算法设计与分析

回溯法（二）

1. 分支限界法介绍
2. 背包问题
3. 最大团问题（MCP）
4. 旅行商问题

目录

组合优化

组合优化：从有限的可行/候选解集合 S 中找到最优解 x。

- 约束条件：$P(x) = 1 \Leftrightarrow x \in S$
- 优化函数 $f(x) \rightarrow$ 定义最优解
  典型的优化函数目标是最大化或最小化

背包问题示例

- $P(x) : 2x_1 + 3x_2 + 4x_3 + 7x_4 \leq 10, x_i \in \mathbb{N}, i \in [4]$
- $f(x) : \max\{x_1 + 3x_2 + 5x_3 + 9x_4\}$

不失一般性，总是假设目标是最大化 $f(x)$

- 因为可以通过求 $g(x) = -f(x)$ 的最大值来找到 $f(x)$ 的最小值

在运筹学、应用数学和理论计算机科学中被广泛研究。

- 旅行商问题（TSP）、最小生成树问题（MST）、背包问题

动机

一般而言，组合优化问题可以通过枚举候选解并逐一测试来求解

- 枚举可以通过暴力搜索状态空间树来完成

$$\text{叶节点} \Leftrightarrow \text{候选解}$$

对于 $\mathcal{NP}$-难问题，状态空间可能是指数级大的。

我们能否改进状态空间树的暴力搜索性能?

分支限界法

[见原文第 5 页图片]
Figure: 1960 年，英国：Ailsa Land 和 Alison Harcourt

求解 $\mathcal{NP}$-难优化问题最常用的技术

[见原文第 5 页剪枝示意图]

分支限界法的关键要素

分支限界算法基于两个原则运作：

- 分支：递归地将搜索空间划分为更小的空间，然后尝试在这些更小的空间中找到最大的 $f(x)$
- 限界：维护一个界值，计算较小空间中 $f(x)$ 的上界，并使用这个界值和上界来" 剪枝" 搜索空间，排除不包含最优解的候选解

良好" 剪枝" 的关键点

- 界值的设置（通常比较自然）
- 上界的计算（相对棘手）

界值

含义：当前可行解的最大优化函数值
初始值：最大化问题为 0，最小化问题为 ∞
更新

- 找到第一个解时
- 找到更好的解时

[见原文第 7 页图]
$B = f(x)$
一个可行解 x

估计/界函数

输入：树的节点，设为 v
输出：以输入节点为根的子树中所有可行解的最大值的上界
可靠性：设 v' 是 v 的子节点

$$E(v) \geq E(v')$$

[见原文第 8 页图]
一个可行解 x

可行解集合缩小
$\Rightarrow$ 上界变小
可靠性 $\Rightarrow$ 不会遗漏解

估计函数的选择不是唯一的，需要在计算成本与准确性之间做权衡

界函数的比喻（代表最乐观的估计）

[见原文第 9 页图片]

回溯与剪枝

当导航到节点 v 时，如果出现以下两种情况之一，算法将停止分支并回溯到父节点：

1. 可行解集合 $A(v)$ 为空
   - 子树中没有叶节点满足约束谓词（与使用默认约束的朴素回溯相同）
2. 估计函数值小于当前界值

$$E(v) < B$$

   - 直接剪掉该子树并回溯

目录

可重复背包问题示例

背包问题实例（重量限制 $W = 10$）

标签	重量	价值
1	2	1
2	3	3
3	4	5
4	7	9

约束谓词 P:

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

优化函数 f: $\max\{x_1 + 3x_2 + 5x_3 + 9x_4\}$

界函数的选择

对于每个节点 $v = (x_1, \ldots, x_k)$，计算子树中可行解的优化函数值的上界

- 预处理：按 v_i/w_i 降序排列，$i \in [n]$

$$\underline{E(v) = K(v) + \Delta(v)}$$

$K(v)$：背包中已装载的价值
$\Delta(v)$：可以进一步装载的最大价值

$\Delta(v = (x_1, \ldots, x_k))$ 的计算：

- $\Delta(v) =$ 剩余重量 $\times\ v_{k+1}/w_{k+1}$

显然，存在更精细的 $E(v)$ 计算方法，例如，找到第一个可装载物品，然后相应地计算 $E(v)$。

背包实例

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

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

重新排列标签使得

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

重排后

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

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

分支限界示例

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

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

[见原文第 15 页图]

