用“匈牙利算法”求解一类最优化问题

常庭懋, 韩中庚

(信息工程大学 信息工程学院, 河南 郑州 450002)

摘要: 文章给出了改进的“匈牙利算法”的一般步骤和 Matlab 实现的通用程序, 可用此来求解著名的指派问题、婚配问题、锁具装箱问题, 以及任何完全或非完全的赋权二分图的最优(大)匹配问题。

关键词: 匈牙利算法; 指派问题; 二分图; 最优匹配

中图分类号: O157.6 文献标识码: A 文章编号: 1671－0673(2004)01－0060－03

Solution to a Class Optimization Problem By Utilizing the “Hungary Calculate Way”

CHANG Ting-mao, HAN Zhong-geng

(Institute of Information Engineering, Information Engineering University, Zhengzhou 450002, China)

Abstract: This artical gives the general step of the improved “Hungary Calculate Way” and the general program realized in Matlab. Then the authors use it to solve the famous appointment problem, marriage problem, the problem of packing locks and the problem of the best (greatest) match in each completely or not complately weighted bivariate graph.

Key words: Hungary Calcuate Way; appointment problem; bivariate graph; problem of the best match

1 问题的提出

“匈牙利算法”最早是由匈牙利数学家 D. Konig 用来求矩阵中 0 元素的个数的一种方法, 由此他证明了“矩阵中独立 0 元素的最多个数等于能覆盖所有 0 元素的最少直线数”。1955 年由 W.W. Kuhn 在求解著名的指派问题时引用了这一结论, 并对具体算法做了改进, 仍然称为“匈牙利算法”。

所谓的指派问题: 共有 n 项任务, 恰好有 n 个人都能去完成其中的每一项任务, 各人完成每项任务所需时间(或效率、成本)不同, 于是产生应该怎样指派这 n 个人, 使完成这 n 项任务所需总时间最少(或效率最高、成本最低)。在实际中, 这是一类广泛的问题, 建立数学模型如下:

设用 $c_{ij}>0(i, j=1, 2, \cdots, n)$ 表示指派第 i 人去完成第 j 项任务时所用的时间, 定义决策变量

$$x_{ij}=\begin{cases}1 & \text{当指派第 } i \text{ 人去完成第 } j \text{ 任务时}\\ 0 & \text{不指派第 } i \text{ 人去完成第 } j \text{ 任务时}\end{cases}$$

则问题可转化为 0－1 线性规划问题

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

$$s.t.\begin{cases}\sum_{i=1}^{n}x_{ij}=1, & j=1, 2, \cdots, n\\ \sum_{j=1}^{n}x_{ij}=1, & j=1, 2, \cdots, n\\ X_{ij}=1 \text{ 或 } 0, & i, j=1, 2, \cdots, n\end{cases} \quad (1)$$

“匈牙利算法”可以用于求解多种形式的指派问题, 其基本思想是寻找独立 1 元素组, 而独立 1 元素组与图论中对集是一个等价概念, 所以与图论中求解赋权二分图最优对集、最大对集的思想是一脉相承的。因此, 实际中凡是能够转化为求解赋权二分图最优对集、最大对集的问题, 均可用“匈牙利算法”来解决。

2 “匈牙利算法”的步骤与 Matlab 的实现

根据指派问题的最优解的性质: “若从系数矩

收稿日期: 2003－06－04

作者简介: 常庭懋(1983－), 男, 重庆人, 信息工程大学本科生, 应用数学专业。

阵 $\boldsymbol{A}=(c_{ij})$ 的一行(列)各元素分别减去该行(列)的最小元素, 得到新矩阵 $\boldsymbol{D}=(c'_{ij})$, 那么以 $\boldsymbol{D}=(c'_{ij})$ 为系数矩阵所对应问题的最优解与原问题的最优解相同”。此时求最优解的问题可转化为寻找系数矩阵的最大对集, 所以可用“匈牙利算法”的思想来求解。

下面首先给出修改的“匈牙利算法”的步骤:

