



Gurobi 入门和进阶网络课程

(课程一： Gurobi 基本操作)

刃之砺信息科技有限公司（上海）有限公司

www.gurobi.cn



大纲

- (1) 安装 Gurobi 软件和 Anaconda IDE 软件，建立可视化 Python 建模环境。
- (2) 结合案例，介绍采用 Python 语言建立优化模型并求解。



为什么 Gurobi ?

Gurobi 是目前数学规划（线性和凸二次规划）优化器性能领袖、性价比领袖。

全球用户超过1600家。国内应用涵盖航空运输、电力、制造、传媒管理、生物医药、通讯、金融等。

性价比更优，技术支持响应更快。

学术许可免费，申请便捷（www.gurobi.cn）

可以求解大规模线性问题，二次型目标问题和混合整数线性和二次型问题

支持多目标优化

支持包括SUM, MAX, MIN, AND, OR等广义约束和逻辑约束

支持并行计算和分布式计算

提供了方便轻巧的接口，支持 C++, Java, Python, .Net, Matlab 和R，内存消耗少

支持多种平台，包括 Windows, Linux, Mac OS X



哪里寻找可靠的 Gurobi 资料

(1) 安装目录 examples, docs 目录下有完整的范例和使用手册、参考手册

(2) 官网 www.gurobi.com

(2.1) 在线手册: <http://www.gurobi.com/documentation/>

(2.2) 视频: <http://www.gurobi.com/resources/seminars-and-videos/seminars-videos>

(2.3) 在线课程: <http://www.gurobi.com/academia/for-online-courses>

(3) 中文网站 www.gurobi.cn

建议不要网上搜索片段化、来源不明、过时的文档



为什么 Python?

很好的编程语言

容易学习，可读性好

跨平台

适合初学者和专业编程人员

简洁、紧凑的语法

开源社区：海量可用的模块库、代码和资料

很棒的数学建模工具

很多受欢迎的数学计算工具库

Jupyter Notebook: 交互式图形化环境

有很多语言表述方式和数据结构，和数学表达式很接近

可扩展性



为什么 Python?

Sep 2018	Sep 2017	Change	Programming Language	Ratings	Change
1	1		Java	17.436%	+4.75%
2	2		C	15.447%	+8.06%
3	5	▲	Python	7.653%	+4.67%
4	3	▼	C++	7.394%	+1.83%
5	8	▲	Visual Basic .NET	5.308%	+3.33%
6	4	▼	C#	3.295%	-1.48%
7	6	▼	PHP	2.775%	+0.57%
8	7	▼	JavaScript	2.131%	+0.11%
9	-	▲	SQL	2.062%	+2.06%
10	18	▲	Objective-C	1.509%	+0.00%
11	12	▲	Delphi/Object Pascal	1.292%	-0.49%
12	10	▼	Ruby	1.291%	-0.64%
13	16	▲	MATLAB	1.276%	-0.35%
14	15	▲	Assembly language	1.232%	-0.41%
15	13	▼	Swift	1.223%	-0.54%
16	17	▲	Go	1.081%	-0.49%
17	9	⚡	Perl	1.073%	-0.88%
18	11	⚡	R	1.016%	-0.80%
19	19		PL/SQL	0.850%	-0.63%
20	14	⚡	Visual Basic	0.682%	-1.07%



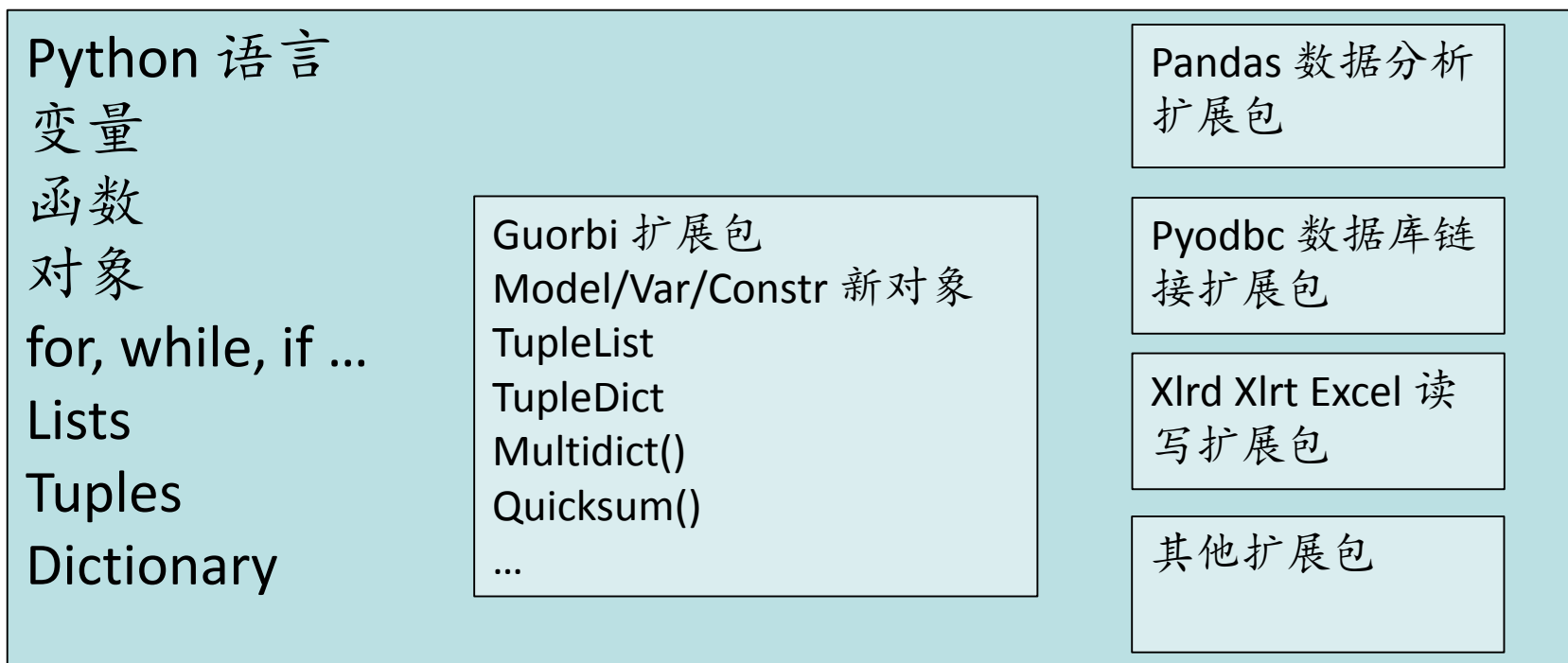
哪里寻找可靠的 Python 资料

- (1) 官网 <https://wiki.python.org/moin/BeginnersGuide>
- (2) 培训视频 <https://www.python.org/doc/av/>

使用 Gurobi 并不需要具备很深的 Python 知识。



Gurobi Python 关系



Gurobi 扩展语法和数据结构的目的:

- (1) 建立Gurobi 特有的数学模型对象 Model/Var/Constr
- (2) 提升数学建模中常见操作（下标搜索和选择、求和、系数相乘等）操作的效率要求



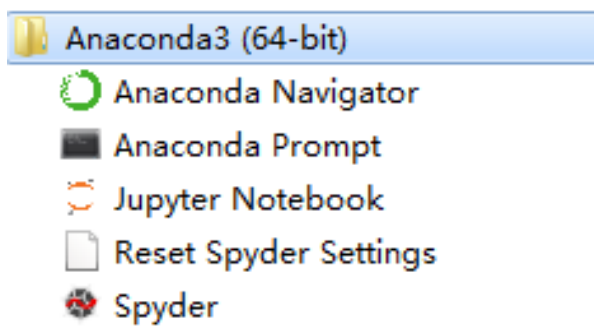
为什么 Anaconda?

最受欢迎的支持 Python 和 R 的数据科学开发、测试、训练平台

支持多种操作系统

一站式管理平台：管理扩展包，管理项目环境

带有多个常用工具：Spyder, Jupyter Notebook 等





哪里寻找可靠的 Anaconda 资料

- (1) 官网 <https://enterprise-docs.anaconda.com/en/latest/>
- (2) 培训视频 <https://www.anaconda.com/videos/>

使用 Gurobi 并不需要具备很深的 Anaconda 知识。



选择 Python 2.x 还是 Python 3.x

Python 2.x 将于 2020年1月1日停止支持。 <https://pythonclock.org/>

Gurobi 支持 Python 2.7, 3.5, 3.6, 3.7

建议选择 Anaconda 3.x 版本



使用 Gurobi + Python 的几种方式

安装包 (1) : Gurobi 安装包, 下载地址 www.gurobi.com 或者 QQ 群 251135672 的群文件

安装包 (2) : Python 安装包, 下载地址 www.python.org

安装包 (3) : Anaconda 安装包, 下载地址 www.anaconda.com

方式	安装方式	使用方式	利	弊
1	(1) Gurobi 里调 Python	gurobi.bat xxxx.py gurobi 交互界面 (Python)	简单	非 IDE 开发, 界面 不友好
2	(1) + (2) Python 里调 Gurobi	双击 <installdir>/bin/pysetup (Windows) 将 gurobipy 模块安装到 python 安装目录下	自由选择 Python IDE 环境, 例如 Canopy, Eric, iep, PyCharm, and PyDev.	需要更深知识
3	(3) 或者 (3) + (1)	conda install gurobi 将 gurobipy 安装 到 Anaconda 环境中	简单管理, 丰富 工具	学习更多工具

推荐方式 3



使用 Anaconda + Gurobi 的安装方式

- (1) Gurobi 单独安装包可以安装，也可以不安装。推荐安装：有帮助文档和案例；如果开发其他语言则需要
- (2) 根据操作系统，选择下载 Anaconda 3.x 适合版本 (www.anaconda.com)，并安装
- (3) 进入到 Anaconda Prompt（类似于 Dos 命令）（有些机器需要以管理员权限打开）
- (4) 利用 conda 扩展包管理工具，进行扩展包的安装，更新，删除等



使用 Anaconda + Gurobi 的安装方式

如果联网：

添加 Gurobi 安装路径

conda config --add channels <http://conda.anaconda.org/gurobi>

安装 Gurobi 扩展包

conda install gurobi

更新Gurobi 扩展包

conda update gurobi

删除 Gurobi 扩展包

conda remove gurobi

查看已经安装的扩展包

conda list



使用 Anaconda + Gurobi 的安装方式

如果不联网：

下载 Gurobi Conda 离线扩展包（注意区分 Python 版本和操作系统）

<https://anaconda.org/Gurobi/gurobi/files>

通过 cd 命令进入到扩展包下载目录

安装 Gurobi 扩展包

`conda install gurobi-8.0.1-py36h0b08b80_0.tar.bz2` （更换成合适的文件名字）

删除 Gurobi 扩展包

`conda remove gurobi`

查看已经安装的扩展包

`conda list`



产生许可文件

如果没有安装 Gurobi 独立安装包，那么需要在 Anaconda Prompt 中运行；如果安装了 Gurobi 独立安装包，也可以在普通 Dos 命令下（CMD）运行，

```
grbgetkey XXXXXXXX-XXXXXX-XXXX-XXXX-XXXXXXXXXXXXXXXXXX
```

需要在线激活。如果 IP 地址不识别为学校，则需要通过 www.gurobi.cn 的步骤申请免 IP 验证的学术许可。

产生后的许可文件 gurobi.lic 放置在 c:\gurobi（Windows）或者 opt/gurobi（Linux）目录下，如果没有这些目录，可以创建一个。以后不管安装多少版本，许可文件都可以放在这个目录下，集中管理。



推荐 Jupyter Notebook

网页应用程序，创建实时交互内容，可以包含实时数据、图表、实时代码、排版公式、文档等丰富内容。

源于 Ipython 项目（2014）

包含在 Anaconda 中

广泛用户基础

在 Anaconda Prompt 下，cd 进入到工作目录，然后运行
jupyter notebook



介绍 Spyder

Python 可视化集成开发工具

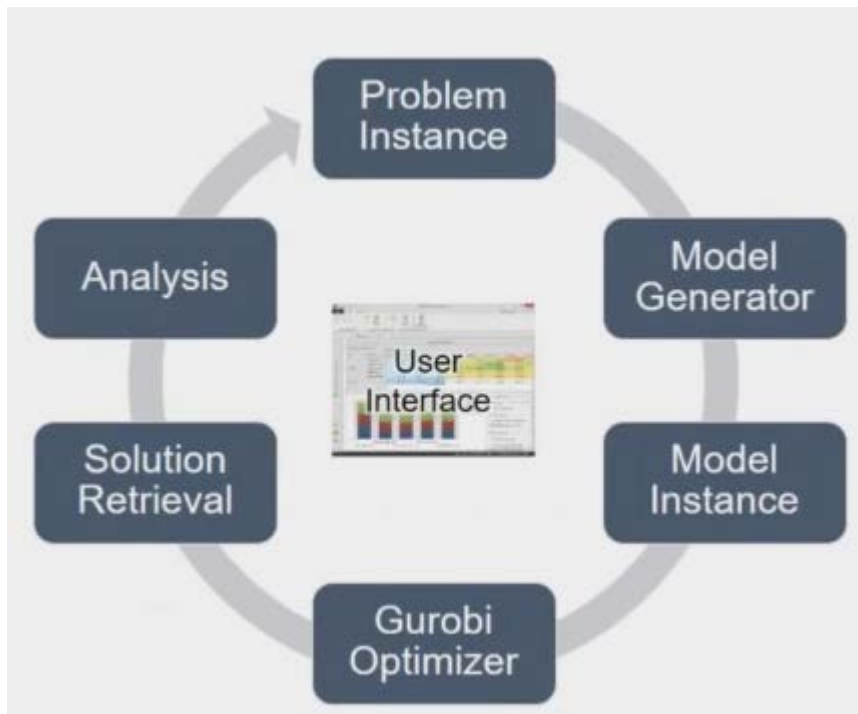
界面友好的代码编辑器

调试工具和交互工具

帮助文档



Gurobi 建模过程



Problem Instance: 待优化问题

Model Generator: 将数据组合成模型，
产生计算机模型对象

Model Instance: 存在于内存的一个完整
数学模型

Gurobi Optimizer: Gurobi 优化求解

Solution Retrieval: 根据需要读取优化结
果

Analysis: 对结果进行分析
循环往复，直到获得满意结果



Gurobi 建模基本概念

Parameter（参数）控制优化器的行为，需要在优化启动前设置。

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

例如：

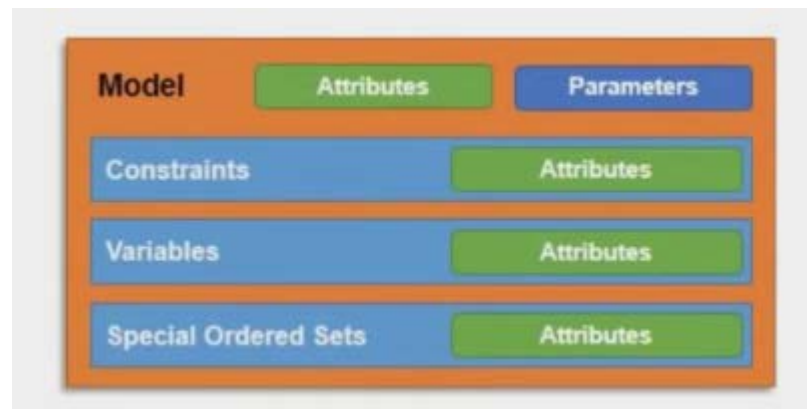
模型 ModelSense

变量 LB/UB

约束 RHS

Environment 是包含模型和全局参数的一个容器，也是许可控制的节点。

Python 中的环境会自动建立。





Python 语法

网上已经有大量培训资料和视频 (www.python.org)。Gurobi 建模对于 Python 语言本身要求不高。

(1) 变量

```
x=5  
y="abc"
```

(2) 对象、属性和方法

```
m=Model("myModel")  
m.ModelSense=
```

(3) for, while 循环

(4) if .. elif



Python 语法

Python List（列表） 用 `[]` 表示，适合做下标、索引、变量、约束等各种对象的集合

一个排序的集合

- `cities = ['A', 'B', 'C', 'D']`

可以修改、添加、删除，排序

- Ex: Add, delete, sort elements

虽然 Python 有 `set` 类型，但 `List` 更有效率



Python 语法

Python Tuple （元组），用()表示

固定、不可修改的组合

- link = ('A', 'B')

适合做多维下标



Python 语法

Python Dictionaries 字典

Python 中的字典可以通过任意**键值**对数据进行映射。任何无法修改的 Python 对象都可以当作键值使用：整数、浮点数、字符串、和元组。

适合用来表示带下标的数值，例如变量

举个例子，以下这些语句创建了一个字典对象 `x`，并将数值 1 与键值 ('Pens', 'Denver', 'New York') 进行关联：

```
gurobi> x = {} # 创建一个空白的字典对象
gurobi> x[('Pens', 'Denver', 'New York')] = 1
gurobi> print x[('Pens', 'Denver', 'New York')] # 1
```

在 Python 中，通过元组访问字典时可以省略圆括号，因此以下语句同样也是同样有效的：

```
gurobi> x = {}
gurobi> x['Pens', 'Denver', 'New York'] = 2
gurobi> print x['Pens', 'Denver', 'New York'] # 2
```




Python 语法

Python List/Tuple/Dictionary 的不足

在建模过程中，经常要对带下标数据做挑选，不同下标的数据进行组合，这样面临着二个处理方法

- (1) 全部循环。多维下标意味着多重循环+ if 条件
这样的处理方法没有效率
- (2) 采用特殊的Gurobi 扩展对象 TupleList 和 TupleDict

Results of tuplelist speed test

N	# elements (K)	Dense count	Tuplelist count
100	1050	0.36 sec	0.0006 sec
200	2100	3.00 sec	0.0013 sec
400	4200	24.17 sec	0.0025 sec
800	8400	195.26 sec	0.0049 sec



Python 语法

Gurobi tuplelist -- Python list 的扩展对象

```
Cities= [('A','B'), ('A','C'), ('B','C'),('B','D'),('C','D')]  
Routes = tuplelist(Cities)
```

使用 tuplelist() 不能忘记 from gurobipy import *
tuplelist 增加了快速筛选 select 功能

```
print(Routes.select('A','*'))
```

```
Result=[]  
for i,j in Cities:  
    if i=='A':  
        Result.append((i,j))  
print(Result)
```



Python 语法

Gurobi tupledict – Python Dictionary 扩展

键值为 tuple（元组），可以使用 select, sum, prod 函数

用于变量和约束（后面案例中体现）



Python 语法

Multidict() 创建 tuplelist 和 tupledict 的便捷方法

```
cities, supply, demand = multidict({  
'A': [100, 20],  
'B': [150, 50],  
'C': [20, 300],  
'D': [10, 200]})
```

返回 cities (tuplelist)、supply (tupledic) 和 demand (tupledict)



Python 语法

创建 list （列表解析 List Comprehension）

```
a=[]
```

```
a.append('A')
```

```
b = [i**2 for i in range(6)] # [0,1,4,9,16,25]
```

```
c = [(i,j) for j in range(4) for i in range(j)] #[(0,1), (0,2), (1,2), (0,3), (1,3), (2,3)]
```

```
d = [i for i in range(10) if i not in b] # [2,3,5,6,7,8]
```

```
Pairs=[]
```

```
for j in range(4):
```

```
    for i in range(j):
```

```
        Pairs.append((i,j))
```



