



HITWH
SE

第七章

Tree Searching Strategies



**HITWH
SE**

参考资料

R.C.T.Lee, S.S.Tseng, R.C.Chang, and Y.T.Tsai,

***Introduction to
the Design and Analysis of Algorithms***

Chapter 5

Page 157~219



- 7.1 Motivation of Tree Searching
- 7.2 Optimal Tree Searching Strategies
- 7.3 Personnel Assignment Problem
- 7.4 0/1 Knapsack Problem



7.1 Motivation of Tree Searching

很多问题的解可以表示成为树.

解为树的节点或路径.

求解这些问题可以转化为树搜索问题



布尔表达式可满足性问题

- 问题的定义

- 输入: n 个布尔变量 x_1, x_2, \dots, x_n

- 关于 x_1, x_2, \dots, x_n 的 k 个析取式

- 输出: 是否存在 x_1, x_2, \dots, x_n 的一种赋值
使得所有 k 个析取式皆为真

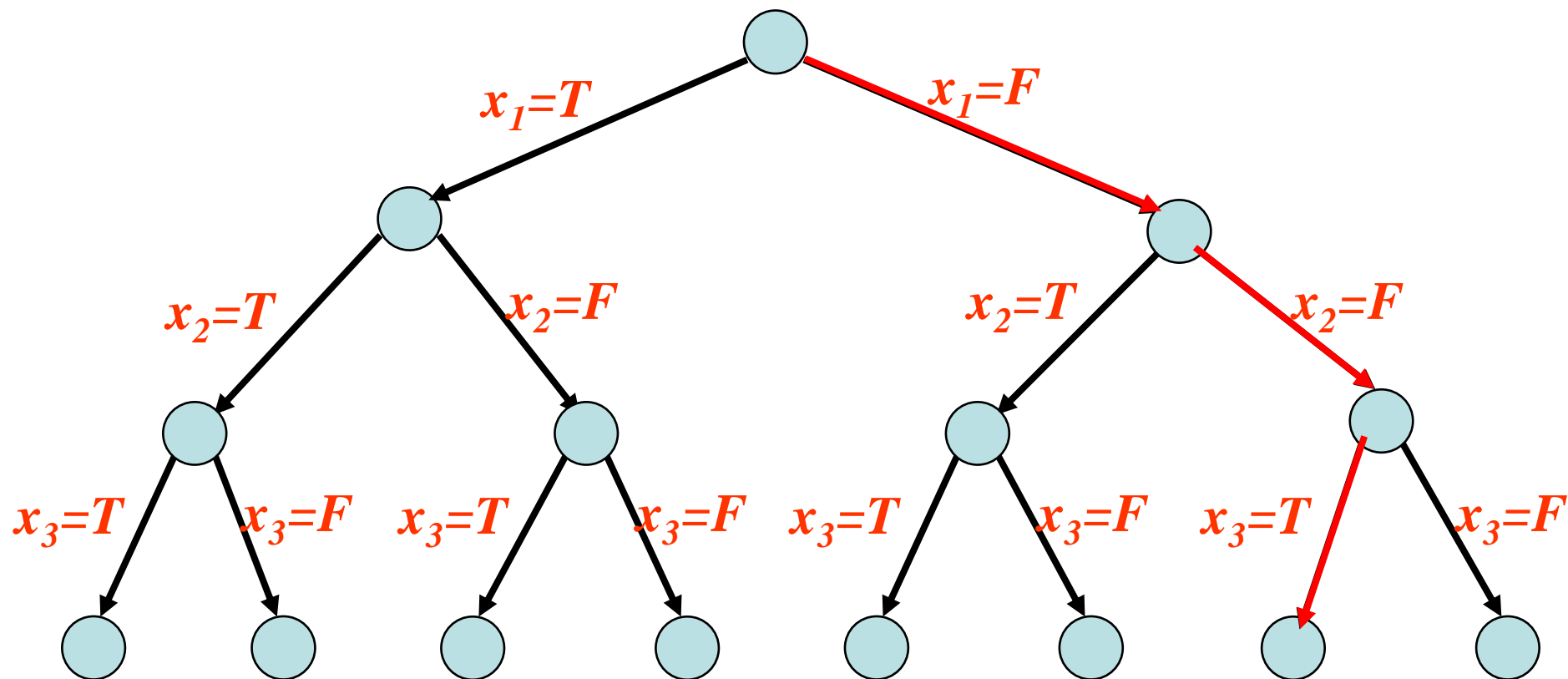
例如: $\neg x_1, x_2 \vee x_3, x_3, \neg x_2$

- 把问题的解表示为树

- 通过不断地为赋值集合分类来建立树

(以 x_1, x_2, x_3 为例)

对于: $\neg x_1, x_2 \vee x_3, x_3, \neg x_2$



问题的解是由根到叶的一条路径



- 问题的定义

- 输入：具有8个编号小方块的魔方

2	3	
5	1	4
6	8	7

- 输出：移动序列，经过这些移动，魔方到达如下状态

1	2	3
8		4
7	6	5

如何构造树？



- 转换为树搜索问题

2	3	
5	1	4
6	8	7

问题的解是
树上的一个节点

2		3
5	1	4
6	8	7

2	3	4
5	1	
6	8	7



7.2 Optimal Tree Searching Strategies

- Hill Climbing
- Best-First Search Strategy
- Branch-and-Bound Strategy



- 基本思想

- 在深度优先搜索过程中，我们经常遇到多个节点可以扩展的情况，首先扩展哪个？
- 爬山策略使用贪心方法确定搜索的方向，是优化的深度优先搜索策略
- 爬山策略使用启发式测度来排序节点扩展的顺序
 - 使用什么启发式测度与问题本身相关



1	2	3
8		4
7	6	5

- 用8-Puzzle问题来说明爬山策略的思想

- 启发式测度函数:

- $f(n)=W(n)$, $W(n)$ 是节点 n 中处于错误位置的方块数.

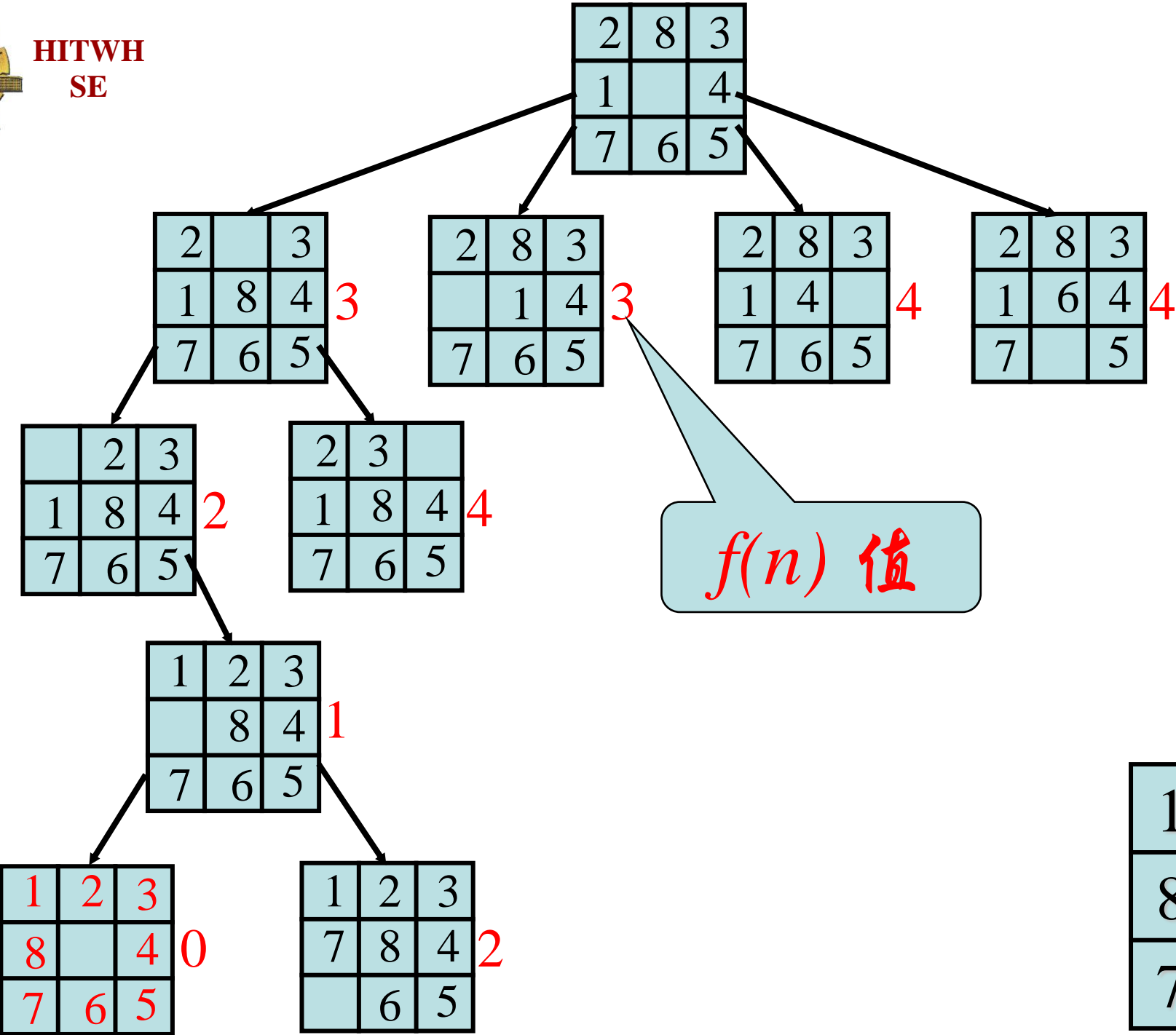
- 例如, 如果节点 n 如下:

