《算法设计与分析》

第七章 分枝—限界法

马丙鹏

2023年11月13日

中国科学院大学
University of Chinese Academy of Sciences

第七章 分枝—限界法

- 7.1 一般方法
- 7.2 LC-检索
- 7.3 15-谜问题
- 7.4 布线问题
- 7.5 LC-检索(续)
- 7.6 分枝-限界算法
- 7.7 0/1背包问题
- 7.8 货郎担问题

7.6 分枝-限界算法

- 成本函数在分枝-限界算法中的应用
  - 假定每个答案结点X有一个与其相联系的c(X)，且找成本最小的答案结点。
  - 最小成本的下界：$\hat{c}(\bullet)$
    $\hat{c}(X)$为X的成本估计函数。
    当$\hat{c}(X) \le c(X)$时，$\hat{c}(X)$给出由结点X求解的最小成本的下界
    ——启发性函数，减少选取E结点的盲目性。

7.6 分枝-限界算法

- 成本函数在分枝-限界算法中的应用
  - 最小成本的上界：U
    - 能否定义最小成本的上界？
    - 定义U为最小成本解的成本上界，则：
      对具有$\hat{c}(X) > U$的所有活结点可以被杀死，从而可以进一步使算法加速，减少求解的盲目性。
    - 注：
      - ✓根据$c(X)$的定义，由那些$\hat{c}(X) > U$的结点X可到达的所有答案结点必有$c(X) \geq \hat{c}(X) \geq U$。
      - ✓故，当已经求得一个具有成本U的答案结点，那些有$\hat{c}(X) > U$的所有活结点都可以被杀死——这些结点不可能具有更小的成本。

7.6 分枝-限界算法

- 成本函数在分枝-限界算法中的应用
  - 最小成本上界U的求取：
    ① 初始值：利用启发性方法赋初值，或置为∞
    ② 每找到一个新的答案结点后修正U，U取当前最小成本值。
  - 注：只要U的初始值不小于最小成本答案结点的成本，利用U不会杀死可以到达最小成本答案结点的活结点。

7.6 分枝-限界算法

- 利用分枝-限界算法求解最优化问题
  - 最优化问题：
    求能够使目标函数取最小(大)值的可行解——最优解。
  - 如何把求最优解的过程表示成与分枝-限界相关联的检索过程？
  - 可行解：类似于**n**-元组的构造，把可行解可能的构造过程用“状态空间树”表示出来。
    答案结点 ⟷ 可行解
  - 最优解：把对最优解的检索表示成对状态空间树答案结点的检索。
    成本最小的答案结点 ⟷ 最优解

7.6 分枝-限界算法

■ 利用分枝-限界算法求解最优化问题

- □成本函数：每个结点赋予一个成本函数，并使得：代表最优解的答案结点的c(X)是所有结点成本的最小值。
- □成本函数的定义：直接用目标函数作为成本函数c(·)：
  - ① 代表可行解结点的c(X)就是该可行解的目标函数值。
  - ② 代表不可行解结点的c(X)=∞；
  - ③ 代表部分解的结点的c(X)是根为X的子树中最小成本结点的成本。
- □成本估计函数 $\hat{c}(X)$，且要求有 $\hat{c}(X) \le c(X)$

  注：$\hat{c}(X)$根据目标函数进行估计分析。

7.6 分枝-限界算法

- 实例：求带限期的作业排序问题
  - 问题描述：带限期的作业排序问题—— 更为一般化的定义
    - 假定n个作业和一台处理机
    - 每个作业i对应一个三元组(p_i, d_i, t_i)
      - ✓ t_i：表示作业i需要t_i个单位处理时间——允许不同作业有不同的处理时间；
      - ✓ d_i：表示完成期限；
      - ✓ p_i：表示若i在期限内未完成招致的罚款。
  - 求解目标：从n个作业的集合中选取子集J，要求J中所有作业都能在各自期限内完成并且使得不在J中的作业招致的罚款总额最小——最优解。

7.6 分枝-限界算法

- 实例：求带限期的作业排序问题
  - 实例：$n = 4$;
    $(p_1, d_1, t_1) = (5, 1, 1)$;　$(p_2, d_2, t_2) = (10, 3, 2)$;
    $(p_3, d_3, t_3) = (6, 2, 1)$;　$(p_4, d_4, t_4) = (3, 1, 1)$;
  - 状态空间：
    问题的解空间由作业指标集(1, 2, 3, 4)的所有可能子集合组成。
  - 状态空间树：两种表示形式，
    - 元组大小可变的表示：
      表示选了哪些作业，用作业编号表示。
    - 元组大小固定的表示：
      表示是否选中作业，每个作业对应一个0/1值。

$(p_1, d_1, t_1) = (5, 1, 1);$

$(p_2, d_2, t_2) = (10, 3, 2);$

$(p_3, d_3, t_3) = (6, 2, 1);$

$(p_4, d_4, t_4) = (3, 1, 1);$

圆形结点都是答案结点(可行解)

