用个人计算机解较大型混合整数线性规划问题*

胡　清　淮
（武汉化工学院）
魏　一　鸣
（北京科技大学）

SOLVING RELATIVELY LARGE SIZE MIXED INTEGER LINEAR PROGRAMMING PROBLEMS USING PERSONAL COMPUTERS

Hu Qinghuai
(*Wuhan Institute of Chemical Industry*)
Wei Yiming
(*Beijing University of Science and Technology*)

Abstract

This paper decribes the main methods and some improvements in the branch－and－bound algorithm and package making for solving relatively large size practical mixed integer linear programming problems using personal computers. A typical application using personal computer to solve a mail compilation problem is presented to demonstrate the effectiveness of the algorithm and package design.

1. 引　言

自从六十年代以来，对整数规划进行了大量关于数学模型及其解算方法的理论研究，并由此出版了一系列具有代表性的著作(见参考文献)．然而在实用编程和计算方面，尤其是针对较大型问题的报导却不够深入．而且解算这类问题仍然非常困难并且大多采用超大型或大型计算机．在改进算法方面所做的努力主要集中于两个方面：一是提高对每一个节点相关的线性规划问题的解算速度，二是采用更好的技术来选择节点和分

* 1994年11月28日收到.

枝以尽快地使其接近最优解．分枝定界法是有前途的方法，作者认为该法对解较大型混合整数或(0, 1)规划问题有许多优点．但是为了更好地实际应用，仍有必要进一步进行研究．

另一方面，个人计算机在工业中的广泛应用，在数学规划领域也表现出来．Stadtler等曾对线性规划问题的个人计算机软件做过比较(1986)，他的工作不论对进一步改进线性规划软件方面还是对采用更高档的个人计算机来开发具有更高水平的整数规划软件方面来说都是极为有益的．应用个人计算机求解较大型整数规划问题的困难主要在于内存有限和求解时间较长．因此，寻求合适的算法和充分利用内存与磁盘空间的技术以及某些程序设计技巧是至关重要的．

对于所谓的较大型问题，根据作者在应用个人计算机解决工业领域的实际问题的经验，认为当问题的规模多至10,000个变量(其中，几百个变量可为整数)或有几百个或更多的约束时，就可算为较大型问题．由于我们使用的是个人计算机，因此这里定义的规模比Forrest和Domlin等定义的中型问题的规模稍小[4]．然而在工业领域所遇到的许多实际问题基本上是这样的规模[6,9]，并可采用个人计算机有效地解决．

本文介绍用个人计算机求解较大型整数线性规划的问题基本算法——基于分枝定界算法[1,12,13]的主要方法，并详细阐述算法中的某些改进技术和编程技术．通过一邮单编辑问题的典型应用实例，说明了作者所采用的算法及设计的软件的有效性．

2. 问题的数学表示

一般有界变量混合型整数线性规划的模型可用矩阵和向量的形式表示如下：

$$\begin{aligned} &\min\ Z=CX+DY \\ &\text{S.T} \\ &\quad AX+EY=b, \\ &\quad L_x\leqslant X\leqslant U_x, \\ &\quad L_y\leqslant Y\leqslant U_y\ (Y\text{为整数}). \end{aligned} \tag{1}$$

通过消除下界约束的变换后，原问题可成为以下形式：

令 $X'=X-L_x$，$Y'=Y-L_y$，则 $b'=b-AL_x-EL_y$．同时将 X 作为 X'，Y 作为 Y'，b 作为 b' 则原问题成为以下形式：

$$\begin{aligned} &\min\ Z=CX+DY \\ &\text{S.T} \\ &\quad AX+EY=b, \\ &\quad X\leqslant U_x, \\ &\quad Y\leqslant U_y, \\ &\quad X\geqslant 0, \\ &\quad Y\geqslant 0\ (Y\text{为整数}). \end{aligned} \tag{2}$$

为简明起见，假定将问题(2)中的所有变量按先安排整数变量，后安排连续变量的排序方式排列在一个变量表中，这样，问题(2)又可改写成如下形式：

$$\begin{aligned}&\min\ Z=CX\\&\text{S.T}\\&\quad AX=b,\\&\quad 0\leqslant X\leqslant U,\\&\quad x_k\ \text{为整数},\ k\in N_1,\end{aligned}\tag{3}$$

其中 N_1 为问题(3)的变量表中的整数变量标记的集合. 问题(3)中的向量 C 包括问题(2)中的向量 C 和 D, 而矩阵 A 则包括问题(2)中矩阵 A 和 E.