2	8	3
1		4
7	6	5

- 则 $f(n)=3$, 因为方块1、2、8处于错误位置.



HITWH
SE



$f(n)$ 值

1	2	3
8		4
7	6	5



- Hill Climbing 算法

1. 构造由根组成的单元素栈 S ;
2. If $Top(S)$ 是目标节点 Then 停止;
3. Pop(S);
4. S 的子节点按照其启发测度由大到小的顺序压入 S ;
5. If S 空 Then 失败 Else goto 2.



Best-First Search Strategy

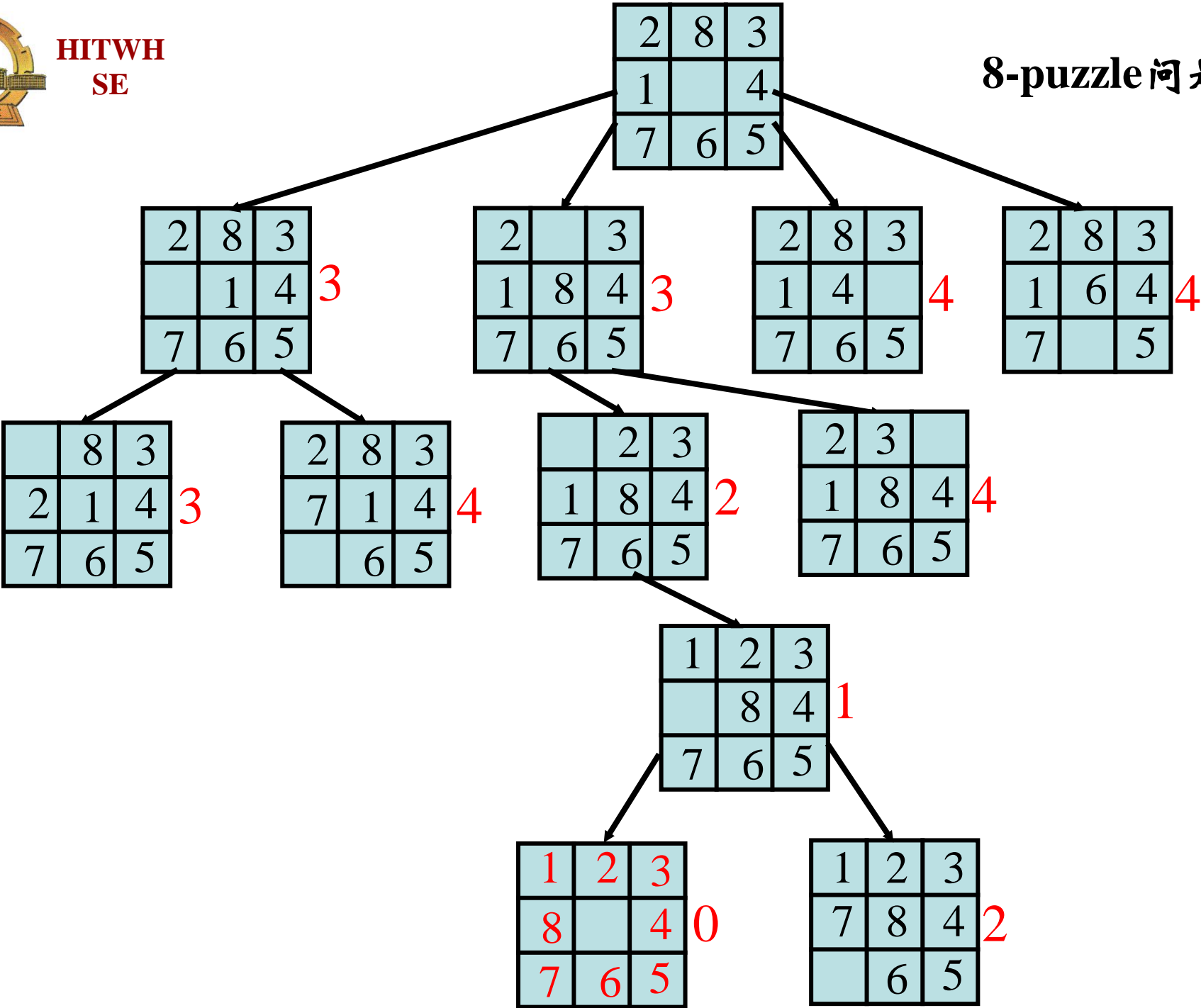
- **基本思想(也称最少代价优先, Least-cost-first)**
 - 结合深度优先和广度优先的优点于一个方法
 - 根据一个评价函数,在目前产生的所有节点中选择具有最小评价函数值的节点进行扩展.
 - 具有全局优化观念,而爬山策略仅具有局部优化观念.



• Best-First Search 算法

1. 使用评价函数构造一个堆 H , 首先构造由根组成的单元素堆;
2. If H 的根 r 是目标节点 Then 停止;
3. 从 H 中删除 r , 把 r 的子节点插入 H ;
4. If H 空 Then 失败 Else goto 2.

Heapsort方法见Intro. to Algo. 第II部分 第6章





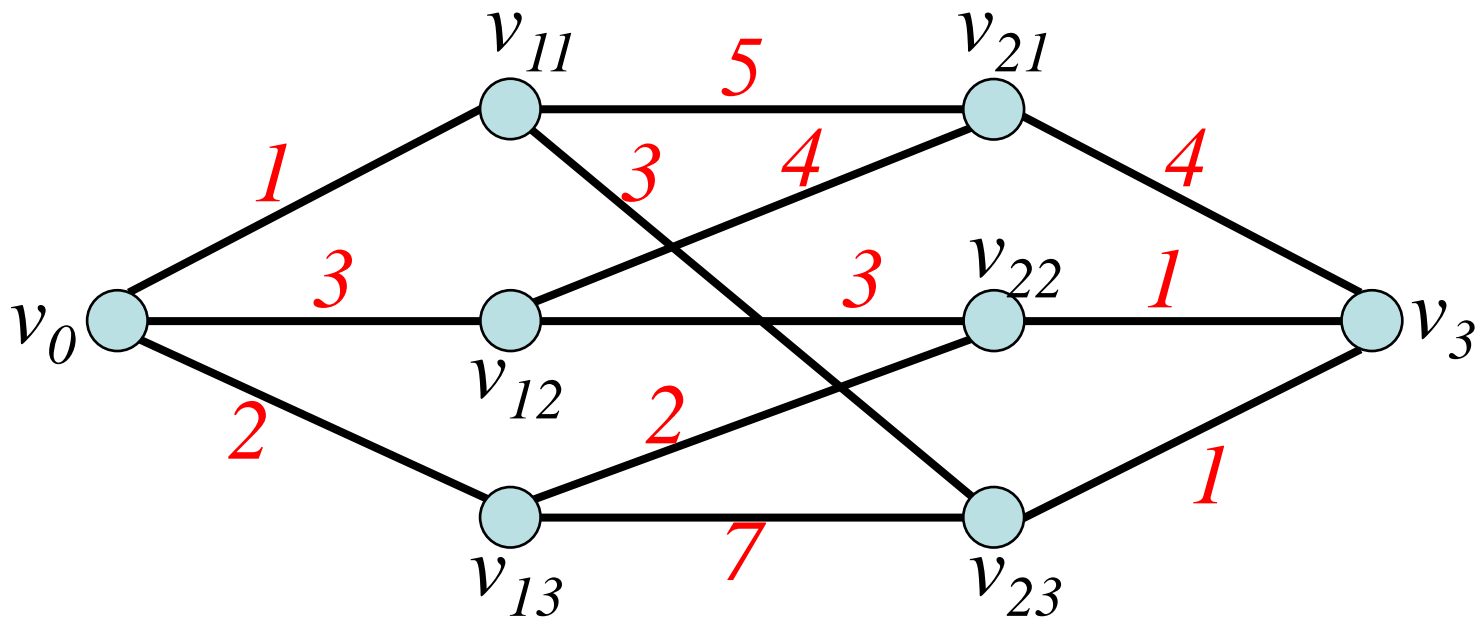
Branch-and-Bound Strategy

- 上述方法很难用于求解优化问题
- 分支界限策略可以有效地求解组合优化问题
- 分支界限基本思想
 - 迅速找到一个可行解
 - 将该可行解作为优化解的一个界限
 - 利用界限裁剪解空间, 提高求解的效率