方形结点是不可行的子集合

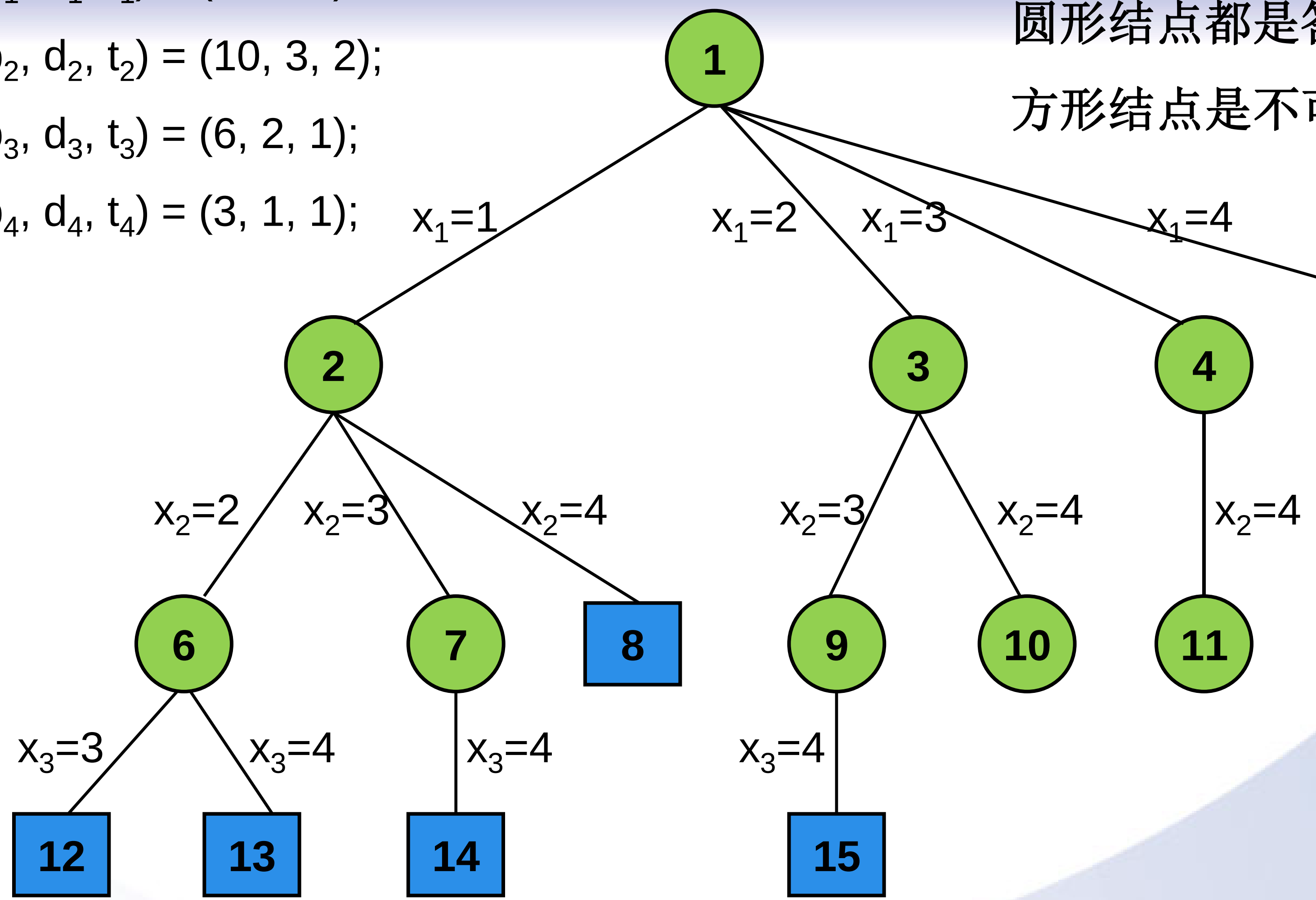


图7.7 采用大小可变的元组表示的状态空间树

$(p_1, d_1, t_1) = (5, 1, 1)$;

$(p_2, d_2, t_2) = (10, 3, 2)$;

$(p_3, d_3, t_3) = (6, 2, 1)$;

$(p_4, d_4, t_4) = (3, 1, 1)$;

