国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第19卷第1期 1997年2月 Vol. 19 No. 1

瓶颈指派问题的一种多项式时间算法*

晓 斌 张千宗

(国防科技大学系统工程与数学系 长沙 410073)

摘 要 本文对瓶颈指派问题给出了一种新的算法, 该算法不需要利用最大流算法, 而类似于解经典指派问题的匈牙利算法。该算法是一个多项式时间算法, 其复杂性为 $O(n^3)$.

关键词 瓶颈指派问题, 多项式时间算法, 阀门算法

分类号 O224

A Polynomial- time Algorithm for the Bottleneck Assignment Problem

Xiao Bin Zhang Ganzong

(Department of Systems Engineering and Mathematics, NUDT, Changsha, 410073)

Abstract In this paper, we give a new algorithm for the bottleneck assignment problem on the basis of Konig´s theorem, and show that the time complexity of the algorithm is $O(n^3)$.

Key words bottleneck assignment problem; polynomial- time algorithm; threshold algorithm.

1 瓶颈指派问题的数学模型

瓶颈指派问题的提法是: 一项工程有 n 件工作分派给 n 个人, 每个人只能分给一项工作, 每项工作只能一人完成。所有工作均同时开始, 应如何分派这 n 项任务, 使整个工程尽早完工? 显然, 工程完工的时间等于最后完工的那个人所花的时间, 问题的优化目标是使这个最长时间最短。

瓶颈指派问题首先由 O. Gross 提出, 给出了一种 "改进圈" 算法[4], 后来 R. S. Garfinkel 利用 Ford- Fulkerson 最大流算法给出了一种 "阀门" 算法[4]. Gross 的算法具有盲目的试探性, 且很难检验最优解已找到。Garfinkel 的算法虽作了改进, 但由于使

* 1995年11月14日收稿。

用最大流算法而不免影响解算效率。

本文利用匈牙利数学家 D. Konig 的一个著名定理对“阀门”算法作了改进，得到的新算法无需引进最大流算法。新算法类似于解经典指派问题的匈牙利算法，并证明了它的复杂性为 $O(n^3)$.

用 c_{ij} 表示第 i 个人完成第 j 项工作所花的时间，矩阵 $C=(c_{ij})_{n\times n}$ 称为指派矩阵或系数矩阵。瓶颈指派问题可表为如下的整数规划模型，记为(BAP)：

$$\min Z=\max_{1\leq i,j\leq n} c_{ij}x_{ij}$$

$$\text{s.t.}\quad \sum_{j=1}^{n} x_{ij}=1,\ (i=1,\cdots,n);\quad \sum_{i=1}^{n} x_{ij}=1,\ (j=1,\cdots,n)$$

$$x_{ij}=0,1\qquad (i,j=1,\cdots,n)$$

这里，$x_{ij}=1$，如果第 j 项工作分派给第 i 个人；否则 $x_{ij}=0$

与经典指派问题相比较，(BAP)的优化目标已发生变化。因此，著名的匈牙利法不能直接求解这类问题，经典指派问题的最优解定理，即每行或每列减去同一个数后最优解不变，也不再成立。但为减少计算机内存，可将系数矩阵的各元素都减去同一个数，(BAP)最优解不变。

2 化为经典指派问题

对系数矩阵元素 c_{ij} 的不同的值按从小到大的顺序排序，$c(k)$ 为第 k 个最小值。用 K' 表示不同 $c(k)$ 值的个数，这里 $1\leq K'\leq n^2$，定义数列 $d(k)$ 如下：

$$d(1)=1$$

$$d(t)=nd(t-1)+1\qquad 2\leq t\leq K' \tag{1}$$

对每一 $i,j(1\leq i\leq n,\ 1\leq j\leq n)$，令 $d_{ij}=d(k)$，这里 k 是满足 $c_{ij}=c(k)$ 的唯一整数，把 $D=(d_{ij})_{n\times n}$ 作为系数短阵，建立如下的经典指派问题(API)：

$$\min \sum_{j=1}^{n}\sum_{i=1}^{n} d_{ij}x_{ij}$$

$$\text{s.t.}\quad \sum_{j=1}^{n} x_{ij}=1,\ (i=1,\cdots,n);\quad \sum_{i=1}^{n} x_{ij}=1,\ (j=1,\cdots,n)$$

$$x_{ij}=0,1\qquad (i,j=1,\cdots,n)$$

定理1 经典指派问题(API)的最优解也是瓶颈指派问题(BAP)的最优解。

证明 设(API)的一个最优解为

$$x_{1t_1}=x_{2t_2}=\cdots=x_{nt_n}=1,\ \text{其余}\ x_{ij}=0 \tag{2}$$

若它不是(BAP)的最优解，则存在 $\{1,2,\cdots,n\}$ 的一个排列 $j_1,j_2,\cdots,j_n$ 满足

$$c_{lj_l}=\max_{1\leq i\leq n}\{c_{ij_i}\}<\max_{1\leq i\leq n}\{c_{ij_i}\}=c_{mt_m}$$