Python 语法

Generator (生成器)

```
SumSquares = sum(i**2 for i in range(6)) # 55
```

Gurobi 中采用 quicksum, 效率更高

```
obj = quicksum(cost[i,j]*x[i,j] for i,j in arcs)
```



Python 语法

tupledict (Gurobi 变量一般都是 tupledict 类型) 有 sum 函数

```
from gurobipy import *  
m=Model()  
x=m.addVars(3,4, vtype=GRB.BINARY, name="x")  
m.addConstrs((x.sum(i,'*')<=1 for i in range(3)), name="con")  
m.update()  
m.write("test.lp")
```

产生如下约束

$$\begin{aligned}x[0,0] + x[0,1] + x[0,2] + x[0,3] &\leq 1 \\x[1,0] + x[1,1] + x[1,2] + x[1,3] &\leq 1 \\x[2,0] + x[2,1] + x[2,2] + x[2,3] &\leq 1\end{aligned}$$



Python 语法

tupledict (Gurobi 变量一般都是 tupledict 类型) 有 prod 函数, 用于变量和系数相乘后累加。

以下二个表达式等效。(x 和 cost 要有相同的键值)

```
obj = quicksum(cost[i,j] * x[i,j] for i,j in arcs)
obj = x.prod(cost)
```




Python 语法

建模建议，尽量采用稀疏方式

采用 tuplelists 筛选和指定合适的下标组合关系

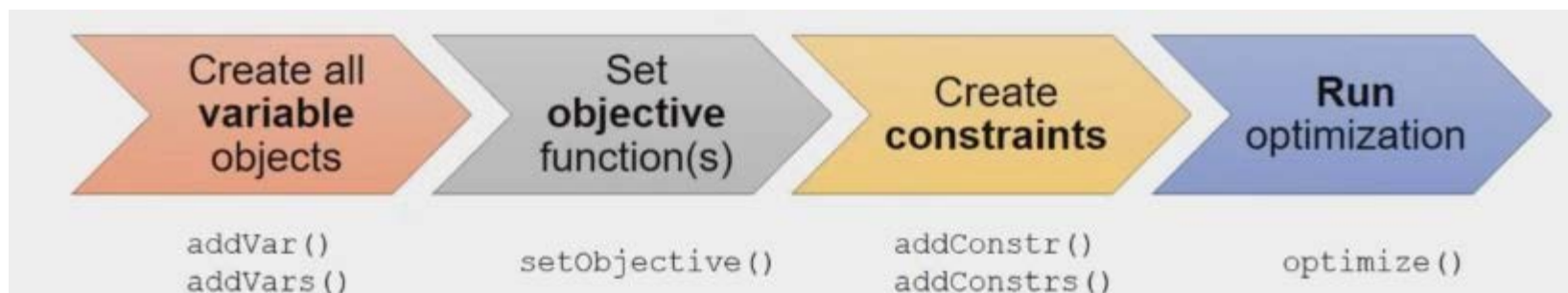
基于这些组合关系建立变量和数据字典

利用 `tuplelist.select()` 以及 `tupledict.select()`, `tupledict.sum()`, `tupledict.prod()` 来

对下标进行组合处理



Gurobi 建模过程





m=Model()

$$\begin{aligned} \max \quad & \sum_{i \in I, j \in J} c_{ij} \cdot x_{ij} \\ \text{s.t.} \quad & \sum_{i \in I} x_{ij} \leq 1 \quad \forall j \in J \\ & \sum_{j \in J} x_{ij} \leq 1 \quad \forall i \in I \\ & x_{ij} \text{ binary} \quad \forall i \in I, j \in J \end{aligned}$$



m.addVar()

m.addVars()

$$\begin{aligned} \max \quad & \sum_{i \in I, j \in J} c_{ij} \cdot x_{ij} \\ \text{s.t.} \quad & \sum_{i \in I} x_{ij} \leq 1 \quad \forall j \in J \\ & \sum_{j \in J} x_{ij} \leq 1 \quad \forall i \in I \\ & x_{ij} \text{ binary} \quad \forall i \in I, j \in J \end{aligned}$$

Decision Variables



m.setObjective()

$$\begin{array}{llll} \max & \sum_{i \in I, j \in J} c_{ij} \cdot x_{ij} & & \text{Objective} \\ \text{s.t.} & \sum_{i \in I} x_{ij} & \leq & 1 \quad \forall j \in J \\ & \sum_{j \in J} x_{ij} & \leq & 1 \quad \forall i \in I \\ & x_{ij} & \text{binary} & \forall i \in I, j \in J \end{array}$$



m.addConstr()
m.addConstrs()

$$\max \sum_{i \in I, j \in J} c_{ij} \cdot x_{ij}$$

Constraints

s.t.

$$\sum_{i \in I} x_{ij}$$

\leq

1

$$\forall j \in J$$

$$\sum_{j \in J} x_{ij}$$

\leq

1

$$\forall i \in I$$

$$x_{ij}$$

binary

$$\forall i \in I, j \in J$$



m.addVar() / m.addVars()
m.addConstr() / m.addConstrs()
m.setObjective()

$$\begin{aligned} \max \quad & \sum_{i \in I, j \in J} c_{ij} \cdot x_{ij} && \text{Data coefficients} \\ \text{s.t.} \quad & \sum_{i \in I} x_{ij} \leq 1 && \forall j \in J \\ & \sum_{j \in J} x_{ij} \leq 1 && \forall i \in I \\ & x_{ij} \text{ binary} && \forall i \in I, j \in J \end{aligned}$$



Lists

TupleList

$$\begin{array}{llll} \max & \sum_{i \in I, j \in J} c_{ij} \cdot x_{ij} & & \text{Index sets} \\ \text{s.t.} & \sum_{i \in I} x_{ij} & \leq & 1 \quad \forall j \in J \\ & \sum_{j \in J} x_{ij} & \leq & 1 \quad \forall i \in I \\ & x_{ij} & \text{binary} & \forall i \in I, j \in J \end{array}$$



TupleList.select()

$$\begin{aligned} \max \quad & \sum_{i \in I, j \in J} c_{ij} \cdot x_{ij} \\ \text{s.t.} \quad & \sum_{i \in I} x_{ij} \leq 1 \quad \forall j \in J \\ & \sum_{j \in J} x_{ij} \leq 1 \quad \forall i \in I \\ & x_{ij} \text{ binary} \quad \forall i \in I, j \in J \end{aligned}$$

Subscripts



+ - *

tupledict.prod()

$$\max \sum_{i \in I, j \in J} c_{ij} \cdot x_{ij}$$

Arithmetic operators

$$\text{s.t.} \quad \sum_{i \in I} x_{ij} \leq 1 \quad \forall j \in J$$

$$\sum_{j \in J} x_{ij} \leq 1 \quad \forall i \in I$$

$$x_{ij} \text{ binary} \quad \forall i \in I, j \in J$$



\leq, \geq

GRB.LESS_EQUAL, GRB.EQUAL, or
GRB.GREATER_EQUAL

$$\begin{array}{llll} \max & \sum_{i \in I, j \in J} c_{ij} \cdot x_{ij} & & \text{Constraint operators} \\ \text{s.t.} & \sum_{i \in I} x_{ij} & \leq & 1 \quad \forall j \in J \\ & \sum_{j \in J} x_{ij} & \leq & 1 \quad \forall i \in I \\ & x_{ij} & \text{binary} & \forall i \in I, j \in J \end{array}$$



List Comprehension 列表解析 中的 for 语句

$$\begin{array}{llll} \max & \sum_{i \in I, j \in J} c_{ij} \cdot x_{ij} & & \text{For-all operators} \\ \text{s.t.} & \sum_{i \in I} x_{ij} \leq 1 & \forall j \in J & \\ & \sum_{j \in J} x_{ij} \leq 1 & \forall i \in I & \\ & x_{ij} & \text{binary} & \forall i \in I, j \in J \end{array}$$



quicksum

$$\begin{array}{ll} \max & \sum_{i \in I, j \in J} c_{ij} \cdot x_{ij} \\ \text{s.t.} & \sum_{i \in I} x_{ij} \leq 1 \quad \forall j \in J \\ & \sum_{j \in J} x_{ij} \leq 1 \quad \forall i \in I \\ & x_{ij} \text{ binary} \quad \forall i \in I, j \in J \end{array}$$

Aggregate sum operators



Gurobi 建模举例 1 (<examples/python/mip1.py>)

$$\begin{array}{ll}\max & x + y + 2z \\ \text{s.t.} & x + 2y + 3z \leq 4 \\ & x + y \geq 1 \\ & x, y, z \in \{0, 1\}\end{array}$$



Gurobi 建模举例 2 营养配方模型 (examples/python/diet.py)

人体需要四种营养: category : calories, protein, fat, sodium

食物来源: foods= hamburger, chicken, hot dog, fries, macaroni, pizza, salad, milk, ice cream

营养吸收每天有上限和下限

单位重量食物价格不同

单位重量食物所含营养成分不同

求达到足够营养花费的代价最小



从外部文件读入数据

数据量比较大的时候，可以直接从 Excel, ODBC 数据库等读入，然后用 `multidict`, `list.append`, `list[n]=xxx`, `dict[k1,ke..]=value` 等方式赋值。

参考 `example/python` 目录下的 `diet` 系列模型



下次课程中将会介绍

- (1) Gurobi 常用的重要参数和属性
- (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

感谢参加。请抽出时间仔细阅读软件自带的范例和手册



Gurobi 入门和进阶网络课程

课程二：Gurobi 功能和操作进阶

刃之砺信息科技有限公司(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

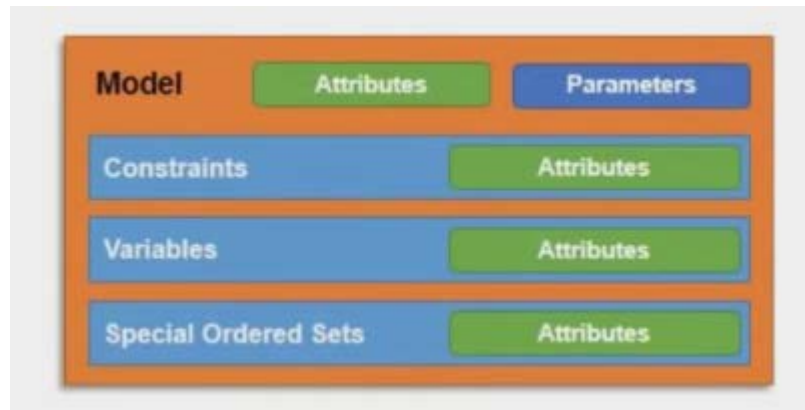
控制控制台输出log记录 LogToConsole

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 $\text{gap} < 0.1$, 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

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

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

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

```
# 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)
```



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: **IIS** 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](https://www.gurobi.com/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 \leq `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

$n = \{0, 1, \dots, \text{TuneResultCount}-1\}$, 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($x = \max(x_1, x_2, 10)$)
- 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($x = \min(x_1, x_2, 10)$)
- 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 $x + y \leq 4$

```
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: $5 \leq x + y + z \leq 10$

```
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:

$$\text{Obj1} = x + y \quad \text{weight1} = 1$$

$$\text{Obj2} = x - 5y \quad \text{weight2} = -2$$

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:

$$\text{Obj1} = x + y \quad \text{priority 1} = 10$$

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

Gurobi 按照优先级大小优化(先优化Obj1), 若没有设定abstol或reitol, 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。

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

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

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

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

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



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

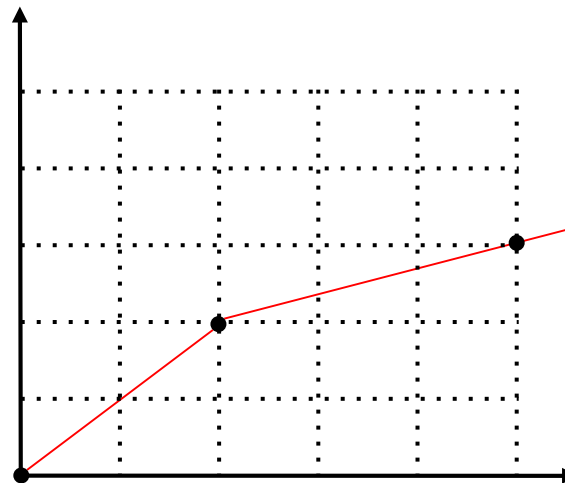
- 目标函数(分段线性目标)

例如:

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

```
m.setPWLObj(x, [0, 2, 5], [0, 2, 3])
```

```
m.setAttr(GRB.Attr.ModelSense, -1)
```





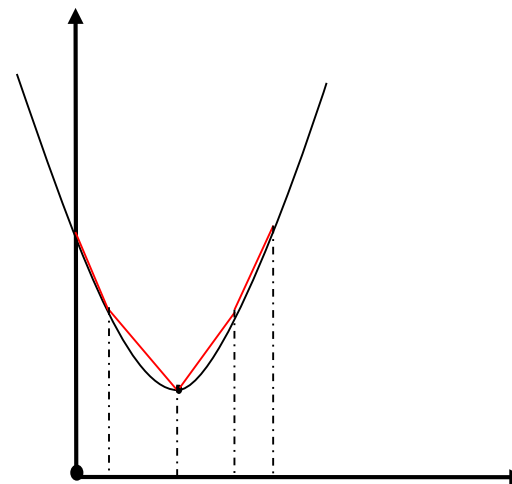
特殊目标函数的表达方式和使用

- 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



Gurobi 入门和进阶网络课程

课程三：Gurobi 高级操作和使用方法

刃之砺信息科技有限公司（上海）有限公司



课程大纲

(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



Callback使用方法

- Callback函数

Callback为用户在求解模型时提供了高级控制功能。它允许用户在Gurobi求解过程中获取信息、终止优化、加入额外约束条件(割平面)、加入自己开发的算法等。

定义CallBack函数: **def 函数名(model, where):**

调用CallBack函数: **m.optimize(callback函数名)**

CallBack函数使用时需要注意两个重要的参数:

where: 回调函数触发点

what: 获取何种信息, **what**能够获取什么取决于**where**



Callback使用方法

- where 参数取值

where	数值	优化器状态
POLLING	0	轮询回调
PRESOLVE	1	预处理
SIMPLEX	2	单纯形
MIP	3	当前Mip
MIPSOL	4	发现新的Mip解
MIPNODE	5	当前探索节点
MESSAGE	6	打印出Log信息
BARRIER	7	当前内点法
MULTIOBJ	8	当前多目标



Callback使用方法

- what参数取值

what 值取决于 where 的取值(使用时一定要正确对应两者关系), 例如 where = MIP 时, what 取值如下:

what	类型	描述
MIP_OBJBST	double	当前最优目标值
MIP_OBJBND	double	当前最优界
MIP_NODCNT	double	当前已探索的节点数
MIP_SOLCNT	int	当前发现可行解的数量
MIP_CUTCNT	int	当前割平面使用次数
MIP_NODLFT	double	当前未搜索的节点数
MIP_ITRCNT	double	当前单纯形迭代步数



Callback使用方法

- what参数取值

当 where = MIPSOL 时, what的取值如下:

what	类型	描述
MIPSOL_SOL	double*	当前解的具体取值
MIPSOL_OBJ	double	新解的目标值
MIPSOL_OBJBST	double	当前最优目标值
MIPSOL_OBJBND	double	当前最优界
MIPSOL_NODCNT	double	当前以搜索的节点数
MIPSOL_SOLCNT	int	当前发现可行解的数量

如果Where = MIPSOL, what 取 MIP_OBJBST 就会报错。



Callback使用方法

- Callback函数

查询一些信息, 例如目标值, 节点数等。使用时注意 what 与 where 的匹配。

cbGet(what)

- what 获取何种信息, what 能够获取什么取决于where

例如, 查询当前单纯形的目标函数值。

```
def mycallback(model, where):
```

```
    if where == GRB.Callback.SIMPLEX:
```

```
        print(model.cbGet(GRB.Callback.SPX_OBJVAL))
```

```
model.optimize(mycallback)
```



Callback使用方法

- Callback函数

查询变量在当前节点的松弛解。

注意 where == GRB.Callback.MIPNODE 且 GRB.Callback.MIPNODE_STATUS == GRB.OPTIMAL 才起作用。

cbGetNodeRel (vars) 函数返回：变量在当前节点的松弛解

- vars 需要查询的变量

```
def mycallback(model, where):
```

```
    if where == GRB.Callback.MIPNODE and model.cbGet(GRB.Callback.MIPNODE_STATUS)== GRB.OPTIMAL :
```

```
        print model.cbGetNodeRel(model._vars)
```

```
model._vars = model.getVars()
```

```
model.optimize(mycallback)
```

callback函数可以通过“_变量名” 获得外部变量的值。



Callback使用方法

- Callback函数

查询可行解变量的取值。

注意 where == GRB.Callback.MIPSOL 或者 GRB.Callback.MULTIOBJ。

cbGetSolution (vars) 函数返回：变量在新可行解中的取值

- vars 需要查询的变量

```
def mycallback(model, where):
```

```
    if where == GRB.Callback.MIPSOL:
```

```
        print (model.cbGetSolution(model._vars))
```

```
model._vars = model.getVars()
```

```
model.optimize(mycallback)
```



Callback使用方法

- Callback函数

在节点添加割平面。

注意 where == GRB.Callback.MIPNODE

参数 **PreCrush=1** (关掉Gurobi预处理对模型约束的转化)。

cbCut (lhs, sense, rhs)

- lhs 左端项
- sense 符号
- rhs 右端项



Callback使用方法

- Callback函数

例如, 通过节点的松弛解信息构造割平面。

```
def mycallback(model, where):
```

```
    if where == GRB.Callback.MIPNODE:
```

```
        status = model.cbGet(GRB.Callback.MIPNODE_STATUS)
```

```
        if status == GRB.OPTIMAL:
```

```
            rel = model.cbGetNodeRel([model._vars[0], model._vars[1]])
```

```
            if rel[0] + rel[1] > 1.1:
```

```
                model.cbCut(model._vars[0] + model._vars[1] <= 1)
```

```
model._vars = model.getVars()
```

```
model.Params.PreCrush = 1
```

```
model.optimize(mycallback)
```




Callback使用方法

- Callback函数

在节点添加 Lazy cut。(与一般cut区别在于只有在被违反的时候才起作用)

注意 Where == GRB.Callback.MIPNODE or GRB.Callback.MIPSOL

使用时必须设定参数LazyConstraints = 1

cbLazy(lhs, sense, rhs)

- lhs 左端项
- sense 符号
- rhs 右端项



Callback使用方法

- Callback函数

例如, 通过可行解的信息构造Lazy cut。

```
def mycallback(model, where):  
    if where == GRB.Callback.MIPSOL:  
        sol = model.cbGetSolution([model._vars[0], model._vars[1]])  
        if sol[0] + sol[1] > 1.1:  
            model.cbLazy(model._vars[0] + model._vars[1] <= 1)  
model._vars = model.getVars()  
model.Params.lazyConstraints = 1  
model.optimize(mycallback)
```



Callback使用方法

- Callback函数

向当前节点导入一个解(完整或部分都可以)。

注意 where == GRB.Callback.MIPNODE

cbSetSolution (vars, solution)

