第5章 整数规划

整数规划解决决策变量是整数值的线性规划和非线性规划问题，是在有限个可供选择的方案中寻找满足一定标准的最好方案。整数规划自1958年由R.E.戈莫里提出割平面法之后形成独立分支，60多年来发展出很多方法。目前比较成功又流行的方法是分枝定界法和割平面法。0-1规划在整数规划中有重要的地位，有界变量的整数规划与0-1规划等价，同时许多实际问题（如背包问题、送货问题、指派问题等）都可以归为这类问题。整数规划的应用范围也极其广泛。它不仅在工业和工程设计、科学研究方面有许多应用，在计算机设计、系统可靠性、编码和经济分析等方面也有新的应用。

5.1 本章内容

本章主要介绍以下内容。

（1）整数规划的基本概念。

（2）实际例子的建模方法。

（3）割平面法及其MATLAB实现。

（4）分支定界法及其MATLAB实现。

（5）0-1整数规划及其MATLAB实现。

解整数规划最典型的做法是逐步生成一个相关的问题，称为原问题的衍生问题。对每个衍生问题又伴随一个更易求解的松弛问题（衍生问题称为松弛问题的源问题）。通过松弛问题的解来确定其源问题的归宿，即确定源问题应被舍弃还是再生成一个或多个它本身的衍生问题来替代它。随后再选择一个尚未被舍弃或替代的原问题的衍生问题。重复以上步骤，至不再有未解决的衍生问题为止。

整数线性规划问题与线性规划问题相类似，只是一些变量要求为整数。要求部分或全部决策变量必须取整数值的规划问题称为整数规划。不考虑整数条件，由余下的目标函数和约束条件构成的规划问题称为该整数规划问题的松弛问题。若松弛问题是一个线性规划，则称该整数规

划为整数线性规划。如果没有连续的变量,则是整数规划问题。

例如,给定矩阵 $\boldsymbol{A}$、$\boldsymbol{B}$ 和向量 $\boldsymbol{b}$、$\boldsymbol{c}$、$\boldsymbol{d}$,则有

$$\min \boldsymbol{c}'\boldsymbol{x}+\boldsymbol{d}'\boldsymbol{y}$$

使得

$$\begin{cases}\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{y}=\boldsymbol{b}\\ \boldsymbol{x},\boldsymbol{y}\geqslant 0\\ \boldsymbol{x}\text{ 是整数}\end{cases}$$

这是一个混合整数线性规划问题,即不是其中所有变量均需要取整数值。

给定如上整数线性规划问题,对应的松弛问题即为

$$\min \boldsymbol{c}'\boldsymbol{x}+\boldsymbol{d}'\boldsymbol{y}$$

使得

$$\begin{cases}\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{y}=\boldsymbol{b}\\ \boldsymbol{x},\boldsymbol{y}\geqslant 0\end{cases}$$

整数线性规划数学模型的一般形式为

$$\max(\text{或 }\min)z=\sum_{j=1}^{n}c_j x_j$$

使得

$$\begin{cases}\sum\limits_{j=1}^{n}a_{ij}x_j\leqslant(\text{或}=,\text{或}\geqslant)b_i, & i=1,2,\cdots,m\\ x_j\geqslant 0, & j=1,2,\cdots,n\\ x_1,x_2,\cdots,x_n\text{ 是整数}\end{cases}$$

整数线性规划问题可以分为下列几种类型。

(1) 纯整数线性规划(pure integer linear programming)。指全部决策变量都必须取整数值的整数线性规划,有时也称为全整数规划。

(2) 混合整数线性规划(mixed integer linear programming)。指决策变量中有一部分必须取整数值,另一部分可以不取整数值的整数线性规划。

(3) 0-1 型整数线性规划(zero-one integer linear programming)。指决策变量只能取值 0 或 1 的整数线性规划。

5.2 建模方法

本节将介绍用于解决线性规划问题的一些建模方法。因为没有一个系统的方法来公式化地离散最优化问题,所以设计一个好的模型显得尤为重要。

5.2.1 二元选择

可以运用一个二进制变量 x，根据两个选项的选择将 x 置为 0 或 1。

若变量只能取值 0 或 1，称其为 0-1 变量。0-1 变量作为逻辑变量(logical variable)，常被用于表示系统是否处于某个特定状态，或者决策时是否取某个特定方案。例如

$$x=\begin{cases}1, & \text{当决策取方案 } P \text{ 时}\\ 0, & \text{当决策不取方案 } P \text{ 时(即取 } \bar{P} \text{ 时)}\end{cases}$$

当问题含有多项要素，且每项要素均有两种选择时，可用一组 0-1 变量来描述。一般地，设问题有有限项要素 $E_1,E_2,\cdots,E_n$，其中每项 E_j 有两种选择 A_j 和 $\bar{A}_j$($j=1,2,\cdots,n$)，则可令

$$x_j=\begin{cases}1, & \text{若 } E_j \text{ 选择 } A_j\\ 0, & \text{若 } E_j \text{ 选择 } \bar{A}_j\end{cases} \qquad j=1,2,\cdots,n$$

在应用中，有时会遇到变量可以取多个整数值的问题。这时，利用 0-1 变量是二进制变量(binary variable)的性质，可以用一组 0-1 变量来取代该变量。例如，变量 x 可取 0 与 9 之间的任意整数时，可令

$$x=2^0x_0+2^1x_1+2^2x_2+2^3x_3\leqslant 9$$

其中，x_0,x_1,x_2,x_3 皆为 0-1 变量。

0-1 变量不仅广泛应用于科学技术问题，在经济管理问题中也有十分重要的应用。

例 5-1 0-1 背包问题。

有 n 个物品，第 j 个物品的重量为 w_j，价值为 c_j。一个背包所能承受重量的上限为 K。如何选择物品使得总价值最大？这里假定不能选取部分物品，这种情况称为 0-1 背包问题(示意图如图 5-1 所示)。

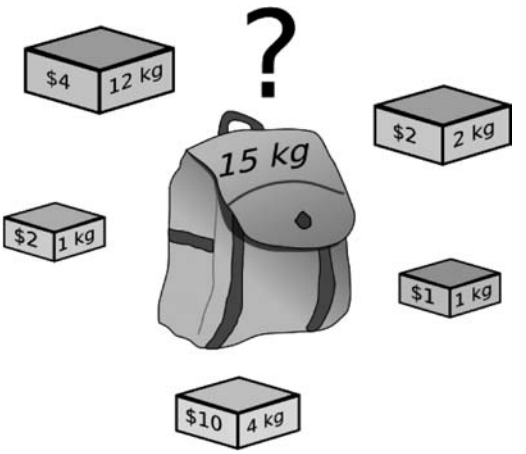

图 5-1 0-1 背包问题示意图

解：为了给这个问题建模，可以引入二进制变量 x_j。如果物品 j 被选择，那么 $x_j=1$，否则 $x_j=0$。这个问题可以公式化为

$$\max \sum_{j=1}^{n} c_j x_j$$

使得

$$\begin{cases} \sum_{j=1}^{n} w_j x_j \leqslant K \\ x_j \in \{0,1\}, \quad j=1,2,\cdots,n \end{cases}$$

例 5-2 固定费用问题。

有 3 种资源(A,B,C)被用于生产 3 种产品，资源量、产品单件可变费用及售价、资源单耗量及组织 3 种产品生产的固定费用见表 5-1。要求制订一个生产计划，使总收益最大。

