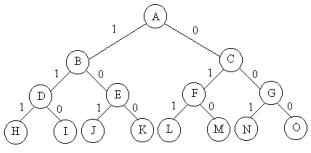
第六章 分支限界法

学习要点

- 理解分支限界法的剪枝搜索策略。
- 通过应用范例学习分支限界法的设计策略。

6.1 分支与限界法的基本思想



n=3时，0/1背包问题的状态空间树

分支与界限法的基本思想：在分支结点上，***预先分别估算***沿着它的各个孩子结点向下搜索的路径中，目标函数可能取得的“界”，然后，把它的这些孩子结点和它们可能取得的“界”保存在一张**结点表**中，再从表中选取“界”最大或最小的结点向下搜索。

分支限界法的基本思想

- 在分支结点上，预先分别估计沿着它的各个孩子结点向下搜索的路径中，目标函数可能取得的“界”，然后将它的这些孩子结点和它们可能取得的界保存在一张结点表中，再从表中选取“界”最大或最小的结点向下搜索。
- **为了在每次进行选取时选则取得极值的结点，因此用优先队列来维护这张表。**

- 在整个搜索过程中，每遇到一个活结点，就对它的各个孩子结点进行目标函数可能取得值得估算，然后把以此来更新**表结点**：丢弃不再需要的结点，加入新的结点。再从表中选取“界”取极值的结点，并重复上述过程。
- **随着过程的不断深入，结点表中所估算的目标函数的极值，越来越接近问题的解。当搜索到一个叶子结点时，如果对该结点所估算的目标函数的值就是结点表中最大或最小值，那么沿叶子结点到根结点的路径确定的解就是问题的解。**

分支限界法不再像回溯法那样盲目的进行搜索，也不是遇到死区才回头。而是依据结点表中不断更新的信息，不断调整搜索的方向，有选择，有目标地进行搜索，回溯也不是单纯的沿着父结点一层层的向上回溯，而是依据结点表中的信息回溯。

分支限界法与回溯法比较

（1）求解目标：回溯法的求解目标是找出解空间树中满足约束条件的所有解，而分支限界法的求解目标则是找出满足约束条件的一个解，或是在满足约束条件的解中找出在某种意义下的最优解。

（2）搜索方式的不同：回溯法以***深度***优先的方式搜索解空间树，而分支限界法则以***广度***优先或***以最小耗费优先***的方式搜索解空间树。

7

两种典型的求解方法

第一种解法

- 假定问题的解向量为$X=(x_1,x_2,...,x_n)$，其中x的取值范围为某个有穷集S_i，$|S_i|=n_i,1<=i<=n$。

8

- 在整个搜索过程中，每遇到一个活结点，就对它的各个孩子结点进行目标函数可能取得值得估算，以此来更新**表结点**：丢弃不再需要的结点，加入新的结点。再从表中选取“界”取极值的结点，并重复上述过程。
- 当从根结点开始向下搜索时，由n_1个孩子结点分别构成n_1棵子树的根，从而组成部分解x_1的n_1种可能取值方式。对这n_1个孩子结点分别估计他们可能取得的目标函数的值$bound(x_1)$。如果是求最小值，就把$bound(x_1)$称为该孩子的下界，意思是沿着孩子结点向下搜索所可能取得的值得最小值不应小于$bound(x_1)$。

9

- 假如$X=(x_1,x_2,...,x_k)$是沿着该孩子结点一层层向下搜索所得的部分解，那么应有：

  $bound(x_1)<= bound(x_1,x_2)<=....<= bound(x_1,x_2,...,x_k)$
- 在求得n_1个孩子结点的下界之后，把它们保存在结点表中，并删除根结点在结点表中的登记项。这时在结点表中登记的结点及其 相应的下界$bound(x_1)$有n_1个，于是从结点表中选取下界$bound(x_1)$最小的孩子结点作为 下一次搜索的起点。

10

分支界限法的另外一种方法

当从根结点开始向下搜索时，预先通过某种方式的处理，从众多孩子结点中挑选一个孩子结点作为搜索树的一个分支结点，而把去掉这个结点之后的其他孩子结点的集合，作为搜索树的另外一个分支结点。

11

- 当从根结点开始向下搜索时，不是如第一种方法那样，对这n_1个孩子结点分别估计他们可能取得的目标函数的值$bound(x_1)$，再选取最大或最小的结点进行分支搜索，**而是预先通过某种方式的处理，从众多孩子结点中选择一个孩子结点作为搜索树的一个分支结点，而把去掉这个结点之后的其他孩子结点集合，作为搜索树的另外一个分支结点。**令$bound(x_1)$是选择孩子结点进行分支搜索时所可能取得的目标函数的界，令$bound(\bar{x}_1)$是不选择该孩子结点时所可能取得的目标函数的界。

12

- 然后选取界最大或最小的分支结点，继续上述处理，直到最后得到界最大或最小的结点为止。
- 该方法每进行一次分支选择，只计算两个目标函数的界。所生成的搜索树是一棵二叉树。
- 关键：如何选择分支和如何计算目标函数的上下界。

6.2 货郎担问题

- 费用矩阵

	0	1	2	3	4
0	∞	25	41	32	28
1	5	∞	18	31	26
2	20	16	∞	7	1
3	10	51	25	∞	6
4	23	9	7	11	∞

6.2.1 费用矩阵的行归约（列归约）

- 费用矩阵c的第i行（或第j列）中的每个元素减去一个正常数lh_i（或ch_j），得到一个新的费用矩阵，使得新矩阵中第i行（或第j列）中的最小值为0，称为费用矩阵的行归约（列归约），称lh_i为行归约常数，称ch_j为列归约常数。

	0	1	2	3	4
0	∞	25	41	32	28
1	5	∞	18	31	26
2	20	16	∞	7	1
3	10	51	25	∞	6
4	23	9	7	11	∞

lh_0=25
lh_1=5
lh_2=1
lh_3=6
lh_4=7

	0	1	2	3	4
0	∞	0	16	7	3
1	0	∞	13	26	21
2	19	15	∞	6	0
3	4	45	19	∞	0
4	16	2	0	4	∞

6.2.2 矩阵的归约常数

- 对费用矩阵的每一行和每一列都进行归约和列归约，得到一个新的费用矩阵，***使得新矩阵中每一行和每一列都至少有一个元素为0，***称为费用矩阵的归约。常数h:

$$h=\sum_{i=0}^{n-1} lh_i+\sum_{j=0}^{n-1} ch_j \qquad (6.2.1)$$

为矩阵的归约常数。

	0	1	2	3	4
0	∞	25	41	32	28
1	5	∞	18	31	26
2	20	16	∞	7	1
3	10	51	25	∞	6
4	23	9	7	11	∞

lh_0=25
lh_1=5
lh_2=1
lh_3=6
lh_4=7

	0	1	2	3	4
0	∞	0	16	7	3
1	0	∞	13	26	21
2	19	15	∞	6	0
3	4	45	19	∞	0
4	16	2	0	**4**	∞

ch_3=4

	0	1	2	3	4
0	∞	0	16	3	3
1	0	∞	13	22	21
2	19	15	∞	2	0
3	4	45	19	∞	0
4	16	2	0	0	∞

$$h=\sum_{i=0}^{n-1} lh_i+\sum_{j=0}^{n-1} ch_j$$
$$=25+5+1+6+7+4$$
$$=48$$

- ***引理***：令$G=(V,E)$是一个有向带权图，l是图G的一条哈密尔顿回路，C是图G的费用矩阵，则回路上的边对应于费用矩阵C中每行每列各一个元素。

	0	1	2	3	4
0	∞	25	41	32	28
1	5	∞	18	31	26
2	20	16	∞	7	1
3	10	51	25	∞	6
4	23	9	7	11	∞