- vars 变量
- solution 变量的值

cbUseSolution ()

计算导入解的目标值。



Callback使用方法

- Callback函数

```
def mycallback(model, where):  
    if where == GRB.Callback.MIPNODE:  
        model.cbSetSolution(vars, newsolution)  
model.optimize(mycallback)
```

对复杂的问题, 可以开发启发式算法找到高质量的解, 然后导入解让Gurobi在其基础上继续求解。

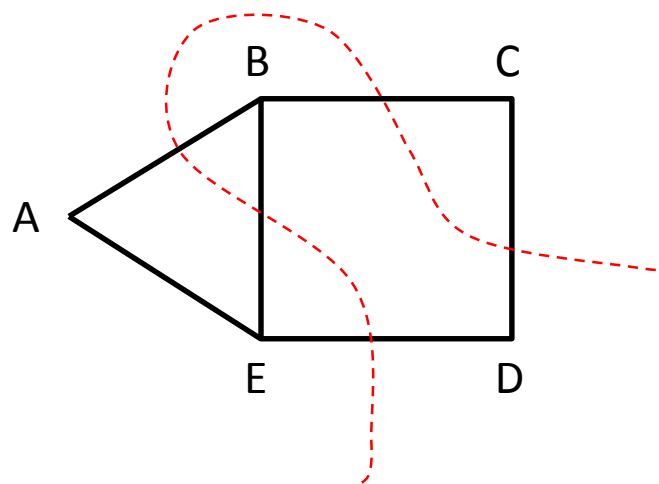


Callback使用方法

- Callback案例

最大割问题(Maximum cut) + RINS heuristic

假设图中线段都被赋上了权重, 希望找到一种方案将顶点分成两个子集(记为 m, n), 使得属于不同子集点的连线权重和最大。例如, 图中的方案 $m = \{A, E, C\}$, $n = \{B, D\}$, 权重总和 = $AB + BE + BC + CD + ED$ 。





Callback使用方法

- Callback 案例

最大割问题(Maximum cut)模型

参数: C_{ij} 线段 i, j 权重, N 顶点数量

变量: $x_i = 1$, if $i \in m$ or $x_i = -1$, if $i \in n$

$$\max \quad \frac{1}{4} \sum_{i=1}^N \sum_{j=1}^N C_{ij} (1 - x_i x_j)$$

$$x_i \in \{-1, 1\}$$

$$x = 2y - 1, \quad y \in \{0, 1\}$$



Callback使用方法

- Callback 案例

RINS heuristic 核心思想

随着整数规划模型的求解进程, 节点松弛模型的解与最优解之间的差距可能会越来越小, 体现在松弛解的部分变量值与最优解对应变量值相等或差距很小。因此利用松弛模型的信息可能会更快发现高质量的可行解。

RINS heuristic 基于上面的想法, 在求解过程中抓取节点松弛解(可能是部分整数, 部分小数), 固定模型中对应的变量的取值构造一个子模型(规模往往远小于原模型), 然后求解子模型。如果发现了更好的可行解, 把解传递给优化器让其在它的基础上继续求解原模型。



Callback使用方法

- Callback案例代码

$$x = 2y - 1, y \in \{0, 1\} \Rightarrow$$

$$\max \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N C_{ij} (y_i + y_j - 2y_i y_j)$$

N = 20

#随机产生线段权重

random.seed(1)

Cmatrix = {(i,j):random.randint(0,100) for i in range(N) for j in range(N)}

m = Model('MaximumCut')

#添加变量

y = m.addVars(N, vtype=GRB.BINARY, name='y')

#构造目标函数

obj = QuadExpr()

for i in range(N):

for j in range(N):

obj = obj+Cmatrix[i,j]*(y[i]+y[j]-2*y[i]*y[j])

m.setObjective(0.5*obj, -1)

#设置求解时间

m.Params.TimeLimit = 600

#外部变量

m._y = y

m._N = N

#求解

m.optimize(RINScallback)

#获得目标值和变量值

print("Obj = ",m.ObjVal)

for i in range(N):

print(y[i].VarName, ' = ', y[i].x)



- Callback案例代码

[illegible]



常用线性化方法

• 广义约束—Max/Min

线性化方法:

取值	0	1
u_1	$x \geq z - M$ 恒成立	$x \geq z$
u_2	$y \geq z - M$ 恒成立	$y \geq z$
u_3	$3 \geq z - M$ 恒成立	$3 \geq z$

$$z = \max(x, y, 3)$$

$$x \leq z, y \leq z, 3 \leq z$$

$$x \geq z - M(1 - u_1)$$

$$y \geq z - M(1 - u_2)$$

$$3 \geq z - M(1 - u_3)$$

$$u_1 + u_2 + u_3 \geq 1$$

$$u_1, u_2, u_3 \in \{0, 1\}$$

$$x \geq z, y \geq z, 3 \geq z$$

至少有一个约束成立

按照 Gurobi 广义约束的写法, 可以直接写为: `m.addConstr(z==max_(x, y, 3))`



常用线性化方法

- 广义约束—Max/Min

线性化方法:

$$z = \min(x, y, 3)$$

$$x \geq z, y \geq z, 3 \geq z$$

$$x \leq z - M(1 - u_1)$$

$$y \leq z - M(1 - u_2)$$

$$3 \leq z - M(1 - u_3)$$

$$u_1 + u_2 + u_3 \geq 1$$

$$u_1, u_2, u_3 \in \{0, 1\}$$

按照 Gurobi 广义约束的方法, 可以直接写为: `m.addConstr(z==min_(x, y, 3))`



常用线性化方法

- 目标函数中存在绝对值

$$\min \sum_i c_i |x_i|$$

$$Ax = b$$

$$x_i \text{ free}, c_i \geq 0, \forall i$$

线性化方法1: $y_i = |x_i|, y_i \geq x_i, y_i \geq -x_i$

$$\min \sum_i c_i y_i$$

$$Ax = b$$

$$y_i \geq x_i, y_i \geq -x_i$$

$$x_i \text{ free}, c_i > 0$$



常用线性化方法

- 目标函数中存在绝对值

线性化方法2: $\forall x, \exists u, v \geq 0$, 使得 $x = u - v, |x| = u + v$, 其中 $u = \frac{|x|+x}{2}, v = \frac{|x|-x}{2}$

$$\begin{aligned} \min \quad & \sum_i c_i(u_i + v_i) \\ & A(u - v) = b \\ & u, v \geq 0 \end{aligned}$$

按照 Gurobi 广义约束的方法, 可以直接写为: `m.addConstr(y==abs_(x))`



常用线性化方法

- Maxmin/Minmax 目标函数

线性化方法：

$$\max(\min_{k \in K} \sum_i c_{ki} x_i)$$

$$\max z$$

$$z \leq \sum_i c_{ki} x_i, \forall k \in K$$

$$\min(\max_{k \in K} \sum_i c_{ki} x_i)$$

线性化方法：

$$\min z$$

$$z \geq \sum_i c_{ki} x_i, \forall k \in K$$



常用线性化方法

- Maxmin/Minmax 目标函数

$$\max(\min \begin{matrix} x + 2y + 10 \\ 3x + y + 1 \end{matrix})$$

线性化方法：

$$\max z$$

$$z \leq x + 2y + 10$$

$$z \leq 3x + y + 1$$



常用线性化方法

- Maxmax/Minmin 目标函数

$$\max(\max_{k \in K} \sum_i c_{ki} x_i)$$

线性化方法：

$$\begin{aligned} \max z \\ \sum_i c_{ki} x_i \geq z - M(1 - y_k), \forall k \in K \\ \sum_k y_k \geq 1 \\ y_k \in \{0,1\} \end{aligned}$$



常用线性化方法

- Maxmax/Minmin 目标函数

$$\max(\max_{x+2y+10} \max_{3x+y+1})$$

线性化方法：

$$\max z$$

$$x + 2y + 10 \geq z - M(1 - u)$$

$$3x + y + 1 \geq z - M(1 - v)$$

$$u + v \geq 1$$

$$u, v \in \{0, 1\}$$

可以去掉: $x + 2y + 10 \leq z$

$$3x + y + 1 \leq z$$



常用线性化方法

- Maxmax/Minmin 目标函数

$$\min(\min_{k \in K} \sum_i c_{ki} x_i)$$

线性化方法：

$$\min z$$

$$\sum_i c_{ki} x_i \leq z + M(1 - y_k), \forall k \in K$$

$$\sum_k y_k \geq 1$$

$$y_k \in \{0,1\}$$



常用线性化方法

- 带fixed cost目标函数

$$\min f(x) = \begin{cases} 0, & x = 0 \\ cx + k, & x > 0, k > 0 \end{cases}$$

线性化方法：

$$\min cx + ky$$

$$x \leq My$$

$$y \in \{0,1\}$$



常用线性化方法

- 分式目标函数

$$\begin{aligned} \min \quad & \sum_i (c_i x_i + \alpha) / \sum_i (d_i x_i + \beta) \\ & \sum_i a_{ij} x_i \leq b_j, \quad \forall j \in J \\ & \sum_i d_i x_i + \beta > 0, \quad x_i \geq 0, \quad \forall j \in J \end{aligned}$$

线性化方法：令 $y = \frac{1}{\sum_i (d_i x_i + \beta)} > 0$

$$\min \sum_i (c_i x_i y + \alpha y)$$

$$\sum_i a_{ij} x_i \leq b_j, \quad \forall j \in J$$

$$\sum_i d_i x_i y + \beta y = 1, \quad \forall j \in J$$

$$y > 0, x_i \geq 0, \quad \forall j \in J$$

$$\boxed{z_i = x_i y}$$

$$\min \sum_i (c_i z_i + \alpha y)$$

$$\sum_i a_{ij} z_i \leq b_j y, \quad \forall j \in J$$

$$\sum_i d_i z_i + \beta y = 1, \quad \forall j \in J$$

$$y > 0, z_i \geq 0, \quad \forall j \in J$$



常用线性化方法

- 分式目标函数

$$\min \frac{2x + y + 1}{x + 3y}$$

$$\begin{aligned} 5x + y &\leq 6 \\ x + 3y &> 0, x, y \geq 0 \end{aligned}$$

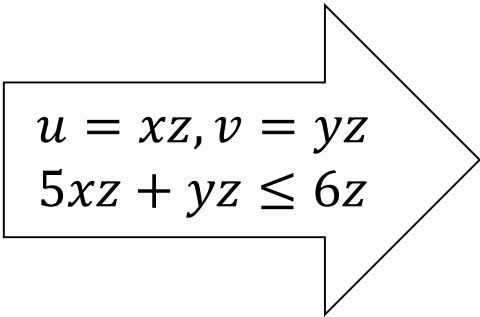
线性化方法：令 $z = \frac{1}{x+3y} > 0$

$$\min (2x + y)z + z$$

$$5x + y \leq 6$$

$$z(x + 3y) = 1$$

$$x, y \geq 0, z > 0$$


$$\begin{aligned} u &= xz, v = yz \\ 5xz + yz &\leq 6z \end{aligned}$$

$$\min 2u + v + z$$

$$5u + v \leq 6z$$

$$u + 3v = 1$$

$$z > 0, u, v \geq 0$$



常用线性化方法

- 逻辑或

$\sum_j a_{1j} x \leq b_1$ 或 $\sum_j a_{2j} x \leq b_2$ (两个约束至少一个成立)

线性化方法：

$$\sum_j a_{ij} x \leq b_i + M(1 - y_i), i = 1, 2$$

$$\sum_i y_i \geq 1$$

$$y_i \in \{0, 1\}, i = 1, 2$$



常用线性化方法

- 逻辑或

$$x + 2y + 10 \leq 15 \text{ 或 } 3x + y + 1 \leq 5$$

线性化方法：

$$x + 2y + 10 \leq 15 + M(1 - u)$$

$$3x + y + 1 \leq 5 + M(1 - v)$$

$$u + v \geq 1$$

$$u, v \in \{0, 1\}$$



常用线性化方法

- 逻辑或

$\sum_j a_{1j} x \leq b_1$ 或 $\sum_j a_{2j} x = b_2$ (两个约束至少一个成立)

线性化方法：

$$\sum_j a_{ij} x \leq b_i + M(1 - y_i), i = 1, 2$$

$$\sum_j a_{2j} x \geq b_2 - M(1 - y_2)$$

$$\sum_i y_i \geq 1$$

$$y_i \in \{0, 1\}, i = 1, 2$$



常用线性化方法

- 逻辑或

$$x + 2y + 10 \leq 15 \text{ 或 } 3x + y + 1 = 5$$

线性化方法：

$$x + 2y + 10 \leq 15 + M(1 - u)$$

$$3x + y + 1 \leq 5 + M(1 - v)$$

$$3x + y + 1 \geq 5 - M(1 - v)$$

$$u + v \geq 1$$

$$u, v \in \{0,1\}$$



常用线性化方法

- 逻辑或

$\sum_j a_{1j} x = b_1$ 或 $\sum_j a_{2j} x = b_2$ (两个约束至少一个成立)

线性化方法：

$$\sum_j a_{ij} x \leq b_i + M(1 - y_i), i = 1, 2$$

$$\sum_j a_{ij} x \geq b_i - M(1 - y_i), i = 1, 2$$

$$\sum_i y_i \geq 1$$

$$y_i \in \{0, 1\}, i = 1, 2$$



常用线性化方法

- 逻辑或

$$x + 2y + 10 = 15 \text{ 或 } 3x + y + 1 = 5$$

线性化方法：

$$x + 2y + 10 \leq 15 + M(1 - u)$$

$$x + 2y + 10 \geq 15 - M(1 - u)$$

$$3x + y + 1 \leq 5 + M(1 - v)$$

$$3x + y + 1 \geq 5 - M(1 - v)$$

$$u + v \geq 1$$

$$u, v \in \{0, 1\}$$



常用线性化方法

- 乘积式

x_1x_2 , 其中 $x_1, x_2 \in \{0, 1\}$

线性化方法: $y = x_1x_2$

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

$$y \leq x_1$$

$$y \leq x_2$$

$$y \geq x_1 + x_2 - 1$$

$$y \in \{0, 1\}$$



常用线性化方法

- 乘积式

x_1x_2 , 其中 $x_1 \in \{0, 1\}$, $x_2 \in [0, u]$

线性化方法: $y = x_1x_2$

x_1	x_2	y
0	$[0, u]$	0
1	$[0, u]$	x_2

$$y \leq ux_1$$

$$y \leq x_2$$

$$y \geq x_2 - u(1 - x_1)$$

$$y \in [0, u]$$



常用线性化方法

- 乘积式

$x_1 x_2$, 其中 $x_1 \in \{0, 1\}$, $x_2 \in [l, u]$

线性化方法: $y = x_1 x_2$

x_1	x_2	y
0	$[l, u]$	0
1	$[l, u]$	x_2

$$\begin{aligned}y &\leq x_2 \\y &\geq x_2 - u(1 - x_1) \\lx_1 &\leq y \leq ux_1\end{aligned}$$



常用线性化方法

- Partial integer variable

$$z \in [0, a] \text{ integer} \quad \text{or} \quad z \in [a, b] \text{ continuous}$$

线性化方法： $x \in [0, a] \text{ integer}$

$y \in [a, b] \text{ continuous}$

$u \in \{0, 1\}$

$$z \leq x + (1 - u)M$$

$$z \geq x - (1 - u)M$$

$$z \leq y + uM$$

$$z \geq y - uM$$



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



Column Generation & Benders Decomposition

基于Gurobi 的列生成和 Benders 分解方法

刃之砺信息科技有限公司（上海）有限公司



课程大纲

(1) 列生成(Column Generation)

首先介绍一些列生成涉及到的基础知识, 然后给出列生成算法的框架, 最后通过Cutting Stock Problem 给出具体的求解步骤。

(2) Benders 分解(Benders Decomposition)

首先会描述一下Benders 分解算法的具体思路, 并且给出经典Benders 分解算法的伪代码。最后通过数值案例详细展示Benders 分解算法的迭代过程。



列生成(Column Generation)

- Reduced Cost

$$\begin{aligned} \min \quad & c^T x \\ \text{st.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

令 $x = [x_B, x_N]$, 其中 x_B 表示基变量, x_N 表示非基变量。相应的 $A = [B, N]$, $c^T = [c_B^T, c_N^T]$

$$Ax = b \Leftrightarrow Bx_B + Nx_N = b \Leftrightarrow x_B = B^{-1}b - B^{-1}Nx_N \quad (\text{非基变量 } x_N = 0, \quad x_B = B^{-1}b)$$

$$\min c^T x \Leftrightarrow \min c_B^T x_B + c_N^T x_N \Leftrightarrow \min c_B^T (B^{-1}b - B^{-1}Nx_N) + c_N^T x_N \Leftrightarrow \min c_B^T B^{-1}b + (c_N^T - c_B^T B^{-1}N)x_N$$

称 $c_N^T - c_B^T B^{-1}N$ 为 **Reduced Cost**

(1) $\exists n \in N$, 使得 $c_n^T - c_B^T B^{-1}n < 0$, 因此可以适当增加非基变量 x_n 的值, 从而降低目标值。

(2) $\forall n \in N$, 都有 $c_n^T - c_B^T B^{-1}n \geq 0$, 获得最优解。



列生成(Column Generation)

- Reduced Cost and Dual Variables

$$\begin{array}{ll} \min & c^T x \\ \text{st.} & Ax = b \\ & x \geq 0 \end{array} \quad \xleftrightarrow{\text{Dual}} \quad \begin{array}{ll} \max & y^T b \\ \text{st.} & y^T A \leq c^T \end{array}$$

假设找到了原问题的最优解 $[x_B, 0] \Leftrightarrow \forall N, c_N^T - c_B^T B^{-1} N \geq 0$, 即 $c_B^T B^{-1} N \leq c_N^T$

$$c_B^T B^{-1} A = c_B^T B^{-1} [B, N] = [c_B^T, c_B^T B^{-1} N] \leq [c_B^T, c_N^T] = c^T \Leftrightarrow c_B^T B^{-1} A \leq c^T \Leftrightarrow c_B^T B^{-1} \text{ 是对偶问题可行解}$$

令 $y^T = c_B^T B^{-1}$, $y^T b = c_B^T B^{-1} b = c_B^T x_B = c^T x$, 根据对偶定理, y^T 为对偶问题最优解。

$$\text{Reduced Cost} = c_N^T - \text{Dual Variables} * N$$



列生成(Column Generation)

- Column Generation 思路

$$\begin{aligned} \min \quad & c_1x_1 + c_2x_2 + \dots \\ \text{s.t.} \quad & a_{11}x_1 + a_{12}x_2 + \dots \leq b_1 \\ & a_{21}x_1 + a_{22}x_2 + \dots \leq b_2 \\ & \vdots \\ & a_{m1}x_1 + a_{m2}x_2 + \dots \leq b_m \\ & x_1, x_2 \dots \text{Integer} \end{aligned}$$

对于一个整数规划模型, 如果问题变量过多(往往很难显性枚举出所有变量), 可以先枚举部分变量构造一个规模相对较小的模型, 然后通过迭代不停地向较小模型里面添加新的变量(列), 直到没有新的变量(列)添加为止。



列生成(Column Generation)

- 案例 Cutting Stock Problem

假设需要长度为3, 7, 9, 16米的钢管各25, 30, 14, 8根, 目前只有长度为20米的钢管若干, 请安排合理地切割方案, 使得消耗的钢管数量最少。

可以很容易给出一种切割方案: 每根钢管只切割一种长度,

3m的钢管 (需求数量25) 需要20米的钢管5根;

7m的钢管 (需求数量30) 需要20米的钢管15根;

9m的钢管 (需求数量14) 需要20米的钢管7根;

16m的钢管 (需求数量8) 需要20米的钢管8根。

上面的切割方案虽然能够满足需求, 但浪费的钢管较多。例如20米的钢管只切7m长度会剩下6m, 余料完全可以用来切3m长度。



列生成(Column Generation)

- 第一种建模方案(常规思路)

参数: I 所需钢管种类集合;
 K 目前未切割的钢管集合;
 D_i 第 i 种长度钢管的需求数量;
 l_i 第 i 种长度钢管的长度;
 L_k 第 k 根未切割的钢管的长度。

变量: x_{ik} 表示第 k 根钢管切割第 i 种长度的数量;
 y_k 表示第 k 根钢管是否使用。

$$\min \sum_k y_k$$

满足需求

$$\sum_k x_{ik} \geq D_i, \forall i \in I$$

钢管切割的长度不能超过本身长度

$$\sum_i l_i x_{ik} \leq L_k y_k, \forall k \in K$$

$$x_{ik} \in N, y_k \in \{0,1\}, \forall i \in I, k \in K$$



列生成(Column Generation)

- 第二种建模方案(列生成思路)

参数: C_{ip} 表示在第 p 种切割模式下, 切出第 i 种长度的钢管数量;
 P 表示切割模式集合。
变量: z_p 表示第 p 种切割模式使用的次数。

$$\begin{aligned} & \min \sum_p z_p \\ & \text{满足需求} \quad \sum_p C_{ip} z_p \geq D_i, \forall i \in I \\ & \quad z_p \text{ Integer}, \forall p \in P \end{aligned}$$

切割模式组合非常多, 因此枚举所有 z_p 几乎是不可行的, 也没有必要。因为并不是所有的切割模式都会用到, 例如, 每根钢管只切割一种长度浪费较多。那么如何去寻找好的切割模式?