表 5-1 资源及产品的相关数据

项目	单耗量			
	产品Ⅰ	产品Ⅱ	产品Ⅲ	资源量
资源 A	2	4	8	500
资源 B	2	3	4	300
资源 C	1	2	3	100
单件可变费用	4	5	6	
固定费用	100	150	200	
单件售价	8	10	12	

解：总收益等于销售收入减去生产上述产品固定费用和可变费用之和。建模的困难主要在于事先不能确切知道某种产品是否生产，进而不能确定相应的固定费用是否发生。下面借助 0-1 变量解决这个困难。

设 x_j 是第 j 种产品的产量，$j=1,2,3$；再设

$$y_i=\begin{cases} 1, & \text{若生产第 } j \text{ 种产品(即 } x_j>0) \\ 0, & \text{若不生产第 } j \text{ 种产品(即 } x_j=0) \end{cases} \qquad j=1,2,3$$

则问题的整数规划模型是

$$\max z=(8-4)x_1+(10-5)x_2+(12-6)x_3-100y_1-150y_2-200y_3$$

使得

$$\begin{cases} 2x_1+4x_2+8x_3 \leqslant 500 \\ 2x_1+3x_2+4x_3 \leqslant 300 \\ x_1+2x_2+3x_3 \leqslant 100 \\ x_1 \leqslant M_1 y_1 \\ x_2 \leqslant M_2 y_2 \\ x_3 \leqslant M_3 y_3 \\ x_j \geqslant 0 \text{ 且为整数}, \quad j=1,2,3 \\ y_j=0 \text{ 或 } 1, \quad j=1,2,3 \end{cases}$$

其中，M_j 为 x_j 的某个上界。例如，根据第 3 个约束条件，可取 $M_1=100$，$M_2=50$，$M_3=34$。

如果生产第 j 种产品，则其产量 $x_j>0$。此时，由约束条件 $x_j\leqslant M_j y_j$，知 $y_j=1$，相应的固定费用在目标函数中将被考虑。如果不生产第 j 种产品，则其产量 $x_j=0$。此时，由约束条件 $x_j\leqslant M_j y_j$ 可知，y_j 可以是 0，也可以是 1。但 $y_j=1$ 不利于目标函数 z 的最大化，因而在问题的最优解中必然是 $y_j=0$，从而相应的固定费用在目标函数中将不被考虑。

5.2.2 强制约束

离散最优化问题的一个共同特点是一些变量之间相互不独立。特别地，假设只能在选择了 B 的情况下才能选择 A，可以引入二进制变量 $x(y)$，如果选择 $A(B)$，则 $x=0(y=0)$。那么 A 与 B 之间的关联就可以表示为

$$x\leqslant y$$

例 5-3 设施选址问题。

如图 5-2 所示，设有 n 个可能建设设施的地点，m 个需要的用户。在 j 地建设设施需要花费 c_j，为用户 i 提供设施 j 需要花费 d_{ij}。如何建设设施使得总花费最少？

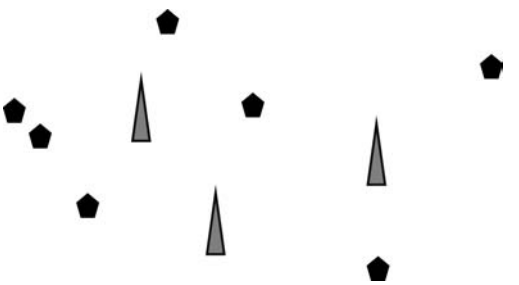

图 5-2 设施选址问题示意图

解：对每个地点 j，设二进制变量 y_j，如果 j 处建设了设备，那么 $y_j=1$，否则 $y_j=0$。设二进制变量 x_{ij}，如果给用户 i 配备设备 j，那么 $x_{ij}=1$，否则 $x_{ij}=0$。设施选址问题公式化为(FL)

$$\min\sum_{j=1}^{n}c_j y_j+\sum_{i=1}^{m}\sum_{j=1}^{n}d_{ij}x_{ij}$$

使得

$$\begin{cases}\sum\limits_{j=1}^{n}x_{ij}=1, & \forall i\\ x_{ij}\leqslant y_j, & \forall i,j\\ x_{ij},y_j\in\{0,1\}, & \forall i,j\end{cases}$$

如果地点 j 处没有设备，即 $y_j=0$，那么用户不可能配备 j 处的设备，则 $x_{ij}=0$。

5.2.3 变量之间的关系

约束条件

$$\sum_{j=1}^{n} x_j \leqslant 1$$

其中所有的变量都是二元的。意味着最多只有一个 x_j 能为 1。类似地，约束条件

$$\sum_{j=1}^{n} x_j = 1$$

那么有且只有一个变量为 1。

5.2.4 析取约束

设 $\boldsymbol{x}$ 为一个非负的决策变量。假设有两个约束条件 $\boldsymbol{a}'\boldsymbol{x} \geqslant \boldsymbol{b}$，$\boldsymbol{c}'\boldsymbol{x} \geqslant \boldsymbol{d}$，其中 $\boldsymbol{a}'$ 和 $\boldsymbol{c}'$ 均为非负。

设二进制变量 y，有以下约束条件

$$\boldsymbol{a}'\boldsymbol{x} \geqslant y\boldsymbol{b}$$
$$\boldsymbol{c}'\boldsymbol{x} \geqslant (1-y)\boldsymbol{d}$$
$$y \in \{0,1\}$$

更一般地，假设有 m 个约束条件 $\boldsymbol{a}_i'\boldsymbol{x} \geqslant b_i$，$i=1,2,\cdots,m$，对任意的 i，有 $a_i \geqslant 0$，并且至少其中 k 个满足。引入 m 个二进制变量 y_i，$i=1,2,\cdots,m$，有以下约束条件

$$\boldsymbol{a}'\boldsymbol{x} \geqslant b_i y_i, \quad i=1,2,\cdots,m$$
$$\sum_{i=1}^{m} y_i \geqslant k$$
$$y_i \in \{0,1\}, \quad i=1,2,\cdots,m$$

5.2.5 值的约束范围

假设变量 x 的值在集合 $\{a_1,a_2,\cdots,a_m\}$ 中取得。可以引入 m 个二进制变量 y_j，$j=1,2,\cdots,m$，有约束条件

$$x = \sum_{j=1}^{m} a_j y_j$$
$$\sum_{j=1}^{m} y_j = 1$$
$$y_j \in \{0,1\}$$

5.2.6 分段线性成本函数

成本函数为定义在$[a_1,a_k]$上的分段线性函数$f(x)$。区间$[a_1,a_k]$上的x可以表示为

$$x=\sum_{i=1}^{k}\lambda_i a_i$$

其中,$\lambda_1,\lambda_2,\cdots,\lambda_k$非负且和为1。

但这样x的表示不是唯一的,假定至多两个相邻的λ_i是非零的,这样任何的$x\in[a_i,a_{i+1}]$可以表示为$x=\lambda_i a_i+\lambda_{i+1}a_{i+1}$且

$$f(x)=\sum_{i=1}^{k}\lambda_i f(a_i)$$

引入二进制变量$b_i,i=1,2,\cdots,k$,公式化至多两个相邻的λ_i非零的条件,得到这个问题的公式化形式为

$$\min\sum_{i=1}^{k}\lambda_i f(a_i)$$

使得

$$\begin{cases}\sum_{i=1}^{k}\lambda_i=1\\ \lambda_1\leqslant y_1\\ \lambda_i\leqslant y_{i-1}+y_i,\quad i=2,3,\cdots,k-1\\ \lambda_k\leqslant y_{k-1}\\ \sum_{i=1}^{k-1}y_i=1\\ \lambda_i\geqslant 0\\ y_i\in\{0,1\}\end{cases}$$

