

# 第6讲 回溯与分支限界 (1/2)

罗国杰

[gluo@pku.edu.cn](mailto:gluo@pku.edu.cn)

2024年春季学期

# 本节概要

- ➡ 回溯与剪枝 backtrack with pruning
  - ▶ 多米诺性质
- ➡ 分支限界 Branch-and-bound
  - ▶ 全局优化的分支限界例子
  - ▶ 解更大的问题?—更快地解小问题
- ➡ 分支限界应用于非凸优化
  - ▶ 通用的算法和收敛性
  - ▶ 非凸优化例子: 混合布尔凸规划
  - ▶ 混合布尔凸规划例子: 最小基数问题

## 回溯 (backtrack)

- 回溯算法
- 分支限界
- 应用实例

# 回溯法：基本概念和适用条件

## ■ 基本概念

- ▶ 搜索问题、搜索空间、搜索策略
- ▶ 判定条件、结点状态、存储结构

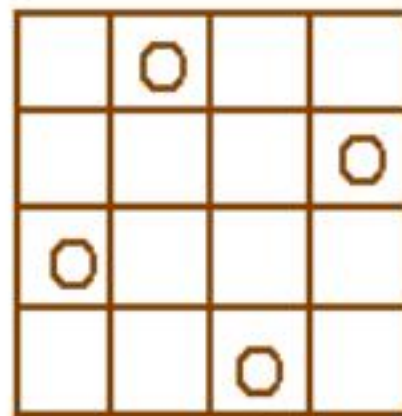
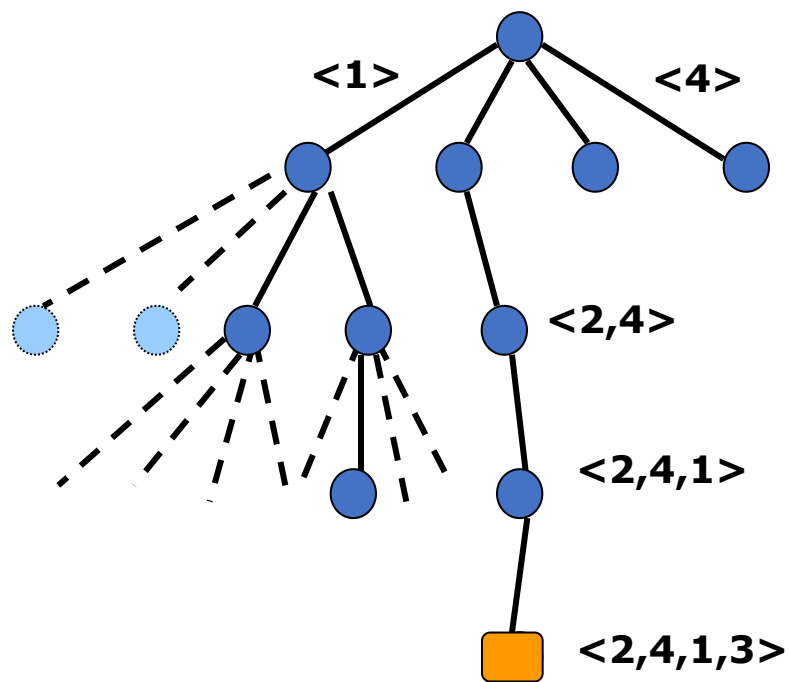
## ■ 必要条件

- ▶ 多米诺性质

# 实例1：四后（4-Queen）问题

解表示成一个4维向量， $\langle x_1, x_2, x_3, x_4 \rangle$ （放置列号）

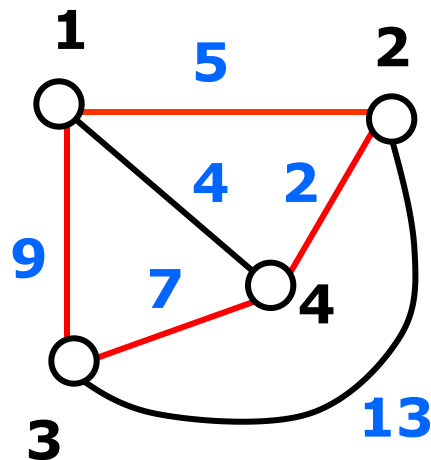
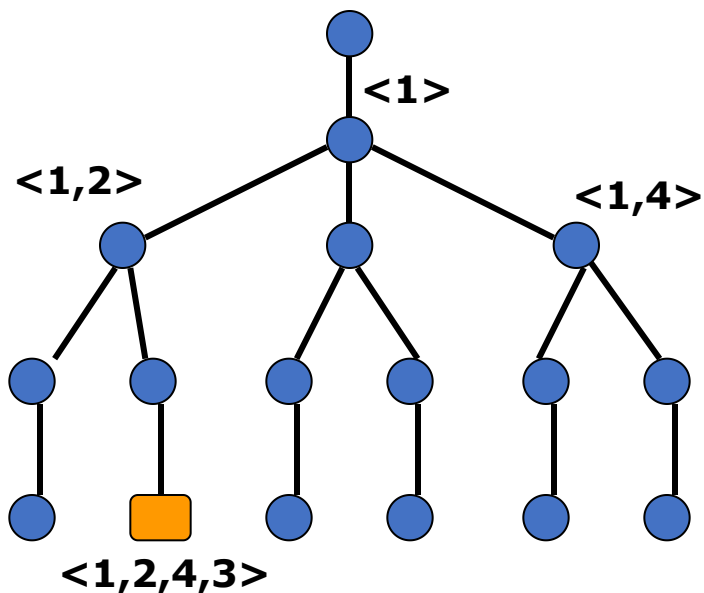
搜索空间：4叉树



## 实例3：旅行商（TSP）问题

$\langle i_1, i_2, \dots, i_n \rangle$  为巡回路线

搜索空间：排列树， $(n-1)!$ 片树叶



$\langle 1, 2, 4, 3 \rangle$  对应于巡回路线： $1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 1$

长度： $5 + 2 + 7 + 9 = 23$

# 回溯法的基本思想 (1/2)

- 适用问题
  - ▶ 求解搜索问题
- 搜索空间
  - ▶ 一棵树
  - ▶ 每个结点对应了部分解向量,
  - ▶ 可达的树叶对应了可行解
- 搜索过程:
  - ▶ 采用系统的方法隐式地遍历搜索树
- 搜索策略:
  - ▶ 深度优先, 宽度优先, 函数优先, 宽深结合等

## 回溯法的基本思想 (2/2)

### ➤ 结点分支判定条件:

- ▶ 满足约束条件——分支扩张解向量
- ▶ 不满足约束条件, 回溯到该结点的父结点

### ➤ 结点状态

#### ▶ 动态生成

- 白结点 (尚未访问)
- 灰结点 (正在访问该结点为根的子树)
- 黑结点 (该结点为根的子树遍历完成)

### ➤ 存储

#### ▶ 当前路径



## 必要条件：多米诺性质

设  $P(x_1, x_2, \dots, x_i)$  为真表示向量  $\langle x_1, x_2, \dots, x_i \rangle$  中  $i$  个皇后放置在彼此不能攻击的位置

$$\begin{aligned} P(x_1, x_2, \dots, x_{k+1}) &\rightarrow P(x_1, x_2, \dots, x_k) \\ \Leftrightarrow \neg P(x_1, x_2, \dots, x_k) &\rightarrow \neg P(x_1, x_2, \dots, x_{k+1}) \end{aligned} \quad 0 < k < n$$

例4 求不等式的整数解

$$5x_1 + 4x_2 - x_3 \leq 10, \quad 1 \leq x_i \leq 3, \quad i=1,2,3$$

$P(x_1, \dots, x_k)$  : 意味将  $x_1, x_2, \dots, x_k$  代入原不等式的相应部分使得左边小于等于10。不满足多米诺性质

变换： 令  $x_3 = 3 - x_3'$ ,

$$5x_1 + 4x_2 + x_3' \leq 13, \quad 1 \leq x_1, x_2 \leq 3, \quad 0 \leq x_3' \leq 2$$

# 回溯算法的设计步骤

- 定义搜索问题的解向量和每个分量的取值范围
  - ▶ 解向量为  $\langle x_1, x_2, \dots, x_n \rangle$
  - ▶ 确定  $x_i$  的可能取值的集合为  $X_i, i = 1, 2, \dots, n$ .
- 当  $x_1, x_2, \dots, x_{k-1}$  确定以后计算  $x_k$  取值集合  $S_k, S_k \subseteq X_k$
- 确定结点儿子的排列规则
- 判断是否满足多米诺性质
- 搜索策略----深度优先、宽度优先等
- 确定每个结点分支约束条件
- 确定存储搜索路径的数据结构

# 一般的回溯算法伪代码

- P: 问题实例
- c: 部分解
- backtrack: 隐式遍历搜索树

```
procedure backtrack(P, c) is
    if reject(P, c) then return
    if accept(P, c) then output(P, c)
    S ← expand(P, c)    # 部分解c的直接扩展集合
    for s in S do
        backtrack(P, s)
```

# 影响算法效率的因素

- 最坏情况下的时间 $W(n)=(p(n)f(n))$ 
  - ▶ 其中 $p(n)$ 每个结点时间,  $f(n)$ 结点个数
- 影响回溯算法效率的因素
  - ▶ 搜索树的结构
    - 分支情况: 分支均匀否
    - 树的深度
    - 对称程度: 对称适合裁减
  - ▶ 解的分布
    - 在不同子树中分布多少是否均匀
    - 分布深度
  - ▶ 约束条件的判断: 计算简单

# 回溯算法的效率估算：估计搜索树的结点数

计数搜索树中遍历的结点，Monte Carlo方法

## Monte Carlo方法

1. 从根开始，随机选择一条路径，直到不能分支为止，即从 $x_1, x_2, \dots$ ，依次对 $x_i$ 赋值，每个 $x_i$ 的值是从当时的 $S_i$ 中随机选取，直到向量不能扩张为止.
2. 假定搜索树的其他 $|S_i| - 1$ 个分支与以上随机选出的路径一样，计数搜索树的点数.
3. 重复步骤 1 和 2，将结点数进行概率平均.

