

第7章 分支限界法

《人工智能算法》

清华大学出版社

2022年7月

提纲

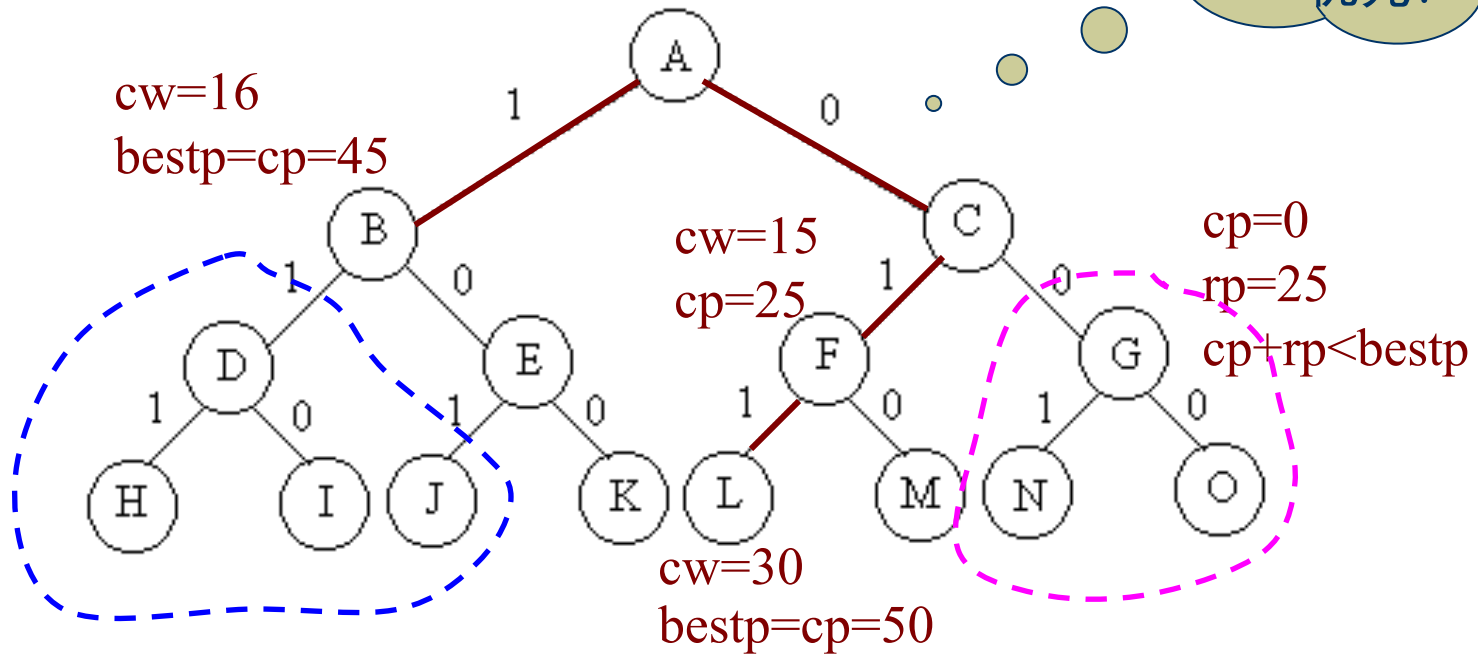
- ◆ 分支限界法的基本思想
- ◆ 0-1背包问题
- ◆ 总结

引例

◆ 0-1背包问题

- $n=3, w=\{16, 15, 15\}, p=\{45, 25, 25\}, c=30$
- 所有可能的情况 vs. 减小了的搜索空间

FIFO vs.
最大可能节点
优先?



分支限界法 vs. 回溯法

◆ 分支限界法与回溯法的区别

(1) 求解目标

- 分支限界法：适于求解满足约束条件的**最优解**
- 回溯法：找出解空间树中满足约束条件的解（一个或多个**可行解**）

(2) 搜索方式

- 分支限界法：广度优先、或最优目标函数优先
- 回溯法：深度优先

◆ 分支限界法的节点生成

- 选择一个活节点为扩展结点
- 生成扩展节点的所有儿子节点
- 可行（可能）的儿子节点加入活节点列表

分支限界法的基本思想 (1)

- 分支限界法中搜索树空间扩展

- (1) 队列式(FIFO)分支限界法

- 按照队列先进先出 (FIFO) 原则选取下一个结点为扩展节点

- (2) 优先队列式(minHeap / maxHeap)分支限界法

- 按照优先队列中规定的优先级选取优先级最高的节点成为当前扩展节点

- “优先队列式分支限界法” 更适用于优化问题?
 - (1) 和 (2) 搜索到叶子结点——找到一个最优解?
 - 确定搜索树 (根据显约束确定内部结点的分支数)
 - 分支限界法的基本步骤?