$l=v_0,v_3,v_1,v_4,v_2,v_0$

$c_{03},c_{31},c_{14},c_{42},c_{20}$

v_i是回路中的任意一个顶点($0\le i\le n-1$)，它在回路中只有一条出边，该边对应于费用矩阵中第i行中的一个元素。根据哈密尔顿回路的定义，v_i在回路中只出现一次，因此第i行中有且仅有一个元素与其对应。

- ***引理***：令$G=(V,E)$是一个有向带权图，l是图G的一条哈密尔顿回路，c是图G的费用矩阵，则回路上的边对应于费用矩阵c中每行每列各一个元素。

	0	1	2	3	4
0	∞	25	41	32	28
1	5	∞	18	31	26
2	20	16	∞	7	1
3	10	51	25	∞	6
4	23	9	7	11	∞

$l=v_0,v_3,v_1,v_4,v_2,v_0$

$c_{03},c_{31},c_{14},c_{42},c_{20}$

设v_i对应于费用矩阵中第i行中第j列的一个元素，即表示在回路中是从v_i出发到达v_j，根据哈密尔顿回路的定义，v_j在回路中只出现一次，因此第j列中不能再有元素出现。

19

- 定理1：令$G=(V,E)$是一个有向带权图，l是图G的一条哈密尔顿回路，c是图G的费用矩阵，$w(l)$是以费用矩阵c计算的这条回路的费用，如果矩阵$\bar{c}$是费用矩阵c的归约矩阵，归约常数为h，$\bar{w}(l)$是以费用矩阵$\bar{c}$计算的这条回路的费用，则有：

$$w(l)=\bar{w}(l)+h \qquad (6.2.2)$$

20

- 定理2：令$G=(V,E)$是一个有向带权图，l是图G的一条哈密尔顿回路，c是图G的费用矩阵，$\bar{c}$是费用矩阵c的归约矩阵，图$\bar{G}$是费用矩阵相对应的图，令$\bar{c}$是图$\bar{G}$的邻接矩阵，则l是图G的一条哈密尔顿回路。

21

- 根据定理1和定理2，求解图G的最短哈密尔顿回路问题，可以先求图G费用矩阵c的归约矩阵$\bar{c}$，得到归约常数h后，再转换求与费用矩阵$\bar{c}$对应的图$\bar{G}$的最短哈密尔顿回路问题。且图G的最短哈密尔顿回路的费用，最少不会少于归约常数h。因此，归约常数h即为状态空间树中根结点的下界。

22

6.2.3 界限的确定

- 搜索方法：

当从根结点开始向下搜索时，预先通过某种方式的处理，从众多孩子结点中挑选一个孩子结点作为搜索树的一个分支结点Y，而把去掉这个结点之后的其他孩子结点的集合，作为搜索树的另外一个分支结点$\bar{Y}$。

23

- Y的界限：

	0	1	2	3	4
0	∞	0	16	3	3
1	0	∞	13	22	21
2	19	15	∞	2	0
3	4	45	19	∞	0
4	16	2	0	0	∞

根结点X的下界：归约常数h。

如果选择从v_1出发到达v_0，则该回路的边必然包括$\bar{c}_{10}$，根据引理1，费用矩阵中第1行和第0列的元素在今后的计算中不再起作用，可以将它们删除。

同时，回路中也不再有从v_0出发到达v_1的边，可以令$\bar{c}_{10}=\infty$

24

	0	1	2	3	4
0	∞	0	16	3	3
1	0	∞	13	22	21
2	19	15	∞	2	0
3	4	45	19	∞	0
4	16	2	0	0	∞

	1	2	3	4
0	∞	16	3	3
2	15	∞	2	0
3	45	19	∞	0
4	2	0	0	∞

	1	2	3	4
0	∞	13	0	0
2	13	∞	2	0
3	43	19	∞	0
4	0	0	0	∞

lh_0=3

ch_1=2

沿着v_1出发到达v_0的回路，其费用肯定不会小于48+2+3=53

$W(Y)=w(X)+h$ （6.2.3）

- $\bar{Y}$的界限：

	0	1	2	3	4
0	∞	0	16	3	3
1	0	∞	13	22	21
2	19	15	∞	2	0
3	4	45	19	∞	0
4	16	2	0	0	∞

根结点X的下界：归约常数h。

因为回路中不再包含边v_iv_j，则可令$\bar{c}_{ij}=\infty$，同时，在后续的选择中，必然包含第i行的最小元素和第j列的最小元素（除c_{ij}之外的）。

令：$d_{ij}=\min_{0\le k\le n-1,k\ne j}\{c_{ik}\}+\min_{0\le k\le n-1,k\ne i}\{c_{kj}\}$ (6.2.4)

则结点$\bar{Y}$的下界为：$w(\bar{Y})=w(X)+d_{ij}$ (6.2.5)

	0	1	2	3	4
0	∞	0	16	3	3
1	0	∞	13	22	21
2	19	15	∞	2	0
3	4	45	19	∞	0
4	16	2	0	0	∞

$w(\bar{Y})=w(X)+d_{ij}=48+4+13=65$

6.2.4 分支的选择

- 在父结点的归约矩阵中，每行每列至少包含一个值为0的元素。于是分支的选取按下面的两个思路进行：

（1）沿着c_{ij}=0的方向选取，使所选路线尽可能短。

（2）沿着d_{ij}最大的方向选取，使得$w(\bar{Y})$尽可能大。

令S是费用矩阵中c_{ij}=0的元素集合，则：

$$D_{kl}=\max_{S}\{d_{ij}\}$$

$$w(\bar{Y})=w(X)+D_{kl}$$

	0	1	2	3	4
0	∞	0	16	3	3
1	0	∞	13	22	21
2	19	15	∞	2	0
3	4	45	19	∞	0
4	16	2	0	0	∞

c_{01}=0， c_{10}=0， c_{24}=0， c_{34}=0， c_{42}=0， c_{43}=0

d_{01}=3+2=5， d_{10}=13+4=17， d_{24}=2+0=2，

d_{34}=4+0=4， d_{42}=0+13=13， d_{43}=0+2=2

6.2.5 求解过程

- 用优先队列存储搜索的结点表，求解过程如下：

（1）初始化优先队列为空；

（2）建立父结点X，费用矩阵为X.c，费用矩阵阶数X.k=n，归约X.c，得到归约常数h，则父结点的下界为X.w=h;