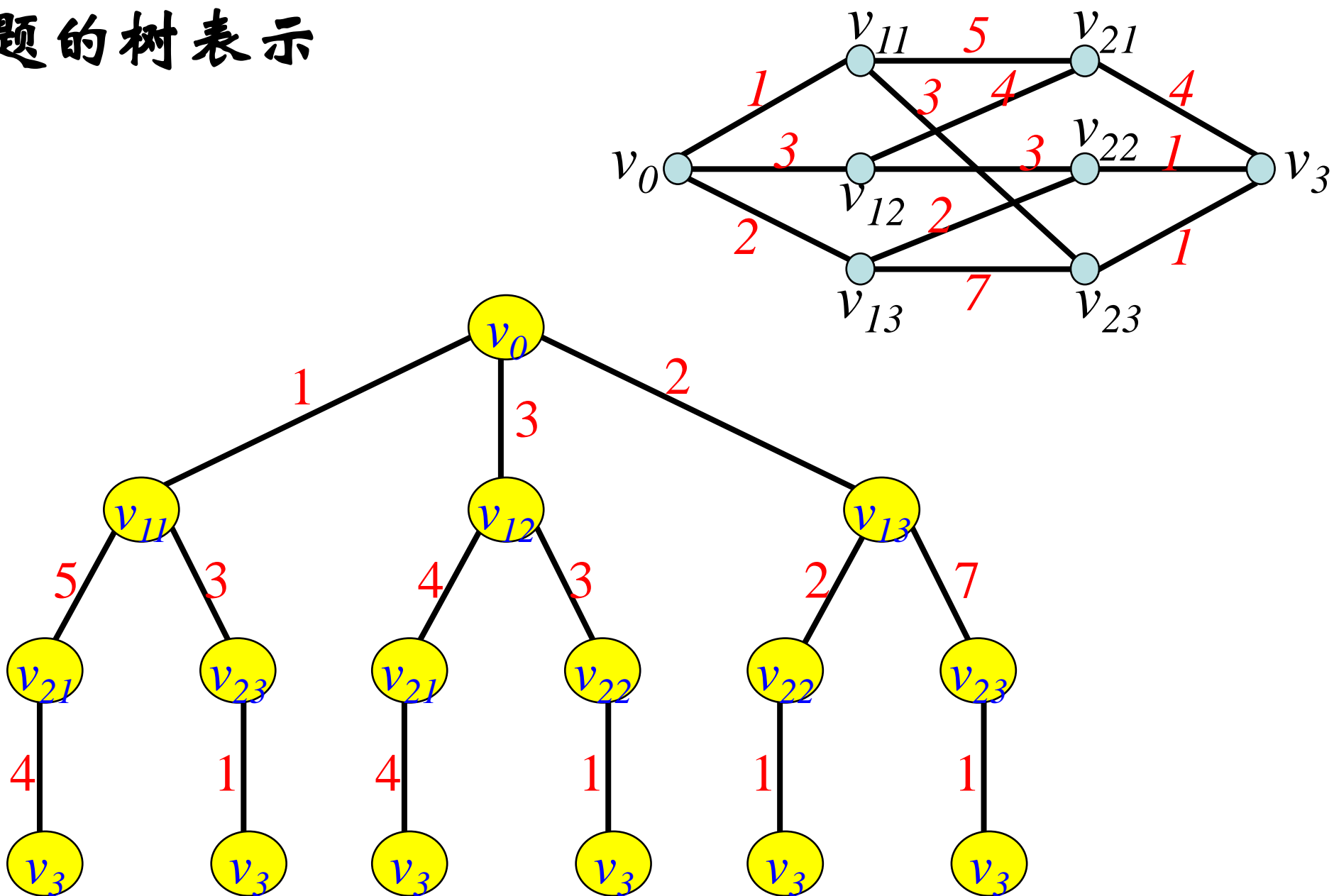
• 多阶段图搜索问题

— 输入：多阶段图



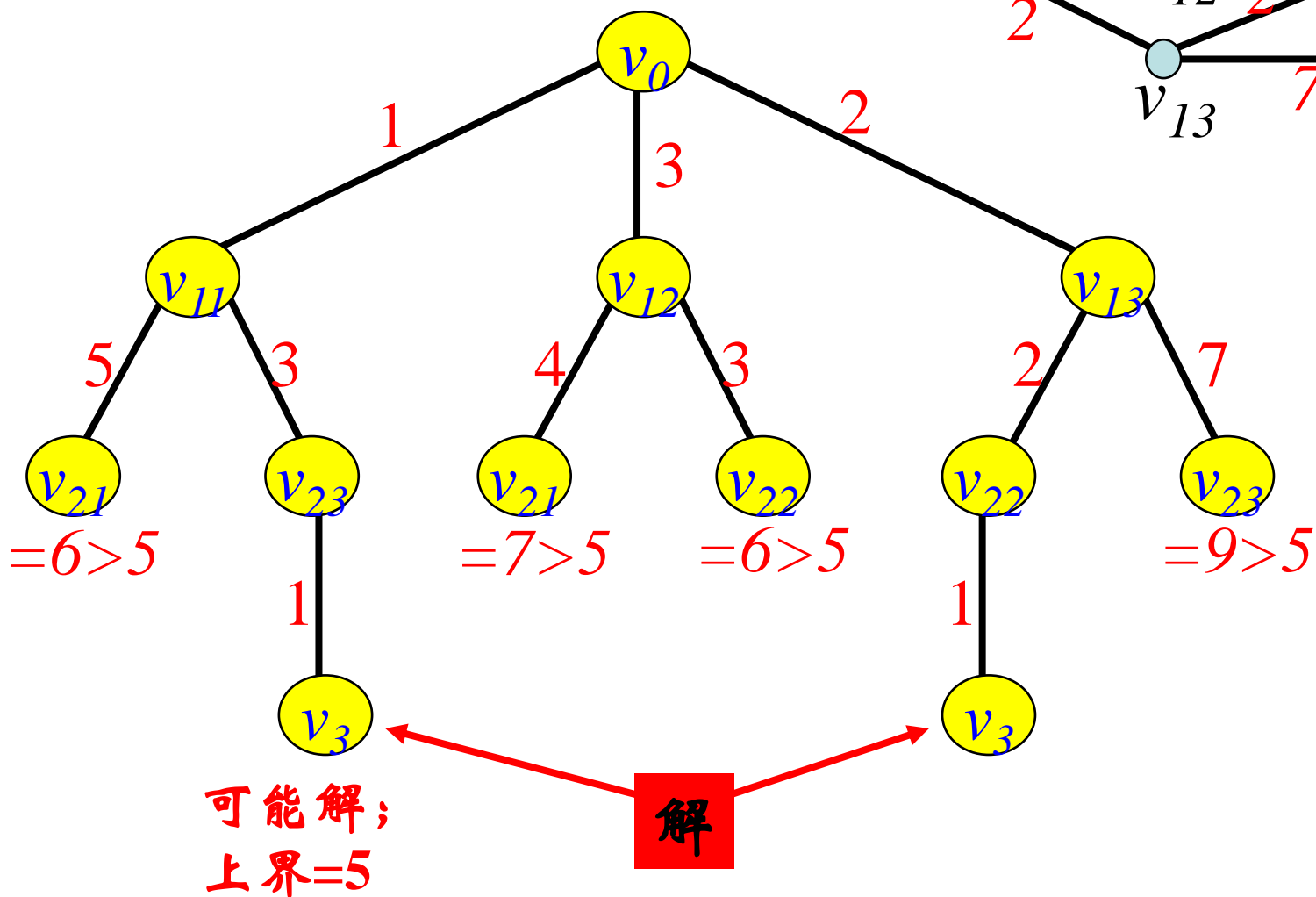
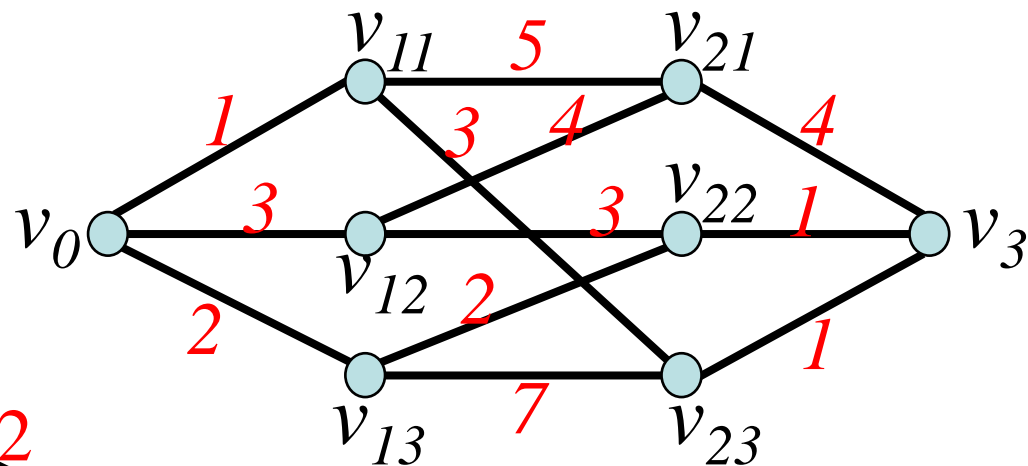
— 输出：从 v_0 到 v_3 的最短路径

