75	7.94	2	72.2	2.9
解决问题数量	53	87	87	83	76	86	32	84
4 线程	**CBC**	**CPLEX**	**FSCIPC**	**FSCIPS**	**GUROBI**	**XPRESS**	**MIPCL**	**SAS**
速度比例	34.8	1.5	9.9	12.1	1	1.66	7.29	3
解决问题数量	66	86	80	79	87	87	84	85
12 线程	**CBC**	**CPLEX**	**FSCIPC**	**FSCIPS**	**GUROBI**	**XPRESS**	**MIPCL**	**SAS**
速度比例	27	1.49	9.8	13	1	1.57	6.53	3.39
解决问题数量	69	87	78	76	87	87	82	82
速度比例为1是最快的速度，其他数值为该速度的倍数。								

支持的模型类型

- 连续和混合整数线性问题
- 凸目标或约束连续和混合整数二次问题
- 非凸目标或约束连续和混合整数二次问题
- 含有对数、指数、三角函数、高阶多项式目标或约束，以及任何形式的分段约束的非线性问题
- 含有绝对值、最大值、最小值、逻辑与或非目标或约束的非线性问题

Gurobi 安装

三种安装方法

- **Pip 安装**（仅支持 Gurobi 9.1 或以后版本）

```
python -m pip install -i https://pypi.gurobi.com gurobipy
或者 pip install -i https://pypi.gurobi.com gurobipy
```

将Gurobi 模块（非Gurobi 完整安装包）安装到当前激活的Python 环境中。

- **官网下载 Gurobi 完整安装包**

- **Anaconda 安装**

```
conda install -c gurobi gurobi
```

Academic Licenses 获取

南京大学 NANJING UNIVERSITY 工程管理学院 SCHOOL OF MANAGEMENT & ENGINEERING

1. 连接校园网或学校VPN

2. 进入Gurobi官网学术许可注册界面

https://www.gurobi.com/downloads/end-user-license-agreement-academic/

填写注册信息

Register for Free

Are you looking for a better optimization solver, with superior support, and a lower end-to-end cost than the leading alternatives?

If so, you've come to the right place.

When you register for an account, you'll get:

- Access to free academic licenses,
- Webinar invitations,
- Product updates and more.

Start your registration by designating your account type as either Commercial or Academic:

Are you an Academic or Commercial user?	*	Academic
First Name:	*	
Last Name:	*	
Company Email Address:	*	
University:	*	
Academic Position:		Select...
Phone Number:	*	
Country:	*	Select...

填写学校邮箱

Check this box if you also consult with commercial businesses: ☐

*Required: The information you provide to us will be used in accordance with the terms of our Privacy Policy.