（3）由（6.2.4），计算全部的c_{ij}=0的d_{ij}。

（4）由（6.2.6），计算D_{kl}，选取边v_kv_l为分支方向，将v_kv_l加入到回路边表中

（5）建立孩子结点$\bar{Y}$，将$\bar{Y}$的下界插入优先队列中。

（6）建立孩子结点Y，若Y.k=2，直接判断最短回路的两条边，使得Y.k=0，否则将Y的下界插入优先队列中。

（7）取优先队列头元素作为X结点，若X.k=0，算法结束，否则，转步骤（3）。

	0	1	2	3	4
0	∞	0	16	3	3
1	0	∞	13	22	21
2	19	15	∞	2	0
3	4	45	19	∞	0
4	16	2	0	0	∞

A W(Y)=48

c_{01}=0， c_{10}=0， c_{24}=0，
c_{34}=0， c_{42}=0， c_{43}=0

d_{01}=3+2=5， d_{10}=13+4=17，
d_{24}=2+0=2， d_{34}=4+0=4，
d_{42}=0+13=13， d_{43}=0+2=2

48 A

回路边表：

31

	0	1	2	3	4
0	∞	0	16	3	3
1	0	∞	13	22	21
2	19	15	∞	2	0
3	4	45	19	∞	0
4	16	2	0	0	∞

A W(Y)=48

(1,0)

	1	2	3	4
0	∞	16	3	3
2	15	∞	2	0
3	45	19	∞	0
4	2	0	0	∞

B

(1,0)

	0	1	2	3	4
0	∞	0	16	3	3
1	∞	∞	13	22	21
2	19	15	∞	2	0
3	4	45	19	∞	0
4	16	2	0	0	∞

C

选取v_1v_0为Y结点

回路边表： v_1v_0

32

	0	1	2	3	4
0	∞	0	16	3	3
1	0	∞	13	22	21
2	19	15	∞	2	0
3	4	45	19	∞	0
4	16	2	0	0	∞

A W(Y)=48

	1	2	3	4
0	∞	13	0	0
2	13	∞	2	0
3	43	19	∞	0
4	0	0	0	∞

lh_0=3
ch_1=2

B

	0	1	2	3	4
0	∞	0	16	3	3
1	∞	∞	0	9	8
2	15	15	∞	2	0
3	0	45	19	∞	0
4	12	2	0	0	∞

lh_1=13
ch_0=4

C

选取v_1v_0为Y结点
并进行归约

33

	0	1	2	3	4
0	∞	0	16	3	3
1	0	∞	13	22	21
2	19	15	∞	2	0
3	4	45	19	∞	0
4	16	2	0	0	∞

A

	1	2	3	4
0	∞	13	0	0
2	13	∞	2	0
3	43	19	∞	0
4	0	0	0	∞

lh_0=3
ch_1=2

B W(Y)=53

53 B

65 C

	0	1	2	3	4
0	∞	0	16	3	3
1	∞	∞	0	9	8
2	15	15	∞	2	0
3	0	45	19	∞	0
4	12	2	0	0	∞

lh_1=13
ch_0=4

C W(Y)=65

建立优先队列

34

	1	2	3	4
0	∞	13	0	0
2	13	∞	2	0
3	43	19	∞	0
4	0	0	0	∞

B W(Y)=53

65 C

c_{03}=0， c_{04}=0， c_{24}=0，
c_{34}=0， c_{41}=0， c_{42}=0，
c_{43}=0

d_{03}=0+0=0， d_{04}=0+0=0，
d_{24}=2+0=2， d_{34}=19+0=19，
d_{41}=0+13=13， d_{42}=0+13=13，
d_{43}=0+0=0

选取v_3v_4为Y结点

35

	1	2	3	4
0	∞	13	0	0
2	13	∞	2	0
3	43	19	∞	0
4	0	0	0	∞

B W(Y)=53

(3,4)

	1	2	3
0	∞	13	0
2	13	∞	2
4	0	0	∞

D W(Y)=55

	1	2	3
0	∞	13	0
2	11	∞	0
4	0	0	∞

ch_2=2

(3,4)

65 C

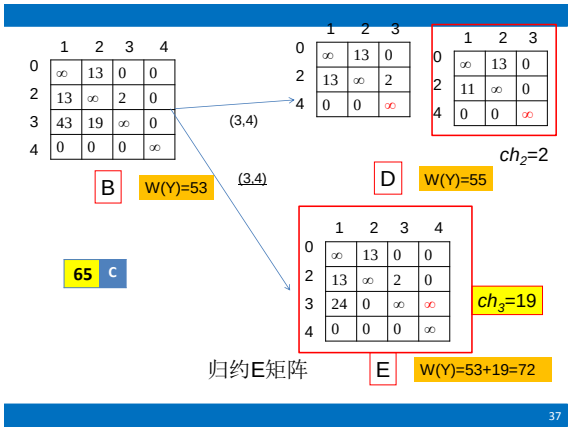
	1	2	3	4
0	∞	13	0	0
2	13	∞	2	0
3	43	19	∞	∞
4	0	0	0	∞