5.3 整数规划的例子

例 5-4 某服务部门各时段(每2h为一时段)需要的最少服务员人数见表5-2。按规定,服务员连续工作8h(即4个时段)为一班。现要求安排服务员的工作时间,使服务部门服务员总数最少。

表 5-2 各时段需要的最少服务员人数

时段	1	2	3	4	5	6	7	8
最少服务员人数	10	8	9	11	13	8	5	3

解：设在第 j 时段开始时上班的服务员人数为 x_j。由于第 j 时段开始时上班的服务员将在第($j+3$)时段结束时下班，故决策变量只需要考虑 x_1、x_2、x_3、x_4、x_5。

问题的数学模型为

$$\min z = x_1 + x_2 + x_3 + x_4 + x_5$$

使得

$$\begin{cases} x_1 \geqslant 10 \\ x_1 + x_2 \geqslant 8 \\ x_1 + x_2 + x_3 \geqslant 9 \\ x_1 + x_2 + x_3 + x_4 \geqslant 11 \\ x_2 + x_3 + x_4 + x_5 \geqslant 13 \\ x_3 + x_4 + x_5 \geqslant 8 \\ x_4 + x_5 \geqslant 5 \\ x_5 \geqslant 3 \\ x_1, x_2, x_3, x_4, x_5 \geqslant 0, \text{且均取整数值} \end{cases}$$

这是一个纯整数规划问题。

例 5-5 现有资金总额为 B。可供选择的投资项目有 n 个，项目 j 所需投资额和预期收益分别为 a_j 和 c_j($j=1,2,\cdots,n$)。此外，由于种种原因，有 3 个附加条件：第一，若选择项目 1，则必须同时选择项目 2，反之则不一定；第二，项目 3 和项目 4 中至少选择一个；第三，项目 5、项目 6 和项目 7 中恰好选择两个。应当怎样选择投资项目，才能使总预期收益最大？

解：每个投资项目都有被选择和不被选择两种可能，为此令

$$x_j = \begin{cases} 1, & \text{对项目 } j \text{ 投资} \\ 0, & \text{对项目 } j \text{ 不投资} \end{cases} \qquad j = 1,2,\cdots,n$$

这样，问题可表示为

$$\max z = \sum_{j=1}^{n} c_j x_j$$

使得

$$\begin{cases} \sum_{j=1}^{n} a_j x_j \leqslant B \\ x_2 \geqslant x_1 \\ x_3 + x_4 \geqslant 1 \\ x_5 + x_6 + x_7 = 2 \\ x_j = 0 \text{ 或 } 1, \quad j = 1,2,\cdots,n \end{cases}$$

这是一个 0-1 规划问题。其中，中间 3 个约束条件分别对应 3 个附加条件。

例 5-6 工厂 A_1 和 A_2 生产某种物资。由于该种物资供不应求，故需要再建一家工厂。

相应的建厂方案有 A_3 和 A_4 两个。这种物资的需求地有 B_1、B_2、B_3、B_4 4 个。各工厂年生产能力、各地年需求量、各工厂至各需求地的单位物资运费 c_{ij} 见表 5-3。

表 5-3　各工厂及各需求地的相关数据

A_i	单位物资运费 c_{ij}/(万元/千吨)				生产能力/(千吨/年)
	B_j				
	B_1	B_2	B_3	B_4	
A_1	2	9	3	4	400
A_2	8	3	5	7	600
A_3	7	6	1	2	200
A_4	4	5	2	5	200
需求量/(千吨/年)	350	400	300	150	

工厂 A_3 和 A_4 开工后，每年的生产费用估计分别为 1200 万元和 1500 万元。现要决定应该建设工厂 A_3 还是 A_4，才能使今后每年的总费用(即全部物资运费和新工厂生产费用之和)最少。

解：这是一个物资运输问题，其特点是事先不能确定应该建 A_3 和 A_4 中的哪一个，因而不知道新厂投产后的实际生产费用。为此，引入 0-1 变量

$$y=\begin{cases}1, & \text{若建工厂 } A_3\\ 0, & \text{若建工厂 } A_4\end{cases}$$

再设 x_{ij} 为由 A_i 运往 B_j 的物资数量($i,j=1,2,3,4$)，单位是千吨；z 表示总费用，单位是万元。

问题的数学模型为

$$\min z=\sum_{i=1}^{4}\sum_{j=1}^{4}c_{ij}x_{ij}+[1200y+1500(1-y)]$$

使得

$$\begin{cases}x_{11}+x_{21}+x_{31}+x_{41}=350\\ x_{12}+x_{22}+x_{32}+x_{42}=400\\ x_{13}+x_{23}+x_{33}+x_{43}=300\\ x_{14}+x_{24}+x_{34}+x_{44}=150\\ x_{11}+x_{12}+x_{13}+x_{14}=400\\ x_{21}+x_{22}+x_{23}+x_{24}=600\\ x_{31}+x_{32}+x_{33}+x_{34}=200y\\ x_{41}+x_{42}+x_{43}+x_{44}=200(1-y)\\ x_{ij}\geqslant 0,\quad i,j=1,2,3,4\\ y=0\text{ 或 }1\end{cases}\tag{5-1}$$

上述数学模型中，目标函数由两部分组成，和式部分为由各工厂运往各需求地的物资总运费，加号后的中括号部分为建工厂 A_3 或 A_4 后相应的生产费用。约束条件式(5-1)中的前 8 个约束条件为供需平衡条件。第 7 个和第 8 个约束条件中含 0-1 变量 y。若 $y=1$，则表示建工厂 A_3。此时，第 7 个约束条件就是对工厂 A_3 的运出量约束。再由第 8 个约束条件，必有 $x_{41}=x_{42}=x_{43}=x_{44}=0$；反之，若 $y=0$，则表示建工厂 A_4。

显然，这是一个混合整数规划问题。

5.4 问题的公式化

因为计算的复杂度是根据变量数 n，限制条件数 m 呈指数增长的，要更好地公式化一个问题，就要尽可能减少使用的变量数和限制条件数。

这个问题就从研究整数线性规划问题的松弛问题入手。松弛问题的解也是整数线性规划问题的解。

在例 5-3 提到的设施选址问题中，可以考虑以下的等价的公式化：

$$\min \sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij}$$

使得

$$\begin{cases} \sum_{j=1}^{n} x_{ij} = 1, & \forall i \\ \sum_{j=1}^{n} x_{ij} \leqslant m y_j, & \forall j \\ x_{ij}, y_j \in \{0,1\}, & \forall i,j \end{cases}$$

这种公式化叫作 aggregate facility location formulation(AFL)。

可以看到限制条件 $\sum_{j=1}^{n} x_{ij} \leqslant m y_j$ 使得当 $y_j=0$ 时，$x_{ij}=0$，如果 $y_j=1$，那么 x_{ij} 可以等于 1。也就是说这个限制条件与之前得出的 $x_{ij} \leqslant y_j, \forall i,j$ 是等价的。所以两组公式化均可解决设施选址问题。原来的公式化得到的结果有 $m+mn$ 个限制条件，而新得到的公式只有 $m+n$ 个限制条件。