对于某节点上的问题，把 A 分解成$[B, N_1, N_2]$, 其中 B 为基，N_1 和 N_2 分别为包含下界 $X_{N_1}=0$、上界 $X_{N_2}=U$ 相对应的非基列所组成的矩阵. 由此，在该节点的问题的基变量和目标函数可以表示为

$$\begin{aligned}&X_B=B^{-1}b-B^{-1}N_1X_{N_1}-B^{-1}N_2X_{N_2},\\&Z=C_BB^{-1}b+(C_{N_1}-C_BB^{-1}N_1)X_{N_1}+(C_{N_2}-C_BB^{-1}N_2)X_{N_2}.\end{aligned}\tag{4}$$

3. 算法及个人计算机程序

3.1 分枝定界法搜索解的一般过程

如前所述，所采用的算法基本上是分枝定界法[4,7,13]的改进方法. 最优解是通过从一个节点到另一个节点的“树形”搜索直到所有可用节点都检查完毕为止，最终解由所有存储的当前问题的解产生(如图 1). 这里一个节点表示问题(2)中不考虑整数要求但带有附加分枝约束的线性规划问题.

问题的限界是当该节点的线性规划解使所有的整数变量为整数而其目标函数值又比当前的界(最开始时取一个最劣值)要优时的该目标函数值. 这时从其他的被选择的节点继续进行搜索. 当得到问题的限界以后，那些目标函数值比这个界要劣的已存储的节点可作为劣节点而弃去. 随后对当前节点，其线性规划问题不存在可行解或者即使可行解存在但其目标函数值比当前问题的界要劣的，也当即弃去(已探测过). 否则，对该节点根据所选择的整数基变量 x_k (x_k 是 X_B 中的一个元素)进行分枝处理，其中 x_k 在当前未取整数值. 也就是说

$$x_k=[x_k^*]+f_k,\ 0<f_k<1,\tag{5}$$

其中$[x_k^*]$是不大于 x_k 的最大整数值. 在 x_k 处的分枝可由以下附加约束条件确定:

$$x_k\leqslant[x_k^*]\ \text{或}\ x_k\geqslant[x_k^*]+1.\tag{6}$$

对该节点带其中一个约束分枝的线性规划问题继续求解，而把带另一个约束分枝的问题构成一个新节点，并将其节点信息存储在计算机内存和磁盘文件中. 反复地执行这个过程，直至所有可能得到的节点都被检查或探测过，直至最终得到最优解(或无可行整数解)为止.

3.2 某些定义及节点识别

为了将算法和程序设计解释得更清楚，有必要对此问题的一些概念进行定义. 如图 1 所示，一个节点仅代表一个如前所述的特定的线性规划问题，而每一个分枝则表示由(6)式中的其中一个分枝约束条件. 节点按顺序编号，且与其在计算机内存中的存储位