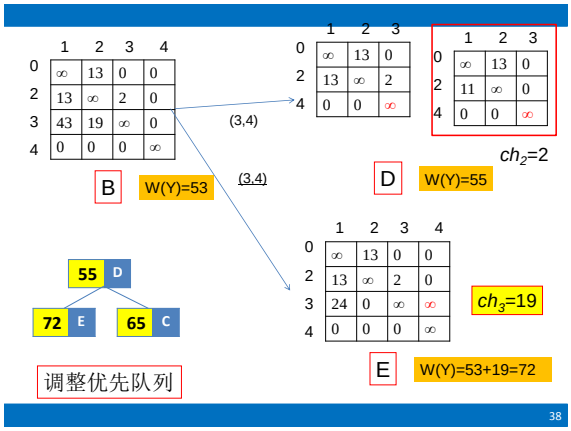
归约E矩阵

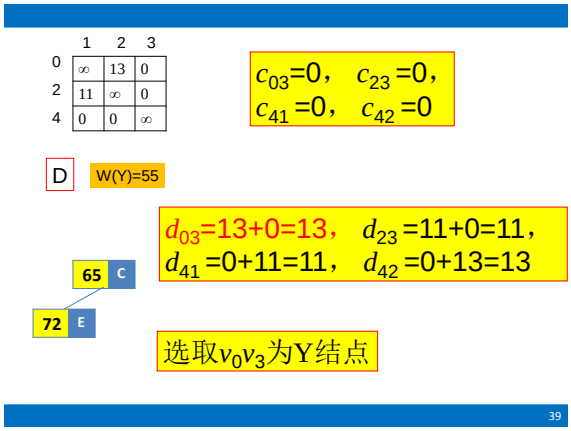
E W(Y)=53

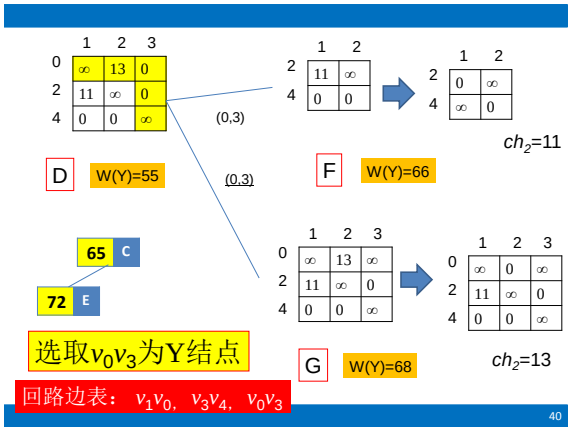
回路边表： v_1v_0、v_3v_4

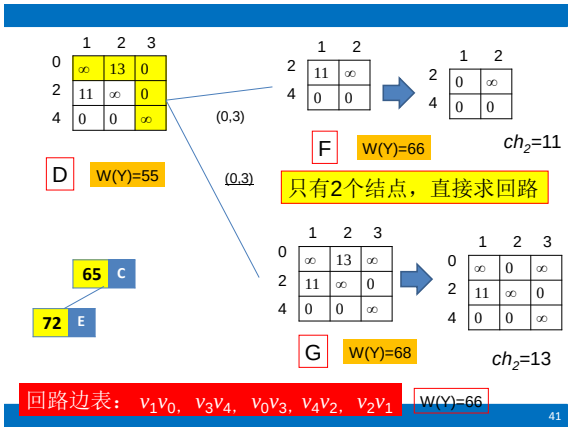
36

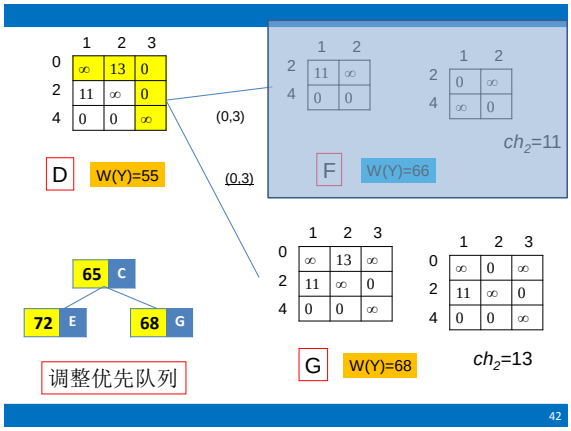

B
W(Y)=53
(3,4)
D
W(Y)=55
ch2=2
65 C
ch3=19
归约E矩阵
E
W(Y)=53+19=72
37


B
W(Y)=53
(3,4)
D
W(Y)=55
ch2=2
55 D
72 E
65 C
ch3=19
调整优先队列
E
W(Y)=53+19=72
38


D
W(Y)=55
c03=0, c23=0,
c41=0, c42=0
d03=13+0=13, d23=11+0=11,
d41=0+11=11, d42=0+13=13
65 C
72 E
选取v0v3为Y结点
39


D
W(Y)=55
(0,3)
F
W(Y)=66
ch2=11
65 C
72 E
选取v0v3为Y结点
G
W(Y)=68
ch2=13
回路边表： v1v0, v3v4, v0v3
40


D
W(Y)=55
(0,3)
F
W(Y)=66
ch2=11
只有2个结点，直接求回路
65 C
72 E
G
W(Y)=68
ch2=13
回路边表： v1v0, v3v4, v0v3, v4v2, v2v1
W(Y)=66
41


D
W(Y)=55
(0,3)
F
W(Y)=66
ch2=11
65 C
72 E
68 G
调整优先队列
G
W(Y)=68
ch2=13
42

	0	1	2	3	4
0	∞	0	16	3	3
1	∞	∞	0	9	8
2	15	15	∞	2	0
3	0	45	19	∞	0
4	12	2	0	0	∞

C W(Y)=65

开始新的搜索！

68 G

72 E

回路边表： v_1v_0, v_3v_4, v_0v_3, v_4v_2, v_2v_1 W(Y)=66

43

	0	1	2	3	4
0	∞	0	16	3	3
1	∞	∞	0	9	8
2	15	15	∞	2	0
3	0	45	19	∞	0
4	12	2	0	0	∞

C W(Y)=65

c_{01}=0， c_{12}=0， c_{24}=0，
c_{30}=0， c_{34}=0， c_{42}=0，
c_{43}=0

68 G

72 E

d_{01}=3+2=5， d_{12}=8+0=8，
d_{24}=2+0=2， d_{30}=0+12=12，
d_{34}=0+0=0， d_{42}=0+0=0，
d_{43}=0+2=2

回路边表：

选取v_3v_0为Y结点

44

	0	1	2	3	4
0	∞	0	16	3	3
1	∞	∞	0	9	8
2	15	15	∞	2	0
3	0	45	19	∞	0
4	12	2	0	0	∞

C W(Y)=65

(3,0)

(3,0)

	1	2	3	4
0	0	16	∞	3
1	∞	0	9	8
2	15	∞	2	0
4	2	0	0	∞

H

	0	1	2	3	4
0	∞	0	16	3	3
1	∞	∞	0	9	8
2	15	15	∞	2	0
3	∞	45	19	∞	0
4	12	2	0	0	∞

I

68 G

72 E

回路边表： v_3v_0

45

	0	1	2	3	4
0	∞	0	16	3	3
1	∞	∞	0	9	8
2	15	15	∞	2	0
3	0	45	19	∞	0
4	12	2	0	0	∞

C W(Y)=65

(3,0)

(3,0)

	1	2	3	4
0	0	16	∞	3
1	∞	0	9	8
2	15	∞	2	0
4	2	0	0	∞

H W(Y)=65

	0	1	2	3	4
0	∞	0	16	3	3
1	∞	∞	0	9	8
2	3	15	∞	2	0
3	∞	45	19	∞	0
4	0	2	0	0	∞

lh_1=12

I W(Y)=77

68 G

72 E

归约E矩阵

回路边表： v_3v_0

46

	0	1	2	3	4
0	∞	0	16	3	3
1	∞	∞	0	9	8
2	15	15	∞	2	0
3	0	45	19	∞	0
4	12	2	0	0	∞

C W(Y)=65

(3,0)

(3,0)

	1	2	3	4
0	0	16	∞	3
1	∞	0	9	8
2	15	∞	2	0
4	2	0	0	∞

H W(Y)=65

	0	1	2	3	4
0	∞	0	16	3	3
1	∞	∞	0	9	8
2	3	15	∞	2	0
3	∞	45	19	∞	0
4	0	2	0	0	∞

I W(Y)=77

65 H

68 G

77 I

72 E

调整优先队列

47

	0	1	2	3	4
0	∞	0	16	3	3
1	∞	∞	0	9	8
2	15	15	∞	2	0
3	0	45	19	∞	0
4	12	2	0	0	∞

C W(Y)=65

(3,0)

(3,0)

	1	2	3	4
0	0	16	∞	3
1	∞	0	9	8
2	15	∞	2	0
4	2	0	0	∞

H W(Y)=65

	0	1	2	3	4
0	∞	0	16	3	3
1	∞	∞	0	9	8
2	3	15	∞	2	0
3	∞	45	19	∞	0
4	0	2	0	0	∞

I W(Y)=77

65 H

68 G

77 I

72 E

回路边表： v_3v_0

48

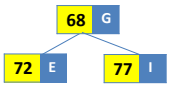
	1	2	3	4
0	0	16	∞	3
1	∞	0	9	8
2	15	∞	2	0
4	2	0	0	∞

H W(Y)=65

c_{01}=0， c_{12}=0， c_{24} =0，
c_{42} =0， c_{43} =0

d_{01}=3+2=5， d_{12}=8+0=8，
d_{24} =2+3=5， d_{42} =0+0=0，
d_{43} =0+2=2

选取v_1v_2为Y结点



回路边表： v_3v_0

49

	1	2	3	4
0	0	16	∞	3
1	∞	0	9	8
2	15	∞	2	0
4	2	0	0	∞

H W(Y)=65

(1,2)

(1,2)