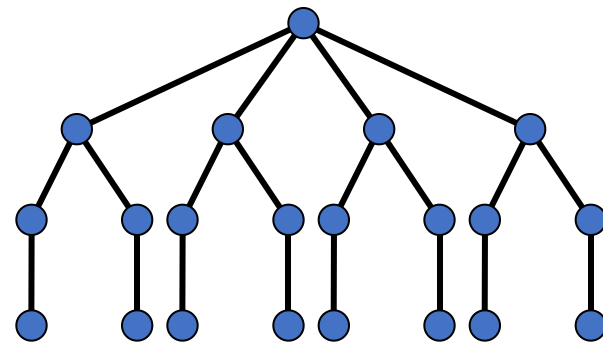
# 平均效率估算：实例

例5 估计四后问题的效率

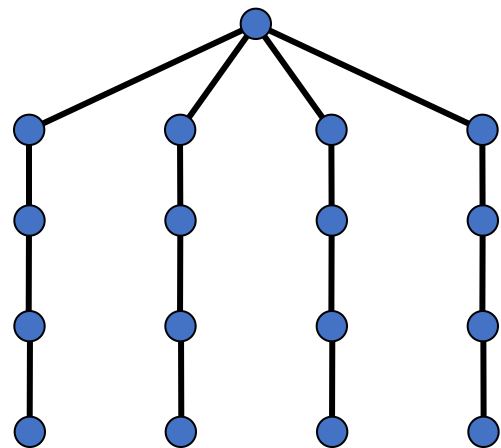
case1.  $\langle 1, 4, 2 \rangle$ :  $1 + 4 + 4 \times 2 + 4 \times 2 = 21$

case2.  $\langle 2, 4, 1, 3 \rangle$ :  $4 \times 4 + 1 = 17$

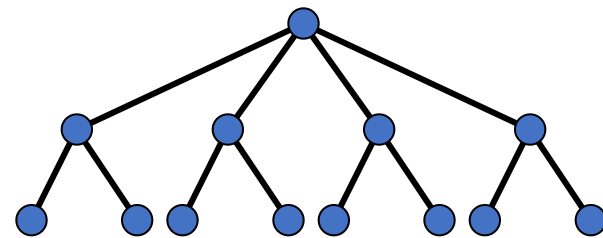
case3.  $\langle 1, 3 \rangle$ :  $1 + 4 \times 1 + 4 \times 2 = 13$



Case1:  $\langle 1, 4, 2 \rangle$



Case2:  $\langle 2, 4, 1, 3 \rangle$



Case3:  $\langle 1, 3 \rangle$

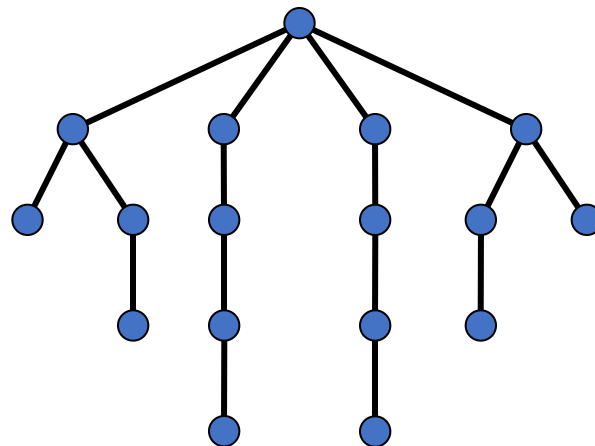
# 平均效率估算：结点数估计

假设 4 次抽样测试：

case1:1次, case2:1次, case3:2次,

平均结点数 =  $(21 \times 1 + 17 \times 1 + 13 \times 2) / 4 = 16$

搜索空间访问的结点数为17



搜索空间

## 平均效率估算：算法实现 (1/2)

Monte Carlo

```
1. sum  $\leftarrow$  0 // sum 为 t 次结点平均数
2. for i  $\leftarrow$  1 to t do // 取样次数 t
3.   m  $\leftarrow$  Estimate(n) // m 为本次结点总数
4.   sum  $\leftarrow$  sum + m
5. sum  $\leftarrow$  sum / t
```



## 平均效率估算：算法实现 (2/2)

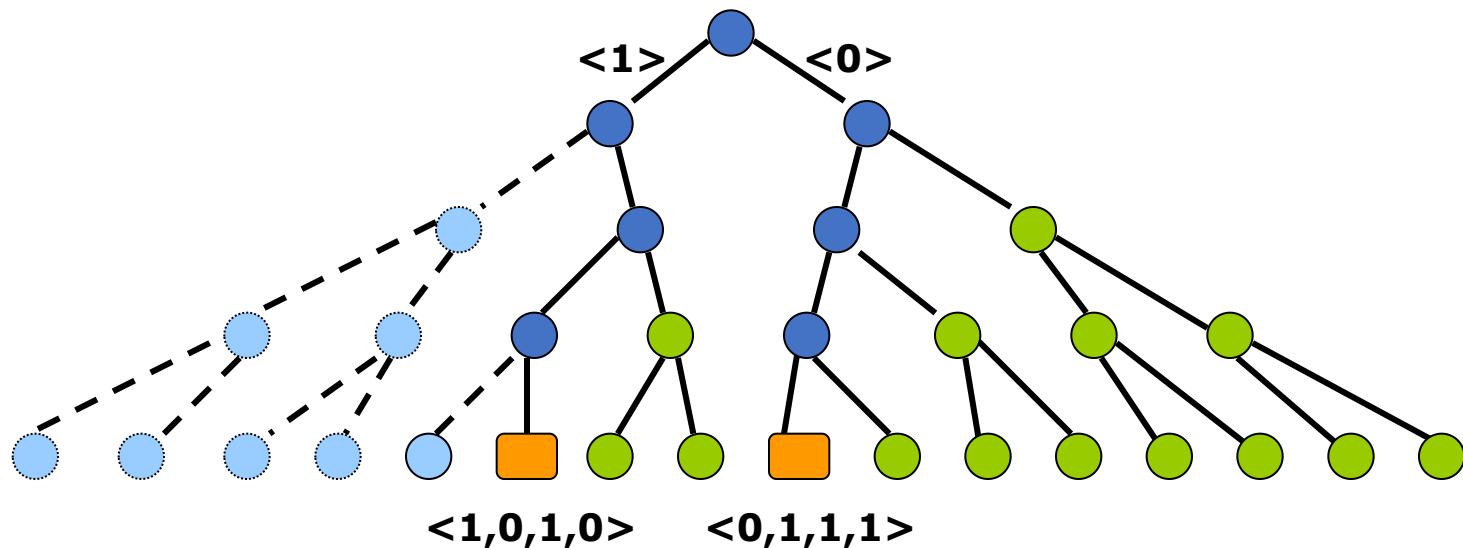
$m$ 为输出——本次取样结点总数， $k$ 为层数， $r_1$ 为本层分支数， $r_2$ 为上层分支数， $n$ 为树的层数

### 算法 Estimate( $n$ )

```
1.  $m \leftarrow 1$ ;  $r_2 \leftarrow 1$ ;  $k \leftarrow 1$  //  $m$  为结点总数
2. While  $k \leq n$  do
3.   if  $S_k = \emptyset$  then return  $m$ 
4.    $r_1 \leftarrow |S_k| * r_2$  //  $r_1$  为扩张后结点总数
5.    $m \leftarrow m + r_1$  //  $r_2$  为扩张前结点总数
6.    $x_k \leftarrow$  随机选择  $S_k$  的元素
7.    $r_2 \leftarrow r_1$ 
8.    $k \leftarrow k+1$ 
```

结点： 向量 $\langle x_1, x_2, x_3, \dots, x_k \rangle$ （子集的部分特征向量）

搜索空间: 子集树,  $2^n$ 片树叶



$\langle 0, 1, 1, 1 \rangle$  可行解:  $x_1=0, x_2=1, x_3=1, x_4=1$ . 重量: 13, 价值: 28