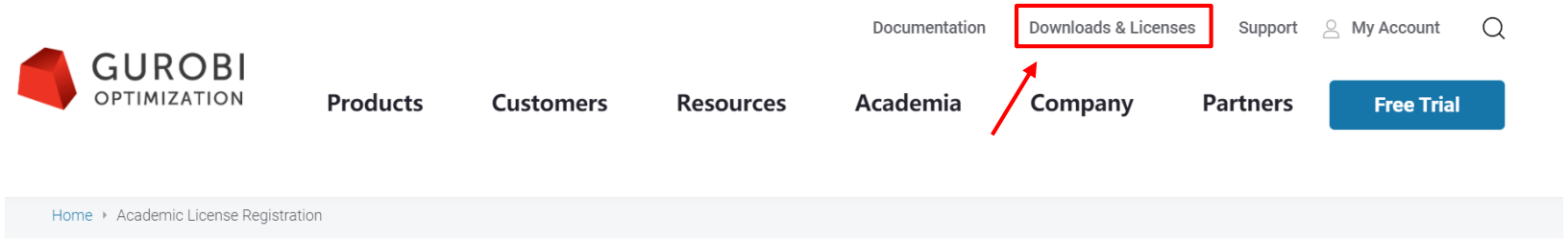
Access Now

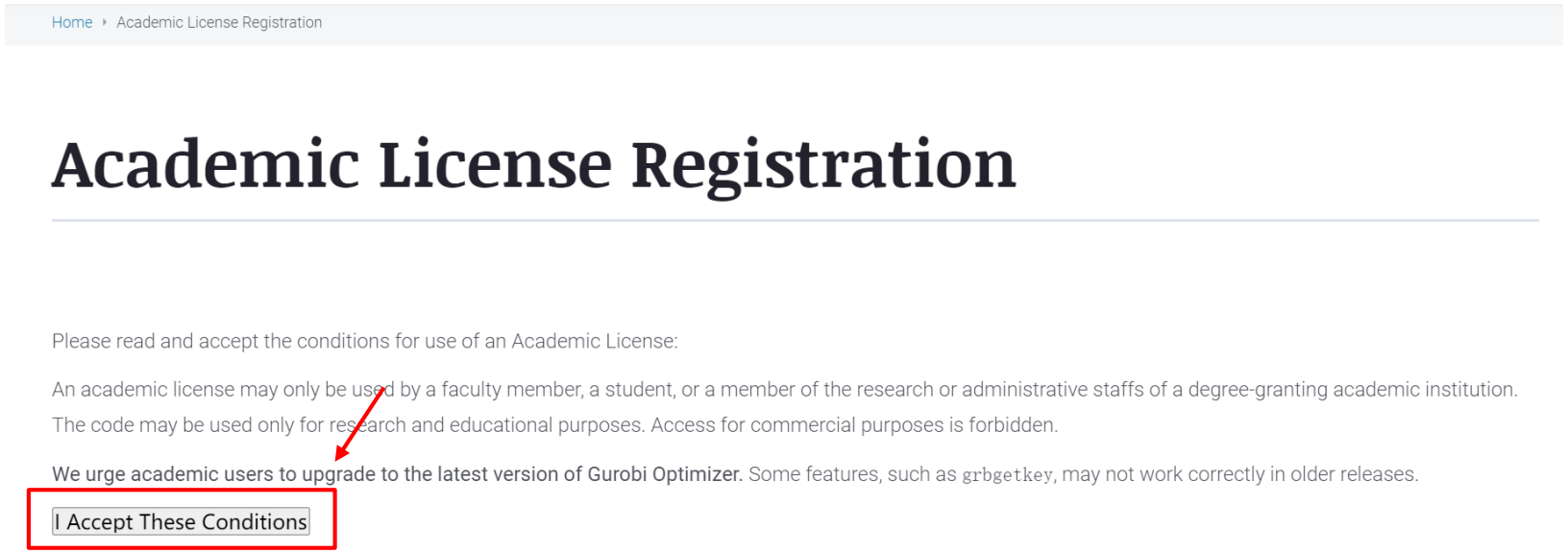
Academic Licenses 获取

3. 根据收到的邮件验证邮箱地址，并设置密码

4. 回到官网，登录账号

5. 申请Academic License





Academic Licenses 获取

南京大学 NANJING UNIVERSITY 工程管理学院 SCHOOL OF MANAGEMENT & ENGINEERING

6. 得到 Liscense ID、grbgetkey

License Details

Information and installation instructions

License ID	516013
Date issued	2020-10-28T22:25:41
Purpose	Trial
License Type	TRIAL
Key Type	TRIAL
Version	9
Expiration Date	2021-04-26
Distributed Limit	0
Host Name	
Host ID	

复制下来

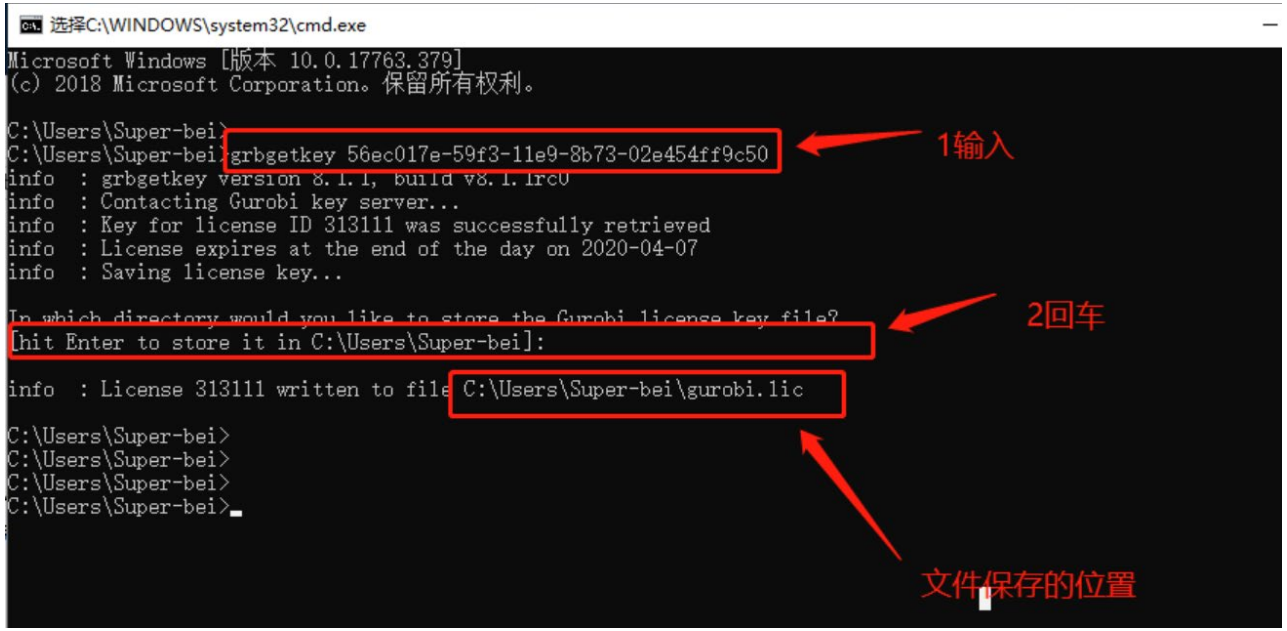
Installation

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):

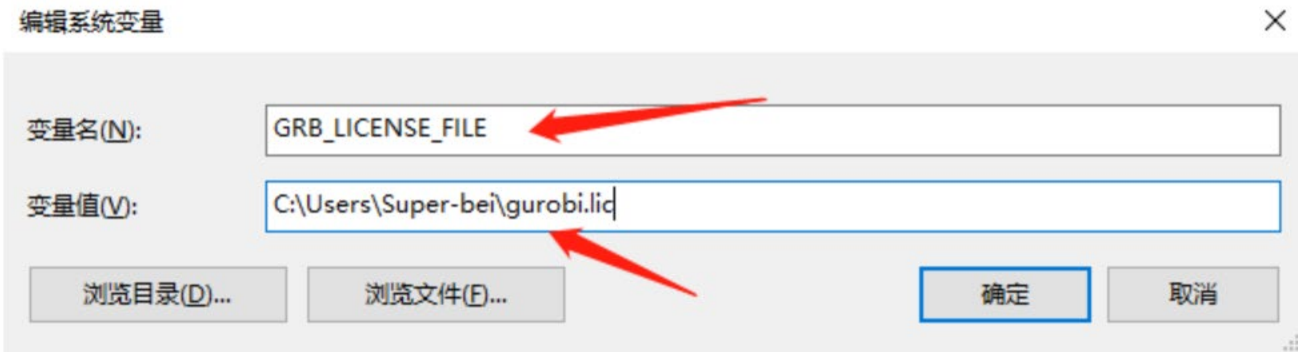
```
grbgetkey 83af988a-196c-11eb-865d-0a7c4f30bdbe
```

The `grbgetkey` command requires an active internet connection. If your computer has no internet access, or you get no response or an error message such as "Unable to contact key server", Please click here for additional instructions.