为了比较两种公式化，考虑与它相对应的线性规划松弛问题。对于松弛问题，整数的限制条件 $x_{ij}, y_j \in \{0,1\}$ 改为 $0 \leqslant x_{ij} \leqslant 1, 0 \leqslant y_j \leqslant 1$。

两个松弛问题的可行解分别为

$$P_{FL} = \left\{ (x,y) \,\middle|\, \sum_{j=1}^{n} x_{ij} = 1, \quad \forall i \right.$$
$$x_{ij} \leqslant y_j, \quad \forall i,j$$
$$\left. 0 \leqslant x_{ij} \leqslant 1, 0 \leqslant y_j \leqslant 1 \right\}$$

$$P_{AFL}=\left\{(x,y)\ \middle|\ \sum_{j=1}^{n}x_{ij}=1,\forall i\right.$$

$$\sum_{j=1}^{n}x_{ij}\leqslant my_j,\forall j$$

$$\left.0\leqslant x_{ij}\leqslant 1,0\leqslant y_j\leqslant 1\right\}$$

显然，$P_{FL}\subset P_{AFL}$。也就是说，FL 公式化方法对应的松弛问题的可行解比 AFL 方法的更加接近整数线性规划问题的解，但是 AFL 公式化方法限制条件的数量有非常明显的减少。

那么什么是整数线性规划问题理想的公式化呢？设 $T=\{x^1,x^2,\cdots,x^k\}$ 是某个整数规划问题的可行整数解的集合。假设 T 是有限的，考虑 T 的凸包

$$\mathrm{CH}(T)=\left\{\sum_{i=1}^{k}\lambda_i x^i\ \middle|\ \sum_{i=1}^{k}\lambda_i=1,\lambda_i\geqslant 0,x^i\in T\right\}$$

集合 $\mathrm{CH}(T)$是一个极值点为整数的多边形，且任何线性规划松弛问题的可行解集合 P 满足 $\mathrm{CH}(T)\subset P$。如果我们准确地知道 $\mathrm{CH}(T)$，即可以将 $\mathrm{CH}(T)$表示为 $\mathrm{CH}(T)=\{x\mid Dx\leqslant d\}$，就可以通过寻找线性规划问题

$$\min \boldsymbol{c}'\boldsymbol{x}$$

使得

$$x\in \mathrm{CH}(T)$$

的极值解来解决线性整数线性规划问题

$$\min \boldsymbol{c}'\boldsymbol{x}$$

使得

$$x\in T$$

理想的公式化得到的线性规划松弛问题的解就是整数线性规划整数可行解的凸包，但这是很难做到的。为得到较好的公式化，希望能够得到尽量接近 $\mathrm{CH}(T)$的多边形。

整数规划问题公式化的质量可以由松弛问题解与整数可行解的凸包 $\mathrm{CH}(T)$来决定。

一般情况下，松弛问题的最优解不会刚好满足变量的整数约束条件，因而不是整数规划的可行解，自然也不是整数规划的最优解。此时，若对松弛问题的这个最优解中不符合整数要求的分量简单取整，则所得到的解不一定是整数规划问题的最优解，甚至也不一定是整数规划问题的可行解。

例 5-7 考虑下面的整数规划问题

$$\max z=x_1+4x_2$$

使得

$$\begin{cases}-2x_1+3x_2\leqslant 3\\ x_1+2x_2\leqslant 8\\ x_1,x_2\geqslant 0\end{cases}$$

解：图 5-3 中四边形 OBPC 及其内部为松弛问题的可行域，其中整数格点为整数规划

问题的可行解。根据目标函数等值线的优化方向，从直观可知，$\boldsymbol{P}$ 点（$x_1=18/7, x_2=19/7$）是其松弛问题的最优解，其目标函数值 $z=94/7$。在 $\boldsymbol{P}$ 点附近对 x_1 和 x_2 简单取整，可得四点：A_1、A_2、A_3 和 A_4。其中，A_1 和 A_2 为非可行解；A_3 和 A_4 虽为整数可行解，但不是最优解。本例整数规划的最优解为 A^* 点（$x_1=4, x_2=2$），其目标函数值 $z=12$。

由于整数规划及其松弛问题之间的上述特殊关系，像例 5-7 中先求松弛问题最优解，再用简单取整的方法虽然直观简单，却并不是求解整数规划的有效方法。

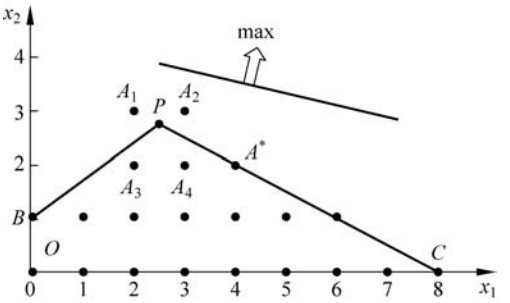

图 5-3　四边形可行域示意图

5.5　割平面法

考虑整数规划问题

$$\max z=\sum_{j=1}^{n} c_j x_j$$

使得

$$\begin{cases} \sum\limits_{j=1}^{n} a_{ij} x_j = b_i, & i=1,2,\cdots,m \\ x_j \geqslant 0, & j=1,2,\cdots,n \\ x_j \text{ 取整数}, & j=1,2,\cdots,n \end{cases} \tag{5-2}$$

设其中 $a_{ij}(i=1,2,\cdots,m; j=1,2,\cdots,n)$ 和 $b_i(i=1,2,\cdots,m)$ 皆为整数（若不为整数时，可乘上一个倍数化为整数）。

割平面法的基本思路如下。

（1）解对应的松弛问题，设 $\boldsymbol{x}^*$ 为可行解。

（2）若 $\boldsymbol{x}^*$ 是整数解，则算法停止。

（3）若不是，增加一个线性不等式的限制条件使得所有的整数解满足，而 $\boldsymbol{x}^*$ 不满足，返回第一步，重复直至停止。

设 $\boldsymbol{x}^*$ 是松弛问题可行的基本解，且至少有一个基本变量。设 N 为非基本变量的集合。任意整数线性规划的解满足对任意 $x \in N$，$x_i=0$。这也是线性规划问题的解，一定与可行

基本解 $\boldsymbol{x}^*$ 相同。所以可行的整数解应该满足

$$\sum_{j\in N} x_j \geqslant 1$$

这是我们加入松弛问题的不等式。任何整数解都满足它，而 $\boldsymbol{x}^*$ 不满足。

下面介绍具体的 Gomory 割平面法。

割平面法于 1958 年由高莫瑞(R. E. Gomory)首先提出，故又称 Gomory 割平面法。在割平面法中，每次增加的用于“切割”的线性约束称为割平面约束或 Gomory 约束。构造割平面约束的方法很多，但下面的方法是最常用的一种，它可以从相应线性规划的最终单纯形表中直接产生。