$\langle 1, 0, 1, 0 \rangle$ 可行解:  $x_1=1, x_2=0, x_3=1, x_4=0$ . 重量: 12, 价值: 21

# 组合优化问题

## ► 相关概念

- ▶ 目标函数 (极大化或极小化)
- ▶ 约束条件
- ▶ 搜索空间中满足约束条件的解称为可行解
- ▶ 使得目标函数达到极大(或极小)的解称为最优解

## ► 实例：物品无限的背包问题

$$\max x_1 + 3x_2 + 5x_3 + 9x_4$$

$$2x_1 + 3x_2 + 4x_3 + 7x_4 \leq 10$$

$$x_i \in N, i = 1, 2, 3, 4$$

# 分支限界技术 (branch-and-bound)

## ➤ 设立代价函数

- ▶ 函数值以该结点为根的搜索树中的所有可行解的目标函数值的上界 (对于最大化问题)
- ▶ 父结点的代价不小于子结点的代价

## ➤ 设立界

- ▶ 代表当时已经得到的可行解的目标函数的最大值
- ▶ 界的设定初值可以设为0
- ▶ 可行解的目标函数值大于当时的界，进行更新

## ➤ 搜索中停止分支的依据

- ▶ 不满足约束条件或者其代价函数小于当时的界

# 实例：物品无限的背包问题

## ► 物品无限的背包问题形式化

$$\begin{aligned}\max & x_1 + 3x_2 + 5x_3 + 9x_4 \\ & 2x_1 + 3x_2 + 4x_3 + 7x_4 \leq 10 \\ & x_i \in \mathbb{N}, i = 1, 2, 3, 4\end{aligned}$$

## ► 对变元重新排序使得

$$\frac{v_i}{w_i} \geq \frac{v_{i+1}}{w_{i+1}}$$

## ► 排序后实例

$$\begin{aligned}\max & 9x_1 + 5x_2 + 3x_3 + x_4 \\ & 7x_1 + 4x_2 + 3x_3 + 2x_4 \leq 10 \\ & x_i \in \mathbb{N}, i = 1, 2, 3, 4\end{aligned}$$

# 代价函数与分支策略确定

► 结点  $\langle x_1, x_2, \dots, x_k \rangle$  的代价函数

$$\sum_{i=1}^k v_i x_i + (B - \sum_{i=1}^k w_i x_i) \frac{v_{k+1}}{w_{k+1}} \quad \text{若对某个 } j > k \text{ 有 } B - \sum_{i=1}^k w_i x_i \geq w_j$$

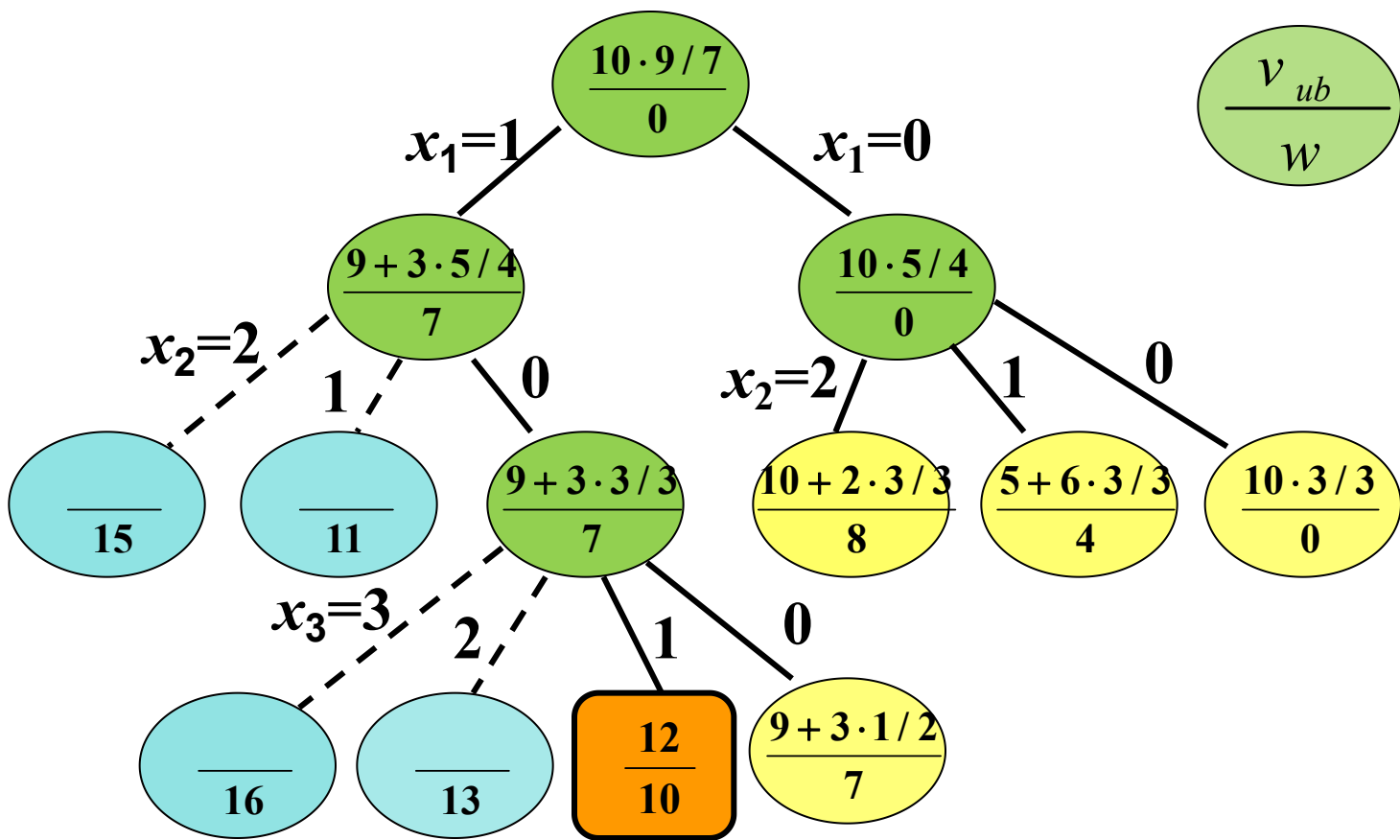
$$\sum_{i=1}^k v_i x_i \quad \text{否则}$$

► 分支策略----深度优先

# 实例

$$\max 9x_1 + 5x_2 + 3x_3 + x_4$$

$$7x_1 + 4x_2 + 3x_3 + 2x_4 \leq 10, \quad x_i \in \mathbb{N}, i = 1, 2, 3, 4$$



# 分支限界在非凸优化的应用

- 凸优化几乎总有全局最优解的高效算法
- 对于一般的非凸问题，只能部分舍弃属性
- 局部优化方法通常很快，在不强求全局最优解时
  - ▶ （即使能得到全局最优解，也很难确认最优性）
- 全局优化方法能求解全局最优解，但需要较长或很长的求解时间



# 分支限界应用于非凸优化

- 非凸问题的全局优化方法
- 非启发式方法
  - ▶ 维护目标函数全局的可证明的下界和上界
  - ▶ 当能确认  $\epsilon$ -次优性时终止运行
- 通常非常慢，最坏情况指数时间
- 但是运气好的时候效果还不错

# 分支限界应用于非凸优化：基本思路

- 根据两组过程（有效地）计算在给定区域目标最优值的上界和下界
  - ▶ 求上界的方法：任取一个点求目标函数值、局部优化方法、.....
  - ▶ 求下界的方法：凸松弛（convex relaxation）、对偶性、Lipschitz界或其他下界、.....
- 基本思路
  - ▶ 将可行集划分成凸的子集，每个子集分别求上/下界
  - ▶ 形成全局的下界和上界
  - ▶ 两者足够接近时终止算法；或者细化划分并重复上述过程

# 分支限界应用于非凸优化：无约束的非凸最小化

- 目标：求解函数  $f: R^m \rightarrow R$  在  $m$ -维长方体区域  $Q_{init}$  的全局最小值
- 对于任意长方体区域  $Q \subseteq Q_{init}$ ，定义  $\Phi_{min}(Q) \subseteq \inf_{x \in Q} f(x)$
- 全局最优值为  $f^* = \Phi_{min}(Q_{init})$
- 通常只需求解使  $|f - f^*| \leq \varepsilon$  的  $f$  值

## 分支限界应用于非凸优化：下界和上界函数

- 记下界和上界函数分别为  $\Phi_{lb}$  和  $\Phi_{ub}$ ，对于任意长方体  $Q \subseteq Q_{init}$ ，满足

$$\Phi_{lb}(Q) \leq \Phi_{min}(Q) \leq \Phi_{ub}(Q)$$

- 假设当长方体区域收缩时，上下界的距离会变窄

$$\forall \varepsilon > 0, \exists \delta > 0, \forall Q \subseteq Q_{init}, \text{size}(Q) \leq \delta \implies \Phi_{ub}(Q) - \Phi_{lb}(Q) \leq \varepsilon$$