只有圆形叶结点是答案结点

方形结点是不可行的子集合

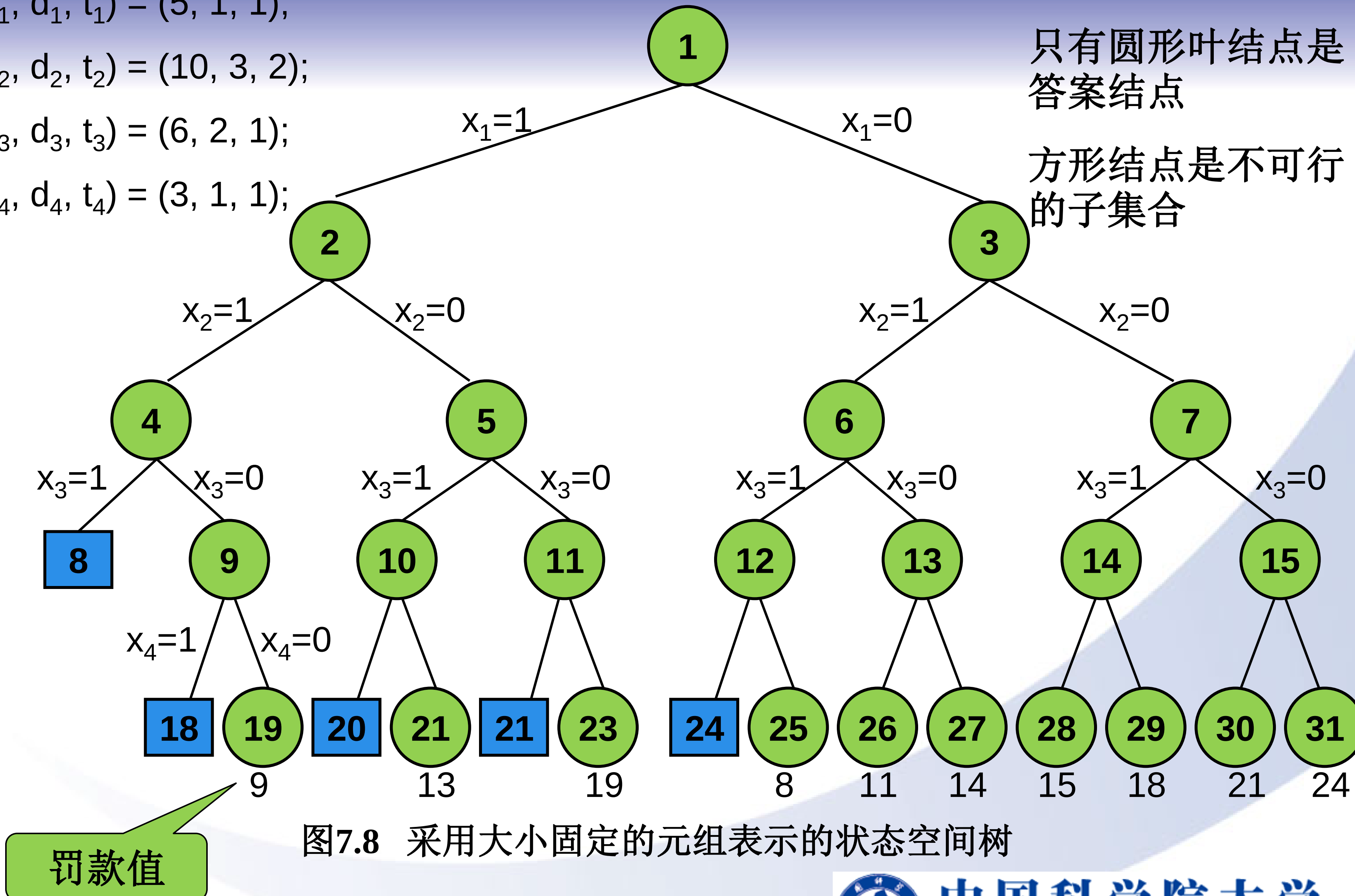


图7.8 采用大小固定的元组表示的状态空间树

7.6 分枝-限界算法

- 实例：求带限期的作业排序问题
  - 成本函数c(·)定义为：
    - 对于圆形结点X，c(X)是根为X的子树中结点的最小罚款；
    - 对于方形结点，c(X) = ∞。
  - 例：如图
    - 图7.7，c(3)=8，c(2)=9，c(1)=8
    - 图7.8，c(2)=9，c(5)=13，c(6)=8，c(1)=8
    - c(1) = 最优解J对应的罚款