置相对应．一个节点的层数表示从该节点直接到达顶节点所穿越过的节点数目．顶节点就是指顶层位置上层数为零的节点．父节点就是由两个分枝直接产生其两个子节点的节点．

显然，从表达式(4)及(6)以及图1，除问题的总信息(例如，矩阵 A，向量 C，b，U 以及 I 集合)外，问题中计算机识别一个节点所必须的信息，可归纳如下：

a) 对应该节点的基变量表；

b) 节点所在的层数；

c) 父节点编号；

d) 节点分枝的整数变量编号(xk)；

e) 分枝值($[x_k^*]$或$[x_k^*]+1$)以及其分枝状态(式(6)中哪一个约束)；

f) 在该节点处于上界的非基变量表；

g) 与父节点相对应的线性规划问题的目标函数值；

h) 非整性偏差的和(S_k，见下面6)．

信息(a)是用于产生当前节点问题的解基逆矩阵，(f)是用于描述所有非基变量是位于上界或下界的状态．在该节点状态下所有相关整数变量的上、下界变化则是利用信息b)—d)通过从当前节点穿过各父节点到达顶层节点的搜索过程来得出．信息(g)是用来快速判断当一个新的问题的界产生后，该节点的弃或留．信息(h)是用于节点选择．

3.3　计算方法和程序设计

1) 每一个节点的线性规划问题的算法用有界变量修正对偶单纯形法

有界变量的对偶单纯形法是在1958年由Wagner提出用以解线性规划问题[16]．本文论述的是有界变量修正对偶单纯形法．此法对解整数规划问题非常有效，他是用个人计算机解较大型整数线性规划问题的重要组成部分．

应用此法，以连续的方式对每个选来测试的节点所对应的线性规划问题求解．这实际上是当取式(6)中的前一式作为分枝时，意味着基变量 x_k 的上界变化，而取式(6)中的后一式作为分枝时意味着基变量 x_k 的下界变化．因此可用该算法快速地对变量上界或下界的变化进行灵敏度分析，并产生新的线性规划解．这是该法运行很快、且成为用个人计算机得到满意求解速度的主要原因之一．

2) 个人计算机存储技术

为了尽可能充分地使用机器的内存空间，除了只有基变量表存储在随机存取磁盘文件外，其他所有的节点信息(参见上述3.2节)都尽可能置于内存中．当一个新的问题的限界产生后，及时清除存储在内存中的无用信息，又收回内存空间．采用随机存取文件存储每一个节点的基变量表，是为了用覆盖无用记录的方法来缩短文件的长度，因此还有必要将这些表的记录号也存储在内存中，以便查找．

3) 节点解基逆矩阵的产生

我们知道，每一节点的线性规划问题都与一个基相对应，但是由于其数量之多，不可能将每一个节点中的解基逆矩阵都存储在机器中．然而可用修正单纯形法的乘积形式，把每一个节点的解基逆矩阵从当前的解基逆矩阵(或初始解基逆矩阵)转换得到．如图1中所示，令当前的解基逆矩阵(节点 A)为 B'^{-1}，根据存储在磁盘文件中的基变量表就可以

计算得到新选择的节点(图1中的 B)的解基逆矩阵 B'^{-1}. 例如，令 xr 为节点 B 问题中的一个基变量，且此变量不在节点 A 问题对应基变量表中. 对应的基变量 x_r 的单元基本矩阵 E_r、及新的解基逆矩阵、可通过替换原来的解基逆矩阵中的基变量 x_t(它不在新问题 B 所对应的基变量表中)，由以下表达式计算而得:

$$Y_r=\begin{Bmatrix} y_{1r} \\ y_{2r} \\ \cdots \\ y_{tr} \\ \cdots \\ y_{mr} \end{Bmatrix}=B'^{-1}A_r, \tag{7}$$

$$E_r=\begin{bmatrix} 1 & 0 & \cdots & -y_{1r}/y_{tr} & \cdots & 0 \\ 0 & 1 & \cdots & -y_{2r}/y_{tr} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & +1/y_{tr} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & -y_{mr}/y_{tr} & \cdots & 1 \end{bmatrix} (t\ row), \tag{8}$$

$$B^{-1}=E_r B'^{-1}. \tag{9}$$

注意这里仅需对不存在于原基变量表中的那些基变量进行计算. 因此，实际上这种处理相当简便.

4) 分枝策略

为了尽可能快速地获得问题初始限界或新限界，采取连续链的分枝策略. 分枝过程是不断地从一个节点到下一个节点，中间不改变问题解基逆矩阵直到新的问题的限界产生或问题被探测过为止. 分枝思想见图2所示.