在松弛问题的最优单纯形表中，记 Q 为 m 个基变量的下标集合，K 为 $n-m$ 个非基变量的下标集合，则 m 个约束方程可表示为

$$x_i + \sum_{j\in K} \bar{a}_{ij} x_j = \bar{b}_i, \quad i \in Q \tag{5-3}$$

而对应的最优解 $\boldsymbol{X}^* = (x_1^*, x_2^*, \cdots, x_n^*)^{\mathrm{T}}$，其中

$$x_j^* = \begin{cases} \bar{b}_j, & j \in Q \\ 0, & j \in K \end{cases} \tag{5-4}$$

若各 $\bar{b}_j (j\in Q)$ 皆为整数，则 $\boldsymbol{X}^*$ 满足 x_j 取整数($j=1,2,\cdots,n$)，因而就是纯整数规划的最优解；若各 $\bar{b}_j (j\in Q)$ 不全为整数，则 $\boldsymbol{X}^*$ 不满足 x_j 取整数($j=1,2,\cdots,n$)，因而就不是纯整数规划的可行解，自然也不是原整数规划的最优解。

用割平面法(cutting plane approach)解整数规划时，若其松弛问题的最优解 $\boldsymbol{X}^*$ 不满足 x_j 取整数($j=1,2,\cdots,n$)，则从 $\boldsymbol{X}^*$ 的非整分量中选取一个，用以构造一个线性约束条件，将其加入原松弛问题中，形成一个新的线性规划，然后求解。若新的最优解满足整数要求，则它就是整数规划的最优解；否则，重复上述步骤，直到获得整数最优解为止。

为最终获得整数最优解，每次增加的线性约束条件应具备两个基本性质：①已获得的不符合整数要求的线性规划最优解不满足该线性约束条件，从而不可能在以后的解中再出现；②凡整数可行解均满足该线性约束条件，因而整数最优解始终被保留在每次形成的线性规划可行域中。

为此，若 $\bar{b}_{i_0} (i_0\in Q)$ 不是整数，在式(5-3)中对应的约束方程为

$$x_{i_0} + \sum_{j\in K} \bar{a}_{i_0,j} x_j = \bar{b}_{i_0} \tag{5-5}$$

其中，x_{i_0} 和 $x_j (j\in K)$ 按 x_j 取整数($j=1,2,\cdots,n$)应为整数；$\bar{b}_{i_0}$ 按假设不是整数；$\bar{a}_{i_0,j} (j\in K)$ 可能是整数，也可能不是整数。

分解 $\bar{a}_{i_0,j}$ 和 $\bar{b}_{i_0}$ 成两部分：一部分是不超过该数的最大整数，另一部分是余下的小数。即

$$\bar{a}_{i_0,j} = N_{i_0,j} + f_{i_0,j}, N_{i_0,j} \leqslant \bar{a}_{i_0,j} \text{ 且为整数}, 0 \leqslant f_{i_0,j} < 1 (j \in K) \tag{5-6}$$

$$\overline{b}_{i_0}=N_{i_0}+f_{i_0},N_{i_0}<\overline{b}_{i_0}\text{ 且为整数},0<f_{i_0}<1 \tag{5-7}$$

把式(5-6)和式(5-7)代入式(5-5),移项得

$$x_{i_0}+\sum_{j\in K}N_{i_0,j}x_j-N_{i_0}=f_{i_0}-\sum_{j\in K}f_{i_0,j}x_j \tag{5-8}$$

式(5-8)中,左边是一个整数,右边是一个小于1的数,因此有

$$f_{i_0}-\sum_{j\in K}f_{i_0,j}x_j\leqslant 0$$

即

$$\sum_{j\in K}(-f_{i_0,j})x_j\leqslant -f_{i_0} \tag{5-9}$$

现在考查线性约束条件(5-9)的性质。

一方面,由于式(5-9)中$j\in K$,所以,如将$\boldsymbol{X}^*$代入,各x_j作为非基变量皆为0,将有

$$0\leqslant -f_{i_0}$$

这和式(5-7)矛盾。由此可见,$\boldsymbol{X}^*$不满足式(5-9)。

另一方面,满足$\sum_{j=1}^{n}a_{ij}x_j=b_i(i=1,2,\cdots,m)$、$x_j\geqslant 0(j=1,2,\cdots,n)$和$x_j$取整数$(j=1,2,\cdots,n)$的任何一个整数可行解$\boldsymbol{X}$一定也满足式(5-3)。式(5-5)是式(5-3)中的一个表达式,当然也满足。因而$\boldsymbol{X}$必定满足式(5-8)和式(5-9)。由此可知,任何整数可行解一定能满足式(5-9)。

综上所述,线性约束条件(5-9)具备上述两个基本性质。将式(5-9)和$\max z=\sum_{j=1}^{n}c_jx_j$、$\sum_{j=1}^{n}a_{ij}x_j=b_i(i=1,2,\cdots,m)$、$x_j\geqslant 0(j=1,2,\cdots,n)$合并,构成一个新的线性规划。记$R$为原松弛问题可行域,$R'$为新的线性规划可行域。从几何意义上看,式(5-9)实际上对R做了一次"切割",在留下的R'中,保留了整数规划的所有整数可行解,但不符合整数要求的$\boldsymbol{X}^*$被"切割"掉了。随着"切割"过程不断继续,整数规划最优解最终有机会成为某个线性规划可行域的顶点,作为该线性规划的最优解而被解得。

