Gurobi 入门和进阶网络课程

课程二：Gurobi 功能和操作进阶

刃之砺信息科技（上海）有限公司

课程大纲

(1) 重要参数和属性

(2) 自动参数调优

(3) 广义约束的表达方式和使用

(4) 多目标优化的表达方式和使用

(5) 多个解集合的实现方式和使用

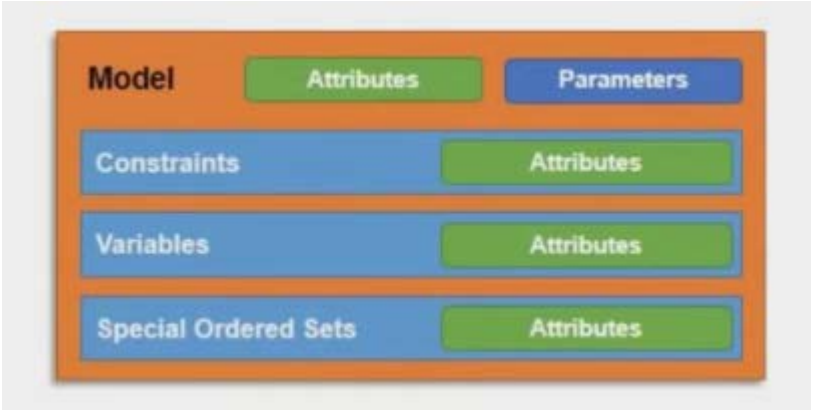
参数和属性

- 参数和属性功能

Parameter(参数)控制优化器的行为,需要在优化启动前设置。

例如：控制求解时间 TimeLimit

控制控制台输出log记录 LogToConsole



Attributes(属性) 控制模型(模型、变量、约束、目标等对象)的特征。

例如：模型 ModelSense

变量 LB/UB

约束 RHS

参数和属性

- 参数类别

(1) Termination 停止参数, 控制求解停止条件。

例如：**TimeLimit** 设定时间；**SolutionLimit** 设定MIP可行解数量。

(2) Tolerances 容差参数, 控制求解器的最优或可行的偏差。

例如：**MIPGap** 设定MIP的gap值；**FeasibilityTol** 设定精度。

(3) Simplex 单纯形参数, 控制单纯形法。

例如：**InfUnbdInfo** 控制是否获取不可行或无界模型的额外信息。

(4) Barrier 障碍法参数, 控制障碍法。

例如：**QCPDual** 控制是否获取二次模型的对偶值。

参数和属性

- 参数类别

(5) Mip 混合整数参数, 控制混合整数规划算法。

例如：**BranchDir** 设定优先分支方向；**Heuristics** 设定启发式算法求解时间所占的比重。

(6) Mip Cuts 割平面参数, 控制割平面。

例如：**Cuts** 设定割平面法的强度。

(7) Tuning 调参参数, 控制调参工具。

例如：**TuneCriterion** 设定调参的准则；**TuneTimeLimit** 设定调参的时间。

(8) Multiple Solutions 多解参数, 尝试寻找多个解。

例如：**PoolSolutions** 决定存储可行解的数量。

参数和属性

- 参数类别

  (9) Distributed algorithms 分布式计算参数

  (10) Compute Server 计算服务器参数

  (11) Cloud 云参数

  (12) Token server 令牌服务器参数

  (13) Other 其他一些参数

完整参数列表和具体作用参见参考手册。 …\docs\refman.pdf

参数和属性

- 参数设置方法

setParam(paramname, newvalue)

- paramname 参数名称。
- newvalue 参数取值, 可以设定为 “default”。

对于 Python,可以简写为 model.Params.xxx。

例如, 设定求解时间:

model. setParam("TimeLimit“, 600)　　model. setParam(GRB.Param.TimeLimit, 600)

model.Params.TimeLimit = 600

参数和属性

- 常用参数

参数名称	作用	取值
TimeLimit	时间设定	单位秒
MIPFocus	设定MIP的求解侧重点	0 默认，均衡搜寻可行解和证明最优； 1 侧重快速找到可行解； 2 侧重证明最优； 3 侧重界的提升（发现界提升缓慢）。
Method	设定线性模型或MIP根节点的求解方法	-1 默认，（3 LP, 2 QP/QCP, 1 MIP）； 0 初始单纯形； 1 对偶单纯形； 2 内点法； 3（0,1,2）并发（线程数，重复运行最优基可能不同）； 4 确定性（0,1,2）并发，重复运行结果相同； 5 确定性（0,1）并发，重复运行结果相同。