列生成(Column Generation)

- 列生成算法求解 Cutting Stock Problem 步骤

首先, 给定几种初始的切割模式 $\bar{P} \in P$ 带入原列生成模型中, 并将变量松弛为连续型:

$$\begin{aligned} \min \quad & \sum_p z_p \\ \sum_p C_{ip} z_p & \geq D_i, \forall i \in I \\ z_p & \geq 0, \forall p \in \bar{P} \end{aligned}$$

求解上述模型获得对应的对偶变量取值, 记为 λ_i 。



列生成(Column Generation)

- 列生成算法求解 Cutting Stock Problem 步骤

然后, 思考是否存在新的切割模式 p_{new} 加到模型进而降低目标函数值? 假设存在, 必然会有变量 $z_{p_{new}}$ 的

Reduced Cost = $1 - \sum_i \lambda_i C_{ip_{new}} < 0$ 。

$$\begin{aligned} \min \quad & \left(\sum_p z_p \right) + z_{p_{new}} \\ \left(\sum_p C_{ip} z_p \right) + C_{ip_{new}} z_{p_{new}} & \geq D_i, \forall i \in I \\ z_p & \geq 0, \forall p \in \bar{P} \cup p_{new} \end{aligned}$$

为了获取更优的目标值, 往往会选择Reduced Cost 最小的切割模式加入模型中。那么如何确定Reduced Cost 最小的切割模式?



列生成(Column Generation)

- 列生成算法求解 Cutting Stock Problem 步骤

接着,通过一个子问题确定Reduced Cost 最小的切割模式。对于任意的切割模式,都需要满足钢管长度限制。

因此可以构造如下模型(省略下标 p_{new}),其中变量 C_i 表示该切割模式切出第 i 种长度的钢管数量;

$$\begin{aligned} \min \quad & 1 - \sum_i \lambda_i C_i \\ \sum_i C_i l_i & \leq L, \forall i \in I \\ C_i & \text{ Integer}, \forall i \in I \end{aligned}$$

如果子问题最优目标值小于0,那么意味着发现更好的切割模式,将新切割模式加入到模型中重复以上步骤直到没有更好的切割模式出现。



列生成(Column Generation)

- 列生成算法求解 Cutting Stock Problem 步骤

初始给出四种切割模式:

模式	第一种	第二种	第三种	第四种
3m	6	0	0	0
7m	0	2	0	0
9m	0	0	2	0
16m	0	0	0	1

$$\min z_1 + z_2 + z_3 + z_4$$

$$6z_1 + 0z_2 + 0z_3 + 0z_4 \geq 25$$

$$0z_1 + 2z_2 + 0z_3 + 0z_4 \geq 30$$

$$0z_1 + 0z_2 + 2z_3 + 0z_4 \geq 14$$

$$0z_1 + 0z_2 + 0z_3 + 1z_4 \geq 8$$

$$z_1, z_2, z_3, z_4 \geq 0$$

求解模型, 获得对应的对偶变量值 $\lambda = [0.16666666666666666, 0.5, 0.5, 1.0]$



列生成(Column Generation)

- 列生成算法求解 Cutting Stock Problem 步骤

通过子问题确定Reduced Cost最小的切割模式:

$$\begin{aligned} \min \quad & 1 - 0.16666666666666666c_1 - 0.5c_2 - 0.5c_3 - c_4 \\ & 3c_1 + 7c_2 + 9c_3 + 16c_4 \leq 20 \\ & c_1, c_2, c_3, c_4 \text{ Integer} \end{aligned}$$

Reduced Cost最小的值为 $-0.3333333333333333 < 0$, 因此发现更好的切割模式:

$$(c_1, c_2, c_3, c_4) = (2, 2, 0, 0)$$

将新的切割模式加入到模型中, 继续迭代。



列生成(Column Generation)

- 列生成算法求解 Cutting Stock Problem 步骤

$$\begin{aligned} \min \quad & z_1 + z_2 + z_3 + z_4 + \mathbf{z_{new}} \\ 6z_1 + 0z_2 + 0z_3 + 0z_4 + \mathbf{2z_{new}} & \geq 25 \\ 0z_1 + 2z_2 + 0z_3 + 0z_4 + \mathbf{2z_{new}} & \geq 30 \\ 0z_1 + 0z_2 + 2z_3 + 0z_4 + \mathbf{0z_{new}} & \geq 14 \\ 0z_1 + 0z_2 + 0z_3 + 1z_4 + \mathbf{0z_{new}} & \geq 8 \\ z_1, z_2, z_3, z_4, \mathbf{z_{new}} & \geq 0 \end{aligned}$$

求解模型，获得对应的对偶变量值 $\lambda = [0, 0.5, 0.5, 1.0]$



列生成(Column Generation)

- 列生成算法求解 Cutting Stock Problem 步骤

继续通过子问题确定 Reduced Cost 最小的切割模式:

$$\begin{aligned} \min \quad & 1 - 0c_1 - 0.5c_2 - 0.5c_3 - c_4 \\ & 3c_1 + 7c_2 + 9c_3 + 16c_4 \leq 20 \\ & c_1, c_2, c_3, c_4 \text{ Integer} \end{aligned}$$

Reduced Cost 最小的值为 0, 因此没能发现更好的切割模式, 终止迭代。列生成完后的模型:

$$\begin{aligned} \min \quad & z_1 + z_2 + z_3 + z_4 + z_{new} \\ & 6z_1 + 0z_2 + 0z_3 + 0z_4 + 2z_{new} \geq 25 \\ & 0z_1 + 2z_2 + 0z_3 + 0z_4 + 2z_{new} \geq 30 \\ & 0z_1 + 0z_2 + 2z_3 + 0z_4 + 0z_{new} \geq 14 \\ & 0z_1 + 0z_2 + 0z_3 + 1z_4 + 0z_{new} \geq 8 \\ & z_1, z_2, z_3, z_4, z_{new} \geq 0 \end{aligned}$$



列生成(Column Generation)

- 完整案例代码

```
from gurobipy import *
TypesDemand = [3, 7, 9, 16]           # 需求长度
QuantityDemand = [25, 30, 14, 8]      # 需求的量
LengthUsable = 20                     # 钢管长度

try:
    MainProbRelax = Model()            # 松弛后的列生成主问题
    SubProb = Model()                 # 子问题

    # 构造主问题模型，选择的初始切割方案 每根钢管只切一种长度
    # 添加变量
    Zp = MainProbRelax.addVars(len(TypesDemand), obj=1.0, vtype=GRB.CONTINUOUS, name = 'z')
    # 添加约束
    ColumnIndex = MainProbRelax.addConstrs(quicksum(Zp[p] * (LengthUsable//TypesDemand[i]) \
    for p in range(len(TypesDemand)) if p==i) >= QuantityDemand[i] for i in range(len(TypesDemand)))
    MainProbRelax.optimize() # 求解
```




列生成(Column Generation)

- 完整案例代码

构造子问题模型

获得对偶值

```
Dualsolution = MainProbRelax.getAttr(GRB.Attr.Pi, MainProbRelax.getConstrs())
```

添加变量

```
Ci = SubProb.addVars(len(TypesDemand), obj=Dualsolution, vtype=GRB.INTEGER, name = 'c')
```

添加约束

```
SubProb.addConstr(quicksum(Ci[i] * TypesDemand[i] for i in range(len(TypesDemand)))) <= LengthUsable)
```

```
SubProb.setAttr(GRB.Attr.ModelSense, -1)      # 设定优化方向
```

```
SubProb.optimize()                             # 求解
```



列生成(Column Generation)

- 完整案例代码

```
# 判断Reduced Cost是否小于零
```

```
while SubProb.objval > 1:
```

```
    # 获取变量取值
```

```
    columnCoeff = SubProb.getAttr("X", SubProb.getVars())
```

```
    column = Column(columnCoeff, MainProbRelax.getConstrs())
```

```
    # 添加变量
```

```
    MainProbRelax.addVar(obj=1.0, vtype=GRB.CONTINUOUS, name="CG", column=column)
```

```
    MainProbRelax.optimize()
```

```
        # 求解
```

```
    # 修改目标函数系数
```

```
    for i in range(len(TypesDemand)):
```

```
        Ci[i].obj = ColumnIndex[i].pi
```

```
    SubProb.optimize()
```



列生成(Column Generation)

- 完整案例代码

```
# 将CG后的模型转为整数，并求解
```

```
for v in MainProbRelax.getVars():
```

```
    v.setAttr("VType", GRB.INTEGER)
```

```
MainProbRelax.optimize()
```

```
for v in MainProbRelax.getVars():
```

```
    if v.X != 0.0:
```

```
        print('%s %g' % (v.VarName, v.X))
```

```
except GurobiError as e:
```

```
    print('Error code ' + str(e.errno) + ": " + str(e))
```

```
except AttributeError:
```

```
    print('Encountered an attribute error')
```



Benders 分解(Benders Decomposition)

- Classic Benders Decomposition 思路

$$\begin{aligned} \min \quad & c^T x + f^T \mathbf{y} \\ \text{s. t.} \quad & Ax + B\mathbf{y} = b \\ & x \geq 0, \mathbf{y} \in Y \end{aligned} \quad (1)$$

假设变量 \mathbf{y} 是一个比较复杂的变量, 如果该变量的值可以提前确定, 剩下的模型往往会比较容易求解。因此通过变量的区分, 可以将原模型拆解为两个相对较小的模型, 然后通过间接迭代求解小模型到达求解原模型的目的。

$$\begin{aligned} \min \quad & f^T \mathbf{y} + g(\mathbf{y}) \\ & \mathbf{y} \in Y \end{aligned} \quad (2)$$

$$\begin{aligned} g(\bar{\mathbf{y}}) = \min \quad & c^T x \\ \text{s. t.} \quad & Ax = b - B\bar{\mathbf{y}} \\ & x \geq 0 \end{aligned} \quad (3)$$

如果模型(3)无界 \Rightarrow 模型(2)无界 \Rightarrow 原模型(1)无界。



Benders 分解(Benders Decomposition)

- Classic Benders Decomposition 思路

如果模型 (3) 有界, 写出模型 (3) 的对偶模型:

$$\begin{aligned} \max \quad & \lambda^T (b - B\bar{y}) \\ & A^T \lambda \leq c \\ & \lambda \text{ free} \end{aligned} \quad (4)$$

可以看到**模型 (4) 的可行域与变量 y 没有关系**, 变量 y 的值只会影响其目标函数值。如果模型 (4) 的可行域为空集, 那么根据对偶性可知模型 (3) 无界或者为空。



Benders 分解(Benders Decomposition)

- Classic Benders Decomposition 思路

假设模型(4)非空,可以枚举出所有的extreme points $(\alpha_p^1, \alpha_p^2, \alpha_p^3, \dots, \alpha_p^I)$ 和 extreme rays $(\alpha_r^1, \alpha_r^2, \alpha_r^3, \dots, \alpha_r^J)$ 。因此求解模型(4)可以等价为:

- 最优: 寻找 α_p^i 最大化 $\alpha_p^{iT}(b - B\bar{y})$ 。

- 排除不可行: 检测 $\alpha_r^{jT}(b - B\bar{y}) \leq 0$ 是否成立。如果 $\alpha_r^{jT}(b - B\bar{y}) > 0$, 模型(4)无界,进而模型(3)不可行。

因此模型(4)可以等价转化为:

$$\begin{array}{ll} \max_{i \in \{1, 2, \dots, I\}} \alpha_p^{iT}(b - B\bar{y}) & \\ \alpha_r^{jT}(b - B\bar{y}) \leq 0, \quad \forall j \in \{1, 2, \dots, J\} & \end{array} \quad \Longrightarrow \quad \begin{array}{ll} \min z & \\ \alpha_p^{iT}(b - B\bar{y}) \leq z, \quad \forall i \in \{1, 2, \dots, I\} & (5) \\ \alpha_r^{jT}(b - B\bar{y}) \leq 0, \quad \forall j \in \{1, 2, \dots, J\} & \\ z \text{ free} & \end{array}$$



Benders 分解(Benders Decomposition)

- Classic Benders Decomposition 思路

进一步将模型(5)带入到模型(2)中,可以得到原模型的一个等价模型(6):

$$\begin{aligned} \min \quad & f^T y + z \\ \alpha_p^{i^T} (b - By) & \leq z, \quad \forall i \in \{1, 2, \dots, I\} \\ \alpha_r^{j^T} (b - By) & \leq 0, \quad \forall j \in \{1, 2, \dots, J\} \\ y & \in Y, z \text{ free} \end{aligned} \quad (6)$$

模型(6)包含变量 y 和 z , 以及大量的约束, 这些约束在实际求解释很难全部枚举, 因此很难直接求解模型(6)。



Benders 分解(Benders Decomposition)

- Classic Benders Decomposition 思路

因此Benders 分解方法不直接求解模型(6), 具体迭代思路如下:

首先, 找到一组候选解 (y^*, z^*) , 将 y^* 带入到模型(3)中, 获得如下模型 (被称为 Benders subproblem, SP)

$$g(y^*) = \min c^T x$$

$$Ax = b - By^* \quad (\text{SP})$$

$$x \geq 0$$

求解SP模型可能有三种结果:

- 最优解且 $g(y^*) = z^*$, 发现原问题最优解, 停止迭代。
- 最优解且 $g(y^*) > z^*$, 可以构造一个形如 $\alpha_p^T (b - By) \leq z$ (benders optimality cut) 加入到 Benders master problem, MP。
- 不可行, 可以构造一个形如 $\alpha_r^T (b - By) \leq 0$ (benders feasibility cut) 加入到 Benders master problem, MP。



Benders 分解(Benders Decomposition)

- Classic Benders Decomposition 思路

进而得到一个随着迭代规模不断增加的Benders master problem Model, MP。

$$\min f^T y + z$$

$$\alpha_p^T (b - By) \leq z \text{ (benders optimality cut)} \quad (\text{MP})$$

$$\alpha_r^T (b - By) \leq 0 \text{ (benders feasibility cut)}$$

$$y \in Y, z \text{ free}$$

接着求解MP, 获得一组新的 (y^*, z^*) , 然后重复上述迭代过程直到 $g(y^*) = z^*$ 。



Benders 分解(Benders Decomposition)

- Classic Benders Decomposition 算法结构

Classic Benders Decomposition

Identify a MP and an easy $SP(y)$

repeat

solve MP obtaining the solution (y^, z^*)*

solve $SP(y^)$*

if $SP(y^)$ is feasible*

if $obj(SP(y^)) = z^*$*

STOP

else

add to MP the benders optimality cut

else

add to MP the benders feasibility cut.

until (end condition)



Benders 分解(Benders Decomposition)

- Benders 分解数值案例

$$\min x_1 + x_2 + x_3 + x_4 + x_5 + 7y_1 + 7y_2 + 7y_3 + 7y_4 + 7y_5$$

$$x_1 + x_4 + x_5 = 8$$

$$x_2 + x_5 = 3$$

$$x_3 + x_4 = 5$$

$$x_1 \leq 8y_1$$

$$x_2 \leq 3y_2$$

$$x_3 \leq 5y_3$$

$$x_4 \leq 5y_4$$

$$x_5 \leq 3y_5$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0, \quad y_1, y_2, y_3, y_4, y_5 \in \{0, 1\}$$



Benders 分解(Benders Decomposition)

- Benders 分解数值案例具体步骤

首先, 确定合适的MP和SP, 针对本数值案例, 变量 x, y 分别为连续变量和 $\{0, 1\}$ 变量, 所以变量 y 相对复杂, 因此构建MP模型如下:

$$\begin{aligned} \min \quad & 7y_1 + 7y_2 + 7y_3 + 7y_4 + 7y_5 + z \\ & z \geq 0, \quad y_1, y_2, y_3, y_4, y_5 \in \{0, 1\} \end{aligned}$$

求解MP模型, 获得最优解 $y^* = (0, 0, 0, 0, 0)$, $z^* = 0$



Benders 分解(Benders Decomposition)

- Benders 分解数值案例具体步骤

将第一组候选解 (y^*, z^*) , 其中 $y^* = (0,0,0,0,0)$, $z^* = 0$ 带入SP模型:

$$\min x_1 + x_2 + x_3 + x_4 + x_5 + 7 * 0 + 7 * 0 + 7 * 0 + 7 * 0 + 7 * 0$$

$$x_1 + x_4 + x_5 = 8$$

$$x_2 + x_5 = 3$$

$$x_3 + x_4 = 5$$

$$x_1 \leq 8 * 0, x_2 \leq 3 * 0, x_3 \leq 5 * 0, x_4 \leq 5 * 0, x_5 \leq 3 * 0$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

显然SP模型不可行, Dual-SP无界, 可以获得Dual-SP的一个extreme ray $(1.0, 0.0, 0.0, -1.0, 0.0, 0.0, -1.0, -1.0)$

可以向MP模型中添加 *benders feasibility cut*

$$8y_1 + 5y_4 + 3y_5 \geq 8$$



Benders 分解(Benders Decomposition)

- Benders 分解数值案例具体步骤

更新MP模型:

$$\min 7y_1 + 7y_2 + 7y_3 + 7y_4 + 7y_5 + z$$

$$8y_1 + 5y_4 + 3y_5 \geq 8$$

$$z \geq 0, y_1, y_2, y_3, y_4, y_5 \in \{0,1\}$$

求解该模型获得最优解为 $y^* = (1,0,0,0,0)$, $z^* = 0$



Benders 分解(Benders Decomposition)