7. 打开cmd命令行，激活Liscense – 输入grbgetkey内容

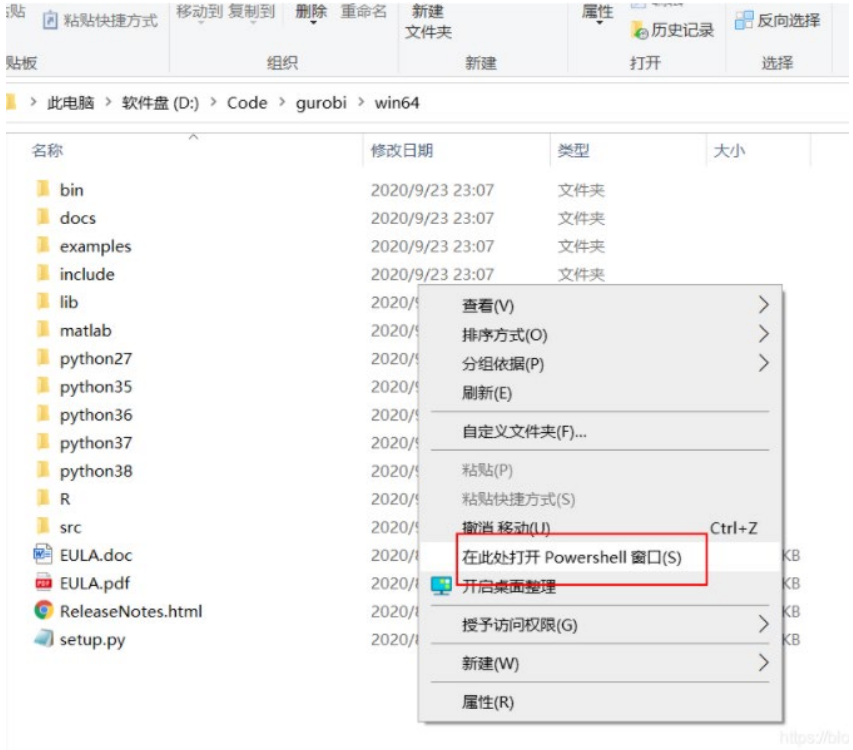


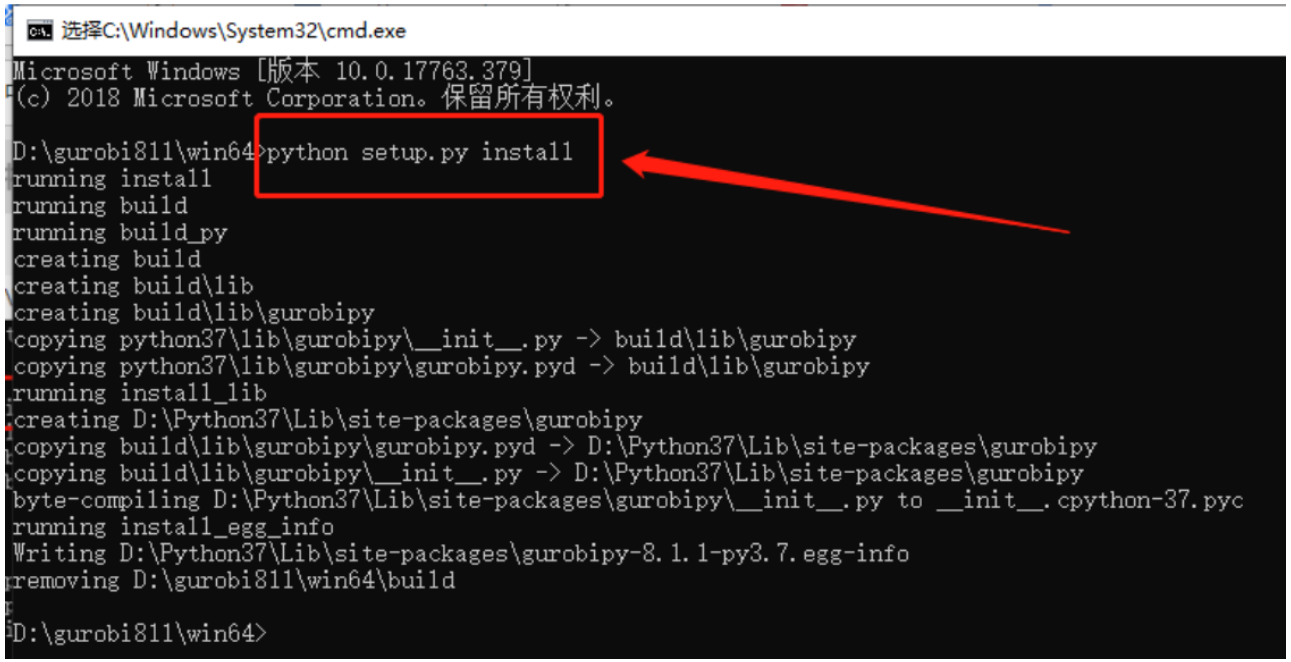
如果文件没有激活，配置一下环境变量



将 Gurobi 配置到 Python

8. 打开 Gurobi 的安装位置，在当前界面shift+右键打开Powershell窗口，运行 python setup.py install





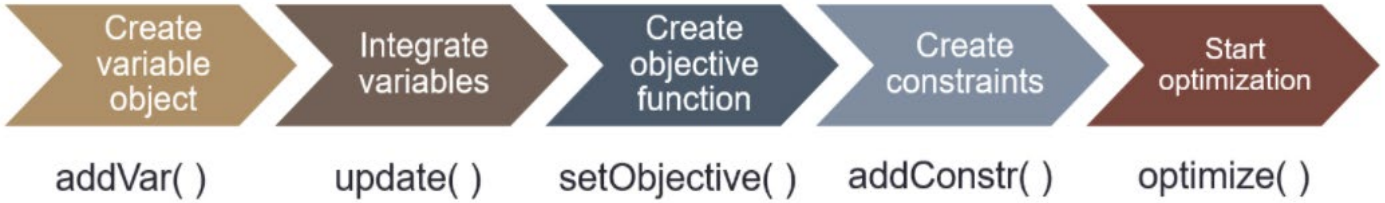
9. 测试。在命令行输入python，进入python环境，输入from gurobipy import * ，不报错即正常

```
(base) C:\Users\YueFeng>python
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from gurobipy import *
>>>
```

Python+Gurobi 建立数学规划模型


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

# 创建模型
MODEL = gp.Model()

# 创建变量
X = MODEL.addVar(vtype=GRB.INTEGER,name="X")

# 更新变量环境
MODEL.update()

# 创建目标函数
MODEL.setObjective('目标函数表达式', GRB.MINIMIZE)

# 创建约束条件
MODEL.addConstr('约束表达式，逻辑运算')

# 执行线性规划模型
MODEL.optimize()

# 输出模型结果
print("Obj:", MODEL.objVal)
for v in X:
    print(f"{v.varName}: {v.X}")
```

通常会按照设置变量、更新变量空间、设置目标函数、设置约束条件、执行最优化的顺序进行。



添加决策变量

- 创建一个变量 *addVar()*