参数和属性

- 常用参数

参数名称	作用	取值
LogToConsole	log记录是否在控制台显示	0 关闭控制台显示; 1 打开控制台显示(默认)。
LogFile	设定log文件名称	默认空字符串。
Presolve	控制预处理的程度	-1 默认, 0 关闭预处理, 1 保守倾向; 2 进取倾向(耗时多, 处理后的模型更紧凑)。
MIPGap	设定gap值	0.0001 默认, 小于给定的值终止计算;
ImproveStartGap	提升策略开始gap值	0.0 默认, 例如取0.1, 当gap < 0.1, 开始提升策略;
ImproveStartNodes	提升策略开始节点值	Infinity 默认, 例如取5,当节点数 > 5, 开始提升策略;
ImproveStartTime	提升策略开始的时间	Infinity 默认, 例如取100, 当运行时间到100s时, 开始提升策略。
ObjNumber	目标函数索引(多目标)	索引从0开始编号。

参数和属性

- 参数使用案例

  来源 ...\examples\python\params.py

  参数 TimeLimit, MIPFocus

```
# Read model and verify that it is a MIP
m = read(sys.argv[1])
if m.isMIP == 0:
    print('The model is not an integer program')
    exit(1)

# Set a 2 second time limit
m.Params.timeLimit = 2

# Now solve the model with different values of MIPFocus
bestModel = m.copy()
bestModel.optimize()
for i in range(1, 4):
    m.reset()
    m.Params.MIPFocus = i
    m.optimize()
    if bestModel.MIPGap > m.MIPGap:
        bestModel, m = m, bestModel # swap models

# Finally, delete the extra model, reset the time limit and
# continue to solve the best model to optimality
del m
bestModel.Params.timeLimit = "default"
bestModel.optimize()
print('Solved with MIPFocus: %d' % bestModel.Params.MIPFocus)
```

m. setParam(GRB.Param.TimeLimit, 2)

m. setParam(GRB.Param.MIPFocus, i)

m. setParam(GRB.Param. TimeLimit, "default")

参数和属性

- 属性类别

(1) Model Attributes 模型属性

例如：**ModelSense** 模型优化方向(最大化或最小化)；**ObjVal** 当前目标值。

(2) Variable Attributes 变量属性

例如：**X** 当前变量的取值； **Start** MIP初始解。

(3) Linear Constraint Attributes 线性约束属性

例如：**Pi** 约束对应的对偶值；**Slack** 约束的松弛量；**RHS** 约束的右端项。

(4) Special-ordered Set constraints Attributes SOS约束属性

例如：**IISSOS** 对不可行的模型,指示约束是否属于IIS (Irreducible Inconsistent Subsystem)。

参数和属性

- 属性类别

(5) Quadratic Constraint Attributes 二次约束属性

例如：**QCRHS** 约束右端项。

(6) General Constraint Attributes 广义约束属性

例如：**GenConstrName** 约束名称。

(7) Quality Attributes 解质量属性

例如：**BoundVio** 最大的界违反； **IntVio** 整数变量离最近整数的最大距离。

(8) Multi-objective Attributes 多目标属性

例如：**ObjN** 对应多目标表达式中变量系数；**ObjNVal** 对应目标函数值

参数和属性

- 属性设置和查询方法

setAttr(attrname, newvalue) 注意并不是所有的属性都可以设置

- attrname 属性名称
- newvalue 属性的值

例如： var.setAttr(GRB.Attr.VType, 'C') 或简写为 var.Vtype = 'C'

getAttr(attrname, objs)