- Benders 分解数值案例具体步骤

将第二组候选解 (y^*, z^*) , 其中 $y^* = (1, 0, 0, 0, 0)$, $z^* = 0$ 带入SP模型:

$$\min x_1 + x_2 + x_3 + x_4 + x_5 + 7 * 1 + 7 * 0 + 7 * 0 + 7 * 0 + 7 * 0$$

$$x_1 + x_4 + x_5 = 8$$

$$x_2 + x_5 = 3$$

$$x_3 + x_4 = 5$$

$$x_1 \leq 8 * 1, x_2 \leq 3 * 0, x_3 \leq 5 * 0, x_4 \leq 5 * 0, x_5 \leq 3 * 0$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

显然SP模型仍然不可行, Dual-SP的一个extreme ray $(0.0, 1.0, 1.0, 0.0, -1.0, -1.0, -1.0, -1.0)$ 。向MP模型再次添加 *benders feasibility cut*

$$3y_2 + 5y_3 + 5y_4 + 3y_5 \geq 8$$



Benders 分解(Benders Decomposition)

- Benders 分解数值案例具体步骤

继续更新MP模型:

$$\min 7y_1 + 7y_2 + 7y_3 + 7y_4 + 7y_5 + z$$

$$8y_1 + 5y_4 + 3y_5 \geq 8$$

$$3y_2 + 5y_3 + 5y_4 + 3y_5 \geq 8$$

$$z \geq 0, y_1, y_2, y_3, y_4, y_5 \in \{0,1\}$$

求解该模型获得最优解为 $y^* = (0,0,0,1,1)$, $z^* = 0$



Benders 分解(Benders Decomposition)

- Benders 分解数值案例具体步骤

将第三组候选解 (y^*, z^*) , 其中 $y^* = (0,0,0,1,1)$, $z^* = 0$ 带入SP模型:

$$\min x_1 + x_2 + x_3 + x_4 + x_5 + 7 * 0 + 7 * 0 + 7 * 0 + 7 * 1 + 7 * 1$$

$$x_1 + x_4 + x_5 = 8$$

$$x_2 + x_5 = 3$$

$$x_3 + x_4 = 5$$

$$x_1 \leq 8 * 0, x_2 \leq 3 * 0, x_3 \leq 5 * 0, x_4 \leq 5 * 1, x_5 \leq 3 * 1$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

SP模型有最优解 $obj = 22 > 0$, 因此可以向MP模型添加 *benders optimality cut*

$$5y_4 + 3y_5 + z \geq 16$$



Benders 分解(Benders Decomposition)

- Benders 分解数值案例具体步骤

继续更新MP模型:

$$\min 7y_1 + 7y_2 + 7y_3 + 7y_4 + 7y_5 + z$$

$$8y_1 + 5y_4 + 3y_5 \geq 8$$

$$3y_2 + 5y_3 + 5y_4 + 3y_5 \geq 8$$

$$5y_4 + 3y_5 + z \geq 16$$

$$z \geq 0, y_1, y_2, y_3, y_4, y_5 \in \{0,1\}$$

求解该模型获得最优解为 $y^* = (0,0,0,1,1)$, $z^* = 8$, 因为 y^* 与上一步值相同, 因此带入SP模型最优解 $obj = 22$ 。

发现了原问题的最优解, 停止迭代过程。

原问题最优解 $obj = 22$, $x = (0,0,0,5,3)$, $y = (0,0,0,1,1)$



Benders 分解(Benders Decomposition)

- 案例代码

```
from gurobipy import *  
def addBendersCuts(SP_Dual_obj, x):  
    if SP_Dual.status == GRB.Status.UNBOUNDED:          #模型无界  
        ray = SP_Dual.UnbdRay  
        MP.addConstr(8*ray[0] + 3*ray[1] + 5*ray[2] + 8*ray[3]*y[0] + 3*ray[4]*y[1] + 5*ray[5]*y[2] + 5*ray[6]*y[3] + 3*ray[7]*y[4]<= 0)  
    elif SP_Dual.status == GRB.Status.OPTIMAL:          #发现最优解  
        MP.addConstr(8*Vdual1[0].x + 3*Vdual1[1].x + 5*Vdual1[2].x + 8*Vdual2[0].x*y[0] + \  
                    3*Vdual2[1].x*y[1] + 5*Vdual2[2].x*y[2] + 5*Vdual2[3].x*y[3] + 3*Vdual2[4].x*y[4] <= z)  
        SP_Dual_obj[0] = SP_Dual.ObjVal                #获取最优解  
        x.append(x1.pi)                                #获取SP模型解  
        x.append(x2.pi)  
        x.append(x3.pi)  
        x.append(x4.pi)  
        x.append(x5.pi)  
    else:                                                #其他状态  
        print (SP_Dual.status)
```



Benders 分解(Benders Decomposition)

- 案例代码

```
try:
    MP = Model()                                #Benders Master Problem
    SP_Dual = Model()                            #dual of Benders SubProblem
    y= MP.addVars(5, obj=7, vtype=GRB.BINARY, name='y')
    z= MP.addVar(obj=1, vtype=GRB.CONTINUOUS, name='z')
    Vdual1 = SP_Dual.addVars(3, lb=-GRB.INFINITY, vtype=GRB.CONTINUOUS, name='Vdual1')
    Vdual2 = SP_Dual.addVars(5, lb=-GRB.INFINITY,ub=0, vtype=GRB.CONTINUOUS, name='Vdual2')
    x1 = SP_Dual.addConstr(Vdual1[0] + Vdual2[0] <=1)
    x2 = SP_Dual.addConstr(Vdual1[1] + Vdual2[1] <=1)
    x3 = SP_Dual.addConstr(Vdual1[2] + Vdual2[2] <=1)
    x4 = SP_Dual.addConstr(Vdual1[0] + Vdual1[2] + Vdual2[3] <=1)
    x5 = SP_Dual.addConstr(Vdual1[0] + Vdual1[1] + Vdual2[4] <=1)
    SP_Dual.Params.InfUnbdInfo = 1                #设置参数 InfUnbdInfo
    iteration = 0
    SP_Dual_obj = [9999]
    x = []
    MP.optimize()
```



Benders 分解(Benders Decomposition)

- 案例代码

```
while z.x < SP_Dual_obj[0]:                                #迭代主循环
    if iteration == 0:
        SP_Dual.setObjective(8*Vdual1[0] + 3*Vdual1[1] + 5*Vdual1[2] + \
                               8*Vdual2[0]*y[0].x + 3*Vdual2[1]*y[1].x + 5*Vdual2[2]*y[2].x + 5*Vdual2[3]*y[3].x + 3*Vdual2[4]*y[4].x, GRB.MAXIMIZE)
        SP_Dual.optimize()
        addBendersCuts(SP_Dual_obj, x)                    #add Benders Cuts
        iteration = 1
    else:
        Vdual2[0].obj = 8*y[0].x                          #更新dual -SP
        Vdual2[1].obj = 3*y[1].x
        Vdual2[2].obj = 5*y[2].x
        Vdual2[3].obj = 5*y[3].x
        Vdual2[4].obj = 3*y[4].x
        SP_Dual.optimize()
        addBendersCuts(SP_Dual_obj, x)                    #add Benders Cuts
        iteration = iteration + 1
MP.optimize()
```



Benders 分解(Benders Decomposition)

- 案例代码

```
for i in range(5):  
    print('x[%d] = %f'%(i, x[i]))
```

#获取变量x的取值

```
for i in range(5):  
    print('y[%d] = %d'%(i, y[i].x))
```

#获取变量y的取值

```
except GurobiError as e:  
    print('Error code ' + str(e.errno) + ": " + str(e))
```

```
except AttributeError:  
    print('Encountered an attribute error')
```



Gurobi 最优算法和启发式算法的融合

顾宗浩

GUROBI CTO, 联合创始人






深圳, 2018年7月13日

- MIP Algorithm
- Heuristics and (vs.) optimization
- Gurobi heuristics
 - Non-LP based
 - LP based
 - Reformulation
 - Improvement
 - SubMIP and recursive
 - Features helping heuristics
- Gurobi heuristic parameters
- User input for Gurobi heuristics
 - MIP start/Multiple MIP starts
 - MIP hint
 - Partition heuristic
 - Heuristic callback
- What to do with too big/hard models

- MIP算法
- 启发式和（vs.）优化
- Gurobi启发式算法
 - 基于非LP
 - 基于LP
 - 模型改建
 - 改进型
 - 子MIP和递归
 - 有助于启发式的功能
- Gurobi 启发式参数
- Gurobi启发式的用户输入功能
 - MIP 起始值/多个 MIP起始值
 - MIP提示
 - 分区启发式
 - 启发式回调
- 如何处理太大/太难的模型

Gurobi MIP Algorithms

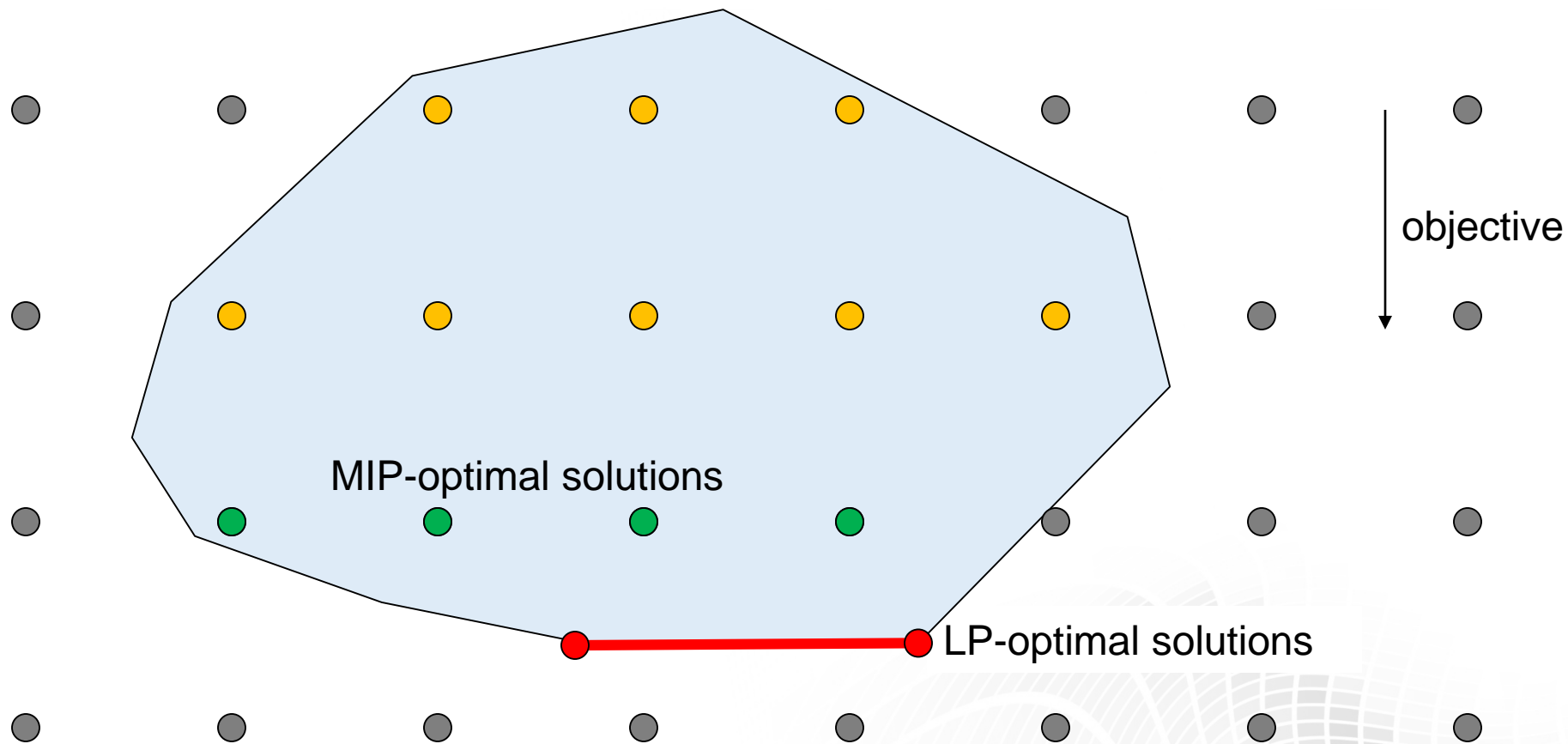
Gurobi 混合整数规划算法

- Presolve  Presolve, PrePasses, AggFill, Aggregate, DualReductions, PreSparsify, ... 预优化
 - Tighten formulation and reduce problem size
- Solve continuous relaxations  Method, NodeMethod 求解连续松弛模型
 - Ignoring integrality
 - Gives a bound on the optimal integral objective
- Cutting planes  Cuts, CutPasses, CutAggPasses, GomoryPasses, CliqueCuts, CoverCuts, FlowCoverCuts, ... 切平面
 - Cut off relaxation solutions
- Branching variable selection  VarBranch 分支变量
 - Crucial for limiting search tree size
- Primal heuristics  Heuristics, MinRelNodes, PumpPasses, RINS, SubMIPNodes, ZeroObjNodes 启发算法
 - Find integer feasible solutions

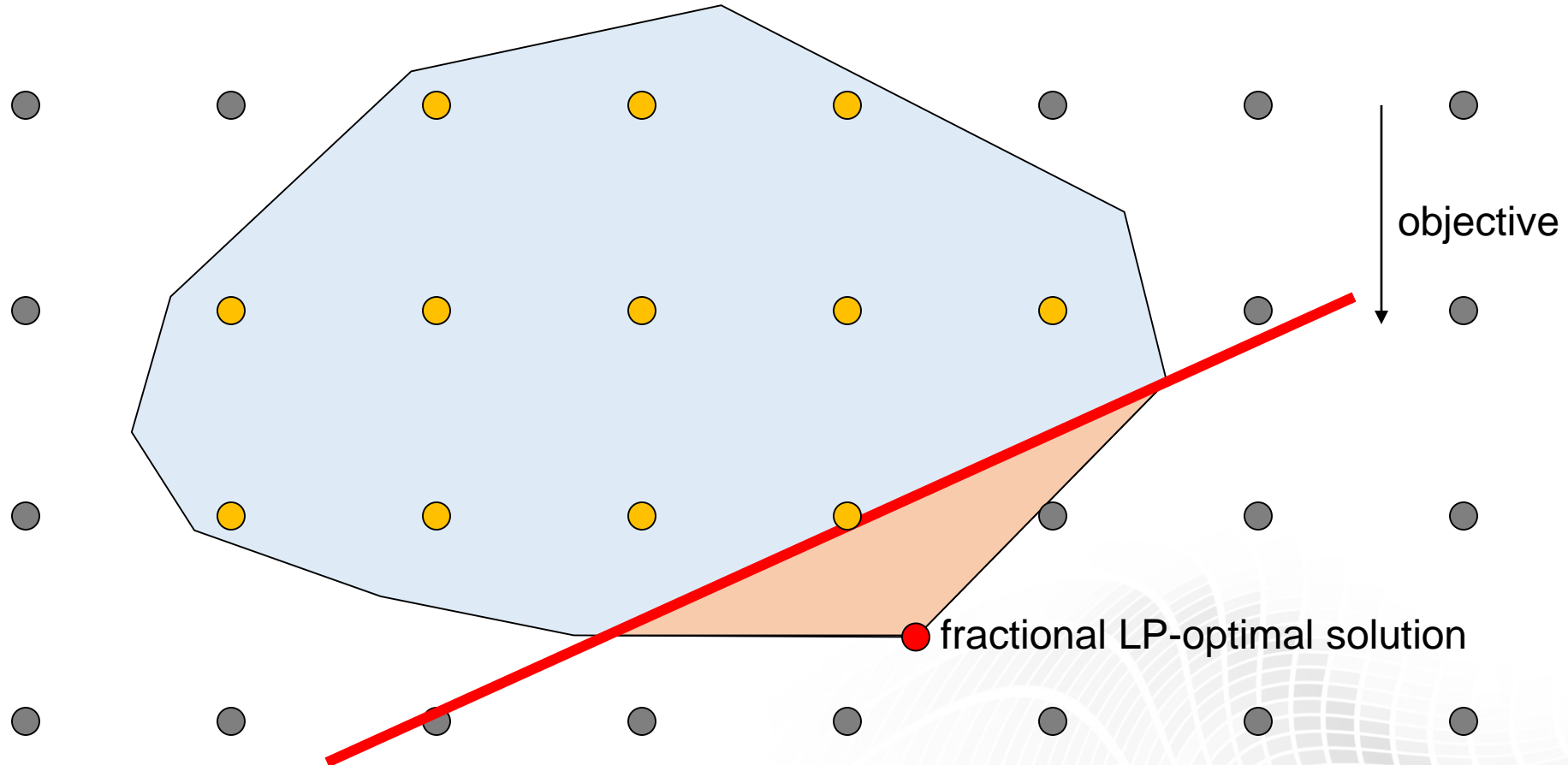
- Goals: 目标
 - Reduce problem size
 - Strengthen LP relaxation
 - Identify problem sub-structures
 - Cliques, implied bounds, networks, disconnected components, ...
- Similar to LP presolve, but more powerful: 与LP 预优化类似, 但更强大
 - Exploit integrality
 - Round fractional bounds and right hand sides
 - Lifting/coefficient strengthening
 - Probing
 - Does not need to preserve duality
 - We only need to be able to uncrush a primal solution
 - Neither a dual solution nor a basis needs to be uncrushed