```
x = MODEL.addVar(lb=0.0, ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name="")
```

  - `lb = 0.0`：变量的下界，默认为 0
  - `ub = GRB.INFINITY`：变量的上界，默认为无穷大
  - `vtype = GRB.CONTINUOUS`：变量的类型，默认为连续型，可改为 `GRB.BINARY` 0-1变量，`GRB.INTEGER` 整型
  - `name = ""`：变量名，默认为空

创建变量 $0 \leq x_1 \leq 1$

```
x1 = MODEL.addVar(lb=0, ub=1, name="x1")
```

- 创建多个变量 *addVars()*

```
x = MODEL.addVars(*indexes, lb=0, ub=GRB.INFINITY, vtype=GRB.CONTINUOUS, name="")
```

创建变量$2*$

3个变量：$x[0,0]$，$x[0,1]$，$x[0,2]$，$x[1,0]$，$x[1,1]$，$x[1,2]$

```
x = MODEL.addVars(2, 3, vtype=GRB.BINARY, name="C")
```

添加目标函数

- 单目标优化

```
MODEL.setObjective(expression, sense=None)
```

- `expression`： 表达式，可以是一次或二次函数类型
- `sense`：求解类型，可以是 `GRB.MINIMIZE` 或 `GRB.MAXIMIZE`

创建目标函数：$\min 8x + 10y + 7z$

```
MODEL.setObjective(8 * x + 10 * y + 7 * z , GRB.MINIMIZE)
```

添加目标函数

- 多目标优化 `MODEL.setObjectiveN(expression, index=0, priority=0, weight=1, abstol=1e-6, reltol=0, name='obj1')`

  - `expression`：表达式，可以是一次或二次函数类型
  - `index`：目标函数对应的序号 (默认 0，1，2，...), 以 index=0 作为目标函数的值, 其余值需要另外设置参数
  - `priority`：分层序列法多目标决策优先级(整数值), 值越大优先级越高
  - `weight`：线性加权多目标决策权重(在优先级相同时发挥作用)
  - `abstol`：分层序列法多目标决策时允许的目标函数值最大的降低量
  - `reltol`：分层序列法多目标决策时允许的目标函数值最大的降低比率 reltol*|目标函数值|

– 合成型

$\text{Obj1} = x + y$ weight = 1

$\text{Obj2} = x - 5y$ weight = -2

即转化为：$(x + y) - 2(x - 5y) = -x + 11y$

```
MODEL.setObjectiveN(x + y, index=0, weight=1, name='obj1')
MODEL.setObjectiveN(x -5 * y, index=1, weight=-2, name='obj2')
```

添加目标函数

- 多目标优化 `MODEL.setObjectiveN(expression, index=0, priority=0, weight=1, abstol=1e-6, reltol=0, name='obj1')`

  - `expression`：表达式，可以是一次或二次函数类型
  - `index`：目标函数对应的序号 (默认 0，1，2，...), 以 index=0 作为目标函数的值, 其余值需要另外设置参数
  - `priority`：分层序列法多目标决策优先级(整数值), 值越大优先级越高
  - `weight`：线性加权多目标决策权重(在优先级相同时发挥作用)
  - `abstol`：分层序列法多目标决策时允许的目标函数值最大的降低量
  - `reltol`：分层序列法多目标决策时允许的目标函数值最大的降低比率 reltol*|目标函数值|

– 分层优化型

$\text{Obj1} = x + y$ priority = 5

$\text{Obj2} = x - 5y$ priority = 1

即转化为：按照 priority 的大小关系，先优化 Obj1，再优化 Obj2

```
MODEL.setObjectiveN(x + y, index=0, priority=5, name='obj1')
MODEL.setObjectiveN(x -5 * y, index=1, priority=1, name='obj2')
```

添加目标函数

- 多目标优化 `MODEL.setObjectiveN(expression, index=0, priority=0, weight=1, abstol=1e-6, reltol=0, name='obj1')`

  - `expression`：表达式，可以是一次或二次函数类型
  - `index`：目标函数对应的序号 (默认 0，1，2，...), 以 index=0 作为目标函数的值, 其余值需要另外设置参数
  - `priority`：分层序列法多目标决策优先级(整数值), 值越大优先级越高
  - `weight`：线性加权多目标决策权重(在优先级相同时发挥作用)
  - `abstol`：分层序列法多目标决策时允许的目标函数值最大的降低量
  - `reltol`：分层序列法多目标决策时允许的目标函数值最大的降低比率 reltol*|目标函数值|

– 混合优化型

$\text{Obj1} = x + y$ priority = 5

$\text{Obj2} = x - 5y$ priority = 1

即转化为：先优化 Obj1，再优化 Obj2，最后相加作为目标值

```
MODEL.setObjectiveN(x + y, index=0, weight=1, priority=5, name='obj1')
MODEL.setObjectiveN(x -5 * y, index=1, weight=-2, priority=1, name='obj2')
```

添加目标函数

- 多目标优化

※ 执行优化结束后，获取目标函数值

```
for i in range(m):
    MODEL.setParam(GRB.Param.ObjNumber, i)    # 第 i 个目标
    print(MODEL.ObjNVal)  # 打印第 i 个目标值
```

创建约束

- 创建一个约束 `MODEL.addConstr(expression, name="")`
  - `expression`：布尔表达式，可以是一次或二次函数类型
  - `name`：约束式的名称

添加约束：$12x + 9y + 25z \geq 60$

```
MODEL.addConstr(12 * x + 9 * y + 25 * z >= 60, "c0")
```

- 创建多个约束 `MODEL.addConstrs(expression, name="")`

添加约束：

$$\sum_{j=0}^{7} x_{ij} \leq 1, \forall i = 0,1,\dots,19$$

$$x_{ij} = 0 \ or \ 1$$

```
x = MODEL.addVars(20, 8, vtype=GRB.BINARY)
# 写法 1
for i in range(20):
    MODEL.addConstr(x.sum(i, "*") <= 1)
# 写法 2
MODEL.addConstrs(x.sum(i, "*") <= 1 for i in range(20))
```

案例 1： python+gurobi 解决线性规划

◆ Gurobi 附带了一个名为 “gurobipy” 的Python扩展模块，该模块为所有 Gurobi 特性提供了方便的面向对象建模的构造和API。

- **求解一个简单的LP模型**

$$
\begin{array}{lrcrcrcl}
\textbf{maximize} & x & + & y & + & 2\,z & & \\
\textbf{subject to} & x & + & 2\,y & + & 3\,z & \leq & 4 \\
 & x & + & y & & & \geq & 1
\end{array}
$$

案例 1： python+gurobi 解决线性规划

$$
\begin{array}{lrcrcrcl}
\text{maximize} & x & + & y & + & 2\,z & & \\
\text{subject to} & x & + & 2\,y & + & 3\,z & \leq & 4 \\
 & x & + & y & & & \geq & 1
\end{array}
$$

➢ 导入 Gurobi 函数并导入 GRB 类

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

➢ 创建一个模型对象

创建一个空的模型对象 “m”，通过传递字符串 “lp1” 作为参数指定模型名称。

一个模型对象“m”包含一个优化问题。它由一组变量、一组约束和目标函数组成。