经验表明,实际解题时若从最优单纯形表中选择具有最小(分)数部分的非整分量所在行构造割平面约束,往往可以提高"切割"效果,减少"切割"次数。

5.6 Gomory 割平面法的 MATLAB 实现

Gomory 割平面法的代码如下:

```
% Gomory 割平面法
function [intx,intf] = Gomory(A,c,b,base)

% 约束矩阵:A
```

```
%目标函数系数向量:c
%约束右端向量:b
%初始基向量:base
%目标函数取最小化时的自变量值:x
%目标函数的最小值:minf

sz = size(A);
nVia = sz(2);
n = sz(1);
xx = 1:nVia;

if length(base)~ = n
   disp('基变量的个数要与约束矩阵的行数相等!');
   mx = NaN;
   mf = NaN;
   return;
end

M = 0;
sigma = -[transpose(c) zeros(1,(nVia - length(c)))];
xb = b;
while 1
   [maxs,ind] = max(sigma);
   if maxs < = 0
      vr = find(c~ = 0,1,'last');
      for l = 1:vr
         ele = find(base == l,1);
         if(isempty(ele))
            mx(l) = 0;
         else
            mx(l) = xb(ele);
         end
      end
      if max(abs(round(mx) - mx))< 1.0e - 7
         intx = mx;
         intf = mx * c;
         return;
      else
         sz = size(A);
         sr = sz(1);
         sc = sz(2);
         [max_x,index_x] = max(abs(round(mx) - mx));
         [isB,num] = find(index_x == base);
         fi = xb(num) - floor(xb(num));
         for i = 1:(index_x - 1)
```

```
    Atmp(1,i) = A(num,i) - floor(A(num,i));
end
for i = (index_x + 1):sc
    Atmp(1,i) = A(num,i) - floor(A(num,i));
end

% 构建对单纯形法的初始表格
Atmp(1,index_x) = 0;
A = [A zeros(sr,1); - Atmp(1,:) 1];
xb = [xb; - fi];
base = [base sc + 1];
sigma = [sigma 0];

% 对偶单纯形法迭代过程
while 1
    if(xb)> = 0
        if max(abs(round(xb) - xb))< 1.0e - 7
            % 用对偶单纯形法求得了整数解
            vr = find(c~ = 0 ,1,'last');
            for l = 1:vr
                ele  = find (base == l,1);
                if(isempty(ele))
                    mx_1(l) = 0;
                else
                    mx_1(l) = xb(ele);
                end
            end
            intx = mx_1;
            intf = mx_1 * c;
            return;
        else
            sz = size(A);
            sr = sz(1);
            sc = sz(2);
            [max_x,index_x] = max(abs(round(mx_1) - mx_1));
            [isB,num] = find(index_x == base);
            fi = xb(num) - floor(xb (num));
            for i = 1:(index_x - 1)
                Atmp(1,i) = a(num,i) - floor(A(num,i));

            end
            for i = (index_x + 1):sc
                Atmp(1,i) = a(num,i) - floor(a(num,i));
            end

            % 下一次对偶单纯形法迭代的初始表格
```

```
                Atmp(1,index_x) = 0;
        A = [A zeros(sr,1); - Atmp(1,:) 1];
        xb = [xb; - fi];
        base = [base sc + 1];
        sigma = [sigma 0];
        continue;
            end


        % 对偶单纯形法的换基变量过程
        else
            minb_1 = inf;
            chagB_1 = inf;
            sA = size(A);
            [br,idb] = min(xb);
            for j = 1:sA(2)
              if A(idb,j)< 0
                 bm = sigma(j)/A(idb,j);
                 if bm < minb_1
                    minb_1 = bm;
                    chagB_1 = j;
                 end
              end
            end
            sigma = sigma - A(idb,:) * minb_1;
            xb(idb) = xb(idb)/A(idb,chagB_1);
            A(idb,:) = A(idb,:)/A(idb,chagB_1);
            for i = 1:sA(1)
              if i ~ = idb
                 xb(i) = xb(i) - A(i,chagB_1) * xb(idb);
                 A(i,:) = A(i,:) - A(i,chagB_1) * A(idb,:);
              end
            end
            base = chagB_1;
        end
    end
end
else
    minb = inf;
    chagB = inf;
    for j = 1:n
        if A(j,ind)> 0
            bz = xb(j)/A(j,ind);
            if bz < minb
                minb = bz;
```

```
                    chagB = j;
                end
            end
        end
        sigma = sigma - A(chagB,:) * maxs/A(chagB,ind);
        xb(chagB) = xb(chagB)/A(chagB,ind);
        A(chagB,:) = A(chagB,:)/A(chagB,ind);
        for i = 1:n
            if i~ = chagB
                xb(i) = xb(i) - A(i,ind) * xb(chagB);
                A(i,:) = A(i,:) - A(i,ind) * A(chagB,:);
            end
        end
        base(chagB) = ind;
    end

    M = M + 1;
    if (M == 1000000)
        disp('找不到最优解!');
        mx = NaN;
        minf = NaN;
        return;
    end
    end
```

下面给出 MATLAB 解法的例子。

例 5-8

$$\min f(x) = x_1 - x_2$$

使得

$$\begin{cases} -x_1 + 2x_2 \leqslant 2 \\ 2x_1 + x_2 \leqslant 4 \\ x_1, x_2 \geqslant 0, \text{且 } x_1, x_2 \text{ 为整数} \end{cases}$$

解：首先引入人工变量 x_3、x_4 将约束条件转换为等式形式，即

$$\min f(x) = x_1 - x_2$$

使得

$$\begin{cases} -x_1 + 2x_2 + x_3 = 2 \\ 2x_1 + x_2 + x_4 = 4 \\ x_1, x_2, x_3, x_4 \geqslant 0, \text{且 } x_1, x_2 \text{ 为整数} \end{cases}$$

在 MATLAB 命令行输入下列命令：

```
A = [ - 1   2   1   0 ;2   1   0   1];
c = [1; - 1];
```

```
b = [2;4];
[intx,intf] = Gomory(A,c,b,[3 4])
```

结果如下：

```
intx =
     0     1
intf =
    -1
```

例 5-9 用割平面法求解纯整数规划

$$\max z = 3x_1 - x_2$$

使得

$$\begin{cases} 3x_1 - 2x_2 \leqslant 3 \\ 5x_1 + 4x_2 \geqslant 10 \\ 2x_1 + x_2 \leqslant 5 \\ x_1, x_2 \geqslant 0 \\ x_1, x_2 \text{ 为整数} \end{cases}$$

解：引入松弛变量 x_3、x_4、x_5，将问题转换为标准形式，用单纯形法解其松弛问题，得最优单纯形表，见表 5-4。

表 5-4 最优单纯形表

c_j			3	−1	0	0	0
$\boldsymbol{C}_B$	$\boldsymbol{X}_B$	$\boldsymbol{b}$	x_1	x_2	x_3	x_4	x_5
3	x_1	13/7	1	0	1/7	0	2/7
−1	x_2	9/7	0	1	−2/7	0	3/7
0	x_4	31/7	0	0	−3/7	1	22/7
$c_j - z_j$			0	0	−5/7	0	−3/7

由于 $\boldsymbol{b}$ 列各分数中 $x_1 = 13/7$ 有最大小数部分 6/7，故从表 5-4 中第一行产生割平面约束。按照式(5-9)，割平面约束为

$$-\frac{1}{7}x_3 - \frac{2}{7}x_5 \leqslant -\frac{6}{7} \tag{5-10a}$$

引入松弛变量 x_6，得割平面方程

$$-\frac{1}{7}x_3 - \frac{2}{7}x_5 + x_6 = -\frac{6}{7} \tag{5-10b}$$

将式(5-10b)并入表 5-4，然后用对偶单纯形法求解，得表 5-5。

表 5-5　表 5-4 中第一行产生割平面约束后的新表

c_j			3	−1	0	0	0	0
$\boldsymbol{C}_B$	$\boldsymbol{X}_B$	$\boldsymbol{b}$	x_1	x_2	x_3	x_4	x_5	x_6
3	x_1	13/7	1	0	1/7	0	2/7	0
−1	x_2	9/7	0	1	−2/7	0	3/7	0
0	x_4	31/7	0	0	−3/7	1	22/7	0
0	x_6	−6/7	0	0	−1/7	0	[−2/7]	1
c_j-z_j			0	0	−5/7	0	−3/7	0
⋮								
3	x_1	1	1	0	0	0	0	1
−1	x_2	5/4	0	1	0	−1/4	0	−5/4
0	x_3	5/2	0	0	1	−1/2	0	−11/2
0	x_5	7/4	0	0	0	1/4	1	−3/4
c_j-z_j			0	0	0	−1/4	0	−17/4

类似地，从表 5-5 中最后一个单纯形表的第四行产生割平面约束

$$-\frac{1}{4}x_4-\frac{1}{4}x_6\leqslant-\frac{3}{4} \tag{5-10c}$$

引入松弛变量 x_7，得割平面方程

$$-\frac{1}{4}x_4-\frac{1}{4}x_6+x_7=-\frac{3}{4} \tag{5-10d}$$

将式(5-10d)并入表 5-5 中最后一个单纯形表，然后用对偶单纯形法解之，得表 5-6。

表 5-6　表 5-5 中第四行产生割平面约束后的新表

c_j			3	−1	0	0	0	0	0
$\boldsymbol{C}_B$	$\boldsymbol{X}_B$	$\boldsymbol{b}$	x_1	x_2	x_3	x_4	x_5	x_6	x_7
3	x_1	1	1	0	0	0	0	1	0
−1	x_2	2	0	1	0	0	0	−1	−1
0	x_3	4	0	0	1	0	0	−5	−2
0	x_5	1	0	0	0	0	1	−1	1
0	x_4	3	0	0	0	1	0	1	−4
c_j-z_j		−1	0	0	0	0	0	−4	−1