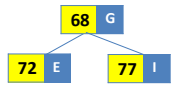
	1	3	4
0	0	∞	3
2	∞	2	0
4	2	0	∞

J W(Y)=65

	1	2	3	4
0	0	16	∞	3
1	∞	∞	9	8
2	15	∞	2	0
4	2	0	0	∞

K W(Y)=65

ch_1=8



回路边表： v_3v_0, v_1v_2

50

	1	2	3	4
0	0	16	∞	3
1	∞	0	9	8
2	15	∞	2	0
4	2	0	0	∞

H W(Y)=65

(1,2)

(1,2)

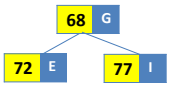
	1	3	4
0	0	∞	3
2	∞	2	0
4	2	0	∞

J W(Y)=65

	1	2	3	4
0	0	16	∞	3
1	∞	∞	1	0
2	15	∞	2	0
4	2	0	0	∞

K W(Y)=73

ch_1=8



回路边表： v_3v_0, v_1v_2

51

	1	2	3	4
0	0	16	∞	3
1	∞	0	9	8
2	15	∞	2	0
4	2	0	0	∞

H W(Y)=65

(1,2)

(1,2)

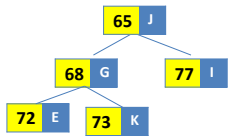
	1	3	4
0	0	∞	3
2	∞	2	0
4	2	0	∞

J W(Y)=65

	1	2	3	4
0	0	16	∞	3
1	∞	∞	1	0
2	15	∞	2	0
4	2	0	0	∞

K W(Y)=73

ch_1=8



回路边表： v_3v_0, v_1v_2

52

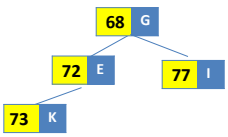
	1	3	4
0	0	∞	3
2	∞	2	0
4	2	0	∞

J W(Y)=65

c_{01}=0， c_{24} =0， c_{43} =0

d_{01}=3+2=5，
d_{24} =2+3=5，
d_{43} =2+2=4

选取v_0v_1为Y结点



回路边表： v_3v_0, v_1v_2

53

	1	3	4
0	0	∞	3
2	∞	2	0
4	2	0	∞

J W(Y)=65

(0,1)

(0,1)

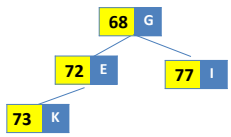
	3	4
2	2	0
4	0	∞

L W(Y)=65

	1	3	4
0	∞	∞	3
2	∞	2	0
4	2	0	∞

M W(Y)=70

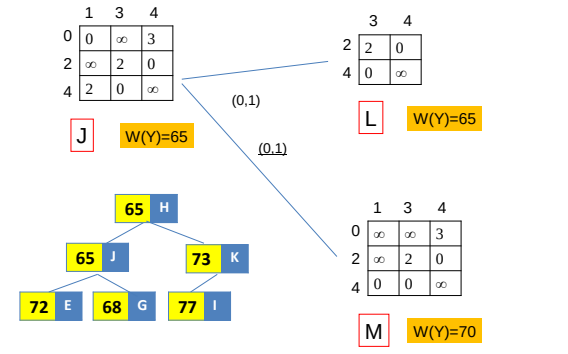
lh_1=2



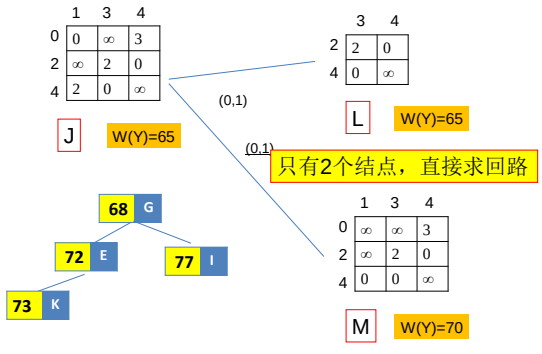
归约E矩阵

回路边表： v_3v_0, v_1v_2, v_0v_1

54



回路边表： $v_3v_0,\ v_1v_2,\ v_0v_1$



回路边表： $v_3v_0,\ v_1v_2,\ v_0v_1,\ v_2v_4,\ v_4v_3$ W(Y)=65

6.3 0/1背包问题

- n个物体重量为$w_0,w_1,\ldots,w_{n-1}$，价值分别为$p_0,p_1,\ldots,p_{n-1}$，背包载重量为M。按价值重量比进行递减排序。排好序的物体序号S={0,1，…,n-1}。将物体划分为3个集合：装入背包的物体集合S_1，不装入背包的物体集合S_2，以及尚待选择的物体集合S_3。初始时，有：

$$S_1(0)=\varphi, S_2(0)=\varphi, S_3(0)=\{0,1,\ldots,n-1\}$$

分支的确定：

- K为当前价值重量比最大的元素，将k装入背包的动作为：

$$\begin{aligned} S_1(k+1) &= S_1(k)\cup\{k\} \\ S_2(k+1) &= S_2(k) \\ S_3(k+1) &= S_3(k)-\{k\} \end{aligned}$$

- K为当前价值重量比最大的元素，不将k装入背包的动作为：

$$\begin{aligned} S_1(k+1) &= S_1(k) \\ S_2(k+1) &= S_2(k)\cup\{k\} \\ S_3(k+1) &= S_3(k)-\{k\} \end{aligned}$$

界限的确定：

- 假设$b(k)$为在选择K时，某个分支结点的背包中物体的价值上界。这时，S_3={k,k+1，…,n-1}。
- 计算两种分支结点的背包中物体价值的上界：

若：

$$M < \sum_{i\in S_1(k)} w_i \quad \text{则} \quad b(k)=0 \qquad (6.3.1)$$

- 若：

$$M=\sum_{i\in S_1(k)}w_i+\sum_{i=k+1}^{l-1}w_i+x\cdot w_l$$
$$0\le x<1,k<l,k\in S_1(k),l\in S_3(k)$$

则：

$$b(k)=\sum_{i\in S_1(k)}p_i+\sum_{i=k+1}^{l-1}p_i+x\cdot p_l \qquad (6.3.2)$$

求解过程：

1. 将物体按价值重量比递减排序；
2. 建立根结点X，X.b=0,X.k=0,X.S_1=Φ, X.S_2=Φ, X.S_2=S;
3. 若X.k=n，算法结束，X.S_1=即为装入背包的物体，X.b即为装入背包的最大值，否则转步骤4；
4. 建立Y结点，令Y.S_1=X.S_1U{X.k}，Y.S_2=X.S_2，Y.S_3=X.S_3-{X.k}，Y.k=X.k+1；按照式（6.3.1），（6.3.2）计算Y.b，将结点Y按Y.b插入堆中；
5. 建立Z结点，令Z.S_1=X.S_1，Z.S_2=X.S_2U{X.k},Z.S_3=X.S_3-{X.k}，Z.k=X.k+1；按照式（6.3.1），（6.3.2）计算Z.b，将结点Z按Z.b插入堆中；
6. 取得堆顶元素为X结点，转入步骤3.