```
m = gp.Model("lp1")
```

➢ 创建变量

变量通过模型对象的 *addVar()* 方法添加。

变量总是与特定的模型相关联。

```
x = m.addVar(vtype=GRB.CONTINUOUS, name="x")
y = m.addVar(vtype=GRB.CONTINUOUS, name="y")
z = m.addVar(vtype=GRB.CONTINUOUS, name="z")
```

案例 1： python+gurobi 解决线性规划

添加约束

与变量一样，约束始终与特定模型相关联。

使用模型对象的 *addConstr()*方法创建，第二个参数指定约束名称。

$$
\begin{aligned}
\text{maximize} \quad & x + y + 2z \\
\text{subject to} \quad & x + 2y + 3z \leq 4 \\
& x + y \geq 1
\end{aligned}
$$

```
m.addConstr(x + 2 * y + 3 * z <= 4, "c0")
m.addConstr(x + y >= 1, "c1")
```

设置目标函数

使用模型对象的 *setObjective()* 方法创建目标函数。

第一个参数指定了目标函数表达式，第二个参数指定当前问题需要最大化目标。

```
m.setObjective(x + y + 2 * z, GRB.MAXIMIZE)
```

优化已创建的模型

使用 *optimize()*方法来解决为模型对象“m”定义的问题。

```
m.optimize()
```

案例 1： python+gurobi 解决线性规划

$$
\begin{array}{lrcrcrcl}
\text{maximize} & x & + & y & + & 2\,z & & \\
\text{subject to} & x & + & 2\,y & + & 3\,z & \leq & 4 \\
 & x & + & y & & & \geq & 1
\end{array}
$$

➢ 显示结果

使用Gurobi/Python API的 *Model. getvars()*方法查询与变量相关的参数 *.varName* 用于查询决策变量的名称，*.x* 用于查询变量的解。

```
for v in m.getVars():
    print('%s %g' % (v.varName, v.x))
```

通过查询模型对象的 *.objVal()*，得到当前目标函数的值。

```
print('Obj: %g' % m.objVal)
```

– 显示结果如下：

```
x 4
y 0
z 0
Obj: 4
```

案例 1： python+gurobi 解决线性规划

- 总体程序

```
import gurobipy as gp
from gurobipy import GRB

try:

    # Create a new model
    m = gp.Model("lp1")

    # Create variables
    x = m.addVar(vtype=GRB.CONTINUOUS, name="x")
    y = m.addVar(vtype=GRB.CONTINUOUS, name="y")
    z = m.addVar(vtype=GRB.CONTINUOUS, name="z")

    # Set objective
    m.setObjective(x + y + 2 * z, GRB.MAXIMIZE)


    # Add constraint: x + 2 y + 3 z <= 4
    m.addConstr(x + 2 * y + 3 * z <= 4, "c0")

    # Add constraint: x + y >= 1
    m.addConstr(x + y >= 1, "c1")

    # Optimize model
    m.optimize()

    for v in m.getVars():
        print('%s %g' % (v.varName, v.x))

    print('Obj: %g' % m.objVal)
```

- 显示结果

```
Optimize a model with 2 rows, 3 columns and 5 nonzeros
Model fingerprint: 0xef13ba7f
Coefficient statistics:
  Matrix range     [1e+00, 3e+00]
  Objective range  [1e+00, 2e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 4e+00]
Presolve removed 2 rows and 3 columns
Presolve time: 0.01s
Presolve: All rows and columns removed
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    4.0000000e+00   0.000000e+00   0.000000e+00       0s

Solved in 0 iterations and 0.01 seconds
Optimal objective  4.000000000e+00
x 4
y 0
z 0
Obj: 4
```

案例 2： python+gurobi 解决线性规划

- 使用矩阵形式求解LP模型

$$
\begin{aligned}
\min \quad & Z = 8x_1 + 10x_2 + 7x_3 + 6x_4 + 11x_5 + 9x_6 \\
s.t. \quad & 12x_1 + 9x + 25x_3 + 20x_4 + 17x_5 + 13x_6 \geq 60 \\
& 35x_1 + 42x_2 + 18x_3 + 31x_4 + 56x_5 + 49x_6 \geq 150 \\
& 37x_1 + 53x_2 + 28x_3 + 24x_4 + 29x_5 + 20x_6 \geq 125 \\
& 0 \leq x_j \leq 1, j = 1, 2, \cdots, 6
\end{aligned}
$$

➢ 导入 Gurobi 函数并导入 GRB 类

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

➢ 输入系数并创建模型对象

```
# 创建模型
c = [8, 10, 7, 6, 11, 9]
p = [[12, 9, 25, 20, 17, 13],
     [35, 42, 18, 31, 56, 49],
     [37, 53, 28, 24, 29, 20]]
r = [60, 150, 125]
MODEL = gp.Model("Example")
```

案例 2： python+gurobi 解决线性规划

$$
\begin{aligned}
\min \quad & Z = 8x_1 + 10x_2 + 7x_3 + 6x_4 + 11x_5 + 9x_6 \\
s.t. \quad & 12x_1 + 9x + 25x_3 + 20x_4 + 17x_5 + 13x_6 \geq 60 \\
& 35x_1 + 42x_2 + 18x_3 + 31x_4 + 56x_5 + 49x_6 \geq 150 \\
& 37x_1 + 53x_2 + 28x_3 + 24x_4 + 29x_5 + 20x_6 \geq 125 \\
& 0 \leq x_j \leq 1, j = 1, 2, \cdots, 6
\end{aligned}
$$

- 创建变量

```
x = MODEL.addVars(6, lb=0, ub=1, name='x')
# 更新变量环境
MODEL.update()
```

- 创建目标函数