第 1 步　对系数矩阵进行变换, 使每行每列都出现 0 元素。

①从系数矩阵 $\boldsymbol{C}$ 中每行减去该行最小元素。

②再在所得矩阵每列减去每列最小元素。

第 2 步　令 $\boldsymbol{D}$ 为上一步所得的矩阵, 矩阵 $\boldsymbol{E}$ 是将矩阵 $\boldsymbol{D}$ 中 0 元素变为 1 元素, 非零元素变为 0 元素。

第 3 步

① 在 E 中选择含有 1 元素的各行中 1 元素最少的行, 比较该行中各 1 元素所在的列中 1 元素的个数, 选择 1 元素最少的那一列的那个 1 元素。

② 将所选的 1 元素所在的行列清“0”。

③ 重复步骤 2 和 3, 直到没有 1 元素为止, 得到一个独立 1 元素组。

第 4 步　如果所得独立 1 元素组还不是原矩阵的最大独立 1 元素组, 那么再用寻找可扩路的方法对其进行扩张, 即可得到原矩阵最大独立 1 元素组。若 1 元素的个数等于矩阵的阶数, 则已得到最优解, 停止计算, 否则进入下一步。

第 5 步　对 $\boldsymbol{D}$ 增加更多的 0 元素。

① 做最少的直线覆盖矩阵 $\boldsymbol{D}$ 的所有 0 元素。

② 在没有被直线覆盖的部分找出最小元素, 在被直线覆盖的各行减去此最小元素, 在被直线覆盖的各列加上此最小元素, 得到新矩阵, 返回第 2 步。

上面的算法是按最小化形式给出的, 如果问题是最大化形式, 即模型(1)中的目标函数换为 $\max z=\sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}x_{ij}$。此令 $M=\max_{i,j}(c_{ij})$ 和 $b_{ij}=M-c_{ij}\geqslant 0$, 则效率矩阵变为 $\boldsymbol{B}=(b_{ij})$。于是考虑目标函数为 $\min z=\sum_{i=1}^{n}\sum_{j=1}^{n}b_{ij}x_{ij}$ 的问题, 仍用上面的方法步骤求解, 所得最小解也就是对应原问题的最大解。

对这类问题的求解, 我们给出了用 Matlab 实现的通用程序, 其中包括 4 个子程序, 子程序“e=*ff*01(*b*, *flag*)”是主子程序, 它需调用其余的 3 个子程序完成计算任务。主子程序中的输入参数 *flag* 取 1 或 0, 分别表示最小化和最大化问题, 输出参数 *e* 表示问题的最优解。子程序“[*jie*, *oue*] = *fc*01(*d*) 和[*e*, *total*] = *fc*02(*xe*, *ye*, *w*)”分别实现计算的第 3 步和第 4 步, 等价于求解二分图的最大对集, *fc*02 中的输出参数 *e* 表示最大对集, *total* 是最大对集包含的边数。子程序“*b*=*ff*02(*b*, *e*)”实现计算的第 5 步。

3　算法的应用

设有 12 名工作人员要完成 12 项工作, c_{ij} 表示第 i 名工作人员去完成第 j 项工作所需要的时间, 问如何指派, 使能完成任务最快。假设每个人完成各项工作的时间矩阵为

$$\boldsymbol{C}=\begin{bmatrix}15&8&16&24&5&23&4&20&22&14&16&7\\19&5&13&10&1&11&10&18&15&19&24&20\\1&9&22&2&8&8&21&4&17&6&12&17\\2&24&19&10&14&5&15&13&23&21&10&24\\21&17&8&2&3&4&1&13&11&17&20&1\\4&16&18&7&21&20&10&22&4&20&2&1\\7&19&10&16&8&16&7&11&15&22&14&18\\7&11&14&24&12&8&3&1&12&9&5&20\\20&5&2&9&10&8&7&24&9&11&9&6\\5&7&7&20&6&8&12&5&19&2&7&17\\14&19&15&3&24&18&10&13&20&23&15&7\\23&15&22&21&24&7&16&23&7&16&23&2\end{bmatrix}$$