5) 分枝选择的罚函数计算

对每个节点计算罚函数是为了估计当前问题的界并为当前检验的节点选择好的分枝. 这种方法在1965年由 Beale 和 Small 首次提出[1,2]，然后1971年 Domlin 又提出了“强罚函数”. 但所有发表的表达式都是一般形式而未考虑有界变量的情况. 本文为有界变量对

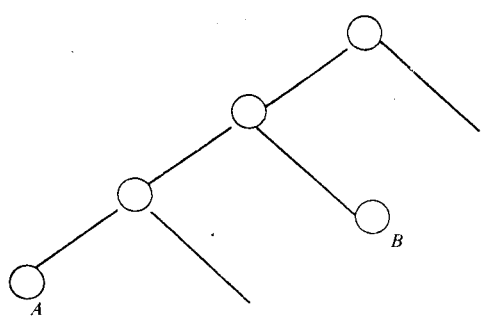

图1 从节点 A 到节点 B 的线性规划问题的转换

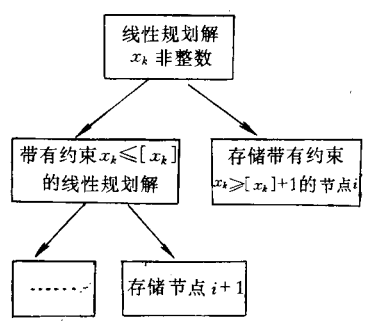

图2 节点上的分枝

偶修正单纯形法的应用，采用改进的罚函数.

当计算出相应于(6)式中上界约束 $x_k \leqslant [x_k^*]$ 的上罚函数 P_u 和相应于越束 $x_k \geqslant [x_k^*]+1$ 的下罚函数 P_d 以及在该节点估计的问题的限界 Z_e 求得，就可以把估计的限界与现有问题的限界进行比较来检查节点是否弃去或分枝. 如果结果优于当前界，则选择罚函数值较小分枝作进一步的测试，而把与另一个分枝相关的节点存储起来作为一个新的节点. 否则此节点就弃去(或已探测过).

6) 节点选择最优投影准则

最优启发式投影准则是已被证明比 LIFO 规则[6]更好用于选择新节点的准则. 也就是按下式确定所谓的每一个节点的非整性偏差总数 s_k

$$s_k = \sum_{i \in N_I} \min\{f_i, 1-f_i\}, \tag{10}$$

$$\lambda = (z^0 - \underline{z}) / s_0,$$

$$z_p^k = z^k - \lambda s_k,$$

其中 N_I 是整数变量的集合，z^0 和 s_0 分别为起始节点的目标函数值和非整性偏差，$\underline{z}$ 是当前整数解的限界，z^k 是当前节点的目标函数值. 取具有最大 z_p^k 的节点来进行下一步测试.

3.4　算法软件包

根据以上算法，作者在 386 微机上开发了“混合整数线性规划”软件包. 该软件包能在 2M 内存机器上处理含 400 个约束(不包括变量有界约束)，10,000 个带上、下界的变量(其中 1,000 个可为整数)的大问题. 当机器有扩展内存时，软件及易扩展用来解决更大型问题.

4. 应 用 实 例

整数规划问题的典型应用实例就是来自文献[9]中的“邮单编辑”问题. 由于数据原因，本文作了一些改变. 该问题是采用两种具有可用资源(工作日数，终端和 CPU 时间)的设施，在一些时间周期内给邮件单编辑工作安排工作进度计划. 也就是说，从几叠候选邮单中，在每周期可用总资源条件下，一个周期一个周期地选择邮单，送交两个设施之一进行编辑. 每一叠邮单均可被分派到任何一个设施中，但不能将其分成两部分. 当某叠邮单的编辑全部完成后，公司才可获得报酬. 每一叠邮单编辑的最早开始时间和最迟完成时间以及允许延续时间都事先确定. 当编辑某叠邮单的编辑开始后，它的编辑必须连续地进行而不允许有超过一个时间周期的间断. 在每一个时间周期内，每一叠邮单的编辑量都加上上限的限制以考虑编辑的连续和规律性. 由于在相连续的时间周期内劳动力使用的任何变化，都会受到取决于变化大小的经济惩罚，因此公司力图稳定劳动力的使用. 将一个年度分成 26 个相等的时间周期安排计划. 其总体目标就是使公司获得最大的利润.