估计值　　分支
当前重量　由支配性质回溯
　　更新界值
　　限界并剪枝

分支限界示例

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

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

[见原文第 16 页图]

$\frac{10 \cdot 9}{7}$

0

分支限界示例

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

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

[见原文第 17 页图]

$\frac{10 \cdot 9}{7}$, 0

$9 + \frac{3 \cdot 5}{4}$, 7, 1

分支限界示例

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

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

[见原文第 18 页图]

分支限界示例

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

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

[见原文第 19 页图]

分支限界示例

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

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

[见原文第 20 页图]

分支限界示例

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

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

[见原文第 21 页图]

分支限界示例

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

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

[见原文第 22 页图]

分支限界示例

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

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

[见原文第 23 页图]

思考与总结

问：预处理步骤是必要的吗?
答：不是。但它可以加速界函数的计算。

问：界函数还有其他可能的选择吗?
答：有。但要确保它易于计算，在剪枝的成本和收益之间取得良好平衡

分支限界法 $\rightsquigarrow$ 组合优化

- 界值的设置与更新
- 估计函数（代表乐观估计）$\Rightarrow$ 保证剪枝不会遗漏解
- 剪枝：比较界值和估计函数值

目录

图的概念

设 $G = (V, E)$ 是一个无向图
子图：$G' = (V', E')$，其中 $V' \subseteq V$，$E' \subseteq E$
补图：$\overline{G} = (V, \overline{E})$，其中 $\overline{E}$ 是 E 关于 V 上完全图的补集
团：G 的一个完全子图
最大团：具有最多顶点数的团。

- MCP 是图论中的经典组合优化问题。

[见原文第 26 页图]

最大团 $= \{1, 3, 4, 5\}$

独立集与团

设 $G = (V, E)$ 是一个无向图
独立集：V 的子集 U，使得 $\forall u, v \in U, (u, v) \notin E$
最大独立集：G 的最大可能大小的独立集。

命题：U 是 G 的最大团当且仅当 U 是 $\overline{G}$ 的最大独立集。

独立集 $\leftrightarrow$ 补图中的团

[见原文第 27 页图]

$\{1, 3, 6\}$
G 的最大团
$\overline{G}$ 的最大独立集

最大团的应用

MCP 的众多应用

- 编码、聚类分析、计算机视觉、经济学、移动通信、VLSI 设计

编码示例：信道中的噪声可能干扰码字传输。考虑混淆图 $G = (V, E)$，V 是有限符号集

$$(u, v) \in E \text{ 或} E(u, v) = 1 \Leftrightarrow u \text{ 和} v \text{ 可能被混淆}$$

[见原文第 28 页图]

编码设计

在编码设计中，我们通常用字符串来编码符号。
码字混淆：我们说两个字符串 xy 和 uv 可能被混淆当且仅当

$$(E(x,u)=1 \wedge E(y,v)=1) \vee$$

$$(x=u \wedge E(y,v)=1) \vee (E(x,u)=1 \wedge y=v)$$

[见原文第 29 页图：G, H 和 $G \times H$]

$G \times H$ 中的顶点是候选码字

- 如果两个码字之间有边，则它们会被混淆

为了减少噪声干扰，我们需要在 $G \times H$ 中找到 MIS（最大独立集）。

最大团问题

问题：给定无向图 $G=(V,E)$，其中 $V=\{1,\ldots,n\}$，找到其最大团。
解：一个 n 维向量 $(x_1,x_2,\ldots,x_n)\in\{0,1\}^n$，$x_k=1$ 当且仅当 k 在 G 的最大团中。
暴力算法：对于 V 的每个子集，检查它是否构成团，即完全子图。

$$\#V \text{ 的子集为} 2^n \rightsquigarrow \text{指数时间复杂度} O(n^2)\cdot 2^n$$

[见原文第 30 页图]