$(p_1, d_1, t_1) = (5, 1, 1);$

$(p_2, d_2, t_2) = (10, 3, 2);$

$(p_3, d_3, t_3) = (6, 2, 1);$

$(p_4, d_4, t_4) = (3, 1, 1);$

$$c(X) = \sum_{i \notin J} p_i$$

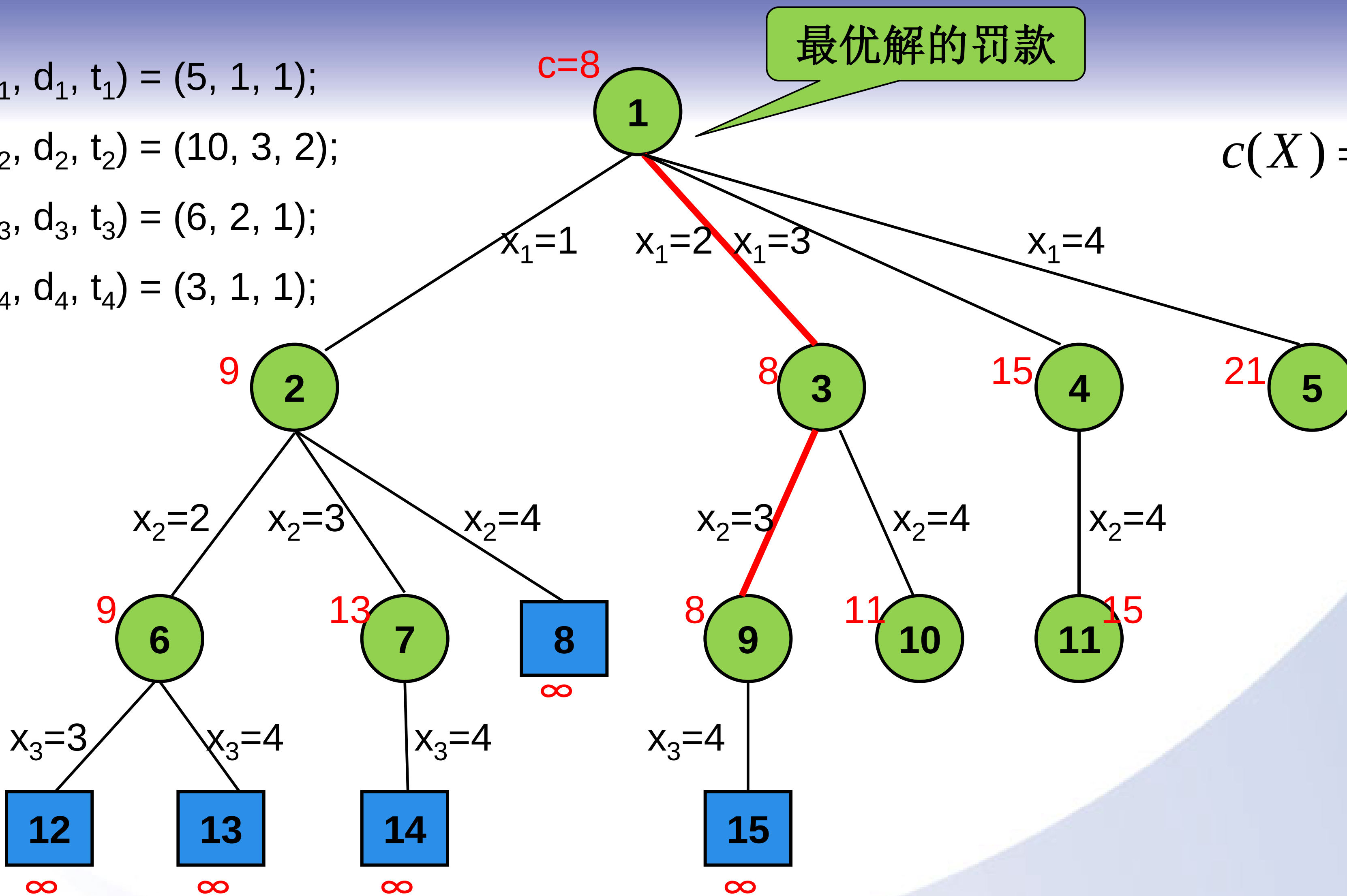


图7.7　采用大小可变的元组表示的状态空间树各结点的c值

$(p_1, d_1, t_1) = (5, 1, 1);$

$(p_2, d_2, t_2) = (10, 3, 2);$

$(p_3, d_3, t_3) = (6, 2, 1);$

$(p_4, d_4, t_4) = (3, 1, 1);$

$$c(X) = \sum_{i \notin J} p_i$$

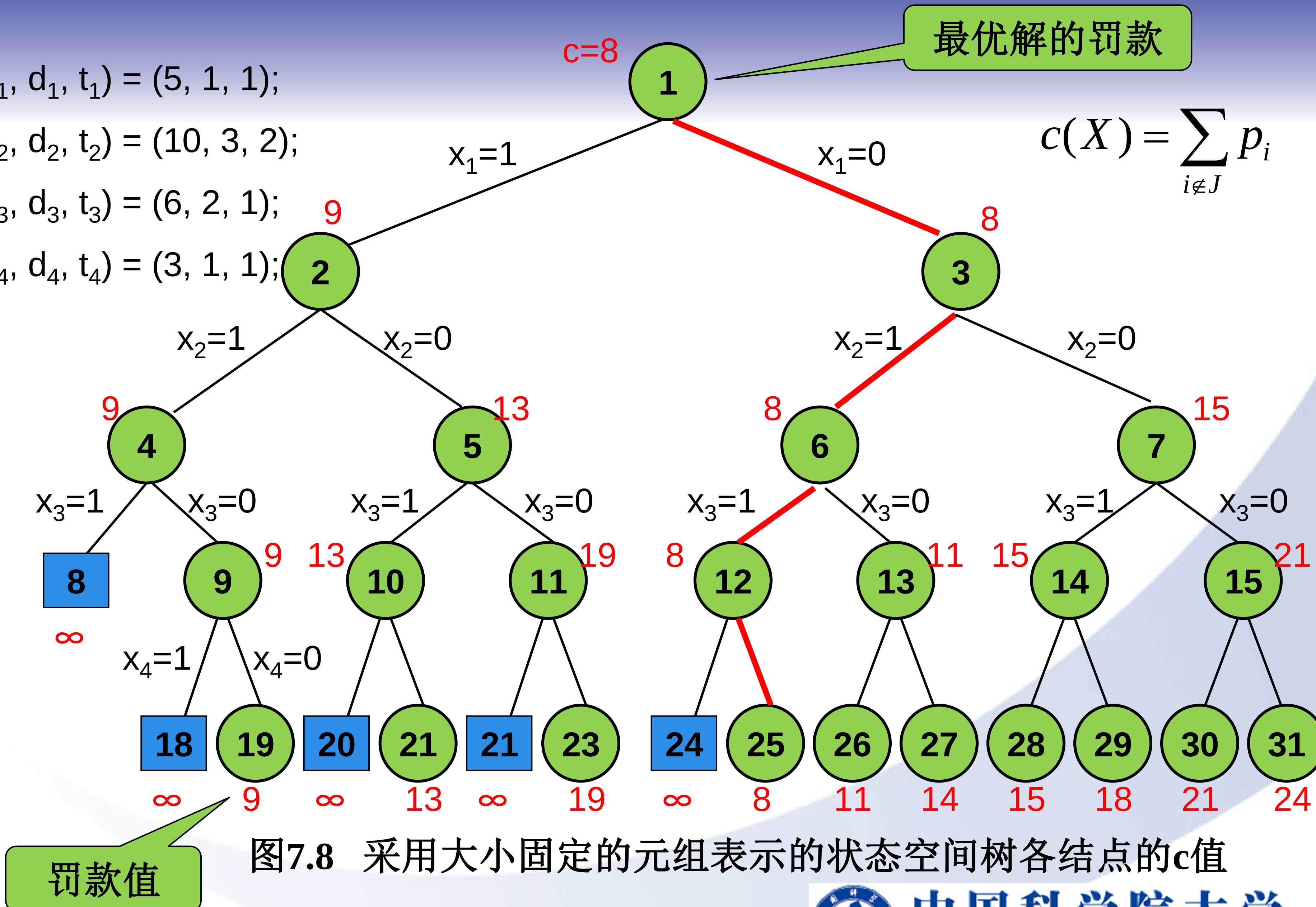


图7.8 采用大小固定的元组表示的状态空间树各结点的c值

7.6 分枝-限界算法

- 实例：求带限期的作业排序问题
  - 成本函数$\hat{c}(\bullet)$的定义
    - 设S_x是在结点X时，已计入J中的作业的集合。
    - 令 $m = \max\{i \mid i \in S_x\}$，则

$$\hat{c}(X) = \sum_{\substack{i<m \\ i \notin S_X}} p_i$$

是使c(X)有$\hat{c}(X) \le c(X)$的估计值(下界)。

- $\hat{c}(X)$的含义：
  - ✓已经被考虑过但没有被计入J中的作业的罚款合计

    ——已确定的罚款数，目前罚款的底线。

$(p_1, d_1, t_1) = (5, 1, 1)$;

$(p_2, d_2, t_2) = (10, 3, 2)$;

$(p_3, d_3, t_3) = (6, 2, 1)$;

$(p_4, d_4, t_4) = (3, 1, 1)$;