为确定问题的模型，令 h 表示邮单 i 中在某一允许时间周期开始编辑、并在取决于开始时间的规定时间内完成编辑的“方式”. 那么，问题可用混合整数线性规划问题的方式表达如下:

表1 两种设施编辑分配表

List	Tol. vol.	Per. 1	2	3	4	5	6	7	8	9	10	11	12
A	.75				*.266*	*.380*	*.104*						
B	.55												
C	1.00			*.340*	*.320*	*.000*	*.340*						
D	.50	.250	.250										
E	4.00												
F	1.50											*.459*	*.271*
G	2.40												
H	1.00						*.179*	*.500*	*.321*				
I	.60		*.200*	*.200*	*.032*	*.168*							
J	3.00								*.217*	*.600*	*.600*	*.383*	*.600*
K	2.00						.500	.500	.288	.212	.500		
L	5.00			.304	.630	.630	.630	.580	.630	.630	.336	.630	
M	1.50												
N	6.00							.628	.786	.860	.860	.286	.860
O	8.00										.000	.800	.800
P	7.50												

List	Tol. vol.	13	14	15	16	17	18	19	20	21	22	23	24	25	26
A	.75														
B	.55	*.000*	*.270*	*.280*											
C	1.00														
D	.50														
E	4.00					.430	.430	.430	.430	.570	.570	.570	.570		
F	1.50	*.271*	*.500*												
G	2.40								*.228*	*4.00*	*.400*	*.228*	*.345*	*.400*	*.400*
H	1.00														
I	.60														
J	3.00	*.600*													
K	2.00														
L	5.00														
M	1.50								*.380*	*.231*	*.231*	*.380*	*.279*		
N	6.00	.860	.860												
O	8.00	.800	.800	.800	.800	.800	.800	.800	.800						
P	7.50				*.460*	*.460*	*.940*	*.940*	*.940*	*.940*	*.940*	*.940*	*.940*		

注意：斜数字为设施2的分配，其他为设施1的分配(续)

$$\text{Max}\{\sum_i\sum_h\sum_k P_i X_{ihk}-\sum_j\sum_k(UPEN_k U_{jk}+DPEN_k V_{jk})\}$$

S.T.

$$\sum_h\sum_k X_{ihk}\leqslant 1,\ \text{对所有 } i,$$

$$\sum_j Y_{ihjk}\leqslant L_i X_{ihk},\ \text{对所有 } i、h、k,$$

$$\sum_i\sum_h Y_{ihk}r_{mik}\leqslant R_{mjk},\ \text{对所有 } m、j、k,$$

$$\sum_i\sum_h(Y_{ihk,j+1}-Y_{ikk,j})r_{lik}=U_{jk}-V_{jk},\ \text{对所有 } j、k,$$

$$X_{ih}=1\text{ 或 }0,\ \text{对所有的 } i、h,$$

$$0\leqslant Y_{ihjk}\leqslant b_{ik},\ \text{对所有 } i、h、j、k,$$

$$U_{jk},\ V_{jk}\geqslant 0,\ \text{对所有 } j、k,$$

其中

X_{ihk}——对邮单 i、“方式”h 及设施 k 中的整数变量;

Y_{ihjk}——对邮单 i、方式 h、在时间周期 j 期间，利用设施 K，所计划的编辑量;

U_{jk}，V_{jk}——度量设施 k 从时间周期 j 到时间周期$(j+1)$，资源(工时)实际利用增加或减少的变量;

P_i——编辑邮单 i 的利润;

$UPEN_k$——设施 K 从一个时间周期到下一个时间周期因工时增加的单位罚款数;

$DPEN_k$——设施 K 从一个时间周期到下一个时间周期因工时减少单位罚款数;

r_{mik}——设施 K 对邮单 i 的资源 m 消耗率;

R_{mjk}——在时间周期 j 内设施 k 可获得资源 m 的总量(其中 $m=1$ 为工时);

b_{ik}——利用设施 k 每时间周期内邮单 i 编辑的上界.