分支限界法

搜索树：子集树（二叉树：从叶节点到根的路径确定一个子集）
节点 $(x_1, x_2, \ldots, x_k)$：已检查节点 $1, 2, \ldots, k$，$x_i = 1$ 表示 i 属于当前团，$i \in [k]$
约束：$x_{k+1} = 1$ 当且仅当它与当前团中的所有节点都相连
界值：当前最大团中的顶点数
估计函数：当前团可能扩展到的最大顶点数：

$$E(v) = C(v) + (n - k)$$

- $C(v)$：当前团中的顶点数（初始值为 0）
- k：v 的深度

E 简单但过于粗糙 $\rightsquigarrow$ 最坏情况复杂度为 $O(n2^n)$，与暴力算法渐近相同

搜索演示（完美二叉树）

[见原文第 32 页图]
图 G：顶点 1,2,3,4,5

搜索演示（完美二叉树）

初始值 $B = 0$

[见原文第 32 页图]
图 G：顶点 1,2,3,4,5

搜索演示（完美二叉树）

初始值 $B = 0$
搜索节点 1

[见原文第 32 页图]
图 G：顶点 1,2,3,4,5

搜索演示（完美二叉树）

初始值 $B = 0$
搜索节点 1
搜索节点 2

[见原文第 32 页图]
图 G：顶点 1,2,3,4,5

搜索演示（完美二叉树）

初始值 $B = 0$
搜索节点 1
搜索节点 2
节点 3: $\perp$（与当前团不全连接）

[见原文第 32 页图]
图 G: 顶点 1,2,3,4,5

搜索演示（完美二叉树）

初始值 $B = 0$
搜索节点 1
搜索节点 2
节点 3: $\perp$（与当前团不全连接）
节点 4, 5: $B = 3$, MC $= \{1, 2, 4\}$

[见原文第 32 页图]
图 G: 顶点 1,2,3,4,5

搜索演示（完美二叉树）

[见原文第 32 页图]
图 G：顶点 1,2,3,4,5

初始值 $B = 0$
搜索节点 1
搜索节点 2
节点 3: $\perp$（与当前团不全连接）
节点 4, 5: $B = 3$, MC $= \{1, 2, 4\}$
$E = 3 \leq B$，回溯

搜索演示（完美二叉树）

[见原文第 32 页图]
图 G：顶点 1,2,3,4,5

初始值 $B = 0$
搜索节点 1
搜索节点 2
节点 3: $\perp$（与当前团不全连接）
节点 4, 5: $B = 3$, MC $= \{1, 2, 4\}$
$E = 3 \leq B$，回溯
继续搜索...

搜索演示（完美二叉树）

[见原文第 32 页图]
图 G: 顶点 1,2,3,4,5

初始值 $B = 0$
搜索节点 1
搜索节点 2
节点 3: $\perp$（与当前团不全连接）
节点 4, 5: $B = 3$, MC $= \{1, 2, 4\}$
$E = 3 \leq B$，回溯
继续搜索...
搜索节点 3, 4, 5: $B = 4$, MC $= \{1, 3, 4, 5\}$

搜索演示（完美二叉树）

[见原文第 32 页图]
图 G: 顶点 1,2,3,4,5

初始值 $B = 0$
搜索节点 1
搜索节点 2
节点 3: $\perp$（与当前团不全连接）
节点 4, 5: $B = 3$, MC $= \{1, 2, 4\}$
$E = 3 \leq B$，回溯
继续搜索...
搜索节点 3, 4, 5: $B = 4$, MC $= \{1, 3, 4, 5\}$
$E = 3 < B$，回溯

搜索演示（完美二叉树）

[见原文第 32 页图]
图 G: 顶点 1,2,3,4,5

初始值 $B = 0$
搜索节点 1
搜索节点 2
节点 3: $\perp$（与当前团不全连接）
节点 4, 5: $B = 3$, MC $= \{1, 2, 4\}$
$E = 3 \leq B$，回溯
继续搜索...
搜索节点 3, 4, 5: $B = 4$, MC $= \{1, 3, 4, 5\}$
$E = 3 < B$，回溯
$E = 4 \leq B$，回溯

搜索演示（完美二叉树）

[见原文第 32 页图]
图 G：顶点 1,2,3,4,5

初始值 $B = 0$
搜索节点 1
搜索节点 2
节点 3: $\perp$（与当前团不全连接）
节点 4, 5: $B = 3$, MC $= \{1, 2, 4\}$
$E = 3 \leq B$，回溯
继续搜索...
搜索节点 3, 4, 5: $B = 4$, MC $= \{1, 3, 4, 5\}$
$E = 3 < B$，回溯
$E = 4 \leq B$，回溯
最终：MC $= \{1, 3, 4, 5\}$

目录

旅行商问题（TSP）

问题：给定 n 个城市以及每对城市之间的距离，访问每个城市并返回起始城市的最短可能路线是什么?