```
MODEL.setObjective(x.prod(c), GRB.MINIMIZE)
```

- 创建约束

```
MODEL.addConstrs(x.prod(p[i]) >= r[i] for i in range(3))
```

- 优化已创建的模型

```
m.optimize()
```

- 显示结果

```
print("Obj:", MODEL.objVal)
for v in MODEL.getVars():
    print(f"{v.varName}: {round(v.x,3)}")
```

案例 2： python+gurobi 解决线性规划

- 总体程序

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

# 创建模型
c = [8, 10, 7, 6, 11, 9]
p = [[12, 9, 25, 20, 17, 13],
     [35, 42, 18, 31, 56, 49],
     [37, 53, 28, 24, 29, 20]]
r = [60, 150, 125]
MODEL = gp.Model("Example")

# 创建变量
x = MODEL.addVars(6, lb=0, ub=1, name='x')

# 更新变量环境
MODEL.update()

# 创建目标函数
MODEL.setObjective(x.prod(c), GRB.MINIMIZE)

# 创建约束条件
MODEL.addConstrs(x.prod(p[i]) >= r[i] for i in range(3))

# 执行线性规划模型
MODEL.optimize()
print("Obj:", MODEL.objVal)
for v in MODEL.getVars():
    print(f"{v.varName}: {round(v.x,3)}")
```

- 显示结果

```
Optimize a model with 3 rows, 6 columns and 18 nonzeros
Model fingerprint: 0x3ead4e64
Coefficient statistics:
  Matrix range     [9e+00, 6e+01]
  Objective range  [6e+00, 1e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [6e+01, 2e+02]
Presolve time: 0.00s
Presolved: 3 rows, 6 columns, 18 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    0.0000000e+00   4.187500e+01   0.000000e+00      0s
       4    3.2154631e+01   0.000000e+00   0.000000e+00      0s

Solved in 4 iterations and 0.00 seconds
Optimal objective  3.215463133e+01
Obj: 32.154631330359486
x[0]: 1.0
x[1]: 0.623
x[2]: 0.343
x[3]: 1.0
x[4]: 0.048
x[5]: 1.0
```

案例 3： python+gurobi 解决混合整数规划

类似地，Gurobi 也可以解决混合整数规划问题。

- **求解一个简单的MIP模型**

$$
\begin{array}{lrcrcrcl}
\textbf{maximize} & x & + & y & + & 2\,z & & \\
\textbf{subject to} & x & + & 2\,y & + & 3\,z & \leq & 4 \\
 & x & + & y & & & \geq & 1 \\
 & & & x,\ y,\ z & \text{binary} & & &
\end{array}
$$

案例 3： python+gurobi 解决混合整数规划

$$
\begin{aligned}
\text{maximize} \quad & x + y + 2z \\
\text{subject to} \quad & x + 2y + 3z \leq 4 \\
& x + y \geq 1 \\
& x, y, z \text{ binary}
\end{aligned}
$$

➢ 导入 Gurobi 函数并导入 GRB 类

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

➢ 创建一个模型对象

创建一个空的模型对象 “m”，通过传递字符串“mip1” 作为参数指定模型名称。

一个模型对象“m”包含一个优化问题。它由一组变量、一组约束和目标函数组成。

```
m = gp.Model("mip1")
```

➢ 创建变量

变量通过模型对象的 *addVar()* 方法添加。

变量总是与特定的模型相关联。

```
x = m.addVar(vtype=GRB.BINARY, name="x")
y = m.addVar(vtype=GRB.BINARY, name="y")
z = m.addVar(vtype=GRB.BINARY, name="z")
```

案例 3： python+gurobi 解决混合整数规划

添加约束

与变量一样，约束始终与特定模型相关联。

使用模型对象的 *addConstr()*方法创建，第二个参数指定约束名称。

$$
\begin{aligned}
\text{maximize} \quad & x + y + 2z \\
\text{subject to} \quad & x + 2y + 3z \leq 4 \\
& x + y \geq 1
\end{aligned}
$$

```
m.addConstr(x + 2 * y + 3 * z <= 4, "c0")
m.addConstr(x + y >= 1, "c1")
```

设置目标函数

使用模型对象的 *setObjective()* 方法创建目标函数。

第一个参数指定了目标函数表达式，第二个参数指定当前问题需要最大化目标。

```
m.setObjective(x + y + 2 * z, GRB.MAXIMIZE)
```

优化已创建的模型

使用 *optimize()*方法来解决为模型对象“m”定义的问题。

```
m.optimize()
```

案例 3： python+gurobi 解决混合整数规划

- 总体程序

```
import gurobipy as gp
from gurobipy import GRB

try:

    # Create a new model
    m = gp.Model("mip1")

    # Create variables
    x = m.addVar(vtype=GRB.BINARY, name="x")
    y = m.addVar(vtype=GRB.BINARY, name="y")
    z = m.addVar(vtype=GRB.BINARY, name="z")

    # Set objective
    m.setObjective(x + y + 2 * z, GRB.MAXIMIZE)

    # Add constraint: x + 2 y + 3 z <= 4
    m.addConstr(x + 2 * y + 3 * z <= 4, "c0")

    # Add constraint: x + y >= 1
    m.addConstr(x + y >= 1, "c1")

    # Optimize model
    m.optimize()

    for v in m.getVars():
        print('%s %g' % (v.varName, v.x))

    print('Obj: %g' % m.objVal)
```

- 显示结果

```
Optimize a model with 2 rows, 3 columns and 5 nonzeros
Model fingerprint: 0x98886187
Variable types: 0 continuous, 3 integer (3 binary)
Coefficient statistics:
  Matrix range     [1e+00, 3e+00]
  Objective range  [1e+00, 2e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 4e+00]
Found heuristic solution: objective 2.0000000
Presolve removed 2 rows and 3 columns
Presolve time: 0.01s
Presolve: All rows and columns removed

Explored 0 nodes (0 simplex iterations) in 0.01 seconds
Thread count was 1 (of 16 available processors)

Solution count 2: 3 2

Optimal solution found (tolerance 1.00e-04)
Best objective 3.000000000000e+00, best bound 3.000000000000e+00, gap 0.0000%
x 1
y 0
z 1
Obj: 3
```