该问题具有 736 个变量(其中 94 个为 0－1 型整数)和 316 个约束条件. 采用本软件在 COMPAQ4 /33 微机上求解此问题，耗时仅为 15 分钟. 计算结果列于表 1. 注意除邮单 4 的可用的工作方式数仅为 2 种外，其他每叠邮件工作方式都为 3 种.

在文献[9]中的原始问题的模型有 229 个约束，806 个连续变量和 65 个 0－1 型变量. 该问题在 CDC6600 机器上，采用商业 OPHELIE－II 程序求解耗费 107 个系统秒.

5. 结　论

整数规划广泛用于各个工业领域. 然而对于大型问题，应用微机求解非常困难. 实践证明，应用本文提出的方法和所设计的软件在个人计算机上求解较大型整数规划、混合整数规划或 0－1 型整数规划问题是非常实用而有效的. 为了获得更满意的结果，在算法和程序设计技术(尤其是分枝技术)相结合方面，还有必要做进一步的改进.

参 考 文 献

[1] E. M. L. Beale, Survey of Integer Programming, Operations Research Quatrerly, Vol.16, (1965) 219—228.

[2] E. M. L. Beale, Mathematical Programming in Practice, Pitman, London, 1968.

[3] H. F. Bruce, S. H. Frederick, The Accelerated Bound-and-Branch Algorithm for Integer Programming, Operations Research, 23:3 (1975) 406—425.

[4] R. J. Dakin, A Tree-Search Algorithm for Mixed Integer Programming Problems. Computer Journal, Vol.8, (1965) 250—255.

[5] E. B. David, A Convergent Duality Theory for Integer Programming, Operations Research, 25:3 (1977) 419—433.

[6] J. J. H. Forrest, J. P. H. Hirst, J. A. Tomlin, Practical Solution of Large Mixed Integer Programming Problems with UMPIRE, Management Science, 20:5 (1974) 736—773.

[7] A. H. Land, A. G. Doig, An Automatic Method for Solving Discrete Programming Problems. Econometrica, Vol. 28 (1960) 497—520.

[8] B. Ley, A. L. Soyster, Integer Programming with Bounded Variables via Canonical Separation, Operations Research Society, Vol.29 (1978) 477—488.

[9] K. Rimas, G. P. Anthony, An Application of Mixed Integer Programming in the Direct Mail Industry, Management Science, 20:5 (1974) 788—792.

[10] L. Schrage, L. Wolsey, Sensitivity Analysis for Branch and Bound Integer Programming, Operations Research, 33:5 (1985) 1008—1023.

[11] H. Stadtler, M. Groeneveld, H. Hermannsen, A Comparison of LP Software on Personal Computers for Industrial Application, European Journal of Operational Research, Vol. 35, (1988) 146—164.

[12] H. A. Taha, Integer Programming, Theory, Applications and Computations, John Wiley & Sons, New York, 1975.

[13] J. A. Tomlin, An Improved Branch-and-Bound Method for Integer Programming, Operations Research, Vol.19, (1971) 1070—1075.

[14] C. A. Trauth, R. E. Woolsey, Integer Linear Programming: A Study in Computational Efficiency, Management Science, Vol.15, (1969) 481—493.

[15] L. E., Jr., Trotter, C. M. Shetty. An Algorithm for the Bounded Variable Integer Programming Problem, Journal of the A. C. M., Vol. 21, (1974) 505—513.

[16] H. M. Wagner, The Dual Simplex Algorithm for Bounded Variables, Naval Research Logistics, Vol. 5, (1958) 257—261.