- 问题的树表示



- 使用爬山策略进行分支界限搜索

评价函数：路径和





- 分支界限策略的原理
 - 产生分支的机制(使用前面的任意一种策略)
 - 爬山法
 - Best-First
 - 产生一个界限(可以通过发现可能解)
 - 进行分支界限搜索,即剪除不可能产生优化解的分支.



7.3 Personnel Assignment Problem

- 问题的定义
- 转换为树搜索问题
- 求解问题的分支界限搜索算法



问题的定义

• 输入

- 人的有序集合 $P = \{P_1, P_2, \dots, P_n\}$, $P_1 < P_2 < \dots < P_n$
- 工作的集合 $J = \{J_1, J_2, \dots, J_n\}$, J 是偏序集合 (\leq)
- 矩阵 $[C_{ij}]$, C_{ij} 是 P_i 被分配工作 J_j 的代价

• 输出

- 矩阵 $[X_{ij}]$, $X_{ij} = 1$ 表示 P_i 被分配 J_j , $\sum_{i,j} C_{ij} X_{ij}$ 最小
 - 不同人分配不同工作
 - 每个人被分配一种工作,
 - $f: P \rightarrow J$ 是工作分配函数, 满足:
 - 如果 $f(P_i) \leq f(P_j)$, 则 $P_i < P_j$,
 - 如果 $i \neq j$, $f(P_i) \neq f(P_j)$.



问题的定义

例. 给定 $P=\{P_1, P_2, P_3\}$, $J=\{J_1, J_2, J_3\}$, $J_1 \leq J_3, J_2 \leq J_3$.
 $P_1 \rightarrow J_1$ 、 $P_2 \rightarrow J_2$ 、 $P_3 \rightarrow J_3$ 是否为可能的解?

$P_1 \rightarrow J_1$ 、 $P_2 \rightarrow J_3$ 、 $P_3 \rightarrow J_2$ 是否为可能的解?



- 问题转化为树搜索问题
 - $J=\{J_1, J_2, \dots, J_n\}$ 的每个拓扑序列对应一个解
 - 用一个拓扑序列树把所有拓扑序列安排在一个树中，每个路径对应一个拓扑序列
 - 问题成为在树中搜索最小路径问题
- 构造拓扑序列树
- 使用分支界限法搜索拓扑序列树求解问题
- 改拓扑序列树构造算法为分支界限求解算法

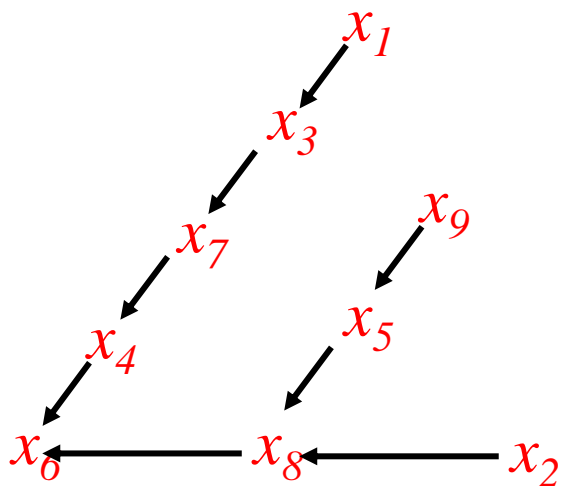


转换为树搜索问题

• 拓扑排序

- 输入: 偏序集合(S, \leq)
- 输出: S 的拓扑序列是 $\langle s_1, s_2, \dots, s_n \rangle$,
满足: 如果 $s_i \leq s_j$, 则 s_i 排在 s_j 的前面.

— 例



$x \rightarrow y$ 表示 $x \leq y$

拓扑排序:

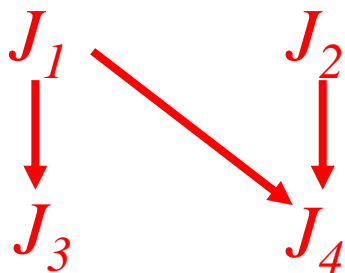
$x_1 \ x_3 \ x_7 \ x_4 \ x_9 \ x_5 \ x_2 \ x_8 \ x_6$

$x_2 \ x_9 \ x_5 \ x_1 \ x_3 \ x_7 \ x_4 \ x_8 \ x_6$

- 问题的解空间

命题1. $P_1 \rightarrow J_{k1}, P_2 \rightarrow J_{k2}, \dots, P_n \rightarrow J_{kn}$ 是一个可能解, 当且仅当 $J_{k1}, J_{k2}, \dots, J_{kn}$ 是一个拓扑排序的序列.

例. $P=\{P_1, P_2, P_3, P_4\}, J=\{J_1, J_2, J_3, J_4\}, J$ 的偏序如下



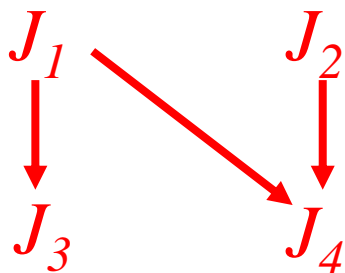
则 J 所有的拓扑排序序列有: $(J_1, J_2, J_3, J_4), (J_1, J_2, J_4, J_3), (J_1, J_3, J_2, J_4), (J_2, J_1, J_3, J_4), (J_2, J_1, J_4, J_3).$

(J_1, J_2, J_4, J_3) 对应于 $P_1 \rightarrow J_1, P_2 \rightarrow J_2, P_3 \rightarrow J_4, P_4 \rightarrow J_3$

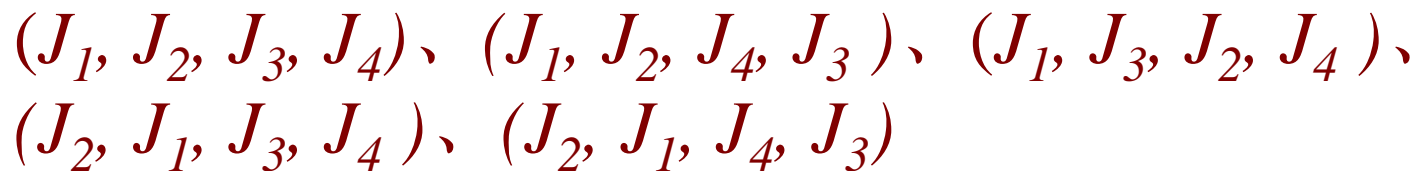
- 问题的解空间

命题1. $P_1 \rightarrow J_{k1}, P_2 \rightarrow J_{k2}, \dots, P_n \rightarrow J_{kn}$ 是一个可能解, 当且仅当 $J_{k1}, J_{k2}, \dots, J_{kn}$ 是一个拓扑排序的序列.