- 例题：五个物体，重量为（8,16,21,17,12），价值为（8,14,16,11,7），背包载重量为37.
- 价值重量比排序：1,0.875,0.76,0.65,0.58

排序	1	0.875	0.76	0.65	0.58
序号	0	1	2	3	4

w	8	16	21	17	12
p	8	14	16	11	7

M=37

排序	1	0.875	0.76	0.65	0.58
序号	0	1	2	3	4

S_1=Φ
S_2=Φ
S_3={0,1,2,3,4}
b=0

A

K=0

S_1={0}
S_2=Φ
S_3={1,2,3,4}

B

$$37<\sum_{i\in S_1(0)}w_i=8$$

不成立

w	8	16	21	17	12
p	8	14	16	11	7

M=37

排序	1	0.875	0.76	0.65	0.58
序号	0	1	2	3	4

S_1=Φ
S_2=Φ
S_3={0,1,2,3,4}
b=0

A

K=0
装入

S_1={0}
S_2=Φ
S_3={1,2,3,4}
b=31.9

B

$$37=\sum_{i\in S_1(0)}w_i+\sum_{i=1}^{l-1}w_i+x\cdot w_l=8+(16)+x\cdot 21$$
$$l=2,x=0.619$$

$$b(0)=\sum_{i\in S_1(0)}p_i+\sum_{i=1}^{2-1}p_i+0.619\cdot p_2$$
$$=8+(14)+0.6198*16=31.9$$

w	8	16	21	17	12
p	8	14	16	11	7

M=37

排序	1	0.875	0.76	0.65	0.58
序号	0	1	2	3	4

S_1=Φ
S_2=Φ
S_3={0,1,2,3,4}
b=0

A

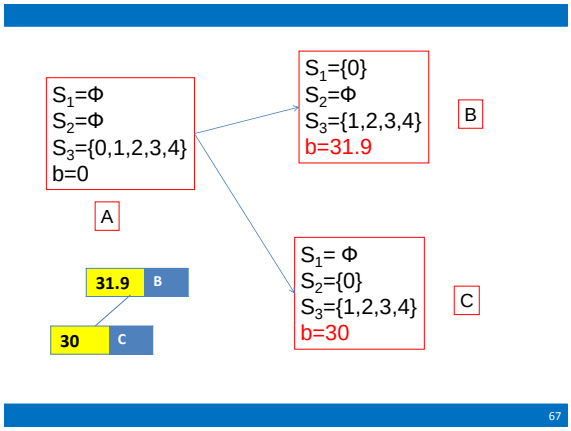
K=0
不装入

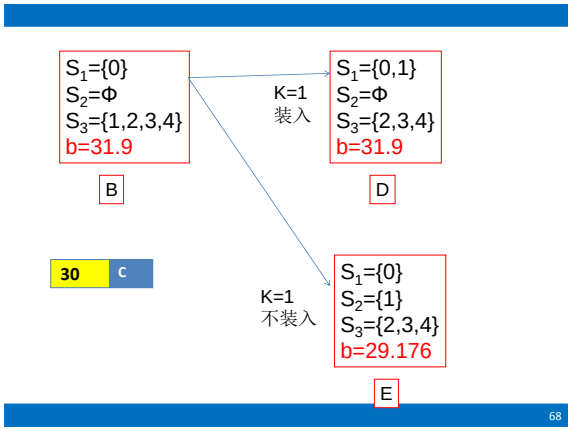
S_1= Φ
S_2={0}
S_3={1,2,3,4}
b=30

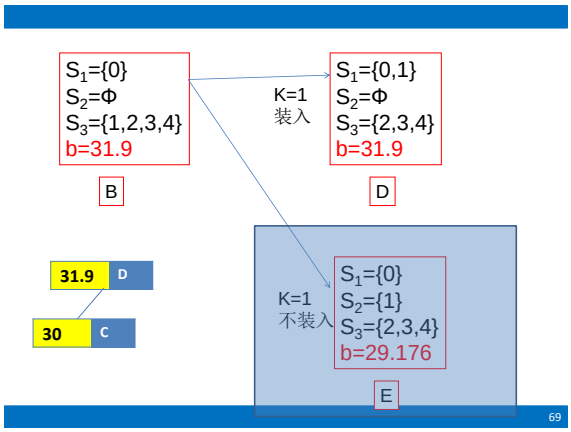
C

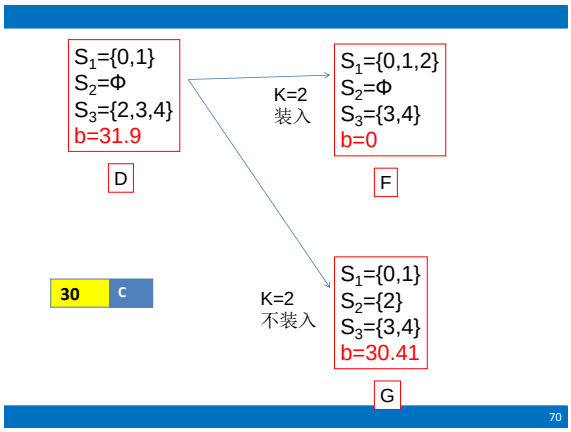
$$37=\sum_{i\in S_1(0)}w_i+\sum_{i=1}^{l-1}w_i+x\cdot w_l=0+(16+21)+0\cdot 17$$
$$l=3,x=0$$

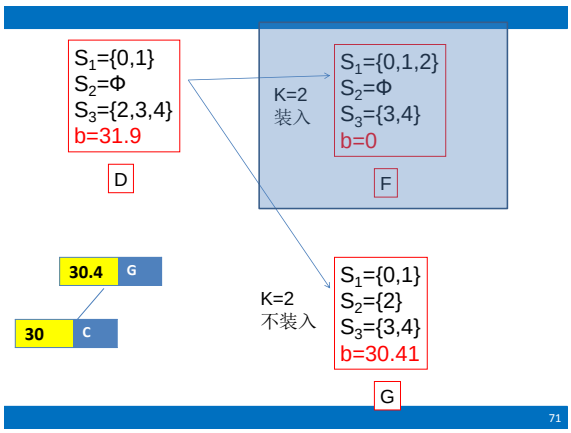
$$b(0)=\sum_{i\in S_1(0)}p_i+\sum_{i=1}^{3-1}p_i+0\cdot p_3$$
$$=0+(14+16)+0*11=30$$

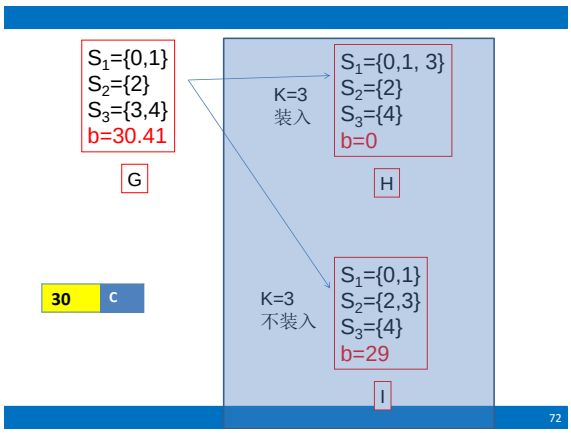

S1=Φ
S2=Φ
S3={0,1,2,3,4}
b=0
A
S1={0}
S2=Φ
S3={1,2,3,4}
b=31.9
B
S1= Φ
S2={0}
S3={1,2,3,4}
b=30
C
31.9 B
30 C
67

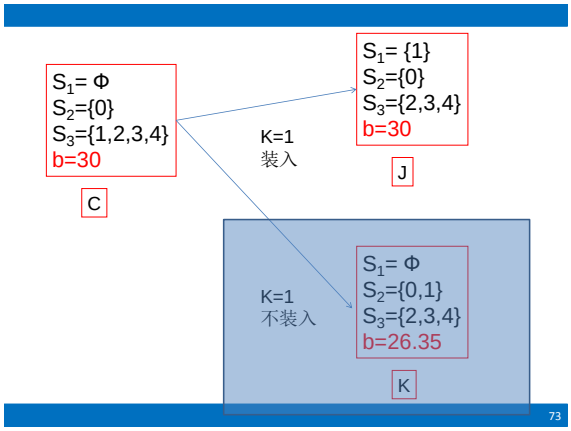

S1={0}
S2=Φ
S3={1,2,3,4}
b=31.9
B
K=1
装入
S1={0,1}
S2=Φ
S3={2,3,4}
b=31.9
D
30 C
K=1
不装入
S1={0}
S2={1}
S3={2,3,4}
b=29.176
E
68