表 5-6 给出的最优解 $(x_1,x_2,x_3,x_4,x_5,x_6,x_7)^{\mathrm{T}}=(1,2,4,3,1,0,0)^{\mathrm{T}}$ 已满足整数要求，故原整数规划问题的最优解为

$$x_1=1,\quad x_2=2,\quad \max z=1$$

如果在先后构造的割平面约束式(5-10a)和式(5-10c)中，将各变量用原整数规划的决

策变量 x_1 和 x_2 表示，则式(5-10a)和式(5-10c)成为 $x_1 \leqslant 1$ 和 $x_1 + x_2 \geqslant 3$。在这种形式下，“切割”的几何意义是显而易见的，如图 5-4 所示。

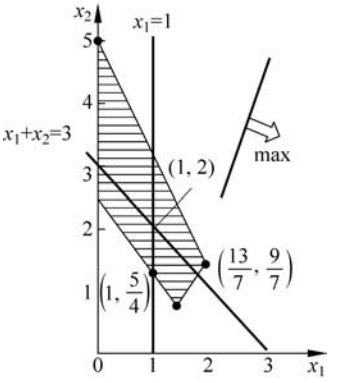

图 5-4 “切割”的几何意义示意图

用 MATLAB 求解。先将限制条件转换为等式

$$\max z = 3x_1 - x_2$$

使得

$$\begin{cases} 3x_1 - 2x_2 + x_3 = 3 \\ 5x_1 + 4x_2 + x_4 = 10 \\ 2x_1 + x_2 + x_5 = 5 \\ x_1, x_2, x_3, x_4, x_5 \geqslant 0 \end{cases}$$

再如例 5-8 中，运用 Gomory 函数即可得到结果。

5.7 分支定界法

分支定界法(branch and bound method)是一种隐枚举法(implicit enumeration)或部分枚举法，它不是一种有效算法，是在枚举法基础上的改进。分支定界法的关键是分支和定界。

若整数规划的松弛问题的最优解不符合整数要求，假设 $x_i = \bar{b}_i$ 不符合整数要求，$[\bar{b}_i]$ 是不超过 $\bar{b}_i$ 的最大整数，则构造两个约束条件：$x_i \leqslant [\bar{b}_i]$ 和 $x_i \geqslant [\bar{b}_i] + 1$。分别将其并入上述松弛问题中，形成两个分支，即两个后继问题。两个后继问题的可行域中包含原整数规划问题的所有可行解。而在原松弛问题可行域中，满足 $[\bar{b}_i] < x_i < [\bar{b}_i] + 1$ 的一部分区域在以后的求解过程中被遗弃了，但它不包含整数规划的任何可行解。根据需要，各后继问题可以类似地产生自己的分支，即自己的后继问题。如此不断继续，直到获得整数规划的最优解。这就是所谓的“分支”。

所谓“定界”，是在分支过程中，若某个后继问题恰巧获得整数规划问题的一个可行解，则其目标函数值就是一个“界限”，可作为衡量处理其他分支的一个依据。因为整数规划问题的可行解集是它的松弛问题可行解集的一个子集，前者最优解的目标函数值不会优于后者最优解的目标函数值。所以，对于那些相应松弛问题最优解的目标函数值劣于上述“界限”值的后继问题，就可以剔除不再考虑了。当然，如果在以后的分支过程中出现了更好的“界限”，则以它来取代原来的界限，这样可以提高求解的效率。

“分支”为整数规划最优解的出现缩减了搜索范围，而“定界”则可以提高搜索的效率。经验表明，在可能的情况下，根据对实际问题的了解，事先选择一个合理的“界限”，可以提高分支定界法的搜索效率。

下面通过例子来阐明分支定界法的基本思想和一般步骤。

例 5-10 求解

$$\max z = x_1 + x_2$$

使得

$$\begin{cases} x_1+\dfrac{9}{14}x_2 \leqslant \dfrac{51}{14} \\ -2x_1+x_2 \leqslant \dfrac{1}{3} \\ x_1,x_2 \geqslant 0 \\ x_1,x_2 \text{ 取整数} \end{cases}$$

解：记整数规划问题为(IP)，它的松弛问题为(LP)。图 5-5 中 S 为(LP)的可行域，黑点表示(IP)的可行解。用单纯形法解(LP)，最优解为 $x_1=3/2, x_2=10/3$，即点 A，$\max z=29/6$。

(LP)的最优解不符合整数要求，可任选一个变量，如选择 $x_1=3/2$ 进行分支。由于最接近 3/2 的整数是 1 和 2，因而可以构造两个约束条件

$$x_1 \geqslant 2 \tag{5-11a}$$

和

$$x_1 \leqslant 1 \tag{5-11b}$$

将式(5-11a)和式(5-11b)分别并入例 5-10 的松弛问题(LP)中，形成两个分支，即后继问题(LP_1)和(LP_2)，分别由(LP)及式(5-11a)和(LP)及式(5-11b)组成。图 5-6 中 S_1 和 S_2 分别为(LP_1)和(LP_2)的可行域。不连通的域 $S_1 \cup S_2$ 中包含了(IP)的所有可行解，S 中被舍去的一部分 $S \backslash S_1 \cup S_2$ 中不包含(IP)的任何可行解。

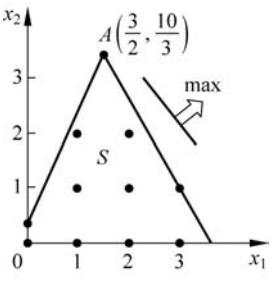

图 5-5 (LP)的可行域和(IP)的可行解

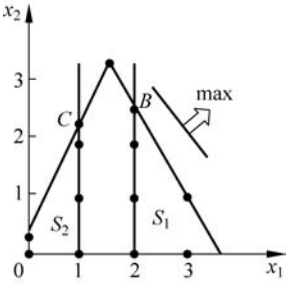

图 5-6 (LP_1)和(LP_2)的可行域

解(LP_1)，最优解为 $x_1=2, x_2=23/9$，即点 B，$\max z=41/9$。点 B 仍不符合整数要求，再解(LP_2)。(LP_2)最优解为 $x_1=1, x_2=7/3$，即点 C，$\max z=10/3$。点 ***C*** 也不符合整数要求，必须继续分支。

由于 $41/9>10/3$，所以优先选择 S_1 分支。因 B 点 $x_1=2$，而 $x_2=23/9$ 不符合整数要求，故可以构造两个约束条件

$$x_2 \geqslant 3 \tag{5-11c}$$

和

$$x_2 \leqslant 2 \tag{5-11d}$$

将式(5-11c)和式(5-11d)分别并入(LP_1),形成两个新分支,即(LP_1)的后继问题(LP_{11})和(LP_{12}),分别由(LP_1)及式(5-11c)和(LP_1)及式(5-11d)组成。图 5-7 中 S_{12} 为(LP_{12})的可行域。由于式(5-11c)和(LP_1)不相容,故(LP_{11})无可行解,也就是说,(LP_{11})的可行域 S_{11} 为空集,所以只需考虑后继问题(LP_{12})。