例. $P=\{P_1, P_2, P_3, P_4\}, J=\{J_1, J_2, J_3, J_4\}, J$ 的偏序如下



解空间是工作集合所有拓扑排序的序列集合,
每个序列对应于一个可能的解



-
- ```
graph TD; 0((0)) --> 1((1)); 0 --> 2((2)); 1 --> 2((2)); 1 --> 3((3)); 2 --> 3((3)); 2 --> 4((4)); 3 --> 2((2)); 4 --> 3((3)); 2 --> 4((4)); 2 --> 3((3)); 4 --> 3((3)); 3 --> 4((4));
```

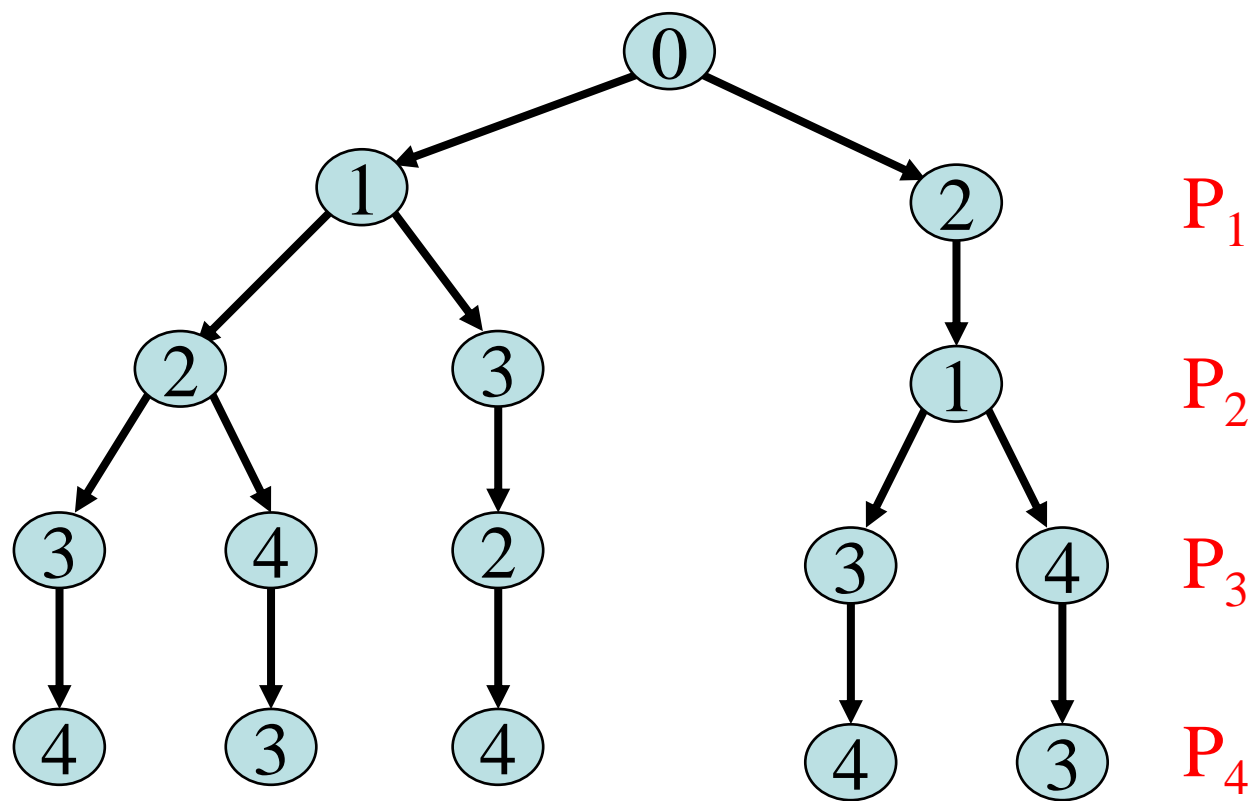
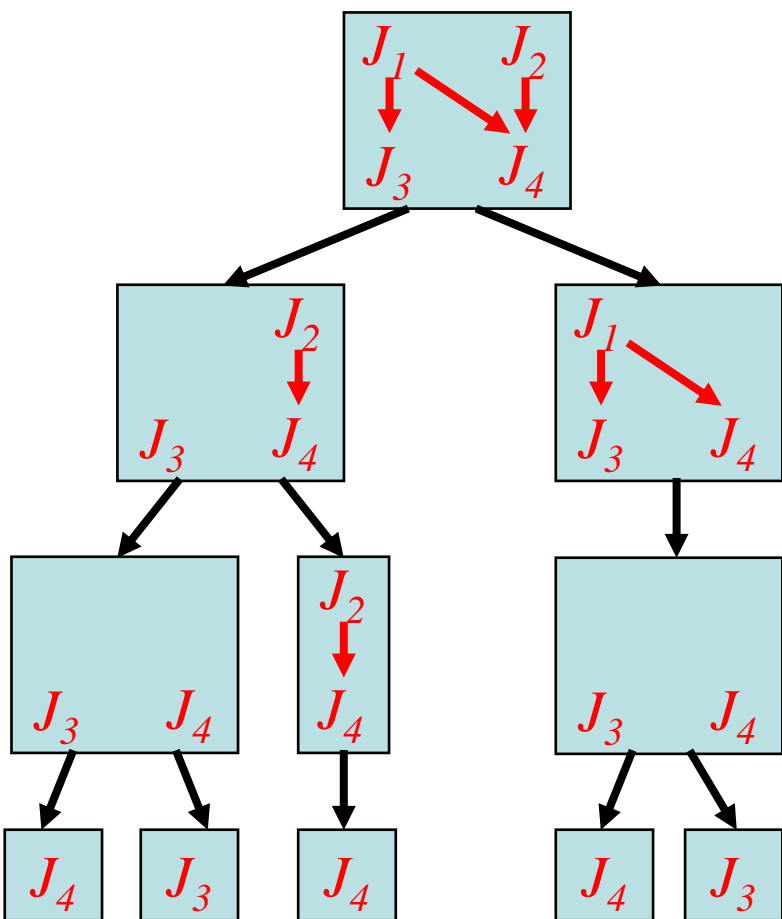
P<sub>4</sub>

## 如何由偏序关系直接生成拓扑序列树呢



$(J_1, J_2, J_3, J_4)$ 、 $(J_1, J_2, J_4, J_3)$ 、 $(J_1, J_3, J_2, J_4)$ 、  
 $(J_2, J_1, J_3, J_4)$ 、 $(J_2, J_1, J_4, J_3)$

• 拓扑序列树的生成算法的基本思想



$P_1$

$P_2$

$P_3$

$P_4$



## ● 拓扑序列树的生成算法

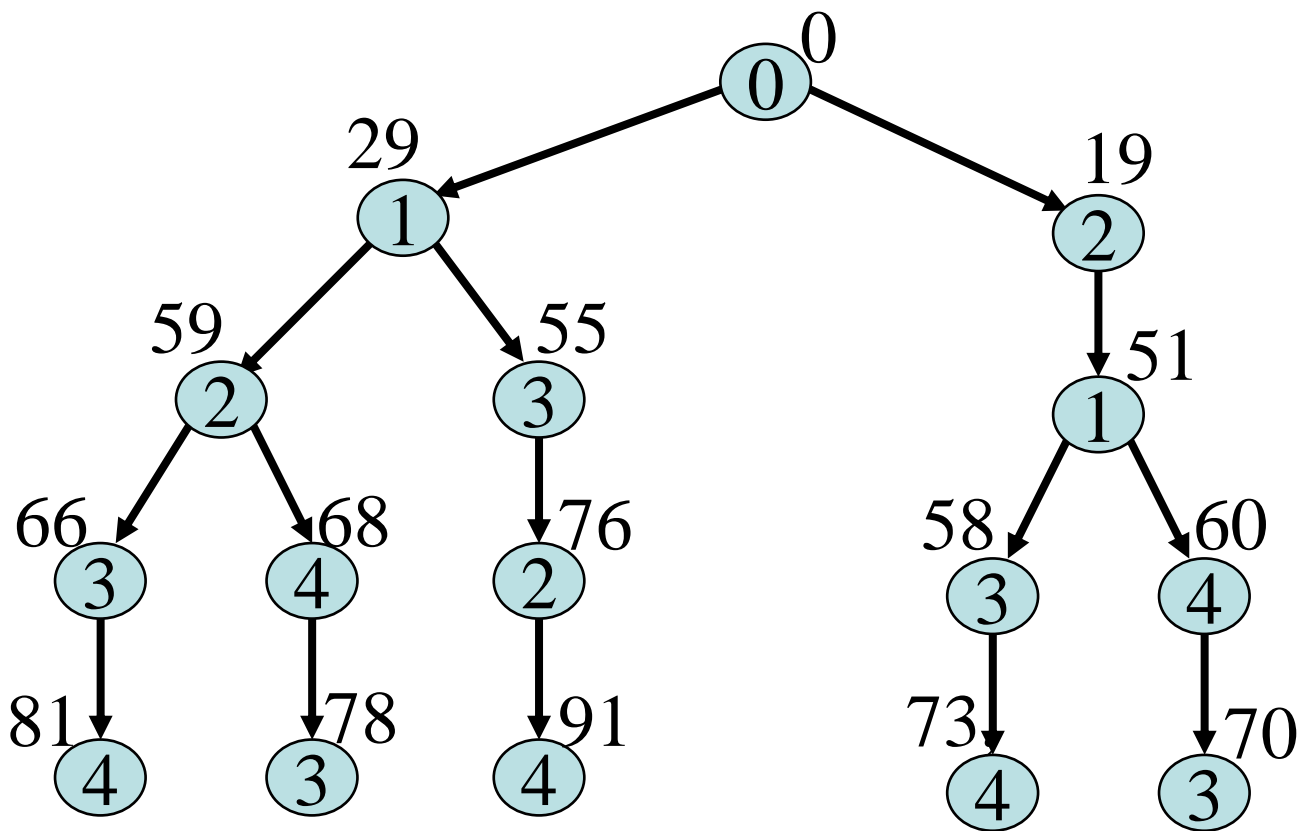
输入：偏序集合 $S$ , 树根 $root$ .