案例 4： 工作分配问题（混合整数规划）

- **工作分配问题：**一家咨询公司有三个空缺的职位：Tester、 Java Developer 和 Architect，三个最佳候选人（资源）是：Carlos、Joe和Monica。咨询公司对每个候选人进行能力测试，以评估他们执行每项职位的能力，测试的结果称为*匹配得分*，下表列出了每位候选人执行每个职位的能力。
- **Q：**假设一个职位仅分配一个候选人，并且每个候选人只能分配一个工作。如何确定人员和职位的分配，使得总匹配得分最大？

Matching Scores	Tester	Java Developer	Architect
Carlos	53%	27%	13%
Joe	80%	47%	67%
Monika	53%	73%	47%

案例 4： 工作分配问题（混合整数规划）

➢ 确定决策变量

为了解决这个分配问题，需要确定哪个人员分配给哪个职位。

为每个可能的 人员→职位 的分配引入一个决策变量，因此，共有9个决策变量。

决策变量	含义
$x_{r,j}$	人员r向职位j的分配情况。对于$r \in R$，$j \in J$，$x_{r,j}$ =1表示将人员r分配给职位j。

Decision Variables	Tester = 1	Java Developer = 2	Architect = 3
Carlos = 1	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
Joe = 2	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$
Monika = 3	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$

案例 4： 工作分配问题（混合整数规划）

➢ 确定约束

– 与职位相关的约束：确保每个职位仅被分配给一个人员

Constraint (Tester=1) $x_{1,1} + x_{2,1} + x_{3,1} = 1$

Constraint (Java Developer = 2) $x_{1,2} + x_{2,2} + x_{3,2} = 1$

Constraint (Architect = 3) $x_{1,3} + x_{2,3} + x_{3,3} = 1$

$$\Rightarrow \quad \sum_{r \in R} x_{r,j} = 1$$

– 与人员（资源）相关的约束：确保每个人员最多分配给一个职位。可能不是所有人员都被录用。

Constraint (Carlos=1) $x_{1,1} + x_{1,2} + x_{1,3} \leq 1.$

Constraint (Joe=2) $x_{2,1} + x_{2,2} + x_{2,3} \leq 1.$

Constraint (Monika=3) $x_{3,1} + x_{3,2} + x_{3,3} \leq 1.$

$$\Rightarrow \quad \sum_{j \in J} x_{r,j} \leq 1.$$

Decision Variables	Tester = 1	Java Developer = 2	Architect = 3	Availability
Carlos = 1	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	1
Joe = 2	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	1
Monika = 3	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	1
Requirement	1	1	1	

案例 4： 工作分配问题（混合整数规划）

➢ 确定目标函数

目标函数是使满足职位和人员（资源）约束的总匹配分数最大化。

$$\text{Maximize} \quad \sum_{j \in J} \sum_{r \in R} s_{r,j} x_{r,j}.$$

Decision Variables	Tester = 1	Java Developer = 2	Architect = 3
Carlos = 1	$53x_{1,1}$	$27x_{1,2}$	$13x_{1,3}$
Joe = 2	$80x_{2,1}$	$47x_{2,2}$	$67x_{2,3}$
Monika = 3	$53x_{3,1}$	$73x_{3,2}$	$47x_{3,3}$

案例 4：工作分配问题（混合整数规划）

Matching Scores	Tester	Java Developer	Architect
Carlos	53%	27%	13%
Joe	80%	47%	67%
Monika	53%	73%	47%

➢ 导入 Gurobi 函数并导入 GRB 类

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

➢ 声明参数

```
# Resource and job sets
R = ['Carlos', 'Joe', 'Monika']
J = ['Tester', 'JavaDeveloper', 'Architect']

# Matching score data
combinations, scores = gp.multidict({
    ('Carlos', 'Tester'): 53,
    ('Carlos', 'JavaDeveloper'): 27,
    ('Carlos', 'Architect'): 13,
    ('Joe', 'Tester'): 80,
    ('Joe', 'JavaDeveloper'): 47,
    ('Joe', 'Architect'): 67,
    ('Monika', 'Tester'): 53,
    ('Monika', 'JavaDeveloper'): 73,
    ('Monika', 'Architect'): 47
})
```

创建 list 表示候选人和空缺职位集合

符号	含义
R	候选人集合，$r \in R$
J	空缺职位集合，$j \in J$
$s_{r,j}$	对所有$r \in R$，$j \in J$，$s_{r,j} \in [0,100]$

使用Gurobi/Python中的 *multidict()函数*：

用一条语句初始化一个或多个字典。

该函数将字典作为其参数，字典的键(key)是combinations，表示人员和职位的可能组合；值(values)是scores，表示候选人每个职位对应的能力。

案例 4： 工作分配问题（混合整数规划）

创建一个模型对象

创建一个空的模型对象 “m”，指定模型名称为“RAP”。

```
m = gp.Model('RAP')
```

创建变量

变量通过模型对象的 *addVar()* 方法添加。变量总是与特定的模型相关联。

决策变量为之前定义的人员向职位的分配 combinations

```
x = m.addVars(combinations, name="assign")
```

– x 的形式：

```
('Carlos', 'Tester'): <gurobi.Var *Awaiting Model Update*>
('Carlos', 'JavaDeveloper'): <gurobi.Var *Awaiting Model Update*>
('Carlos', 'Architect'): <gurobi.Var *Awaiting Model Update*>
('Joe', 'Tester'): <gurobi.Var *Awaiting Model Update*>
('Joe', 'JavaDeveloper'): <gurobi.Var *Awaiting Model Update*>
('Joe', 'Architect'): <gurobi.Var *Awaiting Model Update*>
('Monika', 'Tester'): <gurobi.Var *Awaiting Model Update*>
('Monika', 'JavaDeveloper'): <gurobi.Var *Awaiting Model Update*>
('Monika', 'Architect'): <gurobi.Var *Awaiting Model Update*>
```

案例 4： 工作分配问题（混合整数规划）

添加约束

使用模型对象的 *addConstr()*方法创建。第一个参数使用sum方法， *"x.sum(' * ', j)"*。

如定义职位约束的LHS时：对于职位集 J 中的每个职位 j，取针对所有人员的决策变量的总和。

$$\sum_{r \in R} x_{r,j} = 1$$

```
jobs = m.addConstrs((x.sum('*',j) == 1 for j in J), name='job')
```

$$\sum_{j \in J} x_{r,j} \leq 1.$$

```
resources = m.addConstrs((x.sum(r,'*') <= 1 for r in R), name='resource')
```