$$\hat{c}(X) = \sum_{\substack{i<m \\ i \notin S_X}} p_i$$

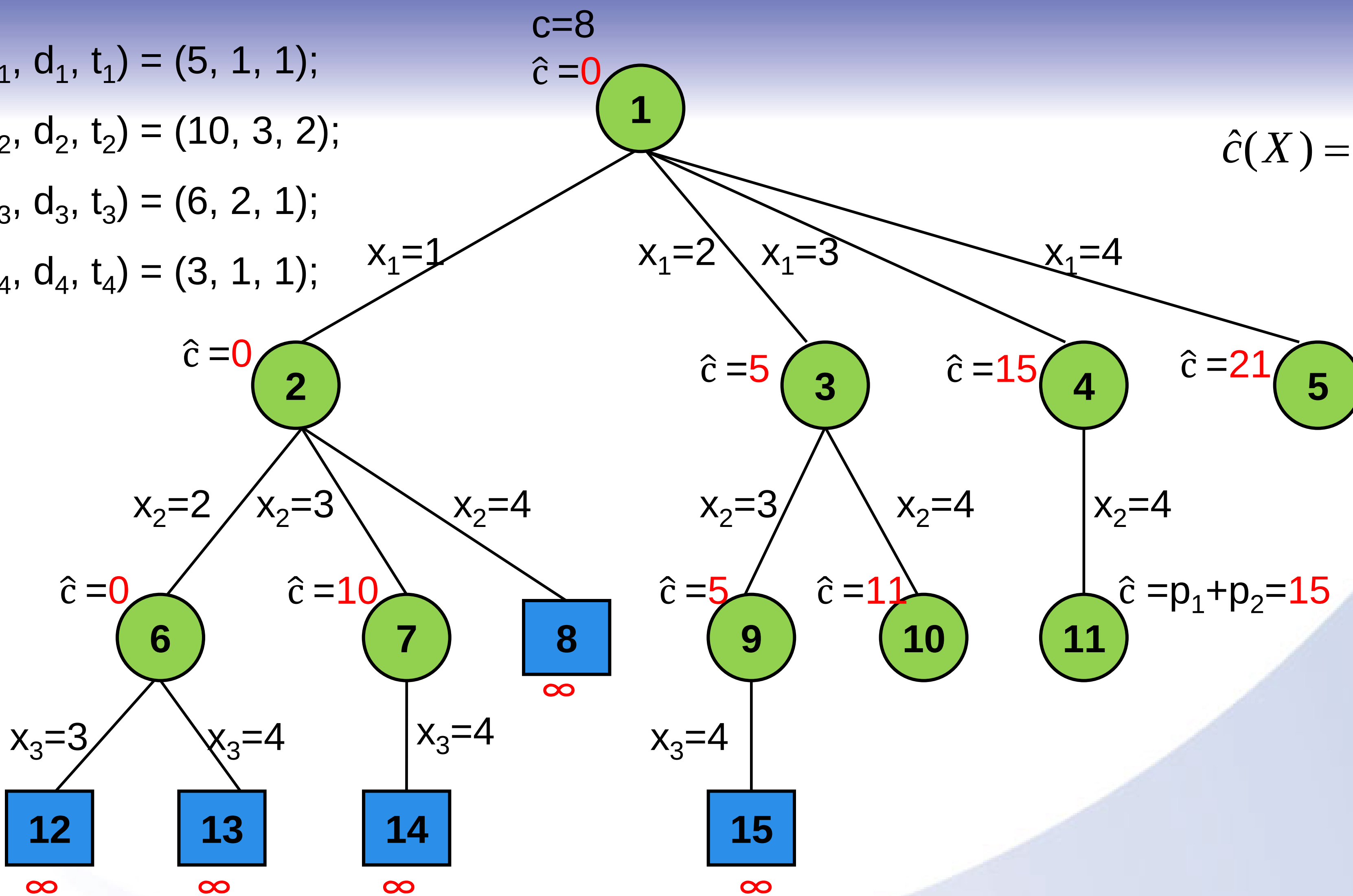


图7.7　采用大小可变的元组表示的状态空间树的成本估计值

$(p_1, d_1, t_1) = (5, 1, 1);$

$(p_2, d_2, t_2) = (10, 3, 2);$

$(p_3, d_3, t_3) = (6, 2, 1);$

$(p_4, d_4, t_4) = (3, 1, 1);$

$$\hat{c}(X) = \sum_{\substack{i<m \\ i \notin S_X}} p_i$$

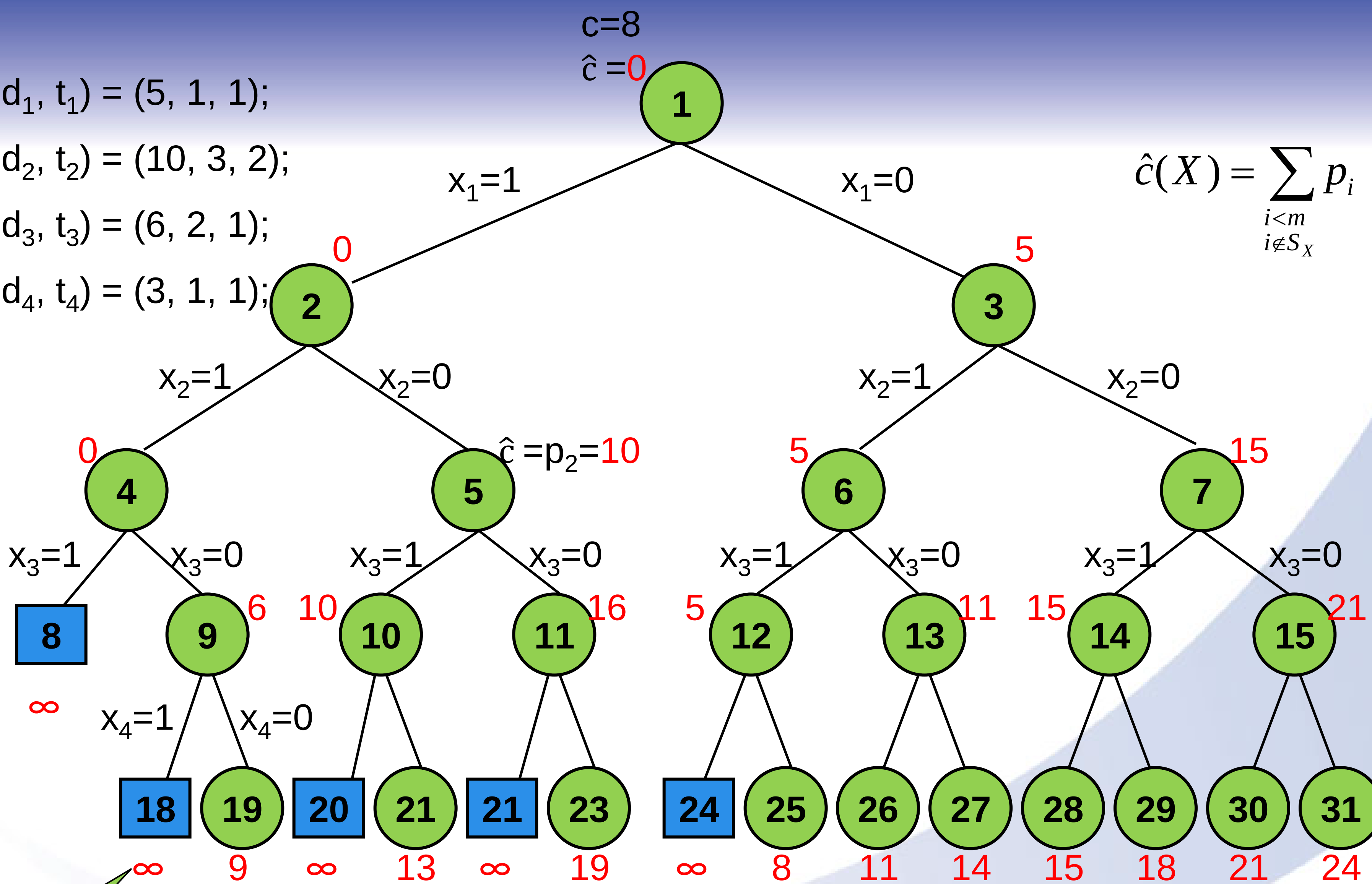


图7.8 采用大小固定的元组表示的状态空间树的成本估计值

7.6 分枝-限界算法

- 实例：求带限期的作业排序问题
  - 成本估计函数上界的定义

$$U(X)=\sum_{i\notin S_X} p_i$$

- 是c(X)的一个“简单”上界。
- U(X)的含义：
  - ✓目前没有被计入到J中的作业的罚款合计，
  - ✓可能的最多罚款，目前罚款的上限。

中国科学院大学