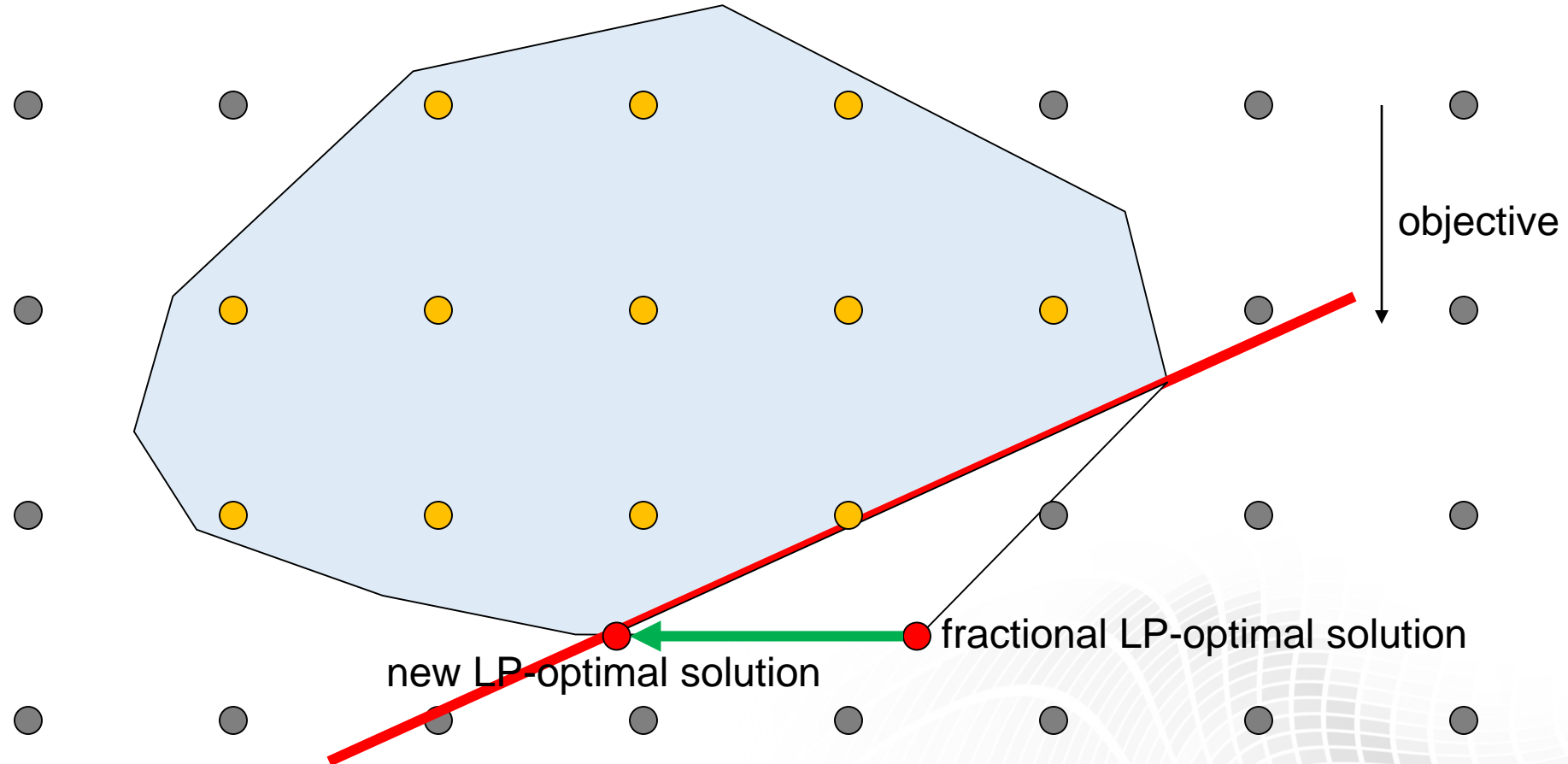
MIP – LP Relaxation LP 松弛问题



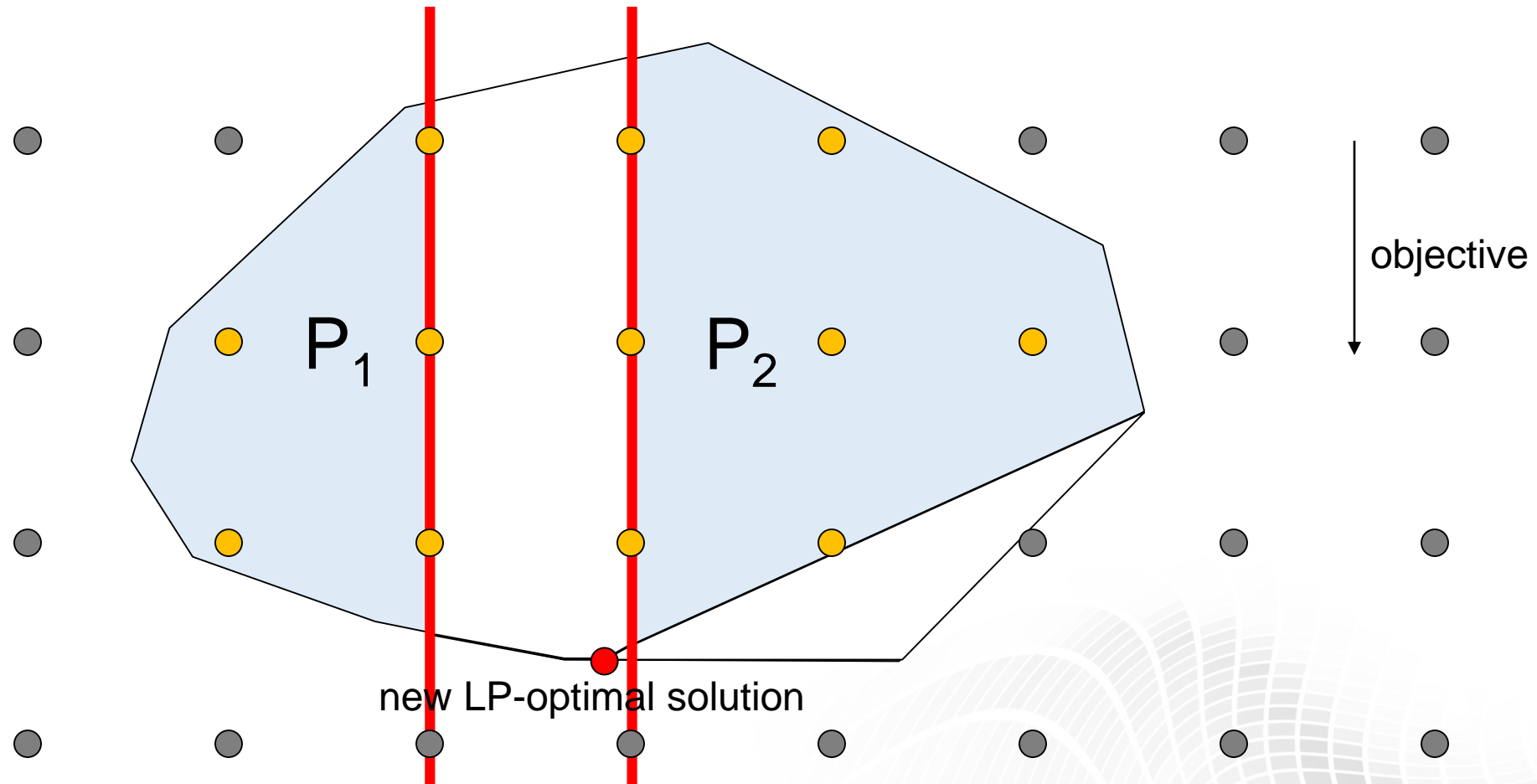
MIP – Cutting Planes 切平面



MIP – Cutting Planes 切平面



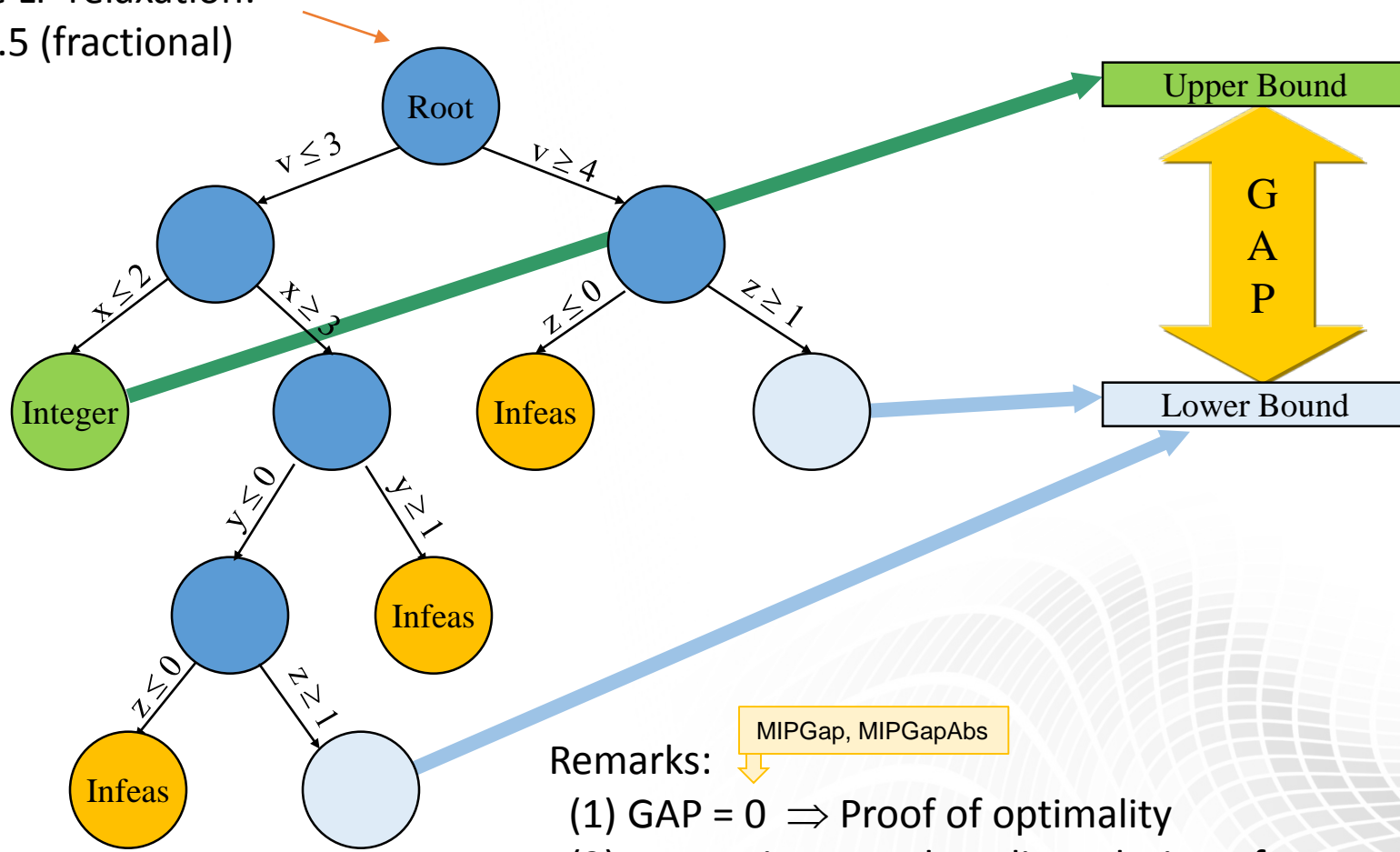
MIP – Branching 分支



LP based Branch-and-Bound 基于LP的分支定界法



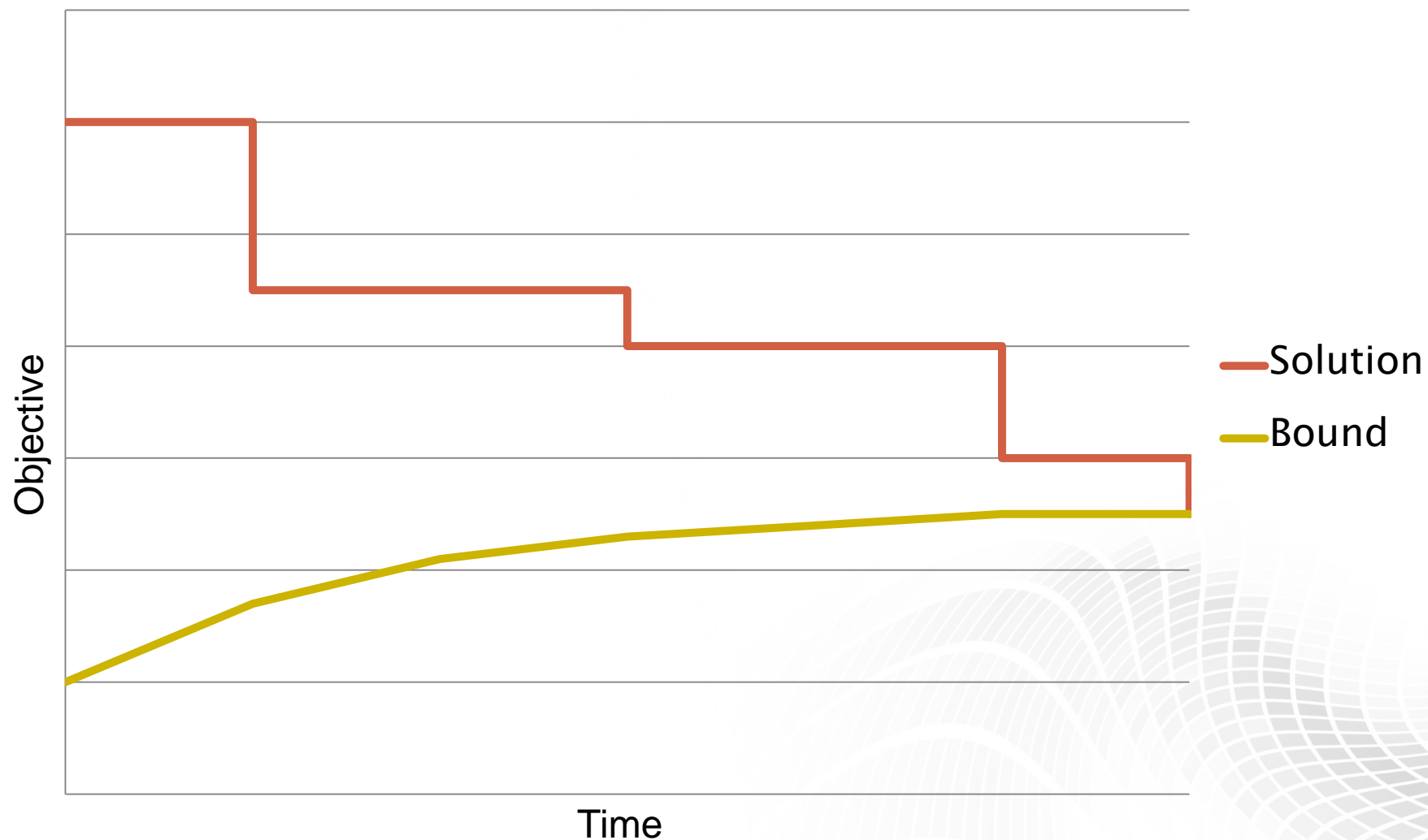
Solve LP relaxation:
 $v=3.5$ (fractional)



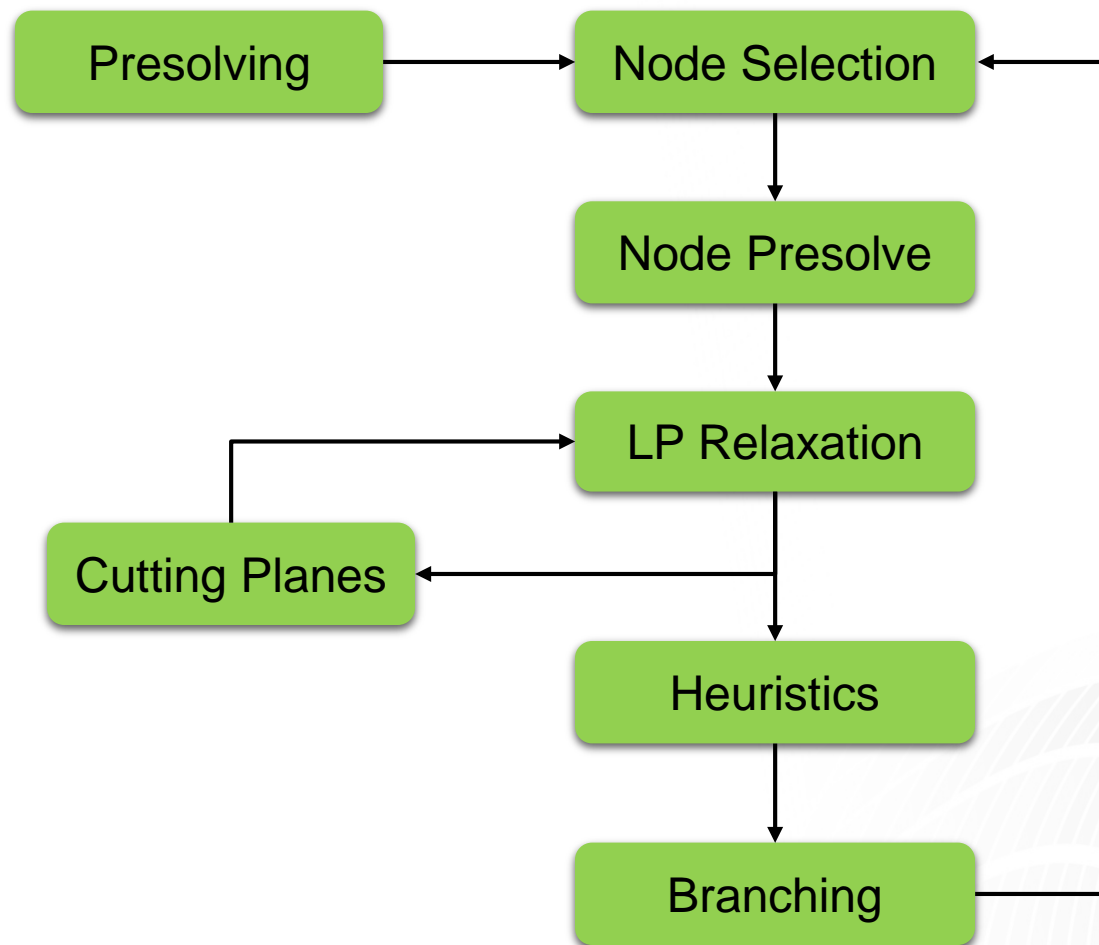
Remarks:

- (1) $\text{GAP} = 0 \Rightarrow$ Proof of optimality
- (2) In practice: good quality solution often enough

Solving a MIP Model 求解混合整数模型



Branch-and-Cut 分支切割



Branch-and-Cut 分支切割

Presolving

```
Gurobi Optimizer version 6.0.0 (linux64)
Copyright (c) 2014, Gurobi Optimization, Inc.

Read MPS format model from file /models/mip/roll3000.mps.bz2
Reading time = 0.03 seconds
roll3000: 2295 rows, 1166 columns, 29386 nonzeros
Optimize a model with 2295 rows, 1166 columns and 29386 nonzeros
Coefficient statistics:
  Matrix range      [2e-01, 3e+02]
  Objective range   [1e+00, 1e+00]
  Bounds range      [1e+00, 1e+09]
  RHS range         [6e-01, 1e+03]
Presolve removed 1308 rows and 311 columns
Presolve time: 0.08s
Presolved: 987 rows, 855 columns, 19346 nonzeros
Variable types: 211 continuous, 644 integer (545 binary)
```