输出：由 $S$ 的所有拓扑排序序列构成的树.

1. 生成树根 $root$ ;
2. 选择偏序集中没有前序元素的所有元素, 作为 $root$ 的子结点;
3. For  $root$ 的每个子结点 $v$  Do
4.      $S = S_v - \{v\}$ ;
5.     把 $v$ 作为根, 递归地处理 $S_v$ .



## •解空间的加权树表示



|       | $J_1$ | $J_2$ | $J_3$ | $J_4$ |
|-------|-------|-------|-------|-------|
| $P_1$ | 29    | 19    | 17    | 12    |
| $P_2$ | 32    | 30    | 26    | 28    |
| $P_3$ | 3     | 21    | 7     | 9     |
| $P_4$ | 18    | 13    | 10    | 15    |

被分配  
的人员

1

2

3

4

裁剪效果不好，  
搜索代价高！！

如何改进？

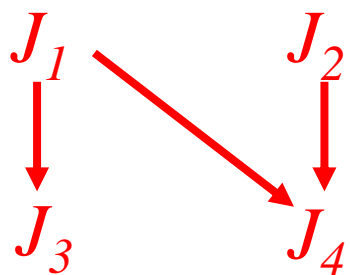




# 求解问题的分支界限搜索算法

## • 计算解的代价的下界

例. 
$$\begin{array}{c} P_1 \\ P_2 \\ P_3 \\ P_4 \end{array} \begin{array}{c} J_1 \\ J_2 \\ J_3 \\ J_4 \end{array} \begin{bmatrix} 29 & 19 & 17 & 12 \\ 32 & 30 & 26 & 28 \\ 3 & 21 & 7 & 9 \\ 18 & 13 & 10 & 15 \end{bmatrix} \begin{array}{c} -12 \\ -26 \\ -3 \\ -10 \end{array} \rightarrow \begin{array}{c} J_1 \\ J_2 \\ J_3 \\ J_4 \end{array} \begin{bmatrix} 17 & 7 & 5 & 0 \\ 6 & 4 & 0 & 2 \\ 0 & 18 & 4 & 6 \\ 8 & 3 & 0 & 5 \end{bmatrix} \begin{array}{c} \\ \\ \\ -3 \end{array} \rightarrow \begin{array}{c} J_1 \\ J_2 \\ J_3 \\ J_4 \end{array} \begin{bmatrix} 17 & 4 & 5 & 0 \\ 6 & 1 & 0 & 2 \\ 0 & 15 & 4 & 6 \\ 8 & 0 & 0 & 5 \end{bmatrix}$$



得到一个具有最小代价  $12+26+3+10+3=54$  的任务分配方案：

$$P_1 \rightarrow J_4, P_2 \rightarrow J_3, P_3 \rightarrow J_1, P_4 \rightarrow J_2$$

它不满足偏序约束，故不是可行解

由此得到可行解的代价下界=54



# 求解问题的分支界限搜索算法

- 计算解的代价的下界

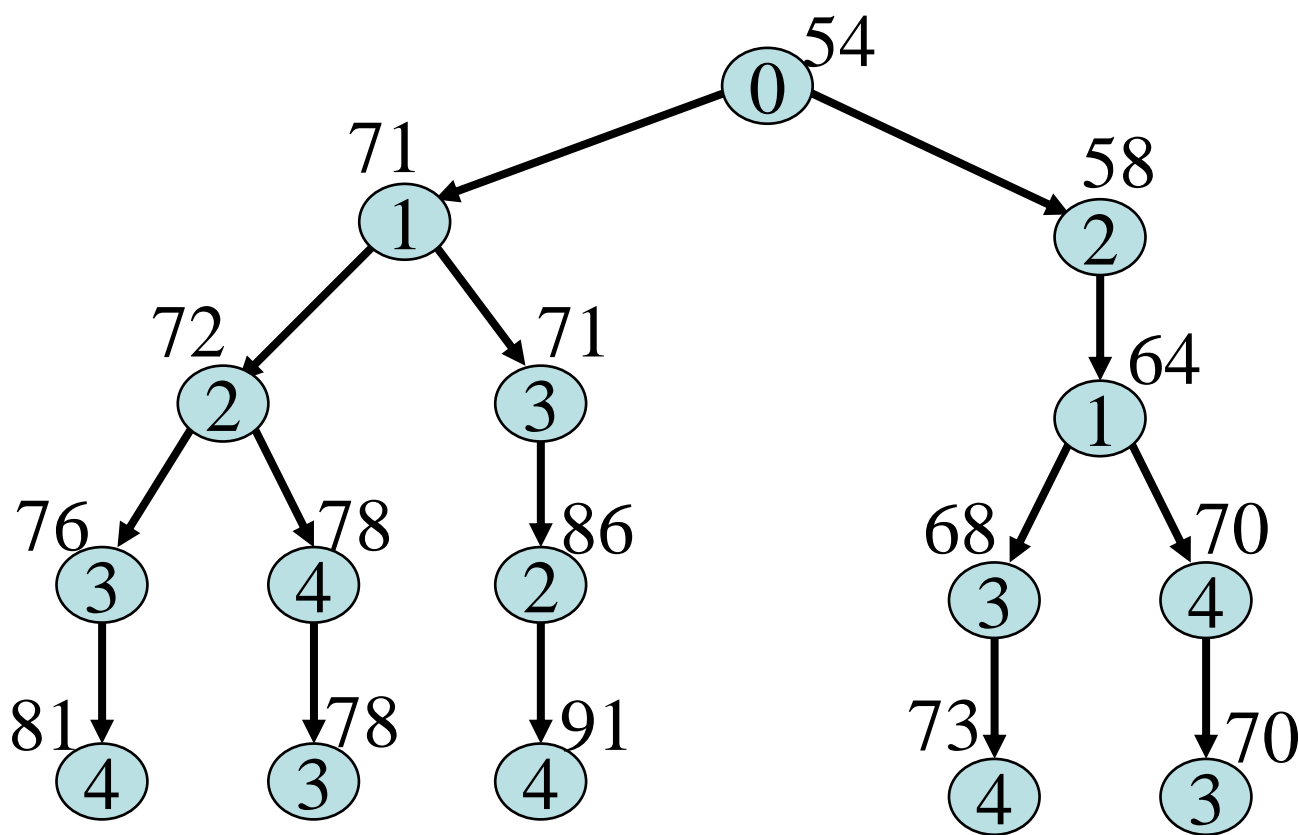
**命题2:** 把代价矩阵每行(列)各元素减去同一个数, 不影响优化解的求解.

- 代价矩阵每行(列)减去该行(列)的最小数, 使得每行和每列至少有一个零, 其余各元素非零
- 如此简化代价矩阵对解无影响  
(因为每个解代价都减去了一个相同的数)
- 每行和列减去的数的总和是解的下界.