这是一个最小化问题, 在 Matlabk 中调用主子程序“*fc*01(*C*, 1)”可得结果

$$\boldsymbol{e}=\begin{bmatrix}1&2&8&9&10&11&3&4&7&5&12&6\\2&5&8&3&10&4&1&6&7&12&9&11\end{bmatrix},$$

$s=40$, 即为问题的最优指派结果和完成任务的最少时间。

4　算法的推广应用

4.1　婚配问题

设有 n 对年龄相当的青年男女, 每个人的基本条件都不相同, 择偶条件也不尽相同, 任意一对男女青年配对, 都有相应的成功率, 问题是如何选择配对方案, 使得

①总的配对成功率最大;

②n 对男女青年都配对成功的成功率最大。

设 c_{ij} 为第 i 名男青年和第 j 名女青年的配对成功率(已知), 则问题①的模型为

$$\max z=\sum_{i=1}^{n}\sum_{j=1}^{n}c_{ij}x_{ij}$$

$$s.t.\begin{cases}\sum_{i=1}^{n}x_{ij}=1, & j=1,2,\cdots,n\\ \sum_{j=1}^{n}x_{ij}=1, & j=1,2,\cdots,n\\ X_{ij}=1\text{ 或 }0,\ i,\ j=1,2,\cdots,n\end{cases}\qquad(2)$$

问题②的目标函数可表示为 $\max z=\prod_{i=1}^{n}\prod_{j=1}^{n}c_{ij}x_{ij}$,

其约束条件与(2)式相同。

对问题①可以直接用该算法求解。问题②先将目标函数转化为和的形式来求解,根据似然方程的求极值的思想,令 $b_{ij}=\log(c_{ij})$,然后考虑相应的目标函数 $\max z=\sum_{i=1}^{n}\sum_{j=1}^{n}b_{ij}x_{ij}$,约束不变,同理求解。

例如:现有10对男女青年,已知配对成功率矩阵为

$$\boldsymbol{C}=\begin{bmatrix}
0.832 & 0.808 & 0.804 & 0.832 & 0.884 & 0.884 & 0.788 & 0.84 & 0.84 & 0.788\\
0.916 & 0.644 & 0.84 & 0.884 & 1 & 0.884 & 0 & 0.936 & 0.852 & 0.82\\
0.744 & 0.724 & 0.772 & 0.744 & 0.756 & 0.704 & 0.704 & 0.72 & 0.636 & 0.712\\
1 & 0.852 & 0.804 & 1 & 0.916 & 0.872 & 0.832 & 0.936 & 0.84 & 0.84\\
0.936 & 0.656 & 0.772 & 0.904 & 0.884 & 0.852 & 0.644 & 0.884 & 0.832 & 0.84\\
0.808 & 0.676 & 0.72 & 0.688 & 0.72 & 0.72 & 0 & 0.772 & 0.788 & 0.756\\
0.936 & 0.644 & 0.84 & 0.936 & 0.884 & 0.84 & 0 & 0.968 & 0.84 & 0.82\\
0.968 & 0.704 & 0.788 & 0.916 & 0.936 & 0.884 & 0.644 & 0.884 & 0.832 & 0.884\\
0.916 & 0.756 & 0.72 & 0.84 & 0.84 & 0.84 & 0.676 & 0.872 & 0.852 & 0.872\\
0.748 & 0.688 & 0.788 & 0.768 & 0.724 & 0.656 & 0.608 & 0.724 & 0.644 & 0.704
\end{bmatrix}$$

注意到,矩阵 $\boldsymbol{C}$ 中有0元素,但Matlab能处理对0取对数的情形。因为是最大化问题,易知0元素的存在对最优解不会产生影响。

求解问题①,运行子程序"ff01(C)"可得到结果

$$\boldsymbol{e}=\begin{bmatrix}2&4&6&7&9&10&3&1&5&8\\5&4&9&8&10&3&2&7&1&6\end{bmatrix},$$

$s=8.7480$。

求解问题②,运行 d=log(C);e=ff01(d);