University of Chinese Academy of Sciences

作业排序问题的FIFO分枝-限界算法图例分析

$(p_1, d_1, t_1) = (5, 1, 1)$;

$(p_2, d_2, t_2) = (10, 3, 2)$;

$(p_3, d_3, t_3) = (6, 2, 1)$;

$(p_4, d_4, t_4) = (3, 1, 1)$;

$$\hat{c}(X) = \sum_{\substack{i<m \\ i\notin S_X}} p_i$$

$$u(1) = \sum_{1\le i\le n} p_i$$

$$u(X) = \sum_{i\notin S_X} p_i$$

$$U = \min(u(X))$$

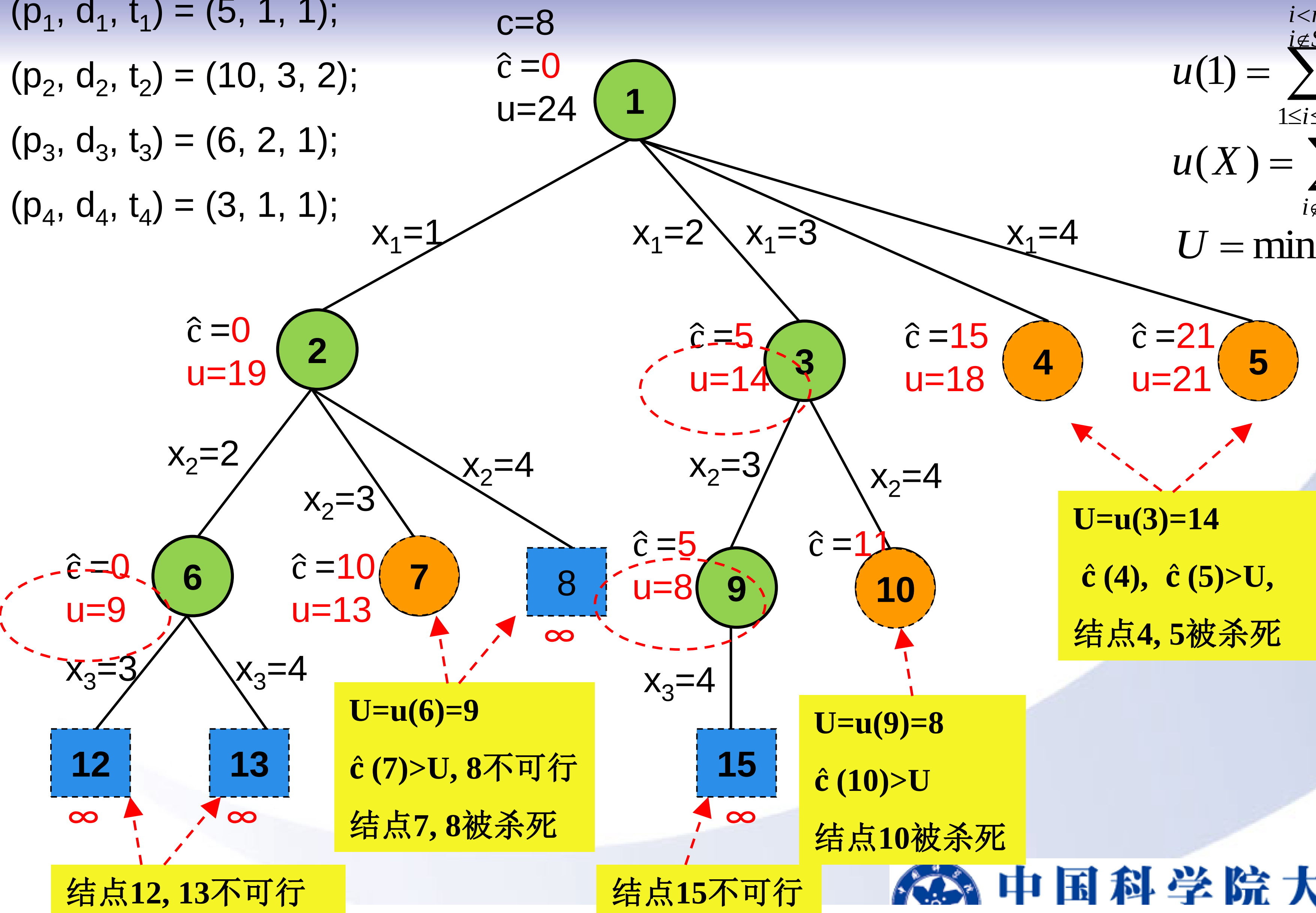


图7.7　采用大小可变的元组表示的状态空间树的成本估计值

7.6 分枝-限界算法

- **作业排序问题的FIFO分枝-限界算法FIFOBB**
  - 设计说明
  - ① 每有更小的u(X)计算出来时修正U值。
  - ② 当结点X从活结点表出来将变成E结点时，
    - 若$\hat{c}(X) > U$
      X被立即杀死，
      继续考虑活结点表中的其它活结点。

