



# GUROBI笔记 (python)

在利用 Python+Gurobi 建立数学规划模型时，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。

[gurobi官网教程](#)

[gurobipy: 函数和类介绍](#)

- [GUROBI笔记 \(python\)](#)
  - [Model类](#)
    - [1. Model 构成](#)
    - [2. 构建模型的方法](#)
    - [3. 添加变量的方法](#)
    - [4. 添加约束的方法](#)

## Model类

Gurobi Python接口中的大多数操作都是通过调用Gurobi对象上的方法来执行的。最常用的对象是 `Model`。

### 1. Model 构成

Model的最低级构建模块是变量、约束和目标。

- 一组决策变量：`Var` 或 `MVar` 类对象，每个变量都有相关的下限、上限、类型（连续、二进制等）和名称
- 变量的线性或二次目标函数：使用 `model.setObjective` 指定
- 变量的一组约束：`CONTR`、`MConstr`、`QConstr` 或 `GenConstr` 类对象，每个线性或二次约束都具有关联的意义（小于或等于、大于或等于或等于）和右侧值

有关变量、约束和目标的更多信息，请参阅[本节 \(网页\)](#)。

有关 `Model` 类的构建方法，请参阅[本节 \(本地\)](#)

### 2. 构建模型的方法

1. `gp.Model(name="", env=defaultEnv)`
  - `name` (string): 新模型的名称。
  - `env` (Env): 创建模型的环境。使用`Env`函数构造环境对象并输入。

```
# 默认环境
model = gp.Model("model")
# 新建环境
env = Env("my.log")
m2 = Model("NewModel2", env)
```

2. 从文件加载模型：`read` [函数](#)

### 3. 添加变量的方法

1. `Model.addVar(lb=0.0, ub=float('inf'), obj=0.0, vtype=GRB.CONTINUOUS, name="", column=None)`

- `lb` (float): 新变量的下限。
- `ub` (float): 新变量的上限。
- `obj` (float): 新变量的目标系数。
- `vtype` (string): 新变量的变量类型 (GRB.CONTINUOUS、GRB.BINARY、GRB.INTEGER、GRB.semicont或GRB.seminint)。

The available variables types are continuous, general integer, binary, semi-continuous, and semi-integer.

Gurobi API 提供了一个符号常量, 允许您指示边界是无限的 (C 和 C++ 中的 GRB\_INFINITY, C#、Java 和 Python 中的 GRB.INFINITY)。任何大于 1e30 的界限都被视为无限。

- `name` (string): 新变量的名称。

注意, 名称将存储为ASCII字符串。强烈建议不要使用包含空格的名称, 因为它们不能写入LP格式文件。

- `column` (Column): `column`对象, 指示新变量参与的约束集以及相关系数。
- 返回值: 新的 `Var` 对象。

```
x = model.addVar()
y = model.addVar(vtype=GRB.INTEGER, obj=1.0, name="y")
z = model.addVar(0.0, 1.0, 1.0, GRB.BINARY, "z")
```

2. `Model.addVars(*indices, lb=0.0, ub=float('inf'), obj=0.0, vtype=GRB.CONTINUOUS, name="")`

- `indices`: 用于访问新变量的索引。
- 参数 `lb`、`ub`、`obj`、`vtype` 和 `name` 类似于 `Model.addVar()`。
- 可以指定值作为标量、列表或字典。对于标量, 该值将用于所有变量; 对于list, 值必须与索引集的顺序相同; 对于dict, 它们必须由变量索引索引。
- 如果为 `name` 指定标量字符串, 则变量 `name` 将自动下标。
- 返回值: 包含新变量作为值的新 `tupledict` 对象, 使用提供的索引作为键。

```
x = model.addVars(3, 4, 5, vtype=GRB.BINARY)
l = tuplelist([(1, 2), (1, 3), (2, 3)])
y = model.addVars(l, ub=[1, 2, 3])
```

3. `Model.addMVar(shape, lb=0.0, ub=float('inf'), obj=0.0, vtype=GRB.CONTINUOUS, name="")`

- `shape` (tuple): 数组的形状。
- `lb` (float): 新变量的下限。
- `ub` (float): 新变量的上限。
- `obj` (float): 新变量的目标系数。
- `vtype` (string): 新变量的变量类型。
- `name` (string): 新变量的名称。给定名称将由生成器表达式的索引下标, 因此如果索引是整数, 将变为 `c0`、`c1` 等。
- 返回值: `MVar` 对象。

```
x = model.addMVar(10) # 包含10个变量的一维数组
y = model.addMVar((3,4), vtype=GRB.BINARY) # 3*4的二进制二维数组
```

## 4. 添加约束的方法

1. `Model.addConstr(constr, name="")`
  - `tc` (TempConstr [网页、本地]): 要添加的约束。
  - `name` (string): 约束的名称。
  - 返回值: 取决于 `tc`
    - `tc` 是线性表达式: `Constr` 对象;
    - `tc` 是二次表达式: `QConstr` 对象;
    - `tc` 是一般约束表达式: `GenConstr` 对象;

```
1 c = model.addConstr(x + y <= 1, "c1")
2 c = model.addConstr(x + y + z == [1, 2], "rgc0")
3 c = model.addConstr(x*x + y*y <= 1)
4 c = model.addConstr(z == and_(y, x, w))
5 c = model.addConstr(z == min_(x, y))
6 c = model.addConstr((w == 1) >> (x + y <= 1), "ic0")
7 c = model.addConstr(A @ x <= b)
```

一个约束只能有一个比较运算符。虽然  $1 \leq x + y \leq 2$  可能看起来像一个有效的约束，但 `addConstr` 不会接受它。

2. `Model.addConstrs(generator, name="")`
    - `generator`: 生成器表达式，每次迭代产生一个约束。
-