由(1)知，$nd_{lj_l}+1\leq d_{mt_m}$。于是，$\sum_{i=1}^{n} d_{ij_i}\leq nd_{lj_l}\leq d_{mt_m}-1<\sum_{i=1}^{n} d_{it_i}$。

与(2)是(API)的最优解相矛盾。

定理1及变换式(1)提供了求解(BAP)可转化为求解(API)的理论与方法。

例 设瓶颈指派问题的系数矩阵为

$$\begin{bmatrix} 4 & 7 & 8 & 10 & 9 \\ 7 & 8 & 4 & 3 & 3 \\ 8 & 4 & 11 & 15 & 12 \\ 10 & 3 & 6 & 7 & 7 \\ 7 & 4 & 9 & 8 & 10 \end{bmatrix}$$

这里 $n=5$，不同的 c_{ij} 共有11个。由(1)得到矩阵 $D=(d_{ij})$ 如下:

$$\begin{bmatrix} 6 & 781 & 3906 & 97656 & 19530 \\ 781 & 3906 & 6 & 1 & 1 \\ 3906 & 6 & 488281 & 12207031 & 2441406 \\ 97656 & 1 & 156 & 781 & 781 \\ 781 & 6 & 19530 & 3906 & 97656 \end{bmatrix}$$

利用匈牙利法求解，得到最优解为 $x_{11}=x_{25}=x_{32}=x_{43}=x_{54}=1$，其余 $x_{ij}=0$，最优值等于8。

化为(API)问题求解的方法，在不同的 c_{ij} 值个数较少，即 K 较小时，算法是有效的。而当 K 较大，例如 $K\geqslant 20$ 时，$d(K)=n^{19}+n^{18}+\cdots+n+1=\dfrac{n^{20}-1}{n-1}$，即使 n 等于5，$d(K)$ 的值也很可观，这将给计算机存储和运算带来很大困难。因此该方法只适合求解 K 较小时的(BAP)问题。

3 改进的阀门算法

定理2 问题(BAP)的目标函数值的下界为 $V_0=\max\left\{\max\limits_{1\leqslant j\leqslant n}\min\limits_{1\leqslant i\leqslant n}c_{ij},\ \max\limits_{1\leqslant i\leqslant n}\min\limits_{1\leqslant j\leqslant n}c_{ij}\right\}$

证明 因目标函数为 $z=\max\limits_{1\leqslant i,j\leqslant n}c_{ij}x_{ij}$，因此

$$z\geqslant\min_{1\leqslant i\leqslant n}c_{ij}\qquad(\forall\, j\in\{1,\cdots,n\})$$

从而

$$z\geqslant\max_{1\leqslant j\leqslant n}\min_{1\leqslant i\leqslant n}c_{ij}$$

同理可知

$$z\geqslant\max_{1\leqslant i\leqslant n}\min_{1\leqslant j\leqslant n}c_{ij}$$

于是

$$z\geqslant\max\left\{\max_{1\leqslant j\leqslant n}\min_{1\leqslant i\leqslant n}c_{ij},\ \max_{1\leqslant i\leqslant n}\min_{1\leqslant j\leqslant n}c_{ij}\right\}$$

由定理2可知，如果问题(BAP)的一个可行解满足 $\max\{c_{ij}\mid x_{ij}=1\}=V_0$，则此可行解即为(BAP)的最优解，$V_0$ 即为目标函数最优值。

定理3 问题(BAP)的系数矩阵 C 中不大于 V(V 为给定的一个常数)的元素的独立个数(指不同行不同列的元素个数)等于能覆盖所有不大于 V 的元素的最少直线数。

此定理为匈牙利数学家 Kōnig 的著名定理——“系数矩阵中独立零元素的个数等于能覆盖所有零元素的最少直线数”的改写。事实上，只需将这些不大于 V 的元素形式上改写为零元素，定理2就是 Kōnig 定理本身。

基于定理2和定理3，我们给出一个关于(BAP)的算法。算法的基本思想是：首先选取 $V=V_0$，在系数矩阵 C 中寻找尽量多的、独立的不大于 V 的元素。若不存在 n 个独立的这样的元素，则寻找新的 V 值；若找到 n 个独立的不大于 V 的元素，则此时的 V 值即为(BAP)的最优值 z^*.

算法步骤如下:

Step1　置$I:=0$, 矩阵$A:=C$.

Step2　找出矩阵A每一行(列)最小的元素, 将这些元素按从小到大的顺序排列, 得到一数列, 设其第$n-I$项为$V_1(V_2)$, 置$V:=\max\{V_1,V_2\}$。

Step3　将矩阵C中所有小于或等于V的元素定为候选元素, 寻找最大数目的独立候选元。具体做法是: 从含候选元素最少的行开始, 圈出此行的一个候选元, 同时划去与该元素同行同列的其它候选元素, 然后对余下一部分重复这一做法(已划去的候选元素不再考虑), 直到查完各行, 所有带圈候选元素便是最大数目的独立候选元。若圈元的数目等于n, 则以这些圈元对应的解矩阵$(x_{ij})_{n\times n}$中的元素为1, 其余为0, 就得到最优解; 若圈元的数目小于n, 转Step4.