- attrname 属性名称
- objs(可选) 列表或字典对象用来存储查询的值

例如：model.getAttr(GRB.Attr.ObjVal) 或简写为 model.ObjVal

参数和属性

- 常用属性

类别	属性名称	作用	取值
模型	ModelSense（可调整）	模型优化方向	1 最小化（默认）； -1 最大化。
	ObjVal（不可调整）	目标函数值	double
	Status（不可调整）	解的状态	1-15
变量	LB/UB（可调整）	变量下界/上界	double
	Obj（可调整）	变量的线性目标系数	double
	VType（可调整）	变量类型	C,B,I,S,N
	X（不可调整）	变量值	double
	Start（可调整）	变量的初始值	double
约束	RHS（可调整）	线性约束右端项	double
	Pi（不可调整）	线性约束对应的对偶变量	double

自动参数调优

- 两种方式调用自动调参工具

(1) 通过命令行：

```
grbtune TuneTimeLimit=100  C:\gurobi801\win64\examples\data\misc07.mps
```

(2) 通过API：

```
model.tune()
```

自动参数调优

- 调参参数

参数名称	作用	取值
TuneCriterion	调整调参准则	-1 默认，缩短发现最优解所需的时间； 1 最优的Gap； 2 最好的可行解； 3 最好的bound。
TuneJobs	分布式并行调参	0 默认。
TuneOutput	控制输出结果的量	0 没有输出； 1 发现最好参数组合时输出； 2 默认，输出试过的参数组合； 3 试过的参数组合和详细的求解器输出。
TuneResults	返回最优参数组合的数量	-1 默认，按照调整参数的个数返回调参结果。
TuneTimeLimit	调参时间	-1 默认，自动选择时间；
TuneTrials	每组参数组合运行的次数	主要目的减小随机因素的影响。

自动参数调优

- 调参案例

来源 ...\examples\python\tune.py

TuneResultCount 模型属性
调参完成后储存的参数组合个数，其值<= TuneResults
若没有发现更好的参数组合则取值为零。

getTuneResult(n) 获得调参结果
n = {0, 1, …, TuneResultCount-1}，最好的结果索引为零。

write('tune.prm')
将调参结果写到prm格式文件中。

```
# Read the model
model = read(sys.argv[1])

# Set the TuneResults parameter to 1
model.Params.tuneResults = 1

# Tune the model
model.tune()

if model.tuneResultCount > 0:

    # Load the best tuned parameters into the model
    model.getTuneResult(0)

    # Write tuned parameters to a file
    model.write('tune.prm')

    # Solve the model using the tuned parameters
    model.optimize()
```

特殊约束的表达方式和使用

- 广义约束—Max

addGenConstrMax(resvar, vars, constant, name)

- resvar 变量(x = max(x1,x2,10))
- vars 一组变量(可以包含常数)
- constant 常数
- name 广义约束名称

一组变量(包含常数)中取最大。

特殊约束的表达方式和使用

- 广义约束—Max

例如：z = max(x, y, 3)

```
m.addGenConstrMax(z, [x, y], 3, "maxconstr")

m.addGenConstrMax(z, [x, y, 3], name="maxconstr")
```

换成一般的约束表达方式：

```
m.addConstr(z == max_([x, y, 3]), "maxconstr")

m.addConstr(z == max_(x, y, 3), "maxconstr")
```

特殊约束的表达方式和使用

- 广义约束—Min

addGenConstrMin(resvar, vars, constant, name)

  - resvar 变量(x = min(x1,x2,10))
  - vars 一组变量(可以包含常数)
  - constant 常数
  - name 广义约束名称

一组变量(包含常数)中取最小。

特殊约束的表达方式和使用

- 广义约束—Min

  例如：z = min(x, y, 3)

  m.addGenConstrMin(z, [x, y], 3, "minconstr")

  m.addGenConstrMin(z, [x, y, 3], name="minconstr")

  换成一般的约束表达方式：

  m.addConstr(z == min_([x, y, 3]), "minconstr")

  m.addConstr(z == min_(x, y, 3), "minconstr")

特殊约束的表达方式和使用

- 广义约束—Abs

addGenConstrAbs(resvar, argvars, name)

  - resvar 变量
  - argvar 变量
  - name 广义约束名称

取绝对值。

特殊约束的表达方式和使用

- 广义约束—Abs

  例如：x = |y|

  m.addGenConstrAbs(x, y, "absconstr")