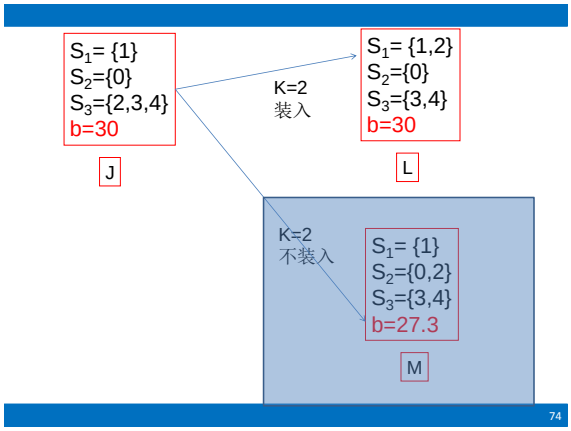

S1={0}
S2=Φ
S3={1,2,3,4}
b=31.9
B
K=1
装入
S1={0,1}
S2=Φ
S3={2,3,4}
b=31.9
D
31.9 D
30 C
K=1
不装入
S1={0}
S2={1}
S3={2,3,4}
b=29.176
E
69

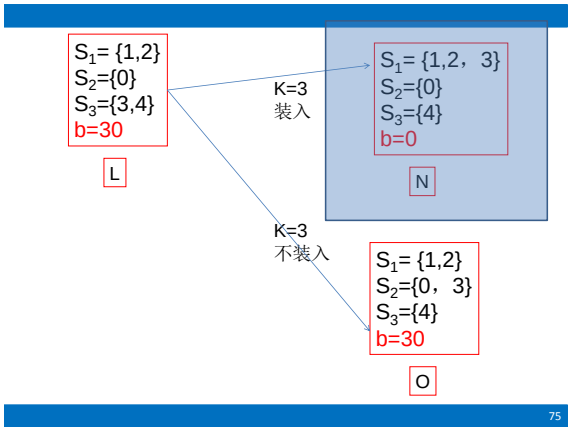

S1={0,1}
S2=Φ
S3={2,3,4}
b=31.9
D
K=2
装入
S1={0,1,2}
S2=Φ
S3={3,4}
b=0
F
30 C
K=2
不装入
S1={0,1}
S2={2}
S3={3,4}
b=30.41
G
70

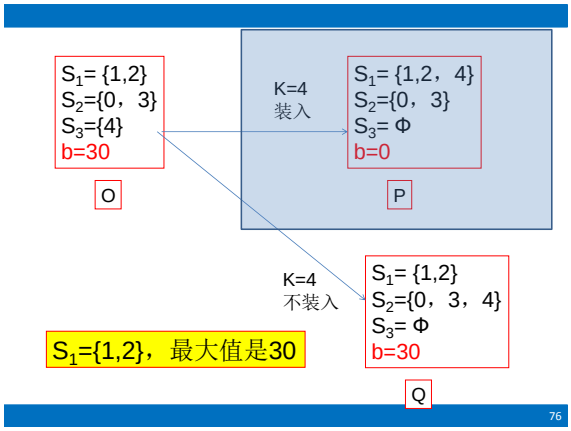

S1={0,1}
S2=Φ
S3={2,3,4}
b=31.9
D
K=2
装入
S1={0,1,2}
S2=Φ
S3={3,4}
b=0
F
30.4 G
30 C
K=2
不装入
S1={0,1}
S2={2}
S3={3,4}
b=30.41
G
71


S1={0,1}
S2={2}
S3={3,4}
b=30.41
G
K=3
装入
S1={0,1, 3}
S2={2}
S3={4}
b=0
H
30 C
K=3
不装入
S1={0,1}
S2={2,3}
S3={4}
b=29
I
72









6.4 作业分配问题

*n*个操作员以*n*种不同时间完成*n*种不同作业，要求分配每一位操作员完成一项工作，使得完成*n*项工作的总时间最少。

77

- 将*n*个操作员编号为0,1,..., *n* -1， *n*个作业编号为0,1,..., *n* -1 。用矩阵*c*来描述每位操作员完成每个作业的时间。用向量x_i描述分配给操作员的作业编号，如表示分配给第*i*为操作员的作业编号。

78

- 按第一种方法，从根结点开始，在整个搜索过程中，每遇到一个e结点，就对它的所有孩子结点计算它们的下界，并把它们登记在结点表中。再从结点表中选取下界最小的结点，重复上述过程。当搜索到一个叶子结点时，如果该结点的下界是结点表中最小的，则该结点就是最优解。否则对下界最小的结点继续进行扩展。

下界的计算

- k表示搜索深度，当k=0时，从结点开始向下搜索。这时它有n个孩子结点对应n个操作员。如果把作业0(k=0)分配给第i位操作员，其余作业分配给其他操作员，则需要的时间至少为：

$$t = c_{i0} + \sum_{j=1}^{n-1} (\min_{k \neq i} c_{kj})$$

操作员 \ 作业	0	1	2	3
0	3	8	4	12
1	9	12	13	5
2	8	7	9	3
3	12	7	6	8

把第0号作业分配给第0位操作员时，c_{00}=3，第1号作业分别由其他3位操作员完成时，最短时间为7，第2号作业的最短时间为6，第三号作业的最短时间为3，因此，当把第0号作业分配给第0位操作员时，所需要的时间不会小于3+7+6+3=19，可以把它看成是根结点下第0个结点的下界。同样，如果把第0号作业分配给第1位操作员时，所需要的最少时间为23，可以把它看成是根结点下第1个结点的下界。

- 当搜索深度为k时，前面0,1,…,k-1号作业已经分配给各个操作员。令S={0,1,…,n-1}表示操作员编号，m_{k-1}={i_0, i_1,…, i_{k-1}}表示作业已分配的操作员编号的集合。当把第k个作业分配给编号为i_k的操作员时， $i_k \in S-m_{k-1}$,所需要至少的时间为：

$$t = \sum_{l=0}^{k} c_{i_l l} + \sum_{l=k+1}^{n-1} (\min_{i \in S-m_k} c_{il}) \qquad (1)$$

作业分配算法

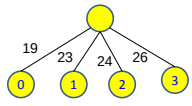
1. 建立根结点X，令根结点$X.k$=0，$X.S$={0,1,…,n-1},$X.m$=ϕ
2. 对所有编号为i的操作员，$i \in X.S$，建立孩子结点Y_i，把结点X的数据复制到结点Y_i
3. 令$Y_i.m=Y_i.m \cup \{i\}$, $Y_i.S=Y_i.S-\{i\}$, $Y_i.x_i=Y_i.k$, $Y_i.k=Y_i.k+1$, 按照公式（1）计算$Y_i.t$
4. 把根结点Y_i插入到最小堆
5. 取堆顶元素作为子树的根X，若$X.k=n$，则算法结束，否则转步骤2.