案例 4：工作分配问题（混合整数规划）

➢ 设置目标函数

使用模型对象的 *setObjective()* 方法创建目标函数。

第一个参数指定了目标函数表达式，第二个参数指定当前问题需要最大化目标。

匹配得分参数 “score” 和决策变量 “x” 都拥有相同的键(key)“combination” 。因此，使用“x.prod(score)”来得到 “score” 矩阵与 “x” 决策变量矩阵的对应元素乘积之和。

```
m.setObjective(x.prod(scores), GRB.MAXIMIZE)
```

– 目标函数的形式：

```
<gurobi.LinExpr: 53.0 <gurobi.Var *Awaiting Model Update*> + 27.0 <gurobi.Var *Awaiting Model Update*> + 13.0 <gurob
i.Var *Awaiting Model Update*> + 80.0 <gurobi.Var *Awaiting Model Update*> + 47.0 <gurobi.Var *Awaiting Model Update*
> + 67.0 <gurobi.Var *Awaiting Model Update*> + 53.0 <gurobi.Var *Awaiting Model Update*> + 73.0 <gurobi.Var *Awaitin
g Model Update*> + 47.0 <gurobi.Var *Awaiting Model Update*>>
```

案例 4： 工作分配问题（混合整数规划）

➢ 优化已创建的模型

使用 *optimize()*方法来解决为模型对象“m”定义的问题。

```
m.optimize()
```

– 输出结果:

```
Gurobi Optimizer version 9.1.2 build v9.1.2rc0 (win64)
Thread count: 8 physical cores, 16 logical processors, using up to 16 threads
Optimize a model with 6 rows, 9 columns and 18 nonzeros
Model fingerprint: 0xb343b6eb
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+01, 8e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve time: 0.01s
Presolved: 6 rows, 9 columns, 18 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    4.6000000e+32   1.800000e+31   4.600000e+02      0s
       5    1.9300000e+02   0.000000e+00   0.000000e+00      0s

Solved in 5 iterations and 0.01 seconds
Optimal objective  1.930000000e+02
```

案例 4： 工作分配问题（混合整数规划）

显示结果

使用Gurobi/Python API的 *Model. getvars()*方法查询与变量相关的参数 *.varName* 用于查询决策变量的名称，*.x* 用于查询变量的解。

```
for v in m.getVars():
    print('%s %g' % (v.varName, v.x))
```

通过查询模型对象的 *.objVal()*，得到当前目标函数的值。

```
print('Obj: %g' % m.objVal)
```

- 输出结果:

```
assign[Carlos,Tester] 1.0
assign[Joe,Architect] 1.0
assign[Monika,JavaDeveloper] 1.0
Total matching score:  193.0
```

案例 4：工作分配问题（混合整数规划）

- 程序

```
import gurobipy as gp
from gurobipy import GRB

# Resource and job sets
R = ['Carlos', 'Joe', 'Monika']
J = ['Tester', 'JavaDeveloper', 'Architect']

# Matching score data
combinations, scores = gp.multidict({
    ('Carlos', 'Tester'): 53,
    ('Carlos', 'JavaDeveloper'): 27,
    ('Carlos', 'Architect'): 13,
    ('Joe', 'Tester'): 80,
    ('Joe', 'JavaDeveloper'): 47,
    ('Joe', 'Architect'): 67,
    ('Monika', 'Tester'): 53,
    ('Monika', 'JavaDeveloper'): 73,
    ('Monika', 'Architect'): 47
})
# Declare and initialize model
m = gp.Model('RAP')
```

```
# Create decision variables for the RAP model
x = m.addVars(combinations, name="assign")

# Create job constraints
jobs = m.addConstrs((x.sum('*',j) == 1 for j in J), name='job')

# Create resource constraints
resources = m.addConstrs((x.sum(r,'*') <= 1 for r in R), name='resource')

# Objective: maximize total matching score of all assignments
m.setObjective(x.prod(scores), GRB.MAXIMIZE)

# Save model for inspection
m.write('RAP.lp')

# Run optimization engine
m.optimize()

# Display optimal values of decision variables
for v in m.getVars():
    if v.x > 1e-6:
        print(v.varName, v.x)

# Display optimal total matching score
print('Total matching score: ', m.objVal)
```

案例 4： 工作分配问题（混合整数规划）

- 显示结果

```
Optimize a model with 6 rows, 9 columns and 18 nonzeros
Model fingerprint: 0xb343b6eb
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+01, 8e+01]
  Bounds range     [0e+00, 0e+00]
  RHS range        [1e+00, 1e+00]
Presolve time: 0.00s
Presolved: 6 rows, 9 columns, 18 nonzeros

Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0    4.6000000e+32   1.800000e+31   4.600000e+02      0s
       5    1.9300000e+02   0.000000e+00   0.000000e+00      0s

Solved in 5 iterations and 0.00 seconds
Optimal objective  1.930000000e+02
assign[Carlos,Tester] 1.0
assign[Joe,Architect] 1.0
assign[Monika,JavaDeveloper] 1.0
Total matching score:  193.0
```

使用cvxpy调用Gurobi

- CVXPY 是一个内嵌 于python 的模型编程语言，解决凸优化问题。它它可以自动转化问题为标准形式，调用求解器求解，而不是用求解器所要求的限制性标准形式。
- CVXPY自带的求解器有：‘ECOS’, ‘GLPK’, ‘GLPK_MI’, ‘OSQP’, ‘SCS’
  - 查看已安装的求解器:

```
import cvxpy as cp
print(cp.installed_solvers())
```

使用cvxpy调用Gurobi

$$\begin{aligned} \min\ & (x-y)^2 \\ \text{s.t.}\ & x+y=1 \\ & x-y\geq 1 \end{aligned}$$

- 总体程序

```
import cvxpy as cvx

#定义优化变量
x = cvx.Variable()
y = cvx.Variable()

# 定义约束条件
constraints = [x + y == 1,
               x - y >= 1]

# 定义优化问题
obj = cvx.Minimize((x - y)**2)

# 定义优化问题
prob = cvx.Problem(obj, constraints)

#求解问题，此处调用GUROBI求解器
prob.solve(solver='GUROBI')

#求解状态
print("status:", prob.status)

#目标函数优化值
print("optimal value", prob.value)

# 优化变量的值
print("optimal var", x.value, y.value)
```

- 结果

```
status: optimal
optimal value 1.0
optimal var 1.0 0.0
```