  换成一般的约束表达方式：

  m.addConstr(x == abs_(y), "absconstr")

特殊约束的表达方式和使用

- 广义约束—And

addGenConstrAnd(resvar, vars, name)

  - resvar 变量
  - vars 一组变量
  - name 广义约束名称

一组变量的值全等于1, 则取1, 否则取0。

注意：所有的变量都被视为0, 1变量, 不论他们之前被定为什么类型。

特殊约束的表达方式和使用

- 广义约束—And

  例如：x = 1且y = 1, 那么z = 1, 否则 z = 0

  ```
  m.addGenConstrAnd(z, [x,y], "andconstr")
  ```

  换成一般的约束表达方式：

  ```
  m.addConstr(z == and_(x, y), "andconstr")
  ```

特殊约束的表达方式和使用

- 广义约束—Or

addGenConstrOr(resvar, vars, name)

  - resvar 变量
  - vars 一组变量
  - name 广义约束名称

一组变量的值有一个等于1, 则取1, 否则取0。

注意：所有的变量都被视为0, 1变量, 不论他们之前被定为什么类型。

特殊约束的表达方式和使用

- 广义约束—Or

  例如：x = 0且y = 0, 那么z = 0, 否则 z = 1

  ```
  m.addGenConstrOr(z, [x,y], "orconstr")
  ```

  换成一般的约束表达方式：

  ```
  m.addConstr(z == or_(x, y), "orconstr")
  ```

特殊约束的表达方式和使用

- 广义约束—Indicator

addGenConstrIndicator(binvar, binval, lhs, sense, rhs, name)

- binvar 指示变量
- binval 指示变量的值{0,1}
- lhs 约束左端项
- sense 约束符号
- rhs 约束右端项
- name 广义约束名称

指示变量的值为1, 约束成立, 否则约束可以被违反。

特殊约束的表达方式和使用

- 广义约束—Indicator

  例如：如果 z = 1, 则 x+y <= 4

  m.addGenConstrIndicator(z, True, x + y, GRB.LESS_EQUAL, 4, 'indicator')

  换成一般的约束表达方式：

  m.addConstr((z == 1) >>(x + y<= 4))

特殊约束的表达方式和使用

- 范围约束

addRange(expr, lower, upper, name)

- expr 表达式
- lower 下界
- upper 上界
- name 约束名称

例如：5 <= x + y + z <= 10

```
m.addRange(x+y+z, 5, 10, "c")
```

换成一般的约束表达方式：

```
m.addConstr(x+y+z==[5,12])
```

特殊约束的表达方式和使用

- SOS约束(Special-Ordered Set)

addSOS(type, vars, wts=None)

- type 约束种类(GRB.SOS_TYPE1 或者 GRB.SOS_TYPE2).
- vars 变量
- wts 变量对应的权重,且权重唯一,默认1, 2, 3 ...

SOS_TYPE1 表示一组有序变量中最多有一个变量取值不为0;

SOS_TYPE2 表示一组有序变量中最多有两个变量取值不为0, 且非零变量相邻。变量是否相邻由权重决定。

```
model.addSOS(GRB.SOS_TYPE2, [x, y, z], [1, 2, 4])
```

特殊目标函数的表达方式和使用

- 目标函数(多个目标)

setObjectiveN(expr, index, priority, weight, abstol, reltol, name)

- expr 目标函数表达式
- index 目标函数对应的序号(0, 1, 2, ….)
- priority 优先级(整数值)
- weight 权重(浮点数)
- abstol 允许的目标函数值最大的降低量 abstol(浮点数)
- reltol 允许的目标函数值最大的降低量reltol*|目标函数值|(浮点数)
- name 目标函数名称

注意：所有的目标函数都为线性的，并且目标函数的优化方向一致(全部最大化或全部最小化)。可以通过乘以 -1 实现不同的优化方向。

特殊目标函数的表达方式和使用

- 目标函数(多个目标)

Gurobi 支持三种多目标模式:

- Blend(合成型) 有权重, 没有优先级。例如优化:

  Obj1 = x + y weight1 = 1

  Obj2 = x – 5y weight2 = -2

  Gurobi会混合这两个目标值形成: 1*(x+y) - 2*(x – 5y) = -x+11y

- Hierarchical(分层型) 有优先级。 例如优化:

  Obj1 = x + y priority 1 = 10

  Obj2 = x – 5y priority 2 = 5

  Gurobi 按照优先级大小优化(先优化Obj1), 若没有设定abstol或reltol, 在优化低优先级目标时,不会改变高优先级的目标值。假设Obj1=10, 在优化 Obj2 时只能在使得 Obj1=10 的所有解中挑选最优解。

- 混合以上两种。

特殊目标函数的表达方式和使用

- 目标函数(多个目标)
  - Blend(合成型)。例如:

```
m.setObjectiveN(x+y+3*z, index=0, weight=0.1, name='obj1')
m.setObjectiveN(2*x+y+3*z, index=1, weight=3, name='obj2')
```

  - Hierarchical 例如:

```
m.setObjectiveN(x+y+3*z, index=0, priority=1, name='obj1')
m.setObjectiveN(2*x+y+3*z, index=1, priority=3, name='obj2')
```

  - 混合以上两种。

```
m.setObjectiveN(x+y+3*z, index=0, weight=0.1,priority=1, name='obj1')
m.setObjectiveN(2*x+y+3*z, index=1, weight=3, priority=3, name='obj2')
```

特殊目标函数的表达方式和使用

- 目标函数(多个目标)

  通过参数 ObjNumber 选择特定的目标, 进而获得对应的目标函数值。

```
for i in range(model.NumObj):
    model.setParam(GRB.Param.ObjNumber, i)
    print('Obj%d = ' %(i+1),  model.ObjNVal)
```

特殊目标函数的表达方式和使用

- 多目标案例(多目标指派问题)

假设工厂需要把N项工作分配给N个工人,且每项工作只能由一个工人做,每位工人也只能做一项工作。已知工人$i\ (i \in N)$处理工作$j\ (j \in N)$需要时间为T_{ij},获得的利润为C_{ij}。找出一种安排方案使得完成所有工作需要的总时间最短且获得的总利润最大。

$$\boldsymbol{obj1}:\ \boldsymbol{min}\quad \sum_{i=1}^{N}\sum_{j=1}^{N} \boldsymbol{T_{ij}x_{ij}}$$

$$\boldsymbol{obj2}:\ \boldsymbol{max}\quad \sum_{i=1}^{N}\sum_{j=1}^{N} \boldsymbol{C_{ij}x_{ij}}$$

$$\sum_{i=1}^{N} \boldsymbol{x_{ij}} = 1,\ \ \forall \boldsymbol{j} \in \boldsymbol{N}$$

$$\sum_{j=1}^{N} \boldsymbol{x_{ij}} = 1,\ \ \forall \boldsymbol{i} \in \boldsymbol{N}$$

$$\boldsymbol{x_{ij}} \in \{\boldsymbol{0},\boldsymbol{1}\},\ \ \forall \boldsymbol{i},\boldsymbol{j} \in \boldsymbol{N}$$

特殊目标函数的表达方式和使用

- 多目标案例(多目标指派问题)

  案例中 T_{ij}和 C_{ij}使用随机数，$obj2$优化方向为最大化，转化为$min\ -obj2$。分别以下两种形式处理多目标：

  - 合成型： $obj1$ 权重 = 0.1　　$obj2$ 权重 = 0.5
  - 分层型： $obj1$ 优先级 = 1　　$obj2$ 优先级 = 2

特殊目标函数的表达方式和使用

- 目标函数(分段线性目标)

  setPWLObj(var, x, y)