Step4　作最少的直线覆盖所有的候选元素, 具体做法是:

(1) 对没有圈元的行打"√"号;

(2) 对已打"√"的行中所有候选元素的列打"√"号;

(3) 再对打有"√"号的列中含圈元的行打"√"号;

(4) 重复(2)(3)直到得不出新的打"√"号的行列为止;

(5) 对没有打"√"号的行画一横线, 对打"√"号的列画一纵线, 这就得到覆盖所有候选元的最少直线数。

Step5　置A为C中未被上述直线覆盖的元素构成的矩阵, I为覆盖所有候选元的最少直线数, 转Step2。

算法的正确性　证明如下: 若所作的覆盖所有候选元素的最少直线数$I<n$, 由定理2知不存在n个独立的候选元素, 因此需增加候选元的数目。矩阵C被直线覆盖的部分中最多只能找出I个独立的元素, 因此C的任意一组n个独立的元素中至少有$n-I$个在矩阵A中。我们选取$V=\max\{V_1,V_2\}$作为新的候选元素的上界, 是因为C中所有比$\max\{V_1,V_2\}$小的元素能被数目小于n的直线所覆盖。事实上, 若$I=n-1$, 显然; 若$I<n-1$, 不妨设$\max\{V_1,V_2\}=V_1$, V取为数列中比V_1小的任意一项, 设A中不大于V的行最小元素所在的行分别为第$k_1,k_2,\cdots,k_m$行, 则$m<n-I$。将这m行的每一行划一直线, 得到m条覆盖所有A中不大于V的直线, 因此矩阵C中所有不大于V的元素能被$I+m$($<n$)条直线覆盖, 由定理2知得不到n独立的候选元素。按算法一步步地进行下去, 一旦得到n个独立的候选元素时, 候选元的上界V即为最优值。

例　(承上例)

Step1与Step2: $V=4$.

Step3: 可得以下有圈元的矩阵:

$$\begin{bmatrix} ④ & 7 & 8 & 10 & 9 \\ 7 & 8 & ④ & 3 & 3 \\ 8 & ④ & 11 & 15 & 12 \\ 10 & 3 & 6 & 7 & 7 \\ 7 & 4 & 9 & 8 & 10 \end{bmatrix}$$

Step4: 作最少直线覆盖所有候选元:

$$\begin{bmatrix} ④ & 7 & 8 & 10 & 9 & \\ 7 & 8 & ④ & 3 & 3 & \\ 8 & ④ & 11 & 15 & 12 & \surd \\ 10 & 3 & 6 & 7 & 7 & \surd \\ 7 & 4 & 9 & 8 & 10 & \surd \end{bmatrix}$$

Step5: $I = 3$, 转入第二次迭代。

Step2: $V = 7$

Step3 与 Step4

$$\begin{bmatrix} 4 & 7 & 8 & 10 & 9 & \surd \\ 7 & 8 & ④ & 3 & 3 & \\ 8 & ④ & 11 & 15 & 12 & \surd \\ 10 & 3 & 6 & ⑦ & 7 & \\ ⑦ & 4 & 9 & 8 & 10 & \surd \end{bmatrix}$$

$\surd \quad \surd$

Step5: $I = 4$, 转入第三次迭代。

Step2: $V = 8$

Step3:

$$\begin{bmatrix} 4 & 7 & ⑧ & 10 & 9 \\ 7 & 8 & 4 & ③ & 3 \\ ⑧ & 4 & 11 & 15 & 12 \\ 10 & 3 & 6 & 7 & ⑦ \\ 7 & ④ & 9 & 8 & 10 \end{bmatrix}$$

得出最优值等于8. 最优解为 $x_{13} = x_{24} = x_{31} = x_{45} = x_{52} = 1$, 其余 $x_{ij} = 0$。

对算法复杂性分析如下: 因阀门算法与匈牙利算法非常相似, 我们将经典的指派问题匈牙利算法迭代过程与阀门算法进行比较。将阀门算法得到的候选元改写为零, 这样, 每迭代一次阀门算法得到的零元不会少于匈牙利算法得到的零元, 在极端的情形下二者相等。故阀门算法与匈牙利算法具有相同的复杂性, 为 $O(n^3)$。

参 考 文 献

1 Mazzola J B. Neebe A W. Bottleneck generalized assignment problem. Costs Prod Econ 14, 1988, 61～65

2 Mazzola J B. Neebe, A W. An algorithm for the bottleneck generalized assignment problem, Computers Ops Res 20, 1993, 355～362

3 Bazaraa, M S. Jarvis, J J. Linear programming and network flows, New York: John wiley & Sons, inc, 1977, 383～391

4 Garfinkel, R S. An improved algorithm for the bottleneck assignment problem. Ops Res. 19k1747–1751, 1971

5 Garfinkel, R S. Rao, M R. The bottleneck transportation problem. Nav, Res. Logist. Q. 18, 1971: 465～472

(责任编辑 潘 生)