► 其中  $\text{size}(Q)$  是长方体最长边的长度

- 实践中，上下界函数  $\Phi_{lb}$  和  $\Phi_{ub}$  需要能快速计算

# 分支限界应用于非凸优化：分支限界算法

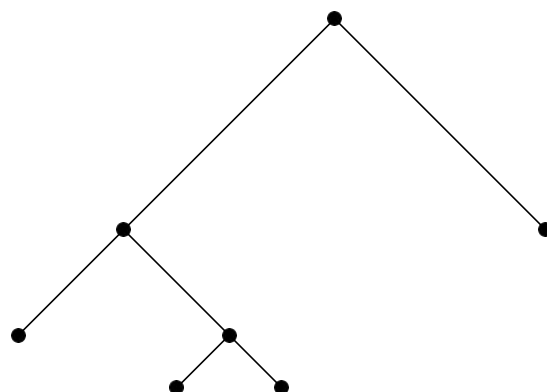
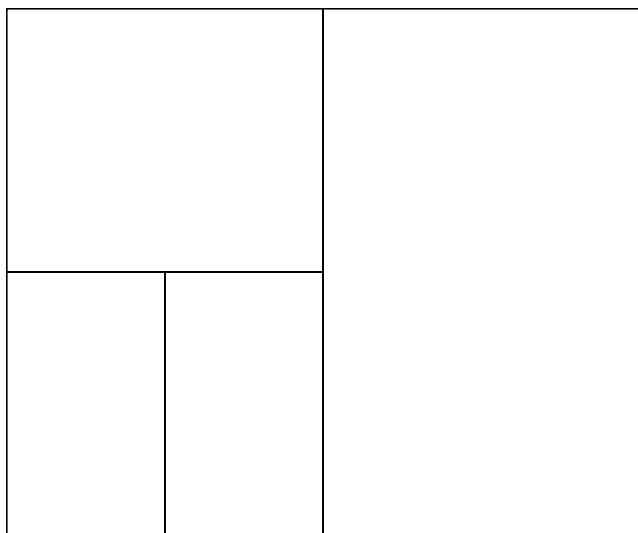
1. 计算  $f^*$  的下界和上界
  - ▶ 计算  $L_1 = \Phi_{lb}(Q_{init})$  和  $U_1 = \Phi_{ub}(Q_{init})$
  - ▶ 若  $U_1 - L_1 \leq \varepsilon$ , 算法终止
2. 将  $Q_{init}$  划分为两个子长方体  $Q_{init} = Q_1 \cup Q_2$
3. 分别计算  $\Phi_{lb}(Q_1), \Phi_{lb}(Q_2), \Phi_{ub}(Q_1), \Phi_{ub}(Q_2)$
4. 更新  $f^*$  的下界和上界
  - ▶ 更新下界:  $L_2 = \min\{ \Phi_{lb}(Q_1), \Phi_{lb}(Q_2) \}$
  - ▶ 更新上界:  $U_2 = \min\{ \Phi_{ub}(Q_1), \Phi_{ub}(Q_2) \}$
  - ▶ 若  $U_2 - L_2 \leq \varepsilon$ , 算法终止
5. 细化划分 (继续划分  $Q_1$  或  $Q_2$ ) , 重复第3步和第4步

# 分支限界应用于非凸优化：分支限界算法

- 不失一般性，可假设  $\{U_i\}$  是非上升序列， $\{L_i\}$  是非下降序列
- 每一步，等效于在扩展一棵二叉树；二叉树的父子节点对应长方体划分过程
- 二叉树的叶子节点，表示  $Q_{init}$  在当前划被分出的子区域
- 每一步需要以下划分规则
  - ▶ 划分哪个长方体？
  - ▶ 划分长方体的哪条边？（划分哪个变量？）
  - ▶ 在哪里划分？（划分边界的取值？）
- 好的划分规则：划分当前能取得最好下界的长方体、沿长边、对半划分、等等

## 分支限界应用于非凸优化：示意图

- 三次迭代后，被划分的矩形、以及对应的二叉树，如下图所示



# 分支限界应用于非凸优化：分支限界

- 在二叉树中  $\Phi_{lb}(Q) > U_2$  的长方体  $Q$  对应的节点，可以排除掉，或称为剪枝
  - ▶ 该长方体内每个点的目标函数值都比当前上界差
  - ▶ 该长方体内肯定不存在最优解
- 剪枝不增加算法实现难度，又能降低计算和存储需求
- 可通过以下指标评估算法的进展
  - ▶ 被剪掉区域的总体积
  - ▶ 被剪掉的二叉树节点数目



## 分支限界应用于非凸优化：收敛性分析

- 记第  $k$  次划分后，记此时的长方体集合为  $\mathcal{L}_k$ （包含被剪枝的长方体）
- 这些长方体的总体积为  $vol(Q_{init})$ ，有

$$\min_{Q \in \mathcal{L}_k} vol(Q) \leq vol(Q_{init})/k$$

- 当  $k$  足够大时，至少有一个长方体体积足够小
- 足够小的体积意味着足够小的边长、以及足够小的上下界距离  $U - L$
- 因此， $U_k - L_k$  将足够小

## 分支限界应用于非凸优化：条件数的范围估算

- m-维长方体  $Q = [l_1, u_1] \times \cdots \times [l_m, u_m]$  的条件数定义为

$$\text{cond}(Q) = \max_i (u_i - l_i) / \min_i (u_i - l_i)$$

- 如果在最长边二分长方体，对于任意长方体，都有

$$\text{cond}(Q) \leq \max\{\text{cond}(Q_{\text{init}}), 2\}$$

- 其他规则（例如按维度顺序轮流划分）也能保证  $\text{cond}(Q)$  的上界

## 分支限界应用于非凸优化：小体积意味着小边长

$$\begin{aligned} \text{vol}(Q) &= \prod_{i=1}^m (u_i - l_i) \geq \max_i (u_i - l_i) \left( \min_i (u_i - l_i) \right)^{m-1} \\ &= (2\text{size}(Q))^m / \text{cond}(Q)^{m-1} \geq (2\text{size}(Q) / \text{cond}(Q))^m \end{aligned}$$

- 则  $\text{size}(Q) \leq (1/2)\text{vol}(Q)^{1/m}\text{cond}(Q)$
- 因此，如果  $\text{cond}(Q)$  有界，当  $\text{vol}(Q)$  足够小时， $\text{size}(Q)$  也可以足够小

# 混合布尔凸优化

$$\begin{array}{ll} \text{minimize} & f_0(x, z) \\ \text{subject to} & f_i(x, z) \leq 0 \quad i = 1, \dots, m \\ & z_j \in \{0, 1\} \quad j = 1, \dots, n \end{array}$$

- $x \in R^p$  是连续变量
- $z \in \{0, 1\}^n$  是布尔变量
- $f_0, \dots, f_m$  对变量  $x$  和  $z$  是凸的
- 最优值记为  $p^*$
- 每组固定的  $z \in \{0, 1\}^n$ , 简化的问题 (对于变量  $x$ ) 是凸的

# 混合布尔凸优化：解决方法

## ► 暴力

- 穷举  $z \in \{0,1\}^n$  的  $2^n$  种可能的取值，每次再求解简化的连续凸优化问题
- 对于  $n \leq 15$  也许可行，但解决不了  $n \geq 20$  规模的问题

## ► 分支限界

- 最坏情况，也需要求解  $2^n$  次凸优化问题
- 寄希望于分支限界对待求解的问题实例能够较快地求解

# 混合布尔凸优化：通过凸松弛估算下界

## ► 松弛凸优化

$$\begin{aligned} & \text{minimize} && f_0(x, z) \\ & \text{subject to} && f_i(x, z) \leq 0 \quad i = 1, \dots, m \\ & && 0 \leq z_j \leq 1 \quad j = 1, \dots, n \end{aligned}$$

- 对于连续变量  $x$  和  $z$  是凸的，容易求解
- 松弛问题的最优值  $L_1$  是原问题最优值  $p^*$  的下界
- $L_1$  可以是  $+\infty$ （意味着原问题无可行解）
- 记松弛问题最优解的布尔变量部分为  $z^{(L_1)}$

# 混合布尔凸优化：上界的估算方法

- 估算  $p^*$  的上界  $U_1$  的几种方法
- 最简单的方法：对每个松弛的布尔变量  $z_i^{(L_1)}$ ，四舍五入到 0 或 1
- 稍复杂的方法：每个布尔变量取整后，重新对  $x$  求解相应的连续凸优化问题
- 随机算法
  - ▶ 以概率  $\text{Prob}(z_i = 1) = z_i^{(L_1)}$  随机生成布尔部分解  $z_i \in \{0,1\}$
  - ▶ （重新求解相应关于  $x$  的连续凸优化问题）
  - ▶ 在多个随机布尔部分解的结果里挑目标函数值最小的
- 上界可以是  $+\infty$ （意味着方法无法求出可行解）
- 如果  $U_1 - L_1 < \varepsilon$ ，算法终止

# 混合布尔凸优化：分支

- 挑一个变量下标  $k$ ，形成两个子问题

- 第一个子问题

$$\begin{array}{ll} \text{minimize} & f_0(x, z) \\ \text{subject to} & f_i(x, z) \leq 0 \quad i = 1, \dots, m \\ & z_j \in \{0, 1\} \quad j = 1, \dots, n \\ & z_k = 0 \end{array}$$

- 第二个子问题

$$\begin{array}{ll} \text{minimize} & f_0(x, z) \\ \text{subject to} & f_i(x, z) \leq 0 \quad i = 1, \dots, m \\ & z_j \in \{0, 1\} \quad j = 1, \dots, n \\ & z_k = 1 \end{array}$$