[见原文第 45 页图]

c_1, c_2, c_3, c_4 及边权重

建模

输入：有限城市集合 $C = \{c_1, c_2, \ldots, c_n\}$，距离 $e(c_i, c_j) = e(c_j, c_i) \in \mathbb{Z}^+$，$1 \leq i < j \leq n$。
解：$(1, 2, \ldots, n)$ 的一个排列 — $(i_1, i_2, \ldots, i_n)$ 使得：

$$\min \left\{ \sum_{i=1}^{n} e(c_{k_i \mod n}, c_{k_{i+1} \mod n}) \right\}$$

状态空间：排列树，节点 $(i_1, i_2, \ldots, i_k)$ 表示已走 k 步的路线
约束：设 $S = \{i_1, i_2, \ldots, i_k\}$，则 $i_{k+1} \in V - S$，因为每个节点只能被访问一次。

界值与估计函数

界值：当前最短路线的长度
估计函数：设与 c_i 相连的最短边长度为 ℓ_i，d_j 是当前路线中第 j 条边的长度

$$E([i_1, \ldots, i_k]) = \sum_{j=1}^{k-1} d_j + \ell_{i_k} + \sum_{i_j \notin S} \ell_{i_j}$$

- 第一部分：已走路线的长度
- 第二部分：剩余路线的下界

界函数示例

[见原文第 48 页图]
顶点 1,2,3,4 及边权重

$$E([i_1,\ldots,i_k]) = \sum_{j=1}^{k-1} d_j + \ell_{i_k} + \sum_{i_j \in V-S} \ell_{i_j}$$

部分路线：$(1,3,2)$，$E([1,3,2]) = (9+13)+2+2 = 26$，$S = \{1,3,2\}$，$V-S = \{4\}$

- $9+13$：已走路线的长度
- 2：与节点 2 相连的最短边长度
- 2：与节点 4 相连的最短边长度

搜索演示

[见原文第 49-56 页图]

搜索演示

[见原文第 49-56 页图]

从节点 1 开始

搜索演示

[见原文第 49-56 页图]

从节点 1 开始
→ 节点 2

搜索演示

[见原文第 49-56 页图]

从节点 1 开始
$\rightarrow$ 节点 2
$\rightarrow$ 节点 3 $\rightarrow$ 节点 4：$B = 29$

搜索演示

[见原文第 49-56 页图]

从节点 1 开始
$\rightarrow$ 节点 2
$\rightarrow$ 节点 3 $\rightarrow$ 节点 4：$B = 29$
$\rightarrow$ 节点 4 $\rightarrow$ 节点 3：$B = 23$

搜索演示

[见原文第 49-56 页图]

从节点 1 开始
$\rightarrow$ 节点 2
$\rightarrow$ 节点 3 $\rightarrow$ 节点 4：$B = 29$
$\rightarrow$ 节点 4 $\rightarrow$ 节点 3：$B = 23$
$\rightarrow$ 节点 3 $\rightarrow$ 节点 2：$E = 26$

搜索演示

[见原文第 49-56 页图]

从节点 1 开始
$\rightarrow$ 节点 2
$\rightarrow$ 节点 3 $\rightarrow$ 节点 4：$B = 29$
$\rightarrow$ 节点 4 $\rightarrow$ 节点 3：$B = 23$
$\rightarrow$ 节点 3 $\rightarrow$ 节点 2：$E = 26$
$\rightarrow$ 节点 4 $\rightarrow$ 节点 2

搜索演示

[见原文第 49-56 页图]

从节点 1 开始
$\rightarrow$ 节点 2
$\rightarrow$ 节点 3 $\rightarrow$ 节点 4：$B = 29$
$\rightarrow$ 节点 4 $\rightarrow$ 节点 3：$B = 23$
$\rightarrow$ 节点 3 $\rightarrow$ 节点 2：$E = 26$
$\rightarrow$ 节点 4 $\rightarrow$ 节点 2
继续搜索所有分支...