## 改进后解空间的加权树表示

|       | $J_1$ | $J_2$ | $J_3$ | $J_4$ |
|-------|-------|-------|-------|-------|
| $P_1$ | 17    | 4     | 5     | 0     |
| $P_2$ | 6     | 1     | 0     | 2     |
| $P_3$ | 0     | 15    | 4     | 6     |
| $P_4$ | 8     | 0     | 0     | 5     |



被分配  
的人员

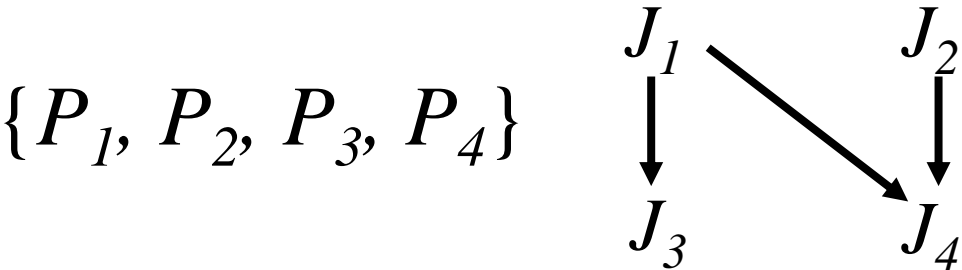
1

2

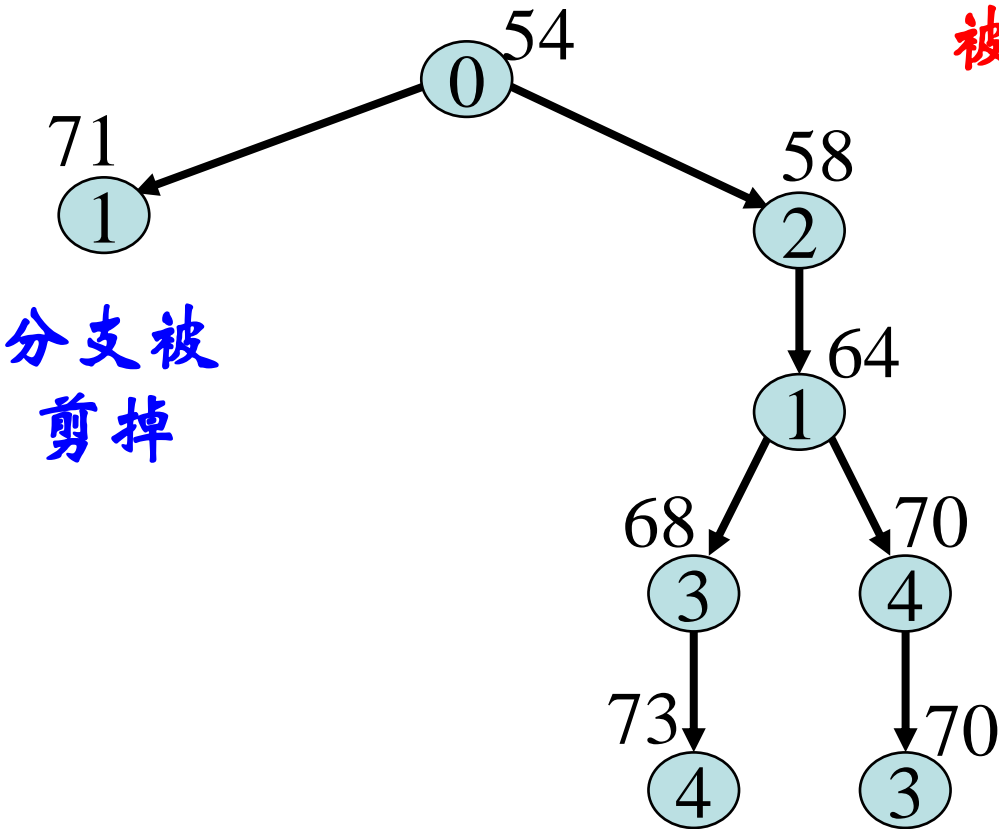
3

4

算法的思想(用爬山法)



|       | $J_1$ | $J_2$ | $J_3$ | $J_4$ |
|-------|-------|-------|-------|-------|
| $P_1$ | 17    | 4     | 5     | 0     |
| $P_2$ | 6     | 1     | 0     | 2     |
| $P_3$ | 0     | 15    | 4     | 6     |
| $P_4$ | 8     | 0     | 0     | 5     |



被分配到的人

1

2

3

4



- 分支界限搜索(使用爬山法)算法

1. 建立根结点,其权值为解代价下界;
2. 使用爬山法,类似于拓扑排序序列树生成算法求解问题,每产生一个结点,其权值为加工后的代价矩阵对应元素加其父结点权值;
3. 一旦发现一个可能解,将其代价作为界限,循环地进行分支界限搜索:剪掉不能导致优化解的子解,使用爬山法继续扩展新增节点,直至发现优化解.

修改拓扑排序算法, 可以写出严格的分支界限搜索算法



## 7.4 0/1 Knapsack

- 问题的定义
- 转换为树搜索问题
- 分支界限搜索算法



给定 $n$ 种物品和一个背包，物品 $i$ 的重量是 $w_i$ ，价值 $v_i$ ，背包承重为 $C$ ，问如何选择装入背包的物品，使装入背包中的物品的总价值最大？

对于每种物品只能选择完全装入或不装入，一个物品至多装入一次。

输入：  $C > 0, w_i > 0, v_i > 0, 1 \leq i \leq n$

输出：  $(x_1, x_2, \dots, x_n), x_i \in \{0, 1\}$ , 满足：

$$(1) \sum_{1 \leq i \leq n} w_i x_i \leq C,$$

$$(2) \sum_{1 \leq i \leq n} v_i x_i \text{ 最大.}$$



- 按照“价值重量比”降序排列 $n$ 个物品
- 空包为根;
- 用爬山法或Best-First递归地划分解空间, 得到二叉树
  - 树中第 $i$ 层的每个节点都代表了 $n$ 个物品中所有符合以下特征的子集:
    - 每个子集对应于序列中前 $i$ 个物品的一个特定选择
    - 这个特定选择是根据从根到该节点的一条路径确定的
      - 向左的分支表示包含下一个物品
      - 向右的分支表示不包含下一个物品
  - 每个节点中记录如下信息
    - 当前装入背包物品的总重量 $w$ 及总价值 $v$
    - 此时背包能够容纳的物品价值上界 $UB$





- 计算节点代价的上界 $UB$ :
  - $UB = v + (C - w) \times (v_{i+1} / w_{i+1})$ , 或者
  - $UB = v + UB'$ 
    - $UB'$ 是部分背包算法在子问题 $(\{i+1, i+2, \dots, n\}, C - w)$ 上的最优解代价
- 0/1背包问题的优化解与部分背包问题优化解之间的关系?
  - 0/1背包问题的优化解是部分背包问题的可行解
  - 部分背包问题的优化解是0/1背包问题优化解上界