7.6 分枝-限界算法

- 作业排序问题的FIFO分枝-限界算法FIFOBB
  - ② 当结点X从活结点表出来将变成E结点时，
    - 若 $\hat{c}(X) = U$，根据U的得来作如下处理：
      - ✓ U是一个已找到的解的成本：杀死X

        注：至少已经有了一个成本等于U的解，这个X对最优解没有任何意义。
      - ✓ U是一个单纯的上界：

        X成为E结点，继续扩展

        注：当前的最好解还没找到，X是有可能导致成本等于U的解结点的。

7.6 分枝-限界算法

■ 作业排序问题的FIFO分枝-限界算法FIFOBB

③ 引进一个很小的正常数ε来区分这两种情况，ε足够小，使得对于任意两个可行的结点X和Y，若u(X)<u(Y)，则u(X)< u(X)+ε < u(Y)

注：引入ε的目的：使U略大于u(x)，便于比较计算

④ 在扩展E结点得到一个儿子结点X时若有**ĉ(X) < U**：

- ✓ 若X是答案结点且c(X) < U时，U = min(c(X), u(X) + ε)

  注：若u(X) < c(X)，则根为X的子树中有更优的解。

- ✓ 若X是单纯的上界，U = min(U, u(X)+ ε)

7.6 分枝-限界算法

- 作业排序问题的FIFO分枝-限界算法FIFOBB
  - ⑤ 在扩展E结点得到一个儿子结点X时若有$\hat{c}(X) \geq U$，儿子结点被杀死
  - ⑥ ADDQ、DELETEQ过程：将结点加入队列或从队列中删除。
  - ⑦ cost(X)：若X是答案结点，则cost(X)是结点X对应的解的成本。
  - ⑧ 可行结点的估计值$\hat{c}(X) \leq c(X) \leq u(X)$
  - ⑨ 不可行结点的$\hat{c}(X) = \infty$

为找出最小成本答案结点检索T。假定T至少包含一个解结点且ĉ(X)≤c(X)≤u(X)

```
line procedure FIFOBB(T, ĉ, u, ε, cost)
E←T; PARENT(E)←0;
if  T是解结点  then  U←min(cost(T), u(T)+ε); ans←T
                  else   U←u(T)+ε; ans←0
endif
将队列置为空
loop
   for E的每个儿子X  do
      if  ĉ(X)<U  then  call  ADDQ(X);  PARENT(X)←E
          case
             :X是解结点 and cost(X)<U: U←min(cost(X), u(X)+ε); ans←X
             :u(X)+ε<U : U←u(X)+ε
          endcase
      endif
   repeat
   loop //从队列中取下一个E-结点
      if 队列为空  then print(“least cost=”, U)
         while  ans≠0 do print (ans);  ans←PARENT(ans)   repeat  return
      endif
      call  DELETEQ(E)
      if ĉ(E)<U then exit  endif
   repeat
repeat
end FIFOBB
```

ans指向最新解

判定X的可解性,修正U

$\hat{c}(E) \geq$ U的结点被从队列中清除(被杀死)

7.6 分枝-限界算法

■ 作业排序问题的FIFO分枝-限界算法FIFOBB

□设计思想

假定状态空间树T至少包含一个解结点，
不可行结点的估计值$\hat{c}(X)=\infty$；
可行结点的估计值 $\hat{c}(X)\le c(X)\le u(X)$。

- 初始化
  - 令根结点T是当前E-结点；
  - 对估计成本上界U和答案解ans赋初值：
    如果T是解结点，$U\leftarrow \min(T的成本, u(T)+\varepsilon)$，$ans\leftarrow T$；
    否则$U\leftarrow u(T)+\varepsilon$, $ans\leftarrow 0$；
  - 活结点队列置为空；
- 对E的每个子结点X进行检验，添加到队列中，并修改U值：
  - 如果$\hat{c}(X)<U$，将X加入到队列中，并对X进一步判断：
    如果X是解结点且X的成本<U，$U\leftarrow \min(X的成本, u(X)+\varepsilon)$，$ans\leftarrow X$；
    如果$u(X)+\varepsilon<U$, $U\leftarrow u(X)+\varepsilon$；
- 若队列为空，打印当前ans；否则从队列中获取下一个活结点E：若$\hat{c}(E)<U$,转到步骤2，否则转到步骤3。