  - var 指定变量的目标函数是分段线性
  - x 定义分段线性目标函数的点的横坐标值(非减序列)
  - y 定义分段线性目标函数的点的纵坐标值

对一些非线性模型，可以使用这一功能去线性逼近。

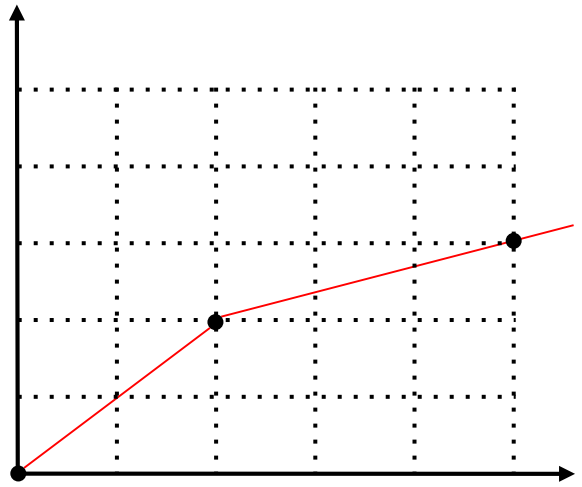
特殊目标函数的表达方式和使用

- 目标函数(分段线性目标)

  例如:

$$Max\ f(x) = \begin{cases} x & (0 \le x \le 2) \\ \frac{1}{3}x + \frac{4}{3} & (2 \le x \le 5) \end{cases}$$

```
m.setPWLObj(x, [0, 2, 5], [0, 2, 3])
m.setAttr(GRB.Attr.ModelSense, -1)
```



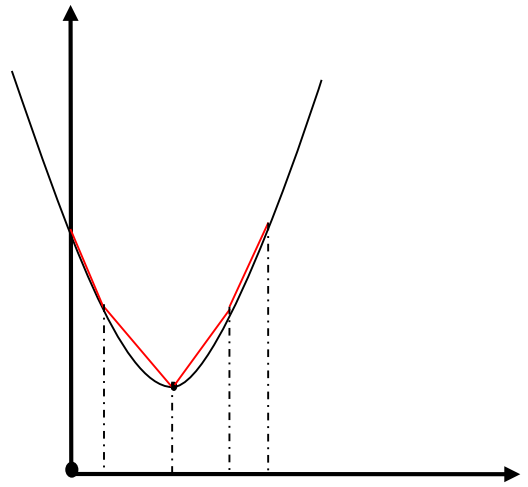
特殊目标函数的表达方式和使用

- 分段目标案例(逼近非线性)

$$min\ f(x) = (x-1)^2+2, x \in [0,10]$$

上面模型可以用Gurobi直接求解,最优解为 $f(1) = 2$。

本案例要求使用线性逼近的方法求解模型。



Solution Pool 的使用

- Solution Pool

Gurobi在搜寻最优解的过程中，会找到一些次优解(sub-optimal solutions)，有时候用户也希望知道次优解的具体情况。因此Gurobi会将计算过程中发现的所有解记录在Solution Pool里供用户查询。

Solution pool 参数

参数名称	作用	取值
PoolSearchMode	发现解的方法	0,默认。搜寻最优解。 1,搜寻更多的解但不保证解的质量。 2,尝试寻找n个解(n由PoolSolutions设定)
PoolGap	设定Pool的Gap值	Infinity, 默认。所有解与最好解的gap小于设定的值。
PoolSolutions	设定Pool存放解的量	10, 默认。设定pool里存储解的个数。

Solution Pool 的使用

- Solution Pool

Solution Pool 里的解按照质量非增排序，并从零开始编号。因此，查询具体解的情况（变量值和对应目标值等）需要先设定参数**SolutionNumber** 的值，然后通过模型属性**PoolObjVal** 和变量属性**Xn** 获得目标值和变量值。

例如：查询pool里面第4个解（索引为3）的目标值和变量值。

```
model.setParam(GRB.Param.SolutionNumber, 3)
print("obj = ",  model.PoolObjVal)
for i in range(model.NumVars):
    print(Vars[i]. VarName, " = ", Vars[i].Xn)
```

下次课程内容

(1) Callback使用方法

用户有时候在求解过程中需要实现一些功能,例如获取一些信息、终止优化、添加约束条件(割平面)、嵌入自己的算法等。

(2) 常用的线性化方法

Maxmin/Minmax 目标函数

带fixed cost目标函数

逻辑或

Partial integer variable

Maxmax/Minmin 目标函数

分式目标函数

乘积式

谢谢各位对 Gurobi 中文网络课程的支持。

我们会不断推出专题培训,敬请关注

www.gurobi.cn