## 混合布尔凸优化：分支（接上页）

- 每个都是  $n-1$  个布尔变量的混合布尔凸优化问题
- 原问题的最优值是子问题两个最优值中的最小者
- 能够通过求解松弛凸优化子问题来估算最优值的下界和上界

## 利用子问题的上下界，更新原问题的上下界

- 记  $\bar{U}$  和  $\bar{L}$  为  $z_k = 0$  时的上下界
- 记  $\hat{U}$  和  $\hat{L}$  为  $z_k = 1$  时的上下界
- $\min \{\bar{L}, \hat{L}\} \geq L_1$
- 不失一般性，假设  $\min \{\bar{U}, \hat{U}\} \leq U_1$
- 因此，我们得到  $p^*$  的新上下界

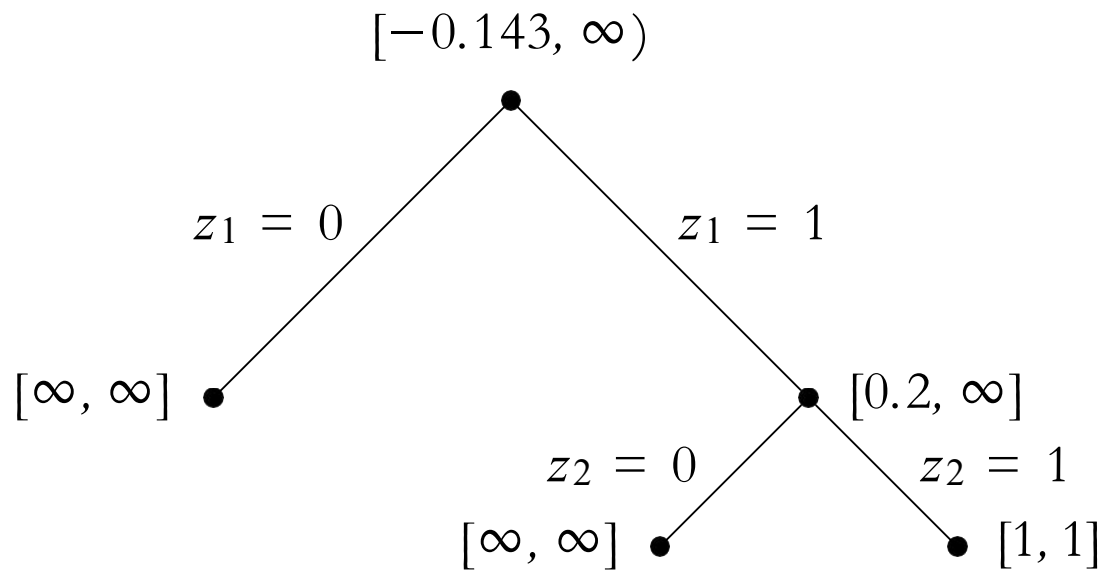
$$L_2 = \min \{\bar{L}, \hat{L}\} \leq p^* \leq U_2 = \min \{\bar{U}, \hat{U}\}$$

# 混合布尔凸优化：分支限界算法

- 通过变量划分、松弛、和子问题上下界的估算，持续（隐式地）扩展二叉树
- 收敛性证明是显然的，在  $U = L$  前不会超过  $2^n$  步
- 能对  $L$  超出当前步  $U_k$  的节点剪枝
- 通常的策略是划分拥有最小  $L$  值的节点（长方体）
- 长方体的划分维度选择
  - ▶ 最“旗帜鲜明”的：选满足  $z_k^{(L)} = 0$  或  $1$ （且对应的拉格朗日乘子最大）的维度  $k$
  - ▶ 最“模棱两可”的：选最小化  $|z_k^{(L)} - 1/2|$  的维度  $k$

# 混合布尔凸优化：小例子

- 节点记录了某个3变量布尔线性规划问题实例的上界和下界



# 混合布尔凸优化的应用：最小基数问题

- 求满足线性不等式的最稀疏（非零元素数目最少）的  $x$

$$\begin{array}{ll} \text{minimize} & \mathbf{card}(x) \\ \text{subject to} & Ax \leq b \end{array}$$

- 等价于求解混合布尔线性规划问题

$$\begin{array}{ll} \text{minimize} & \mathbf{1}^T z \\ \text{subject to} & L_i z_i \leq x_i \leq U_i z_i \quad i = 1, \dots, n \\ & Ax \leq b \\ & z_i \in \{0, 1\} \quad i = 1, \dots, n \end{array}$$

- 其中常数  $U_i$  和  $L_i$  是  $x_i$  的上下界（见下页）

# 最小基数问题：变量上下界的计算

- 变量  $x_i$  下界  $L_i$  是以下线性规划的最优值

$$\begin{array}{ll}\text{minimize} & x_i \\ \text{subject to} & Ax \leq b\end{array}$$

- 变量  $x_i$  上界  $U_i$  是以下线性规划的最优值

$$\begin{array}{ll}\text{maximize} & x_i \\ \text{subject to} & Ax \leq b\end{array}$$

- 求解  $2n$  个线性规划问题，得到所有变量的上下界
- 如果  $L_i > 0$  或  $U_i < 0$ ，置  $z_i = 1$

# 最小基数问题：松弛问题

## ► 松弛问题

$$\begin{aligned}
 & \underset{x, z}{\text{minimize}} && \mathbf{1}^T z \\
 & \text{subject to} && L_i z_i \leq x_i \leq U_i z_i \quad i = 1, \dots, n \\
 & && Ax \leq b \\
 & && 0 \leq z_i \leq 1 \quad i = 1, \dots, n
 \end{aligned}$$

## ► （假设 $L_i < 0$ 且 $U_i > 0$ ）松弛问题等价于

$$\begin{aligned}
 & \underset{x, z}{\text{minimize}} && \sum_{i=1}^n ((1/U_i)(x_i)_+ + (-1/L_i)(x_i)_-) \\
 & \text{subject to} && Ax \leq b
 \end{aligned}$$

## ► 目标函数是非对称的加权 $\ell_1$ 范数

（细节请参考 Stanford EE364b: “L1 methods for convex-cardinality problems”; 略）

## 最小基数问题：更多细节

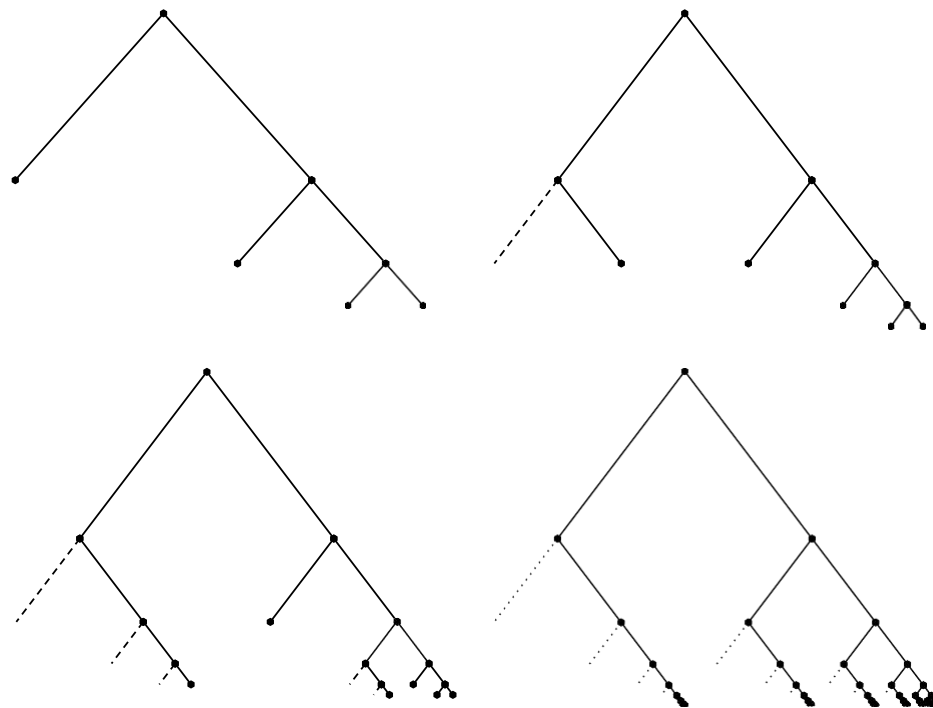
- 松弛问题在凸集上最小化  $\ell_1$  范数，是求解稀疏问题常用的启发式方法
- 取松弛问题的最优值  $\tilde{y}$  作为原问题最优值的下界
  - ▶ 该下界可进一步增强至  $\lceil \tilde{y} \rceil$
  - ▶ 例如，若知道原问题最优值  $\text{card}(x^*) \geq \tilde{y} = 23.4$ ，易得  $\text{card}(x^*) \geq 24$
- 取松弛问题的最优解  $\tilde{x}$  部分的  $\text{card}(\tilde{x})$  作为原问题最优值的上界
- 每次迭代，划分拥有最小下界的长方体，沿最模棱两可的分量的维度