分支限界法的基本思想 (2)

◆ 分支限界法解决优化问题的基本思路

- 确定解空间树的结构
- 确定**目标函数**，作为结点扩展的依据
- 确定优先队列和**优先级**：最大堆/最小堆（目标函数最优）
- **最优目标函数优先+剪枝函数**

◆ 常用剪枝函数

- 用**约束函数**在扩展节点处剪去不满足约束的子树（问题本身的约束）
- 用**限界函数**剪去得不到最优解的子树
 - ✓ 上界/下界 限界函数
 - ✓ 互相控制的目标函数约束
 - ✓ 将可能导致最优解的活结点加入优先队列中

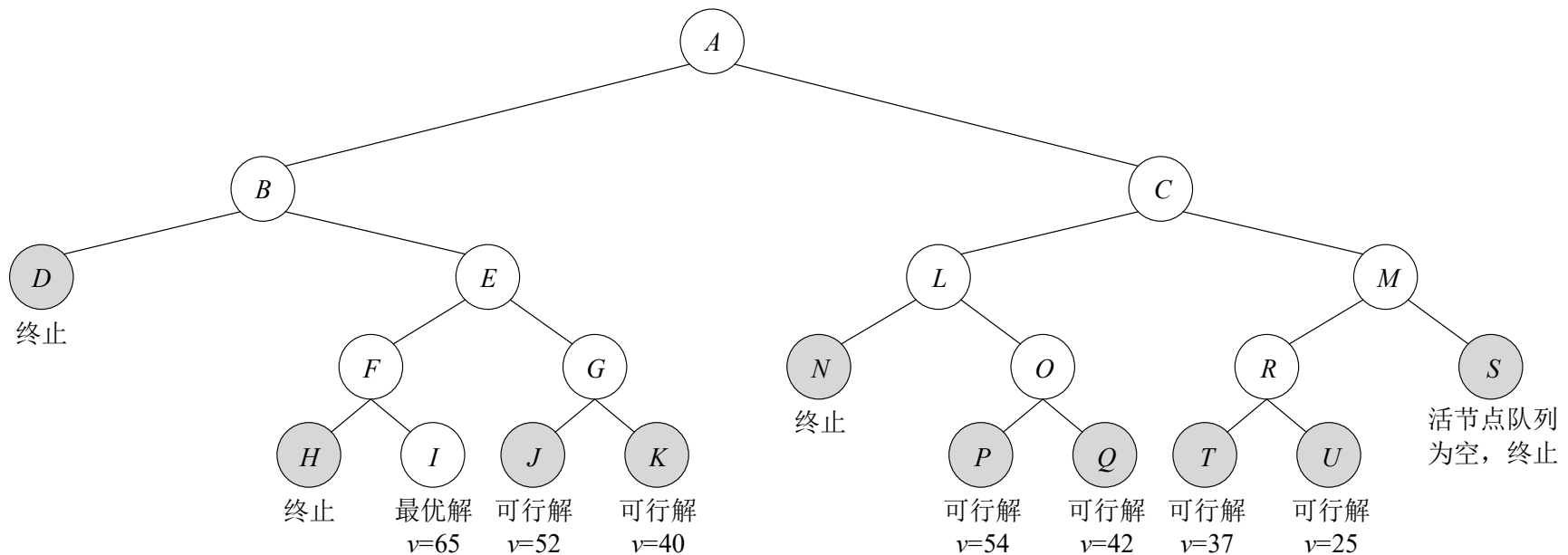
提纲

- ◆ 分支限界法的基本思想
- ◆ 0-1背包问题
- ◆ 总结

0-1背包问题 (1)

◆ 基本思想

- **解空间树**：子集树，一个物品要么装入（左孩子）、要么不装入（右孩子）
- 4种物品的重量和价值分别为{4, 7, 5, 3}和{40, 42, 25, 12}，背包容量为10



0-1背包问题 (2)

- 搜索空间扩展：优先队列——最大堆

- 优先级

节点*i*的价值上界 ub =已装入物品的价值+剩余空间装满获得的最大价值

- 剪枝策略

<1> 左子树：装入 $w[i]$ ，若 $ew+w[i]<c$ ，则可行

若 $cp+p[i]>bestp$ 则 $bestp\leftarrow cp+p[i]$

下一层活节点优先级： $heap.addNode(ub, cp+p[i], cw+w[i], i+1)$

<2> 右子树：不装入 $w[i]$ ， $ub\leftarrow bound(i+1)$

若 $ub>bestp$ ，则可行

下一层活节点优先级： $heap.addNode(ub, cp, cw, i+1)$

预处理：
类似背包问题


0-1背包问题 (3)

◆ 算法主要步骤:

```
if  $cw+w[i] \leq c$  then  
  if  $cp+p[i] > bestp$  then  
     $bestp \leftarrow cp+p[i]$   
     $heap.addNode(ub, cp+p[i], cw+w[i], i+1)$   
  end if
```

```
 $ub \leftarrow bound(i+1)$   
if  $ub > bestp$  then  
   $heap.addNode(ub, cp, cw, i+1)$   
end if
```

```
 $node \leftarrow heap.removeMax()$   
 $cw \leftarrow node.weight$   
 $cp \leftarrow node.profit$   
 $p \leftarrow node.ub$   
 $i \leftarrow node.level$ 
```



如何实现
bound的计算?