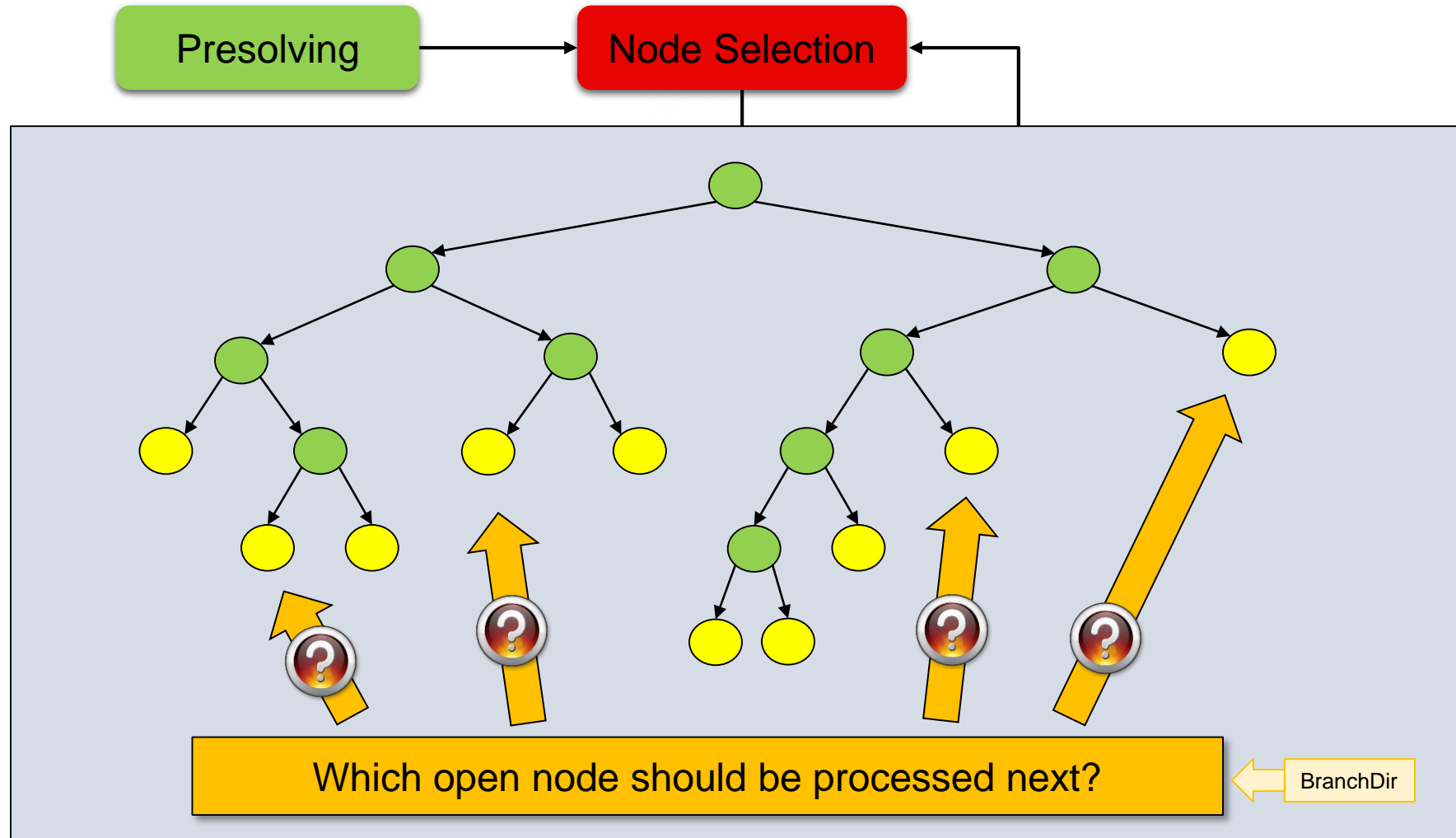
Cutting Planes

```
Root relaxation: objective 1.112003e+04, 1063 iterations, 0.03 seconds
```

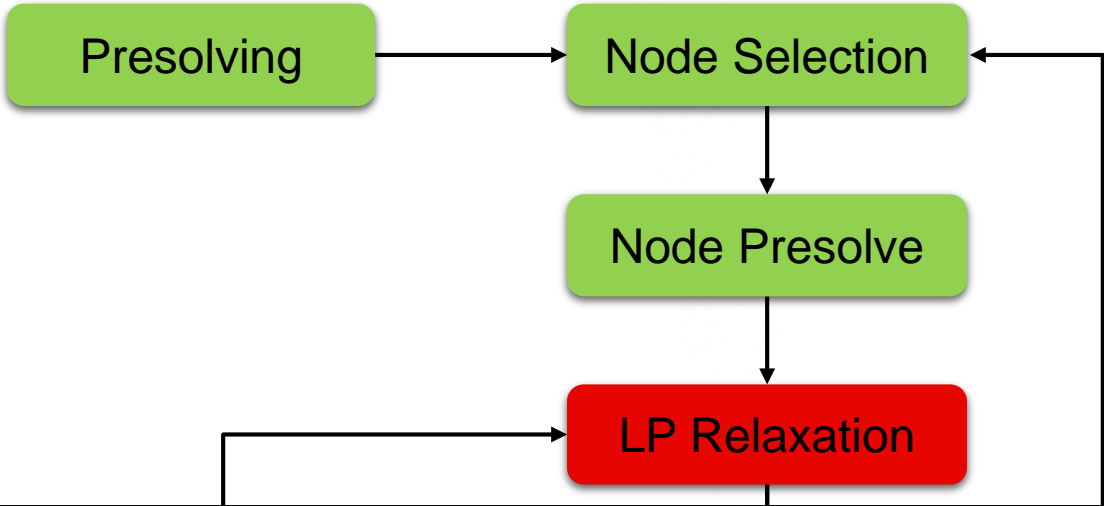
Nodes		Current Node			Objective Bounds			Work	
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
0	0	11120.0279	0	154	-	11120.0279	-	-	0s
0	0	11526.8918	0	207	-	11526.8918	-	-	0s
0	0	11896.9710	0	190	-	11896.9710	-	-	0s

Branching

Branch-and-Cut 分支切割



Branch-and-Cut 分支切割



Presolved: 987 rows, 855 columns, 19346 nonzeros										
Variable types: 211 continuous, 644 integer (545 binary)										
Root relaxation: objective 1.112003e+04, 1063 iterations, 0.03 seconds										
Nodes		Current Node				Objective Bounds			Work	
Expl	Unexpl	Obj	Depth	IntInf		Incumbent	BestBd	Gap	It/Node	Time
0	0	11120.0279	0	154		- 11120.0279		-	-	0s
0	0	11526.8918	0	207		- 11526.8918		-	-	0s
0	0	11896.9710	0	190		- 11896.9710		-	-	0s
...										
H 327	218					13135.000000	12455.2162	5.18%	42.6	1s
H 380	264					13093.000000	12455.2162	4.87%	41.6	1s
H 413	286					13087.000000	12455.2162	4.83%	41.4	1s
1066	702	12956.2676	31	192	13087.0000	12629.5426		3.50%	37.7	5s

Branch-and-Cut 分支切割

Presolving

Presolved: 987 rows, 855 columns, 19346 nonzeros
Variable types: 211 continuous, 644 integer (545 binary)

Root relaxation: objective 1.112003e+04, 1063 iterations, 0.03 seconds

Nodes		Current Node			Objective Bounds			Work	
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
0	0	11120.0279	0	154	-	11120.0279	-	-	0s
0	0	11526.8918	0	207	-	11526.8918	-	-	0s
0	0	11896.9710	0	190	-	11896.9710	-	-	0s
0	0	12151.4022	0	190	-	12151.4022	-	-	0s
0	0	12278.3391	0	208	-	12278.3391	-	-	0s
...									
5485	634	12885.3652	52	143	12890.0000	12829.0134	0.47%	54.5	25s

Cutting Planes

Cutting planes:
Learned: 4
Gomory: 46
Cover: 39
Implied bound: 8
Clique: 2
MIR: 112
Flow cover: 27
GUB cover: 11
Zero half: 91

Explored 6808 nodes (357915 simplex iterations) in 27.17 seconds
Thread count was 4 (of 8 available processors)

Branch-and-Cut

Presolved: 987 rows, 855 columns, 19346 nonzeros
Variable types: 211 continuous, 644 integer (545 binary)

Root relaxation: objective 1.112003e+04, 1063 iterations, 0.03 seconds

Nodes		Current Node			Objective Bounds			Work	
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
0	0	11120.0279	0	154	-	11120.0279	-	-	0s
0	0	11526.8918	0	207	-	11526.8918	-	-	0s
0	0	11896.9710	0	190	-	11896.9710	-	-	0s
...									
0	0	12448.7684	0	181	-	12448.7684	-	-	0s
H	0				16129.000000	12448.7684	22.8%	-	0s
H	0				15890.000000	12448.7684	21.7%	-	0s
	0	2	12448.7684	0	181	15890.0000	21.7%	-	0s
H	142	129			15738.000000	12450.7195	20.9%	43.8	1s
H	212	189			14596.000000	12453.8870	14.7%	42.3	1s
H	217	181			13354.000000	12453.8870	6.74%	42.6	1s
*	234	181		40	13319.000000	12453.8870	6.50%	42.1	1s
H	254	190			13307.000000	12453.8870	6.41%	41.3	1s
H	284	194			13183.000000	12453.8870	5.53%	42.6	1s
H	286	194			13169.000000	12453.8870	5.43%	42.7	1s

Presolving

Cutting Plane

Heuristics

Branching

Branch-and-Cut分支切割

Presolving

Presolved: 987 rows, 855 columns, 19346 nonzeros
Variable types: 211 continuous, 644 integer (545 binary)
Root relaxation: objective 1.112003e+04, 1063 iterations, 0.03 seconds

Cutting Plane

Nodes		Current Node			Objective Bounds			Work	
Expl	Unexpl	Obj	Depth	IntInf	Incumbent	BestBd	Gap	It/Node	Time
0	0	11120.0279	0	154	-	11120.0279	-	-	0s
0	0	11526.8918	0	207	-	11526.8918	-	-	0s
0	0	11896.9710	0	190	-	11896.9710	-	-	0s
...									
H	0	0			15890.000000	12448.7684	21.7%	-	0s
	0	2	12448.7684	0	181	15890.0000	12448.7684	21.7%	0s
...									
1066	702	12956.2676	31	192	13087.0000	12629.5426	3.50%	37.2	5s
1097	724	12671.8285	8	147	13087.0000	12671.8285	3.17%	41.6	10s
1135	710	12732.5601	32	126	12890.0000	12727.1362	1.26%	44.6	15s
3416	887	12839.9880	46	136	12890.0000	12780.7059	0.85%	49.7	20s
5485	634	12885.3652	52	143	12890.0000	12829.0134	0.47%	54.5	25s

Branching

- 优化是NP-hard
 - 我们只应考虑启发式，但可行性问题也是NP-hard
 - 理论上, 启发式和优化难度是一样的
- 许多实际问题经常被解到最优
- 我的优化问题很难解
 - 所以我只开发了自己的启发式算法
 - 那么你就是这次演讲的最佳听众
- 启发式算法和优化算法并不对立, 可以融合
- 我们将展示
 - Gurobi启发式的想法(ideas), 或许可以帮助您开发和改进您的启发式算法
 - 如何融合Gurobi优化算法和您的启发式算法为您找到更好的解决方案

GUROBI Heuristics

GUROBI启发式算法

- Gurobi has more than 30 heuristics
- Different types
 - Non-LP based
 - Enumerate, search, **greedy**, ...
 - LP based
 - **Rounding**, fixing & diving, ...
 - Reformulation
 - **Zero-objective**, **min relaxation**, ...
 - Improvement
 - **RINS** ..
 - SubMIP and recursive
 - Target heuristic, RINS, ...
 - Problem specific
 - Fixed charge network heuristic
 - Features helping heuristics
 - **Pump reduce**

有30多种启发式算法

不同种类

基于非LP

基于LP

模型改建

改进型

子MIP和递归

针对具体问题的启发式

有助于启发式的功能

Non-LP Based Heuristics: Greedy

基于非LP启发式算法: 贪婪算法



- Famous algorithms 著名算法

- Optimal 最优的: shortest path最短的路径, min spanning tree 最小生成树

- Not optimal非最优的: 0-1 Knapsack 背包

Max $10u + 8v + 11x + 7y + 5z$

s.t. $3u + 4v + 6x + 4y + 3z \leq 14$

u, v, x, y, z are binary variables

Sorting variables based on the ratio of the obj. coefficient and constraint coefficient, already sorted

Setting variables to one based on the order until it become infeasible. Here setting y to one become infeasible

So the solution is $(u, v, x, y, z) = (1, 1, 1, 0, 0)$ with obj. value 29

Optimal solution $(u, v, x, y, z) = (1, 1, 0, 1, 1)$ with obj. value 30

- Gurobi blind heuristics 盲目启发式 (blind means not using LP relaxation solution)

- Sort binary/integer variables based on some measure 用某种度量对二进制/整数变量进行排序

- Fixing them in the greedy order 按贪婪顺序固定它们

- Propagate fixing and bound changes for each fix 传播变量固定和收紧界值

- Without it, it is almost impossible to find a feasible solution

- Solve the remaining LP model, if there are continuous variables

- LP based greedy heuristics 基于LP的贪婪算法

- It is often more effective to use relaxation solution to sort variables 用松弛解对变量排序通常更有效

Non-LP Based Heuristics 基于非LP启发式算法



- They can find integer solutions quickly 有可能可以快速找到整数解
- The quality of the solutions is often very poor 解的质量通常很差
- One poor solution is good enough, poor solutions often won't help overall optimization 一个差的解就足够了, 差的解往往无法帮助整体优化
- Multi cores and difficulty to parallelize the root node are the reasons to pay some attention to non-LP based heuristics 多核电脑和难以对根节点并行化是关注基于非LP的启发式算法的原因

LP Based Heuristics 基于LP启发式算法



- Rounding 取整法
 - Solve LP relaxation, round the solution values to nearest integer values 解LP松弛问题, 将解值四舍五入到最近的整数值
 - 0-1 knapsack example (same example) 0-1背包例子(相同例子)
Max $10u + 8v + 11x + 7y + 5z$
s.t. $3u + 4v + 6x + 4y + 3z \leq 14$
 u, v, x, y, z are binary variables
Optimal LP relaxation solution $(u, v, x, y, z) = (1, 1, 1, \frac{1}{4}, 0)$
Rounded solution $(u, v, x, y, z) = (1, 1, 1, 0, 0)$ is integer feasible with obj. value 29
 - Simple rounding won't work well, especially for models with equalities 简单的取整不会很好, 特别是具有等式约束的模型
 - Consider integer values on both sides 考虑两边的整数值
 - Rounding with propagating, fixing variables and tightening bounds 取整时需传播变量固定和收紧界值
 - Gurobi has several different versions of rounding heuristics Gurobi有多种不同版本的取整启发式算法
- Most Gurobi heuristics are LP based or need LP relaxation solutions 大多数Gurobi启发式算法都是基于LP或需要LP松弛解
 - LP relaxation solution is very important for heuristics to get high quality solutions! LP松弛解对于启发式算法获取高质量解非常重要!

Reformulation 模型改建

- Example 例子

$$\text{Min } 3u + 8v + 3w + 2x + 7y + 5z$$

$$\text{s.t. } 3u + 4v - 4w + 8x + 4y + 3z \leq 9$$

$$5u + 2v + 4x + 7y + 9z = 15$$

u, v, x, y, z are non negative integer variables, w is a binary variable

- Zero-objective heuristic 去目标启发式

- Remove the objective and solve it as a feasible problem 删除目标并将其为可行问题去解
- Hope that presolve can have more reductions and the resulting presolved model is easier to solve 希望预预优化可以使模型变的更小, 并且最终的预优化模型更容易解
- For this example, the reformulated model is 对于这个例子, 重新改建的模型是

$$\text{Min } 0$$

$$\text{s.t. } 3u + 4v - 4w + 8x + 4y + 3z \leq 9$$

$$5u + 2v + 4x + 7y + 9z = 15$$

u, v, x, y, z are non negative integer variables, w is a binary variable

- Variables x and v are parallel, x can be fixed to 0
- Variable w can be fixed to 1, which will only help the feasibility of the first constraint

- Minimum relaxation heuristic 最小松弛启发式
 - For each inequality, add one penalty variable for the constraint violation 对于每个不等式，加一个违反约束的惩罚变量
 - For each equality, add two penalty variables for the two directions of the violation 对于每个等式，加两个违反约束的惩罚变量，两个方向各一个
 - Then minimize the sum of violations 然后对违约总和求最小化
 - If the optimal solution has the sum = 0, then we find a feasible solution; otherwise the original model is infeasible 如果最优总和为0，那我们找到一个可行解；否则原始模型是不可行的
 - For this example, the reformulated model is 对于这个例子，重新改建的模型是

Min $r + s + t$

s.t. $3u + 4v - 4w + 8x + 4y + 3z - r \leq 9$

$5u + 2v + 4x + 7y + 9z + s - t = 15$

u, v, x, y, z are non negative integer variables

w is a binary variable

r, s, t are non negative continuous variables

- Relaxation induced neighborhood search (RINS) 松弛诱导邻域搜索
- Given the incumbent (the best integer solution found so far) and the current fractional solution of the node relaxation 给定现任整数解和节点松弛的当前分数解
- Fix a variable if its incumbent value and its relaxation value agrees 如果变量的整数解值和松弛解值一致，则固定变量
- Solve the partially fixed model as a subMIP 将部分固定的模型作为子MIP去解
- It is an improvement heuristics 这是一种改进型的启发式算法
- It is the our most effective heuristic 这是我们最有效的启发式算法

SubMIP and Recursive Solve 子MIP和递归

- Many Gurobi heuristics will 许多Gurobi启发式算法会
 - Have a target to fix some percentage of variables, say 80% 设一个目标来固定一定比例的变量, 比如80%
 - Fix one variable and then propagate 固定一个变量然后传播
 - Repeat fixing and propagating until the target is reached or it becomes infeasible 重复固定和传播, 直到达到目标或它变得不可行
 - Solve it as a subMIP 将其为子MIP去解
 - In the subMIP, it will call the same heuristics, so recursively 在子MIP中,它将以递归方式调用相同的启发式算法
 - It often works well and finds feasible solutions quickly 它通常非常有效, 可迅速找到可行解

Feasibility Pump Heuristic 泵式缩减启发式



- Fischetti, Glover and Lodi, 2004
- Solve the relaxation and round the solution 解松弛问题并舍入到整数解
- Replace the objective to minimize the distance to the rounded solution (quadratic) 目标换成到舍入整数解距离最小 (二次)
- Use L1 norm ($\sum |x_j - x_j^*|$), where x^* is the rounded solution (linear) 使用L1范数($\sum |x_j - x_j^*|$), 其中 x^* 是取整解 (线性)
 - If a binary variable $x_j = 0.3$, then $x_j^* = 0$, then the objective part for x_j is $|x_j - 0| = x_j$, i.e. obj. coefficient is 1
 - If a binary variable $x_j = 0.7$, then obj. coefficient will be -1
- Solve the modified LP and repeat 解修改后的LP并重复
- Until it hits some limit or the relaxation solution is integer feasible 直到它达到一定限值或松弛解是整数可行的
- Setting the limit to e.g. 10, i.e. solving the LP 10 times is expensive and it usually won't be lucky 例如将限值设为10, 则需解LP10次, 很耗时, 通常很难运气好

- Motivated by feasibility pump heuristic 受泵式缩减启发式算法的启发
- Observation 观察
 - Most models are dual degenerate, i.e. relaxation has alternative optimal solutions 大多数模型对偶退化,即松弛问题有多个的最优解
- Goal 目标
 - A relaxation solution with less fractional integer variables 有较少整数变量取分数值的松弛解
 - Possible zero fractional integer variables, but not the goal, so it isn't heuristic 可能没有分数整数变量,但不是目标,所以它不是启发式的
 - Such relaxation solution helps heuristics and b&b significantly to find integer feasible solutions 这样的松弛解很显著地帮助启发式和b&b找到可行的整数解
- Steps 步骤
 - Solve the relaxation and fix all variables with nonzero reduced costs, making sure to stay in the optimal space 解松弛问题, 固定非零递减成本的所有变量, 确保保持在最优空间
 - Round the relaxation solution, replace the objective with L1 norm distance to the rounded solution 舍入松弛解, 目标换成到舍入整数解距离最小 (L1 范数)
 - Solve the modified LP, round and repeat 解修改后的LP并重复
 - Until it hits some limit or the number of fractional integer variables doesn't go down 直到它达到一定限值或取分数值的整数变量的数量不下降