可得结果

$$\boldsymbol{e}=\begin{bmatrix}2&4&6&7&5&9&8&1&10&3\\5&4&9&8&1&10&6&7&3&2\end{bmatrix},$$

$s=0.2474$。

4.2 锁具装箱问题(CUMCM—94B)

设一种锁具有5个槽,每个槽有6种高度,但要求必须有2个不同槽高,而且相邻2个槽的高度不能为3。2个锁具能够互开,当且仅当有4个对应的槽高度相同,最后一个槽高度差1。销售时每60个锁具装一箱,求出最大不能互开的锁具数。

首先,我们知道满足要求的合格锁具共5880种组合,可令$(x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5})$表示5个槽槽高分别为 $x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5}$的一个锁具。记 H_1 是所有槽高之和为奇数的锁具集合,H_2 是所有槽高之和为偶数的锁具集合。现有奇数个槽,每个槽有偶数高度的情况下,集合 H_1 和 H_2 之间存在双射 ϕ:$H_1\alpha H_2$,其对应关系为$(x_{i1}, x_{i2}, x_{i3}, x_{i4}, x_{i5})\alpha(x'_{i1}, x'_{i2}, x'_{i3}, x'_{i4}, x'_{i5})$。其中 $x'_{ij}=7-x_{ij}$,$(j=1, 2, \cdots, 5)$,所以 H_1 和 H_2 各有2940个元素。然后,将每个合格锁具都看作一个顶点,能够互开的2个顶点连一条边,可计算出互开总对数为22778,所以共有22778条边。因为能互开的锁具奇偶性不同,所以 H_1 和 H_2 之间有边相连,而 H_1, H_2 内部无边相连,即构成一个二分图。

最大不能互开锁具数,即为此图的最大独立顶点数,由图论的知识可知,"最大独立顶点数=顶点总数-最大对集的边数"。由于此图的邻接关系比较复杂,所以从理论上求出最大对集是很困难的,在这里我们可以用"匈牙利算法"来求解最大不能互开锁具数。

求出 H_1, H_2 和它们之间的邻接矩阵 $\boldsymbol{W}$(此略),再运行[jie, oue]=fc01($\boldsymbol{W}$);[e, $total$]=$fc02$(jie, oue, $\boldsymbol{W}$);

可得到最大对集 e(此略)及其边数 $total=2940$。所以最大独立顶点数=5880-2940=2940,即为最大不能互开锁具数。

子程序 fc01, fc02 也适用在2个分部顶点个数不相等的条件下寻找最大对集的问题。这时为了更快求得最大对集,在寻找可扩路时,应该从顶点个数较少的分部进行寻找。例如:若将"锁具装箱"的问题变为有6个槽,每个槽有6个槽高。此时合格锁具为35080种,H_1 有177720种,H_2 有17360种,互开总对数为168984。然而,邻接矩阵的维数为17720×17360,规模太大,不能用满矩阵存储,但在Matlab中可使用稀疏矩阵存储,可求出最大对集边数为17360,所以最大独立顶点数=35080-17360=17720。

4.3 求解完美对集和最大对集的问题

将上面的处理问题的方法可以推广到求任意赋权(完全或非完全)二分图的完美对集或最大对集问题。例如:设有二分类(X, Y)的赋权二分图 G, $X=\{x_1, x_2, \cdots, x_n\}$, $Y=(y_1, y_2, \cdots, y_n)$。如果边 x_iy_j 有权,$\omega_{ij}=\omega(x_i, y_j)$,则问题就是在这个赋权二分图中寻找一个具有最大权的完美对集问题。如果二分图每条边的权值相同,则问题就转化为求最大对集。

参考文献:

[1] 钱颂迪. 运筹学(修订版)[M]. 北京:清华大学出版社,1998.

[2] 全国大学生数学建模竞赛组委会. 全国大学生数学建模竞赛优秀论文集汇编[M]. 北京:中国物价出版社,2002.

[3] 代西武. 锁具装箱问题的补充讨论[J]. 数学的实践与认识,1996,(1):74—79.

[4] 杜瑞甫. 运筹图论[M]. 北京:北京航空航天大学出版社,2000.