# 最小基数问题：小例子

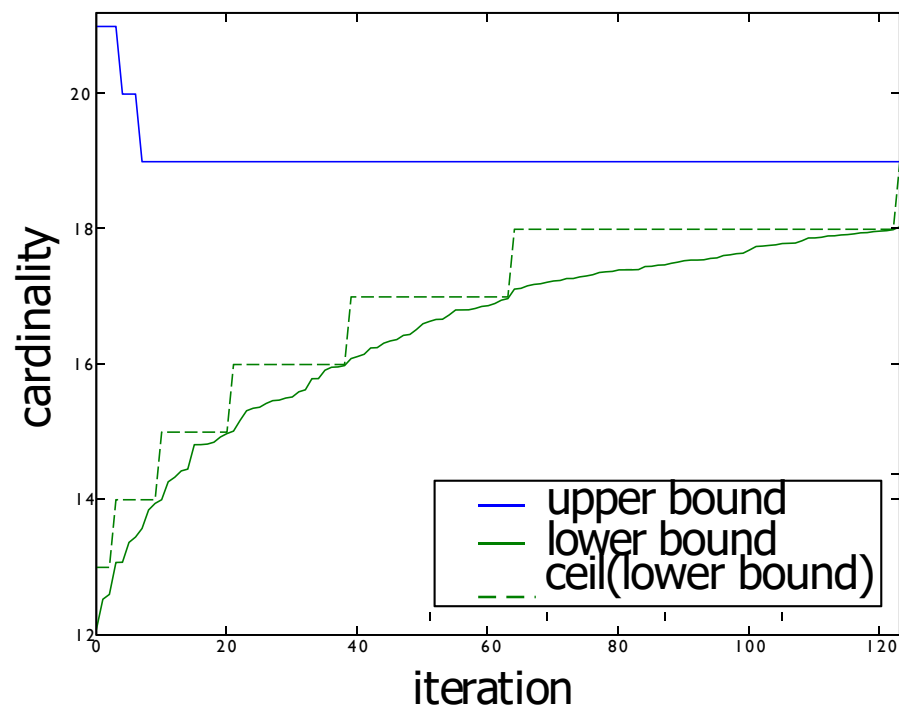
- 随机实例，30个布尔变量，100个约束
- 解空间大小：  $2^{30} \approx 10^9$
- 8次迭代即可求得全局最小基数（19）
- 但需要花124迭代证明19是最优解
- 调用 309 次线性规划求解器（包括60次计算每个变量的上下界）