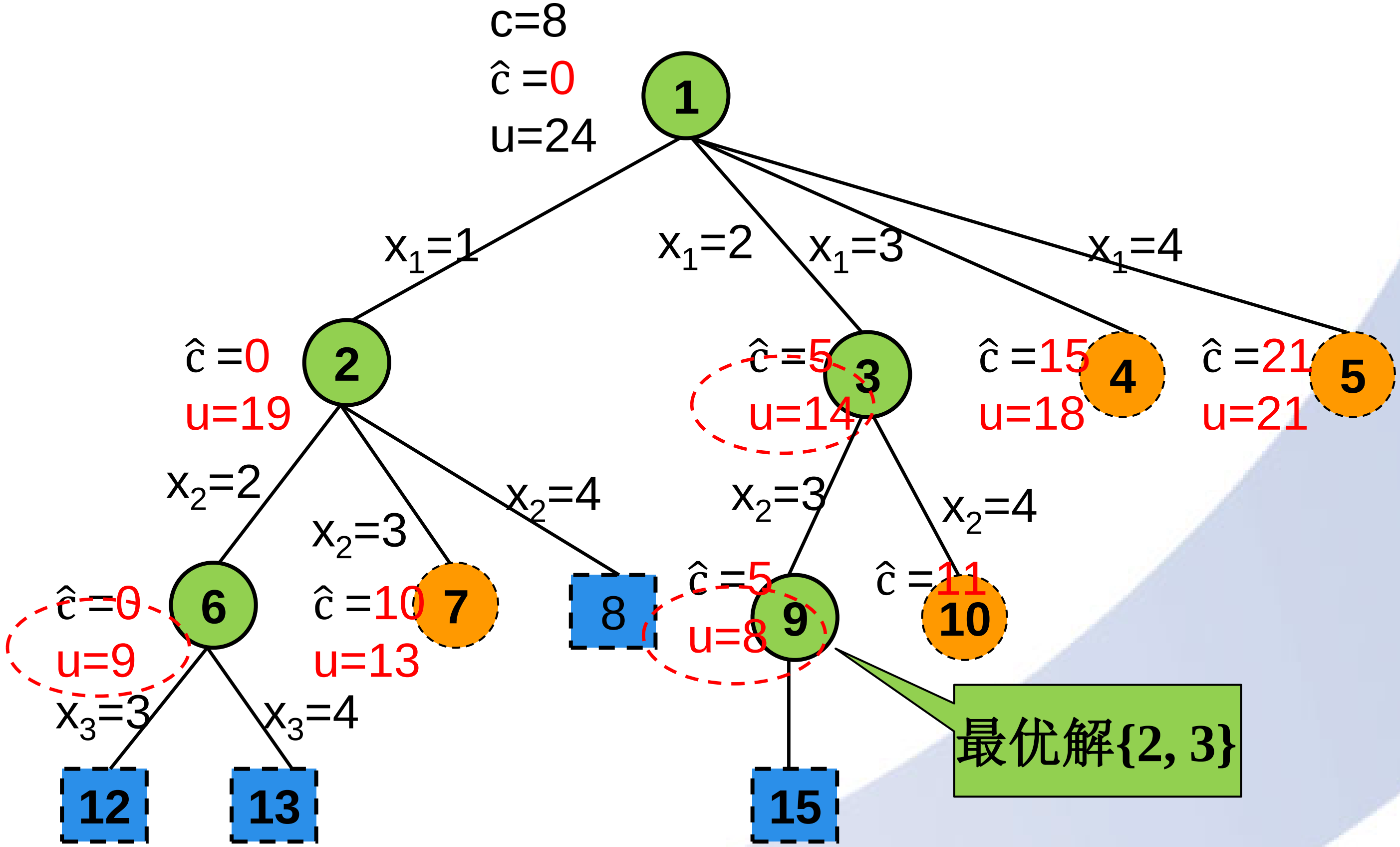
中国科学院大学
University of Chinese Academy of Sciences

$(p_1, d_1, t_1) = (5, 1, 1)$; $(p_2, d_2, t_2) = (10, 3, 2)$;

$(p_3, d_3, t_3) = (6, 2, 1)$; $(p_4, d_4, t_4) = (3, 1, 1)$;

ε=0.1

U	ans	E
24	1	1
19	2	
14	3	2
9	6	3
8	9	6
		9



队列 2 3 6 9

7.6 分枝-限界算法

- 作业排序问题的LC分枝-限界算法LCBB
  - 设计说明：
    ① 采用min-堆保存活结点表
    ② ADD、LEAST过程：
    将结点加入到min-堆或从min-堆中删除。
    ③ 其余约定同于FIFO检索
    ④ 当min-堆中没有活结点或下一个E结点有情况下算法终止。
  - 算法描述：

中国科学院大学
University of Chinese Academy of Sciences

为找出最小成本答案结点检索T。假定T至少包含一个解结点且$\hat{c}(X) \le c(X) \le u(X)$

```
line procedure LCBB(T, ĉ, u, ε, cost)
E←T; PARENT(E)←0;
if  T是解结点  then  U←min(cost(T), u(T)+ε); ans←T
                 else   U←u(T)+ε; ans←0
endif
将活结点表初始化为空   min堆
loop
  for E的每个儿子X  do
    if  ĉ(X)<U  then  call  ADDQ(X);  PARENT(X)←E
          case
               :X是解结点 and cost(X)<U: U←min(cost(X), u(X)+ε);ans←X
               :u(X)+ε<U : U←u(X)+ε
          endcase
    endif
  repeat
  if 不再有活结点 or  下一个E-结点有ĉ≥U  then print("least cost=",U)
     while  ans≠0 do   print (ans);  ans←PARENT(ans)   repeat  return
  endif
  call  LEAST(E)
repeat
end LCBB
```

判定X的可解性修正U

活结点表中剩余结点都有$\hat{c} \ge U$

中国科学院大学
University of Chinese Academy of Sciences

7.6 分枝-限界算法

- 效率分析
  - 上下界函数的选择是决定分枝-限界算法效率的主要因素
  - 对U选择一个更好的初值是否能减少所生成的结点数？(否，根据定理7.4)
  - 扩展一些 $\hat{c}(X) > U$ 的结点是否能减少所生成的结点数？(否，根据定理7.5)
  - 假定有两个成本估计函数 $c_1\hat{}(\cdot)$ 和 $c_2\hat{}(\cdot)$，对于状态空间树的每一个结点X，若有 $c_1\hat{}(\cdot) \le c_2\hat{}(\cdot) \le c(X)$，则称 $c_2\hat{}(\cdot)$ 比 $c_1\hat{}(\cdot)$ 好。是否用较好的成本估计函数生成的结点数要少呢？
    (否，根据定理7.6和定理7.7)

作业-课后练习24

- 问题描述
  - 给定一个带期限的作业排序问题(n=5)：
    $(p_1, p_2, p_3, p_4, p_5)=(6, 3, 4, 8, 5)$,
    $(t_1, t_2, t_3, t_4, t_5)= (2, 1, 2, 1, 1)$,
    $(d_1, d_2, d_3, d_4, d_5)= (3, 1, 4, 2, 4)$,
  - 应用FIFOBB求使总罚款数最小的可行作业集J，求出对应于最优解的罚款值，并画出状态空间树（采用大小可变的元组）。
  - 应用LCBB求使总罚款数最小的可行作业集J，求出对应于最优解的罚款值，并画出状态空间树（采用大小固定的元组）。
- 要求
  - 作业提交到课程网站上

End