操作员 \ 作业	0	1	2	3
0	3	8	4	12
1	9	12	13	5
2	8	7	9	3
3	12	7	6	8

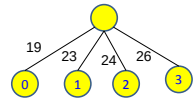
当k=0时，有：

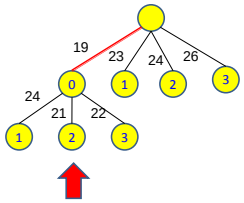
t_{00}=3+7+6+3=19
t_{10}=9+7+4+3=23
t_{20}=8+7+4+5=24
t_{30}=12+7+4+3=26

于是，在根结点下建立4个孩子结点，把第0号作业分配给第0,1,2,3号操作员，其下界分别是19,23,24,26。



操作员 \ 作业	0	1	2	3
0	3	8	4	12
1	9	12	13	5
2	8	7	9	3
3	12	7	6	8

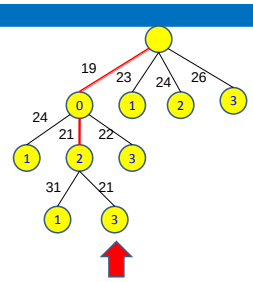




将这些结点插入到优先队列中，这时，结点0的下界最小，可以从该结点开始进行搜索。又生成3个孩子结点。

t_{11}=3+12+6+3=24
t_{21}=3+7+6+5=21
t_{31}=3+7+9+3=22

操作员 \ 作业	0	1	2	3
0	3	8	4	12
1	9	12	13	5
2	8	7	9	3
3	12	7	6	8

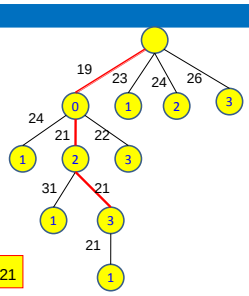


将这些结点插入到优先队列中，这时，结点0的下界最小，可以从该结点开始进行搜索。又生成3个孩子结点。

t_{11}=3+12+6+3=24
t_{21}=3+7+6+5=21
t_{31}=3+7+9+3=22

t_{12}=3+7+13+8=31
t_{32}=3+7+5+6=21

操作员 \ 作业	0	1	2	3
0	3	8	4	12
1	9	12	13	5
2	8	7	9	3
3	12	7	6	8



t_{13}=3+7+5+6=21

0号操作员分配0号作业，2号操作员分配1号作业，
3号操作员分配2号作业，1号操作员分配3号作业。

6.5 同题异策—“算法”与“策略”

- 没有刻意区分
- 策略
  - 一种思想，提出解决问题的思路
  - 从战略高度上面向问题
- 算法
  - 对一个/类具体问题的解决方法
  - 面向具体实现，可以用任何方式描述
  - 策略对算法有指导性
  - 在一种策略的指导下，可以有多种不同的算法实现
    - 如排序问题，使用贪婪策略，列出3种有效算法

同题异策—“算法”与“策略”

- 迭代(递推)
- 枚举
- 贪婪
- 动态规划
- 分治

同题异策—“算法”与“策略”

- 迭代(递推)
  - 中心思想：重复使用迭代(递推)公式，根据变量的旧值推出新值；
  - 适用问题：具有明确迭代公式的问题，主要是数值计算等。

同题异策－“算法”与“策略”

- 贪婪
  - ✓中心思想：通过一系列的局部选择来得到一个问题的解，所作的每一步选择：“只顾眼前最优，不管将来好坏”；
  - ✓适用问题：
    1)贪婪选择性质；
    2)最优子结构性质。

这项要求很高

- 动态规划
  - ✓ 中心思想：把求解的问题分成许多阶段/子问题，然后按顺序求解各阶段/子问题；记录每个阶段决策得到的结果序列。最后阶段的解就是初始问题的解。
  - ✓ 适用问题：
    1)最优子结构； (必须有)
    2)无后向性；
    3)子问题重叠性质。 (体现优势)

同题异策－“算法”与“策略”

- 分治
  - 中心思想：将整个问题分解成若干个小问题后分而治之；分解，解决，合并。
  - 适用问题：
    1)问题的规模缩小到一定程度就可容易解决；
    2)问题可以分解为若干个规模较小的相似问题；
    3)子问题的解可以合并为原问题的解；
    4) 子问题相互独立。

- 枚举
  - 中心思想：枚举出问题的所有可行解，找出其中的最优解；
  - 适用问题：其它策略难以奏效。

6.6 同题异策－算法策略的总结

- 对问题进行分解的策略：“分治法”与“动态规划法”

- “分治法”与“动态规划法”都是递归思想的应用之一，是找出大问题与小的子问题之间的关系,直到小的子问题很容易解决，再由小的子问题的解导出大问题的解。

- 动态规划的实质：
  分治算法思想+解决子问题冗余情况

同题异策－算法策略的总结

- 多阶段逐步解决问题的策略：“贪婪算法”、“递推法”、“递归法”和“动态规划法”

- 多阶段过程就是按一定顺序(从前向后或从后向前等)一定的策略,逐步解决问题的方法。
  - “贪婪算法”每一步根据策略得到一个结果传递到下一步，自顶向下，一步一步地作出贪心选择。
  - “动态规划法”则根据一定的决策,每一步决策出的不是一个结果，而只是使问题的规模不断的缩小，如果决策比较简单,是一般的算法运算,则可找到不同规模问题间的关系，使算法演变成“递推法”、“递归法”算法。
  - “递推法”、“递归法”更注重每一步之间的关系,决策的因素较少。

同题异策－算法策略的总结

- 全面逐一尝试：比较“枚举法”、“递归回溯法”

- 有这样一类问题，问题中不易找到信息间的相互关系，也不能分解为独立的子问题,似乎只有把各种可能情况都考虑到,并把全部解都列出来之后,才能判定和得到最优解。
- 对于规模不大的问题，这些策略简单方便;而当问题的计算复杂度高且计算量很大时,还是考虑“动态规划法”这个更有效的算法策略。

- 实现
  - 循环次数固定的问题：通过循环嵌套枚举问题中各种可能的情况,如八皇后问题能用八重循环嵌套枚举。
  - 不固定的问题：靠递归回朔法来“枚举”或“遍历”各种可能情况。比如n皇后问题只能用“递归回朔法”通过递归实现（当然可以通过栈,而不用递归）。

求解组合优化问题的算法设计技术比较

技术	动态规划	分支限界	贪心法
使用条件	优化原则 多步判断	多步判断	贪心选择+优化原则 多步判断
选择依据	子问题结果	约束条件和界	局部最优性质
计算过程	看子问题结果选择 自底向上	选择后生成子问题 自顶向下	选择后生成子问题 自顶向下
数据结构	二维表	树、队列	线性表
解	一个最优解	一个和多个最优解	一个最优或近似解
关键问题	递推方程 空间复杂性高	设定代价函数 时间复杂性高	贪心选择性质证明 近似解的误差估计