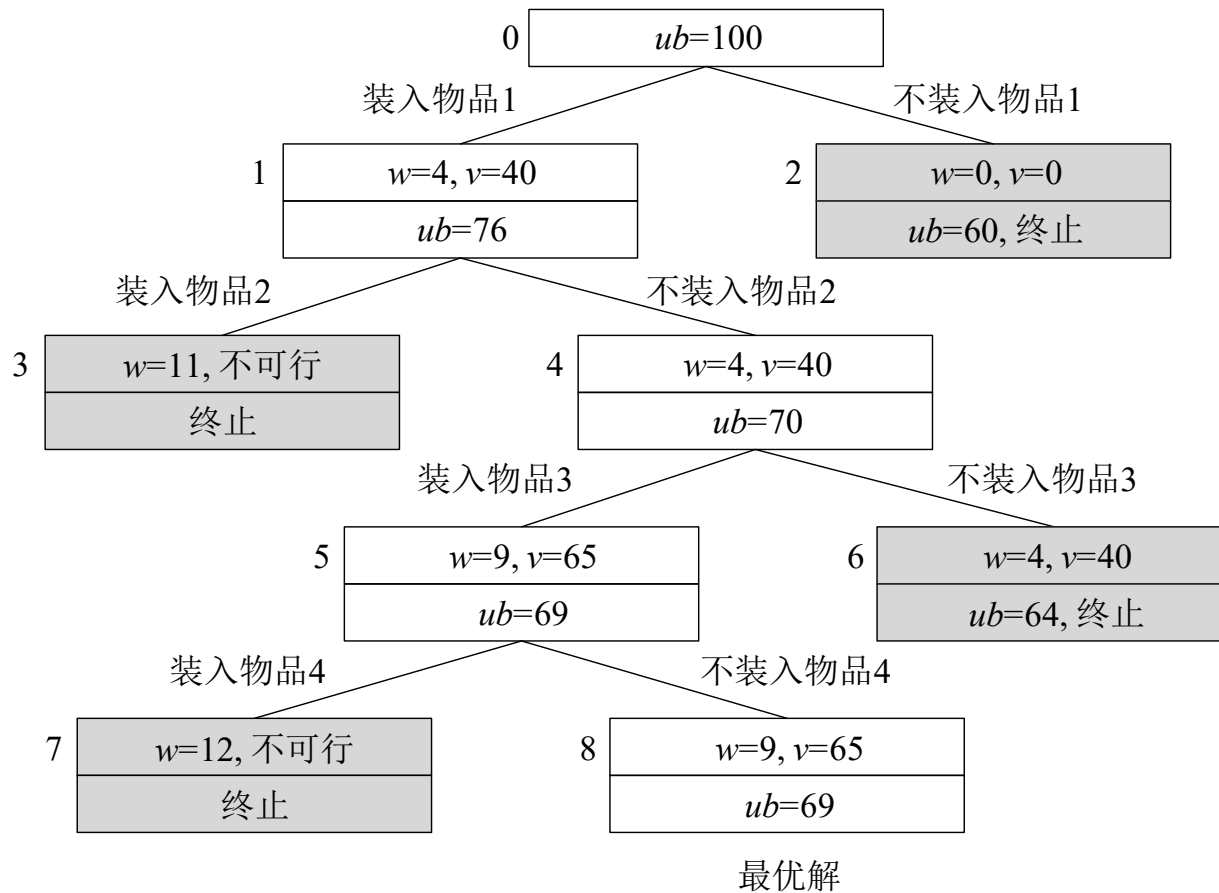
0-1背包问题 (4)

◆ 上界bound的计算

- 预处理：将输入按照单位重量价值的顺序排序
- 计算bound:

```
cleft ← c - cw
while i ≤ n and w[i] ≤ cleft do
    cleft ← cleft - w[i]
    b ← b + p[i];
    i ← i + 1
end while
if i ≤ n then
    b ← b + p[i] / w[i] * cleft
end if
return b
```

0-1背包问题 (5)



提纲

- ◆ 分支限界法的基本思想
- ◆ 0-1背包问题
- ◆ 总结

总结 (1)

- ◆ 分支限界法的基本思想
- ◆ 分支限界法与回溯法的区别
- ◆ 分支限界法解决优化问题的关键步骤
 - 解空间树+搜索扩展策略
 - 剪枝函数+限界函数
- ◆ 重要的分支限界算法实例：0-1背包问题

总结 (2)

- ◆ **基于优先队列分支限界法解决优化问题的关键**
 - 确定扩展结点选择的目标函数及搜索空间
 - 最大/最小堆优先队列
 - 剪枝策略
 - ✓ 最有希望的代价下界或价值上界
 - ✓ 互相控制的目标函数约束
 - 分支限界法解决优化问题的特征：
 - ✓ 循环结束条件
 - ✓ 优化问题需找到一个最优解：只需找到一个解，解停止搜索
- ◆ **FIFO的广度优先分支限界法类似回溯法，一般不适宜用于求最优值（最优解）**



结语

谢谢！