GUROBI Heuristic Parameters

Gurobi 启发式参数

Heuristic Parameters 启发式参数



- Main MIP parameter 主要MIP参数, MIPFocus
- Main heuristic parameter 主要启发式参数, Heuristics
- Individual heuristic parameters 个别启发式参数
- Other parameters affecting feasible solutions 影响可行解的其他参数

- Define high-level solution strategy 定义解高层策略
- Default 默认, balance between finding new feasible solutions and proving that the current solution is optimal. 在找新的可行解和证明最优性之间取得平衡
- = 1, more interested in finding feasible solutions quickly 更注重找到可行解
- = 2, more attention on proving optimality 更注重证明最优性
- = 3, focus on the objective bound 更注重目标界值

- Main heuristic parameter 主要启发式参数
- The parameter value is roughly the fraction of time that we will spend on heuristics 参数值大致是我们在启发式上花费时间的部分值
- Default value 默认值 = 0.05
- > 0.05 , more aggressive, 1 most aggressive
- < 0.05 , less aggressive, 0 no heuristics

Individual heuristic parameters 个别启发式参数



- Pump reduce 泵式缩减 (or degenerate simplex moves), **Degenmoves**
- Feasibility pump heuristic 泵式缩减启发式, **PumpPasses**
- Improvement heuristic parameters 改进型的启发式参数
 - **ImproveStartGap**
 - **ImproveStartNodes**
 - **ImproveStartTime** (warning: not deterministic)
- Minimum relaxation heuristic 最小松弛启发式, **MinRelNodes**
- RINS heuristic **RINS**启发式, **RINS**
- Zero objective heuristic 去目标启发式, **ZeroObjNodes**

Other Parameters Affecting Heuristics

影响启发式的其他参数



- Nodes explored by sub-MIP heuristics, **SubMIPNodes**
- Branch direction preference, **BranchDir**
 - Setting the value to 1 may help MIP diving to find a feasible solution more quickly
- Tuning criterion, **TuneCriterion**
 - = 2 objective value, i.e. focusing more on finding good feasible solutions

User Input for GUROBI Heuristics

启发式的用户输入功能

MIP Start / Multiple MIP Starts MIP起始值/多个MIP起始值



- User can provide a MIP start or multiple MIP starts (new in 8.0) 用户可以提供一個或多个MIP起始值(多个为8.0的新功能)
- A good MIP start, even a partial solution often can produce a good feasible solution instantly 良好的MIP起始值, 即使是部分解, 也可以立即产生好的可行解
- Useful when you have multiple partial solutions 有多个部分解可能会很有用
 - MIP solver will try to complete them, and will store the ones it finds
- For distributed MIP, MIP starts will be evaluated on different machines 对于分布式MIP, 将在不同的机器上评估MIP起始值

- Provide hints to the solver about which variable should take which value 向优化器提示哪个变量应采用哪个值
- Guides heuristics and branching 指导启发式和分支
- VarHintVal attribute 属性
 - Specifies a value for a variable 指定变量的值
- VarHintPri attribute 属性
 - Specifies a level of confidence in this particular variable value 指定此特定变量值的置信度
- Comparison to MIP start 与MIP起始值比较
 - MIP start is used to provide an initial feasible solution to the solver MIP起始值用于为优化器提供初始可行解
 - Is evaluated prior to starting the solution process
 - Provides incumbent if feasible
 - Does not influence solution process if it is not feasible
 - Variable Hints guide the search 变量提示指导搜索
 - High quality hints should lead to a high quality solution quickly
 - Either through heuristics or through branching
 - Affects the whole solution process

- User-specified local improvement heuristic 用户指定的局部改进型启发式算法
- RINS is our most effective heuristic RINS是我们最有效的启发式算法
- It is a *sub-MIP* heuristic 这是一个子MIP启发式算法
 - Fix a subset of the variables to incumbent values 将变量的子集固定为现任整数解的值
 - Solve the resulting MIP (recursively) 解生成的MIP(递归)
 - Reoptimizes over just that portion of the problem
- Sub-MIP heuristics extremely effective in general 子MIP启发式算法一般非常有效
- How to choose the sub-problem to reoptimize? 如何选择子问题进行重新优化?
 - RINS chooses automatically RINS自动选择
 - This feature allows user to make the choice 此功能允许用户做出选择
 - Example sub-problems:
 - All decisions related to a single time period
 - All decisions related to a single machine
 - All decisions related to physical sub-regions (e.g., Western US, Eastern US, etc.)

- Motivations
 - Our MIP solver is mostly a black box solver 我们的MIP求解器主要是黑盒求解器
 - We try to recognize some common structures, but very limited
 - Users know the structure of their model 用户知道他们模型的结构
 - Relaxation solutions help heuristics a lot 松弛解对启发式算法很有帮助
 - Knowledge of problem structure and the relaxation solutions often mean fast good feasible solutions 对问题结构和松弛解的了解通常意味着快速找到可行解
- Heuristic callback
 - At each node, Gurobi will call back 在每个节点，Gurobi都会回调
 - Users can query the relaxation solution and use it to guide their heuristics 用户可以查询松弛解并使用它来指导他们的启发式算法
 - Users can provide a full or partial solution vector to Gurobi through callback 用户可以通过回调向Gurobi提供完整或部分解
 - If it is partial, Gurobi will try to complete it 如果它是部分的，Gurobi将尝试完成它

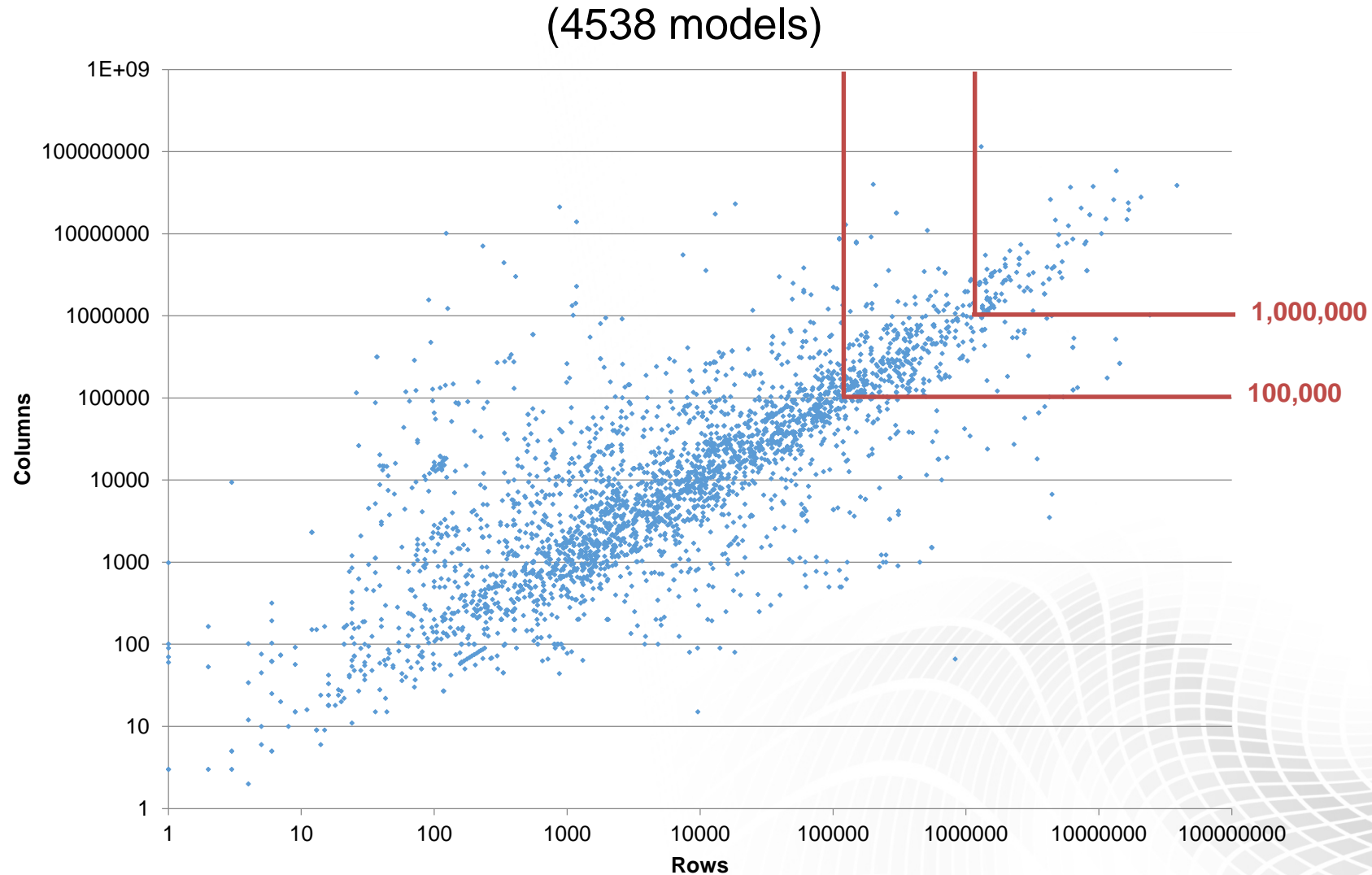
What to do with too big/hard models

如何处理太大/太难的模型

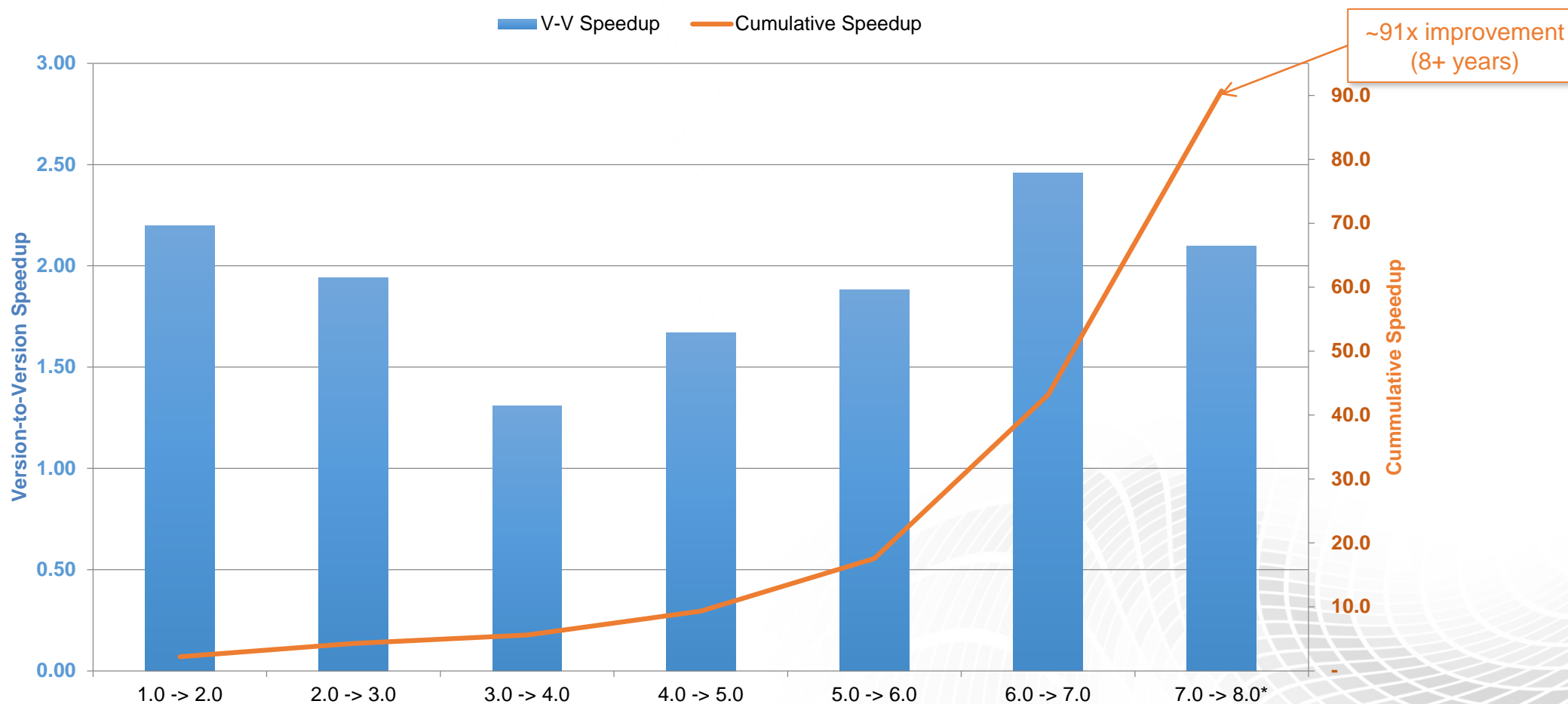
Too Big/Hard Models, Really? 太大/太难的模型, 真的吗?



- “My model is too big or too hard, I have no choice but heuristic”, really? “我的模型太大或太难, 我别无选择, 只有启发式算法”, 真的吗?
- Old MIP experiences don't count 旧的经验不算数
 - At the end of 80's and earlier 90's, people in electrical power industry concluded that MIP was a nice tool, which couldn't solve real unit commitment model
 - Close to 2000, people revisited MIP technology and people now solve the unit commitment model routinely
 - The similar stories happened more and more
- I tried open source solvers, they are hopeless 我试过开源优化器, 没有希望解我的问题
 - We have a lot of users, who send us their models, since the open solver they used couldn't find a feasible solution in hours. Gurobi often solved the models in less than one second
 - All open source solvers are way behind the state of art commercial solvers
- Gurobi users often solve their MIP models with millions of variables/constraints Gurobi用户经常解有数百万个变量/约束的MIP模型
 - Our customer model sets have a lot of such models, many of them we can solve or find good solutions within 10% MIPGap.



MIP速度不断提高, 主要版本每次提高几乎两倍



- Have you tried to solve the relaxation? 你试过解松弛问题吗?
 - LP relaxation is polynomial-time solvable LP松弛问题是多项式时间可解的
 - Gurobi has solved LP models with 100M+ variables/constraints Gurobi解过许多超过几亿个变量/约束的LP模型
 - Relaxation solution is often very useful for heuristics 松弛解对启发式算法很有帮助
 - The objective value of the relaxation solution provides the bound, without it, it is hard to know how good a heuristic solution is 松弛解提供目标界值，没有它，很难知道启发式解有多好

- Have you tried to reduce the models?你试过减小模型吗?
 - Aggregate汇总
 - Daily schedule -> weekly schedule日计划->周计划
 - Decompose big model into smaller pieces 将大模型分解为较小的部分
 - World -> America, Europe and Asia 世界 ->美洲,欧洲和亚洲
 - Local improvement 局部改进
 - Use heuristic to generate an initial solution 使用启发式算法生成初始解
 - Use MIP to reoptimize over a portion of the model, like RINS 使用MIP重新优化模型的一部分, 如RINS
 - No lower bound, but often produces very high quality global solutions 没有下限, 但通常会产生非常高质量的**整体解**

- Successful stories to combine optimization and heuristics融合优化和启发式的成功案例
 - MIP based heuristics 基于MIP的启发式算法
 - Rolling horizon heuristics 滚动时段启发式算法
 - Relax integrality of future periods
 - May aggregate future time periods
 - Solve smaller LP/MIP
 - Air taxi and mining
 - Local search heuristics 局部搜索启发式算法
 - In group of periods, machines etc, solve smaller LP/MIP
 - Lenstra et al., local search in combinatorial optimization
 - Lin-Kernighan heuristic for TSP 货郎担问题的启发式算法
 - Solve relaxation and use reduced costs to guide
 - Etc.

Always Try MIP

总是试试MIP

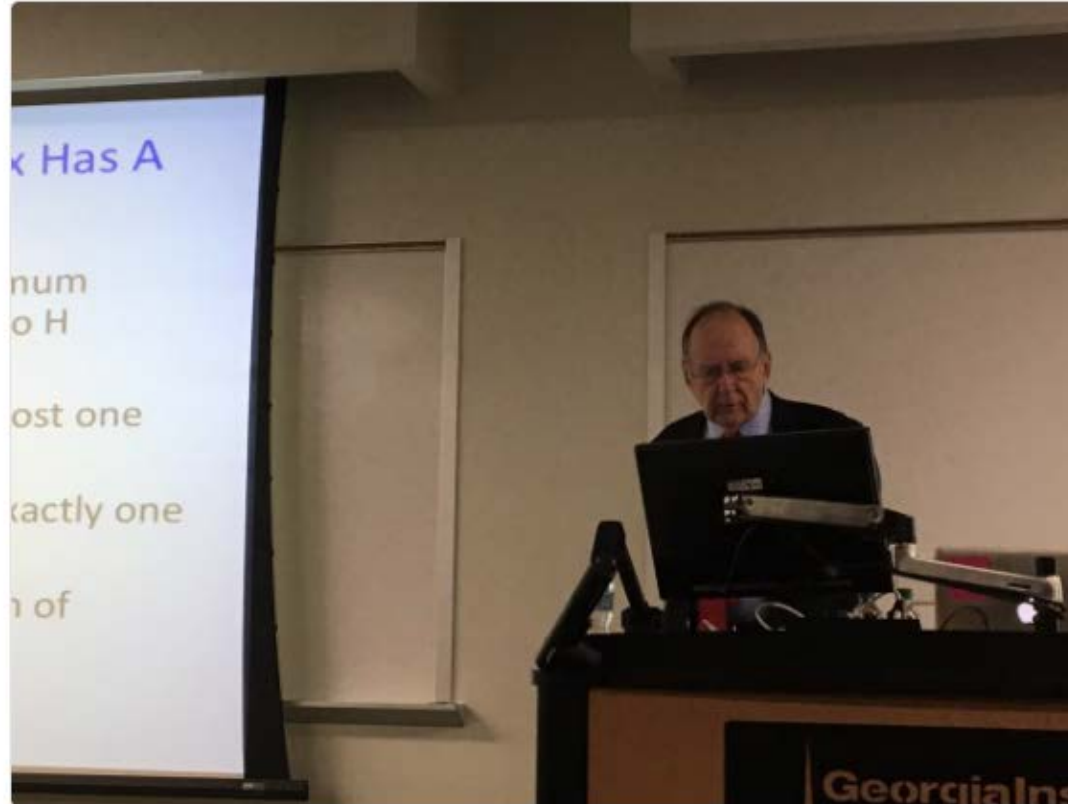


Shabbir Ahmed
@Shabbir0Ahmed

Follow



Richard Karp quotes a colleague "Always try integer programming, it might work"



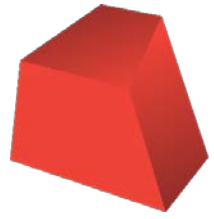
8:27 AM - 13 Mar 2017

Copyright © 2018, Gurobi Optimization, LLC

Conclusion

结论

- Always try Gurobi, it should be better than pure heuristics
试试Gurobi, 它应该比纯启发式更好!



GUROBI
OPTIMIZATION

Thank you – Questions?

谢谢，请提问题