右上角：5次迭代后

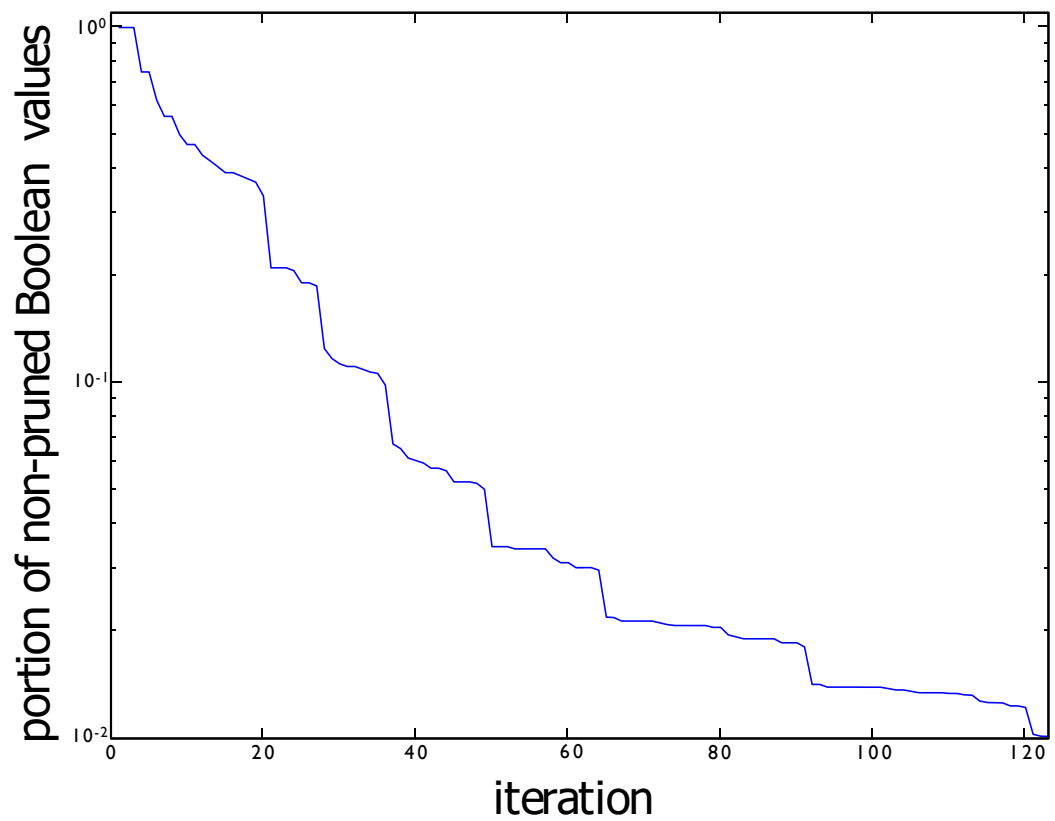


右下角：124次迭代后

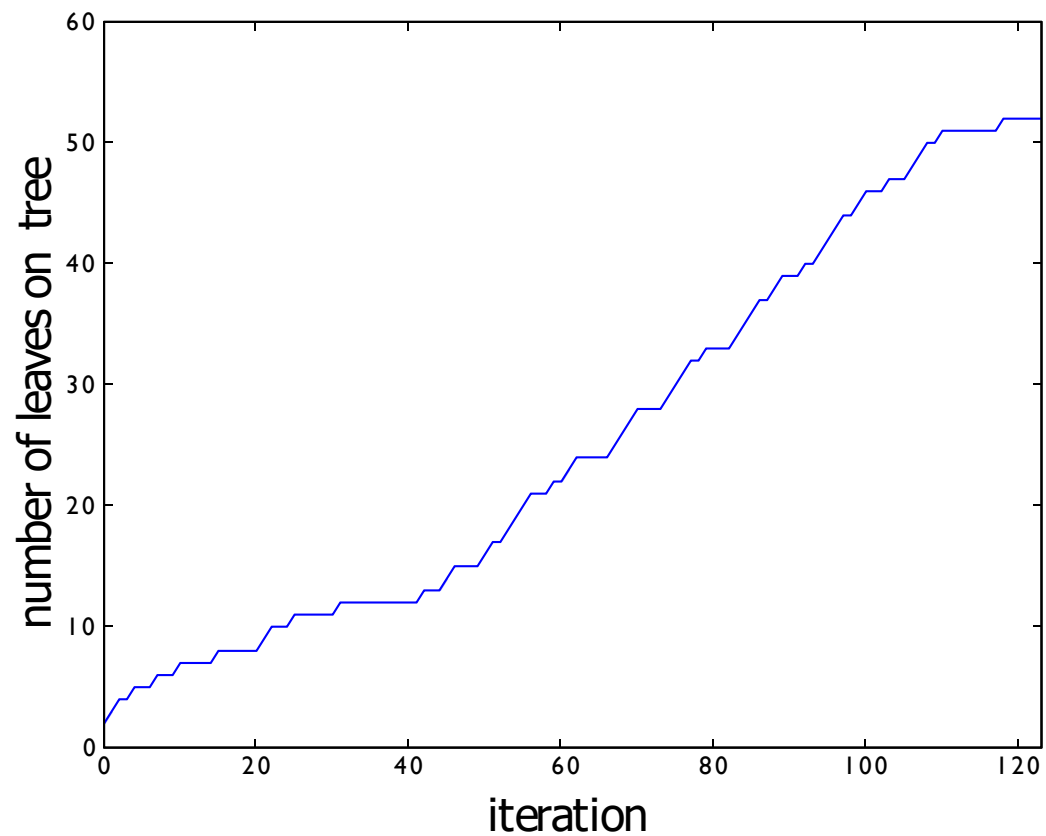
# 最小基数问题：全局的上界和下界（小例子）



# 最小基数问题：未被剪枝候选解的比例（小例子）



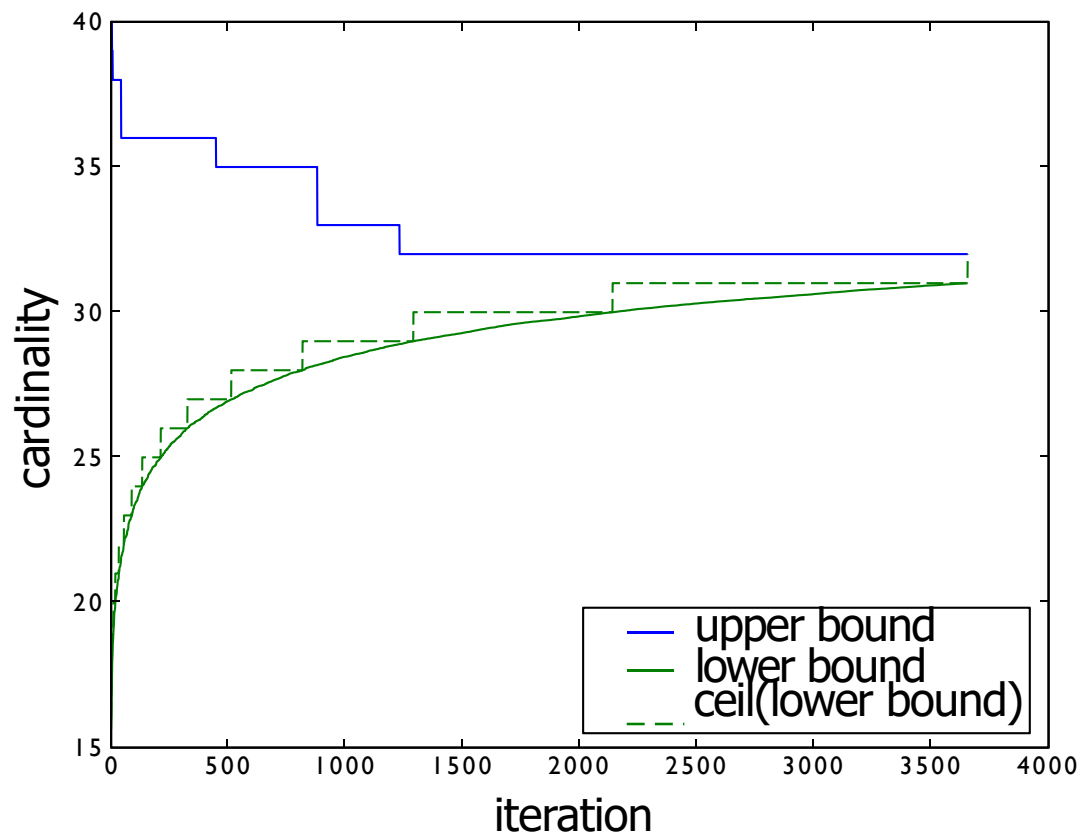
# 最小基数问题：二叉树的活跃叶子数目（小例子）



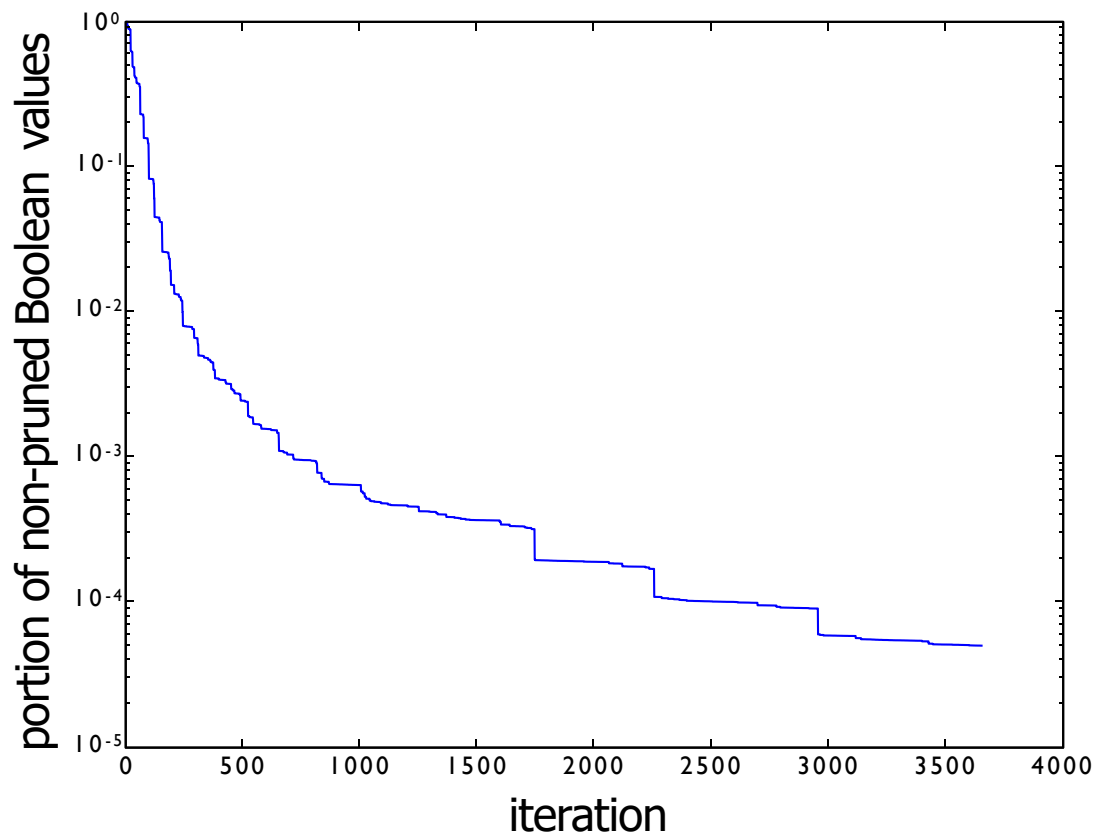
# 最小基数问题：大例子

- 随机实例，50个布尔变量，100个约束
- 解空间大小：  $2^{50} \approx 10^{15}$
- 消耗 3665 次迭代（第1300次即找到最优解）
- 最小基数 31