复杂度分析（1/2）

叶节点数：$(n-1)!$

- 每个叶节点对应一条路线
- 每条路线（实际上是一个环）有 n 个城市 $\rightsquigarrow$ 循环移位等价 $\rightsquigarrow$ 最多 $(n-1)!$ 条不同路线

[见原文第 57 页图]

复杂度分析（1/2）续

叶节点数：$(n-1)!$

- 每个叶节点对应一条路线
- 每条路线（实际上是一个环）有 n 个城市 $\rightsquigarrow$ 循环移位等价 $\rightsquigarrow$ 最多 $(n-1)!$ 条不同路线

进一步观察：解是无向图中的环 $\rightsquigarrow$ 顺时针和逆时针对称 $\rightsquigarrow$ 最多 $(n-1)!/2$ 条不同路线（两种等价不重叠）

[见原文第 58 页图]

复杂度分析（2/2）

$E(\cdot)$ 的复杂度为 $O(1)$ $\rightsquigarrow$ 遍历每条路线需要 $O(n)$

$$E(i_1, \ldots, i_k) = \sum_{j=1}^{k-1} d_j + \ell_{i_k} + \sum_{i_j \in V-S} \ell_{i_j}$$

复杂度分析（2/2）续

$E(\cdot)$ 的复杂度为 $O(1) \rightsquigarrow$ 遍历每条路线需要 $O(n)$

$$E(i_1, \ldots, i_k) = \sum_{j=1}^{k-1} d_j + \ell_{i_k} + \sum_{i_j \in V-S} \ell_{i_j}$$

- 移动到子节点 i_{k+1} 时更新（加上 $d_k - \ell_{i_k}$），其中 $d_k = e(i_k, i_{k+1})$

$$E(i_1, \ldots, i_k, i_{k+1}) = \sum_{j=1}^{k} d_j + \ell_{i_{k+1}} + \sum_{i_j \in V-(S+i_{k+1})} \ell_{i_j}$$

$$= \sum_{j=1}^{k} d_j + \sum_{i_j \in V-S} \ell_{i_j}$$

- 初始值为 $E([i_1]) = \sum_{j=1}^{k} \ell_{i_j}$

复杂度分析（2/2）续

$E(\cdot)$ 的复杂度为 $O(1)$ $\rightsquigarrow$ 遍历每条路线需要 $O(n)$

$$E(i_1, \ldots, i_k) = \sum_{j=1}^{k-1} d_j + \ell_{i_k} + \sum_{i_j \in V-S} \ell_{i_j}$$

- 移动到子节点 i_{k+1} 时更新（加上 $d_k - \ell_{i_k}$），其中 $d_k = e(i_k, i_{k+1})$

$$E(i_1, \ldots, i_k, i_{k+1}) = \sum_{j=1}^{k} d_j + \ell_{i_{k+1}} + \sum_{i_j \in V-(S+i_{k+1})} \ell_{i_j} = \sum_{j=1}^{k} d_j + \sum_{i_j \in V-S} \ell_{i_j}$$

- 初始值为 $E([i_1]) = \sum_{j=1}^{k} \ell_{i_j}$

总体最坏情况复杂度为 $O(n!)$

总结（1/3）

求解组合优化问题的一般步骤

- 解 $\rightsquigarrow$ 向量
- 状态空间 $\rightsquigarrow$ 搜索树（部分向量是内部节点，向量是叶节点）
- 搜索顺序（DFS、BFS）

暴力算法：遍历整棵树（回顾不等式问题的整数解）

[见原文第 62 页图片]

我们能否更聪明地实现暴力算法?

总结（2/3）

是的。回溯技术！回溯需要准则

基本回溯

- 从默认约束推导准则：确保多米诺性质成立

[见原文第 63 页图片]

额外优化技巧

- 可以利用对称性来减小搜索树的规模

示例：装载问题、图着色问题

总结（3/3）

高级回溯

- 分支限界法：除了默认准则外，引入界值和估计函数来剪枝搜索树（利用已获得的结果）$\rightsquigarrow$ 进一步降低复杂度 $\rightsquigarrow$ 精细化准则

[见原文第 64 页图片]

示例：MCP、TSP

应用分支限界法时，需要在收益和成本之间找到权衡