HITWH  
SE

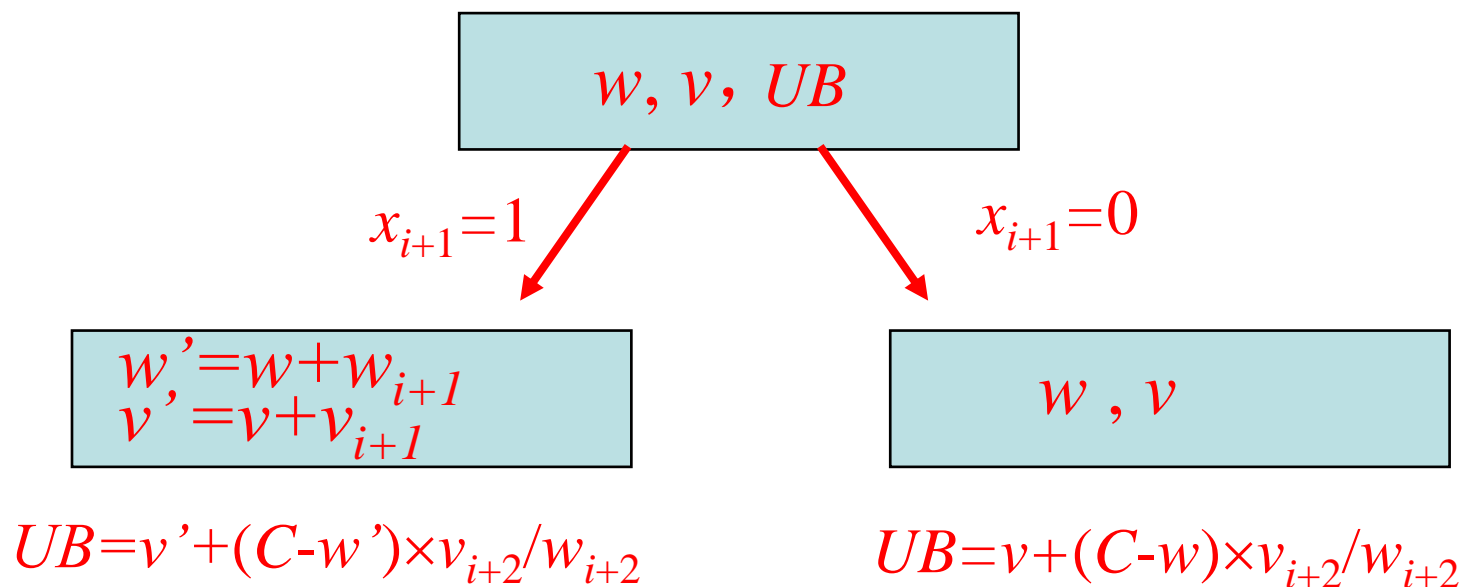
- 根节点及其代价上界

$$w=0, v=0$$

$$\begin{aligned} UB &= v + (C - w) \times v_1 / w_1 \\ &= C \times v_1 / w_1 \end{aligned}$$



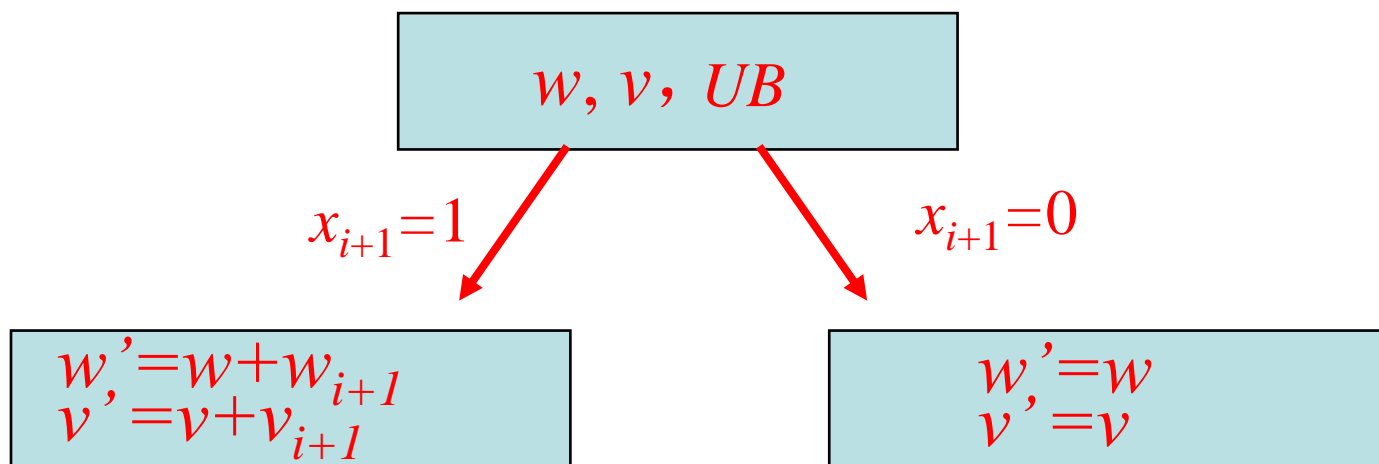
- 第 $i$ 层节点的搜索及其代价上界



根据左右儿子节点的代价的上界，确定下一次待搜索的节点



## • 第 $i$ 层节点的搜索及其代价上界



$$UB = v' + (C - w') \times v_{i+2} / w_{i+2}$$

$$UB = v + (C - w) \times v_{i+2} / w_{i+2}$$

## • 对于 $i+1$ 层节点:

- 若  $w' > C$ , 停止搜索, 由根到该节点对应路径不是可行解一部分
- 若其  $UB$  小于当前优化解下界, 则由根到该节点对应的路径一定不会产生最优解, 停止搜索
- 否则, 优先选择  $UB$  较大的节点, 继续爬山法搜索

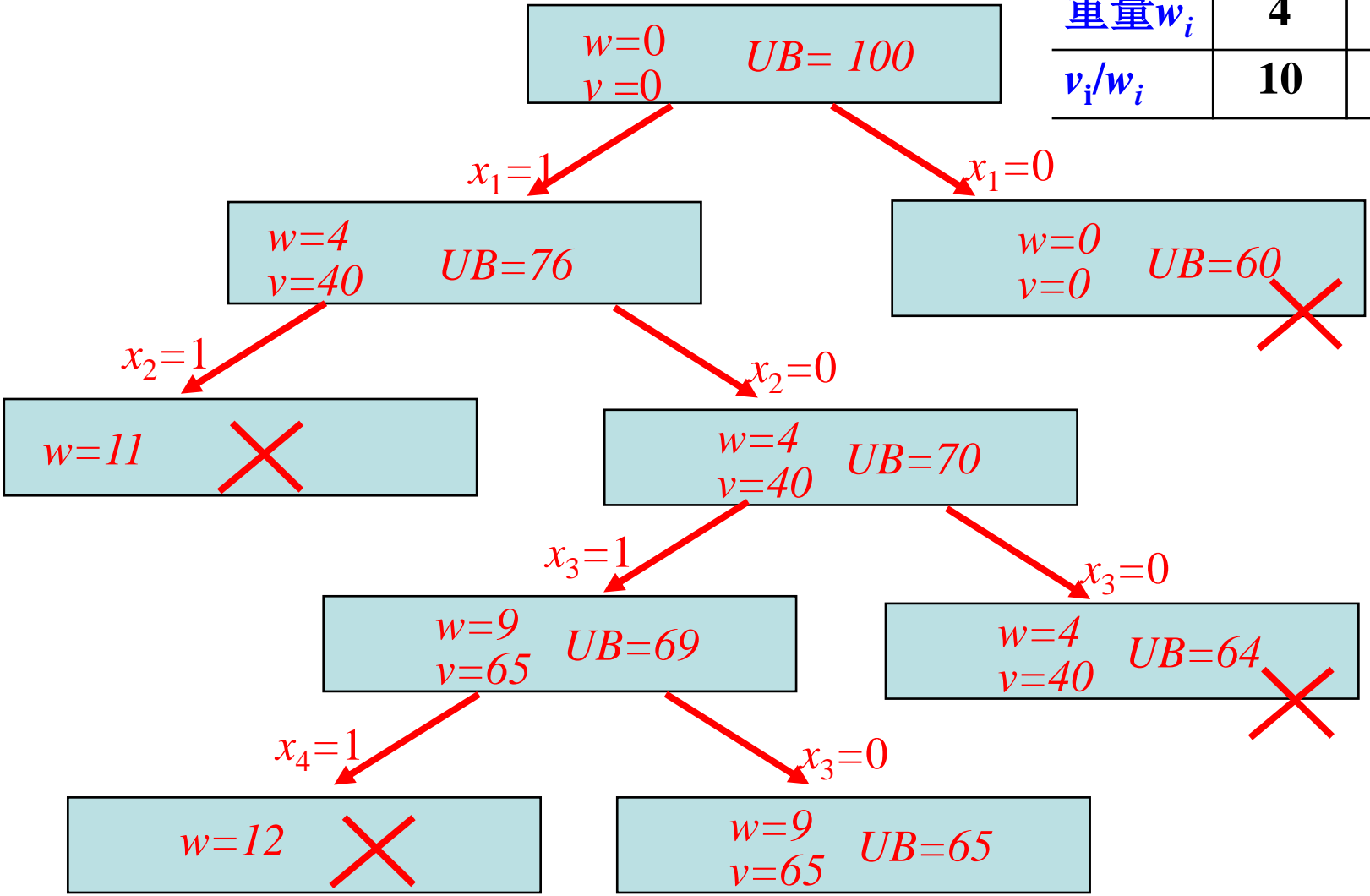


HITWH  
SE

● 举例

$C=10$

| 编号 <i>i</i>                        | 1  | 2  | 3  | 4  |
|------------------------------------|----|----|----|----|
| 价值 <i>v<sub>i</sub></i>            | 40 | 42 | 25 | 12 |
| 重量 <i>w<sub>i</sub></i>            | 4  | 7  | 5  | 3  |
| <i>v<sub>i</sub>/w<sub>i</sub></i> | 10 | 6  | 5  | 4  |



最优解!