# 最小基数问题：全局的上界和下界（大例子）

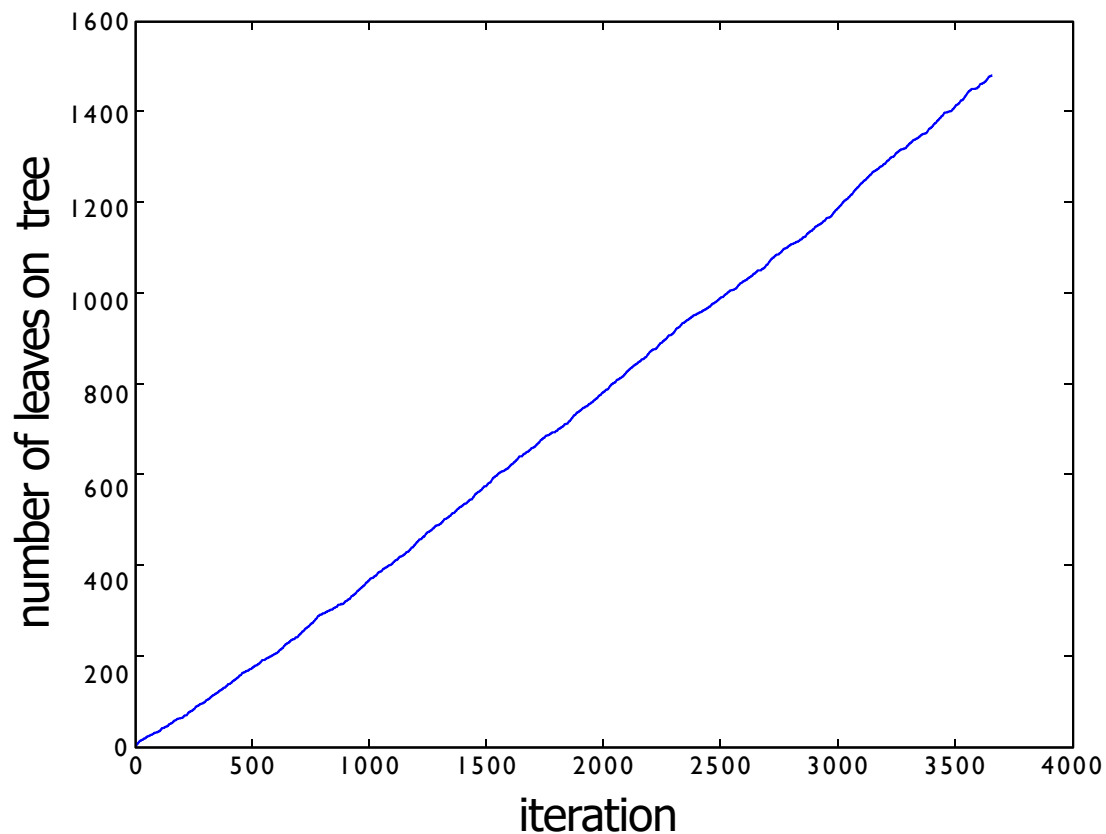


# 最小基数问题：未被剪枝候选解的比例（大例子）





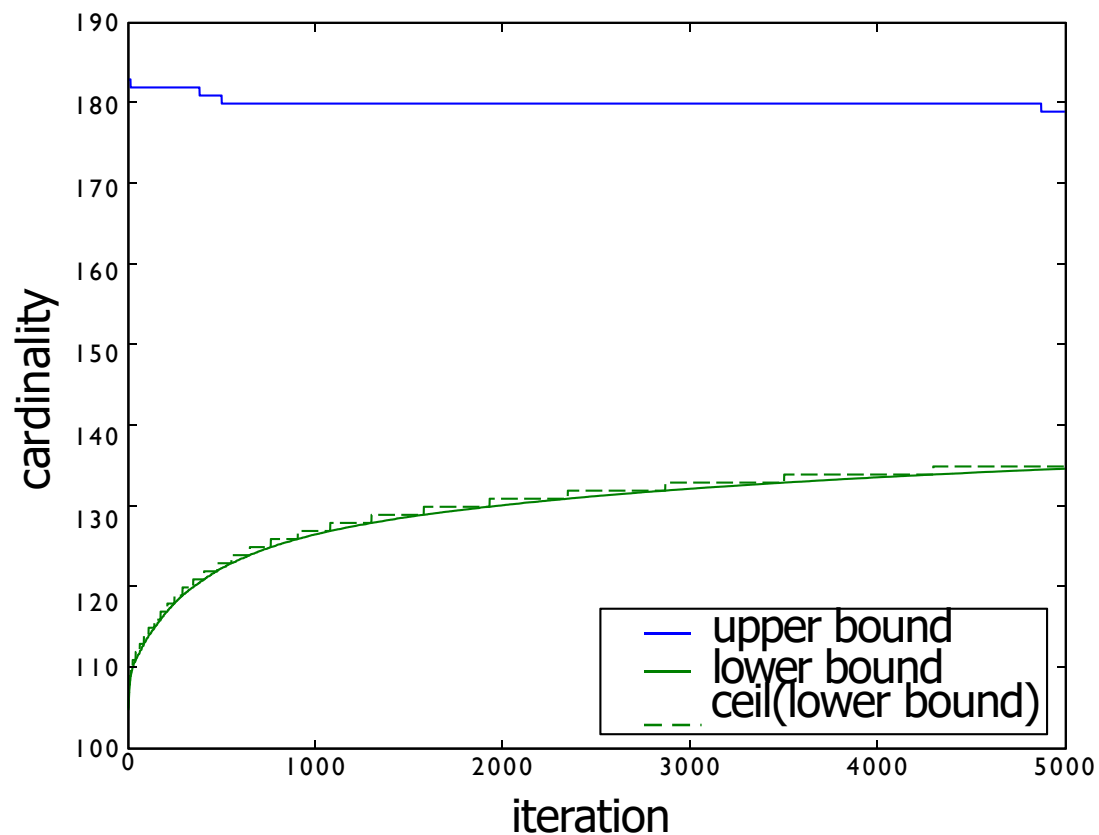
# 最小基数问题：二叉树的活跃叶子数目（大例子）



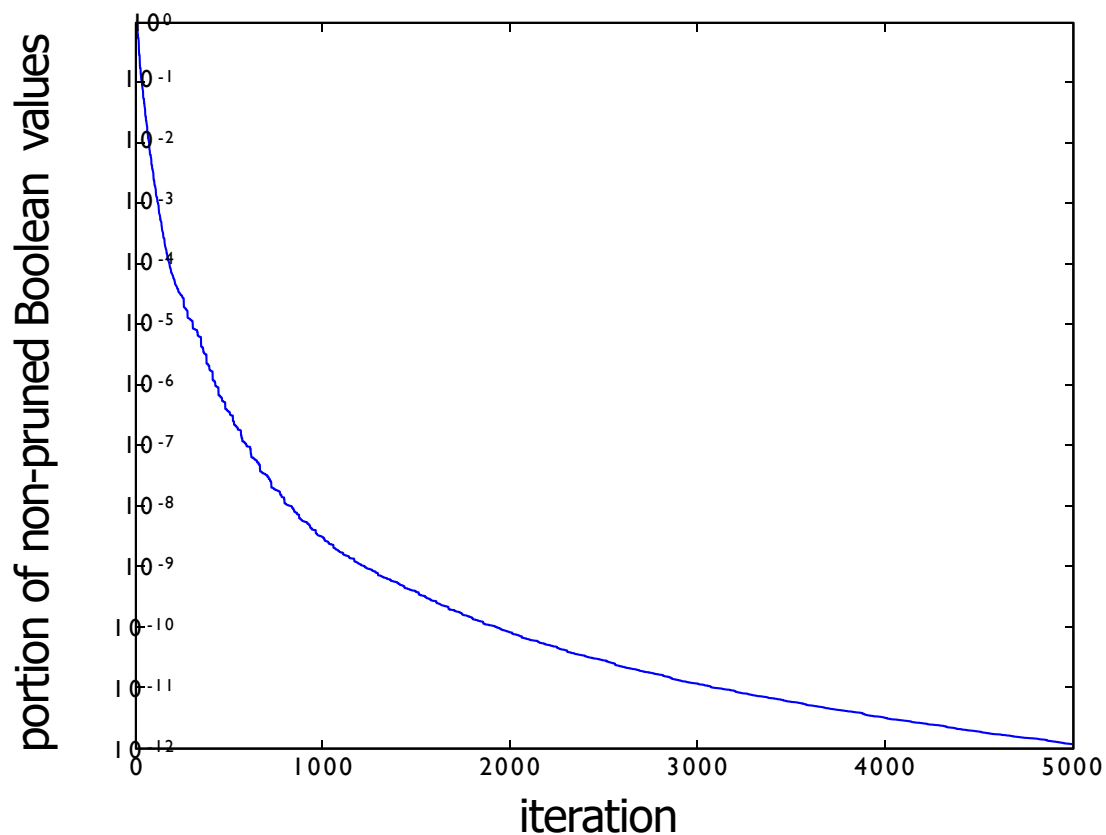
## 最小基数问题：超级大例子

- 随机实例，200个布尔变量，400个约束
- 解空间大小： $2^{200} \approx 1.6 \times 10^{60}$
- 运行 10000 次迭代后终止（消耗50小时，带1GB内存的单处理器上）
- 只能知道最优基数在 135 至 179 之间
- 但是能够将候选解的数目降低至原来的  $1/10^{12}$

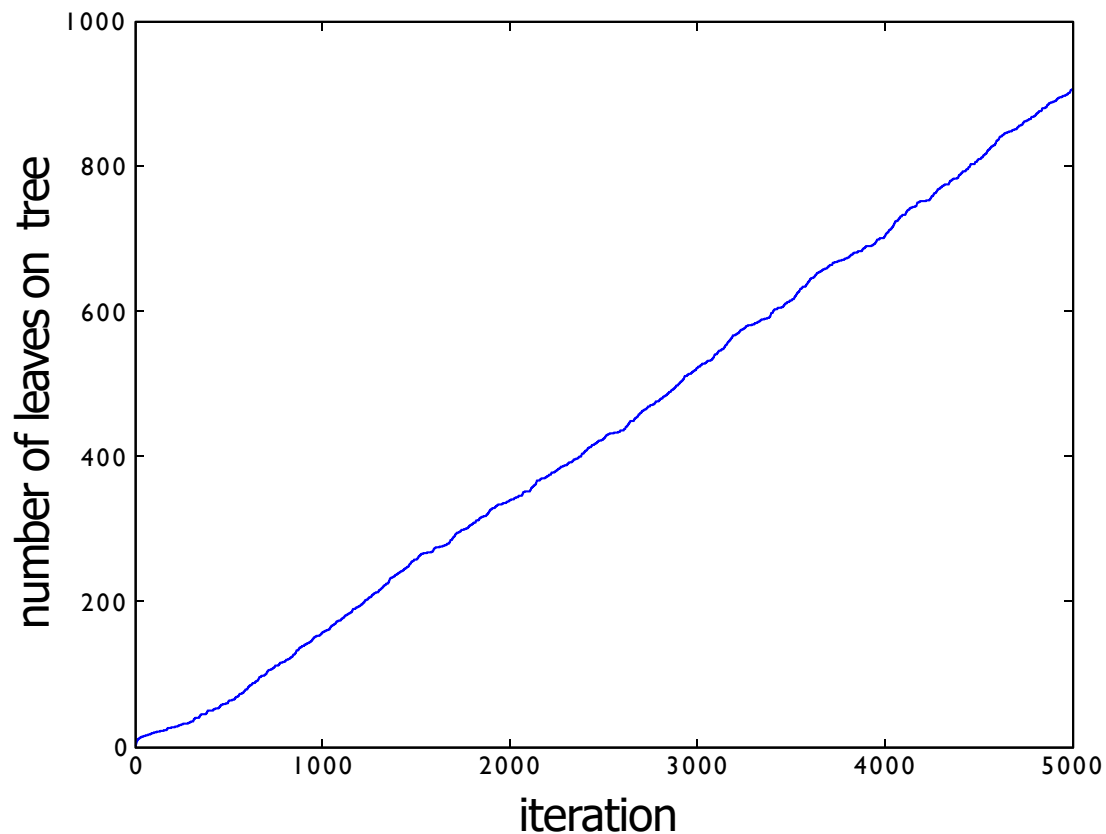
# 最小基数问题：全局的上界和下界（超级大例子）



# 最小基数问题：未被剪枝候选解比例（超大例子）



# 最小基数问题：二叉树活跃叶子数目（超大例子）



## 本节小结

- ➡ 回溯与剪枝 backtrack with pruning
  - ▶ 多米诺性质
- ➡ 分支限界 Branch-and-bound
  - ▶ 全局优化的分支限界例子
  - ▶ 解更大的问题?—更快地解小问题
- ➡ 分支限界应用于非凸优化
  - ▶ 通用的算法和收敛性
  - ▶ 非凸优化例子: 混合布尔凸规划
  - ▶ 混合布尔凸规划例子: 最小基数问题