在 S_{12} 上解(LP_{12}),最优解为 $x_1=33/14, x_2=2$,即图 5-7 中点 D,$\max z=61/14$。

对于原整数规划(IP)来说,至此还剩两个分支:后继问题(LP_2)和(LP_{12})。因为(LP_{12})的最优解目标函数值比(LP_2)大,所以优先考虑对(LP_{12})进行分支。

两个新约束条件为

$$x_1 \geqslant 3 \tag{5-11e}$$

和

$$x_1 \leqslant 2 \tag{5-11f}$$

类似地,形成(LP_{12})的两个后继问题(LP_{121})和(LP_{122})。图 5-8 中 S_{121} 和 S_{122} 分别为它们的可行域,其中 S_{122} 是一条直线段。

(LP_{121})的最优解是 $x_1=3, x_2=1$,即图 5-8 中的点 E,$\max z=4$;(LP_{122})的最优解是 $x_1=2, x_2=2$,即图 5-8 中的点 F,$\max z=4$。这两个解都是(IP)的可行解,且目标函数值相等。至此,可以肯定两点:第一,在 S_{121} 和 S_{122} 中不可能存在比 E 点和 F 点更好的(IP)的可行解,因此不必再在它们中继续搜索;第二,既然点 E 和点 F 都是(IP)的可行解,它们的目标函数值 $z=4$ 就可看作(IP)最优解的目标函数值的一个界限(对于最大化问题,是下界;对于最小化问题,是上界)。

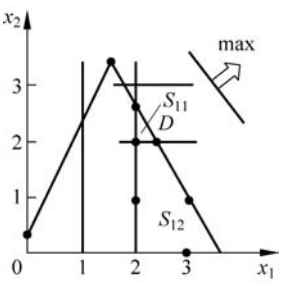

图 5-7 (LP_{12})的可行域

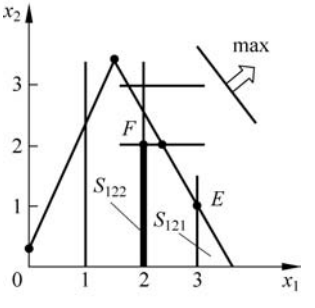

图 5-8 (LP_{121})和(LP_{122})的可行域

现在,尚未检查的后继问题只有(LP_2)了。但(LP_2)的最优解的目标函数值是 10/3,比界限 4 小。因此,S_2 中不存在目标函数值比 4 大的(IP)的可行解,也就是说,不必再对(LP_2)进行分支搜索了。

综上所述,我们已经求得了整数规划(IP)的两个最优解。它们分别是 $x_1=3, x_2=1$ 和 $x_1=2, x_2=2$,$\max z=4$。

上述分支定界法求解的过程可用图 5-9 表示。

分支定界法解整数规划的一般步骤如下。

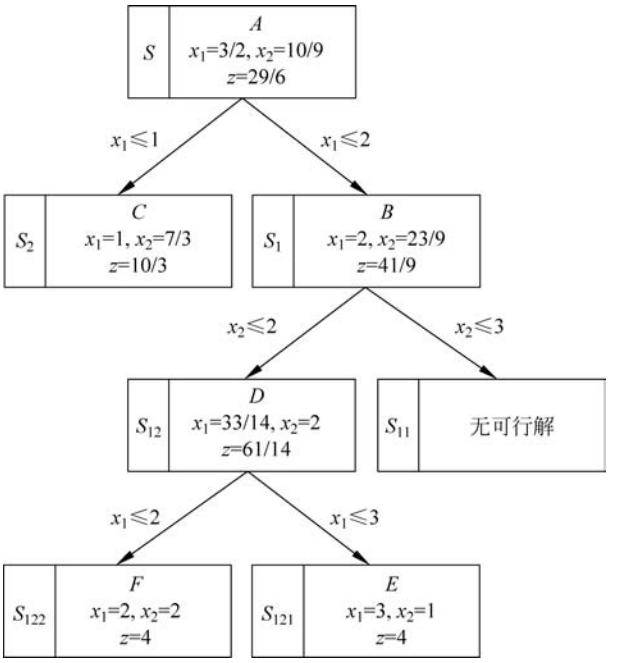

图 5-9　分支定界法求解的过程

步骤 1　称整数规划问题为问题 A，它的松弛问题为问题 B，以 z_b 表示问题 A 的目标函数的初始界（如已知问题 A 的一个可行解，则可取它的目标函数值为 z_b）。对最大化问题 A，z_b 为下界；对最小化问题 A，z_b 为上界。解问题 B。转步骤 2。

步骤 2　如问题 B 无可行解，则问题 A 也无可行解；如问题 B 的最优解符合问题 A 的整数要求，则它就是问题 A 的最优解。对于这两种情况，求解过程到此结束。如问题 B 的最优解存在，但不符合问题 A 的整数要求，则转步骤 3。

步骤 3　对问题 B，任选一个不符合整数要求的变量进行分支。设选择 $x_j=\bar{b}_j$，且设 $[\bar{b}_j]$ 为不超过 $\bar{b}_j$ 的最大整数。对问题 B 分别增加下面两个约束条件中的一个：

$$x_j \leqslant [\bar{b}_j] \text{ 和 } x_j \geqslant [\bar{b}_j]+1$$

从而形成两个后继问题。解这两个后继问题。转步骤 4。

步骤 4　考查所有后继问题，如其中有某几个存在最优解，且其最优解满足问题 A 的整数要求，则以它们中最优的目标函数值和界 z_b 作比较。若比界 z_b 更优，则以其取代原来的界 z_b，并称相应的后继问题为问题 C。否则，原来的界 z_b 不变。转步骤 5。

步骤 5　不属于 C 的后继问题中，称存在最优解且其目标函数值比界 z_b 更优的后继问题为待检查的后继问题。

若不存在待检查的后继问题，当问题 C 存在时，问题 C 的最优解就是问题 A 的最优解；当问题 C 不存在时，和界 z_b 对应的可行解就是问题 A 的最优解。z_b 即为问题 A 的最

优解的目标函数值，求解到此结束。

若存在待检查的后继问题，则选择其中目标函数值最优的一个后继问题，改称其为问题 B。回到步骤 3。

分支定界法是求解整数规划的较好方法，很多求解整数规划的计算机软件是根据分支定界法原理编写的，同时这种方法也适用于求解混合整数规划问题，在实际中有着广泛应用。

5.8 分支定界法的 MATLAB 实现

根据分支定界算法的步骤，可以写出其具体实现代码：

```
% By Sherif A. Tawfik, Faculty of Engineering, Cairo University
% [x,val,status] = IP1(f,A,b,Aeq,beq,lb,ub,M,e)

% this function solves the following mixed - integer linear programming problem
%  min f * x
%  subject to
%       A * x <= b
%       Aeq * x = beq
%       lb <= x <= ub
%       M is a vector of indeces for the variables that are constrained to be integers
%       e is the integarilty tolerance

% the return variables are :
% x : the solution
% val: value of the objective function at the optimal solution
% status = 1 if successful
%      = 0 if maximum number of iterations reached in he linprog function
%      = -1 if there is no solution

% Example:
%      maximize 17 x1 + 12 x2
%      subject to
%              10 x1 + 7 x2 <= 40
%                x1 +    x2 <= 5
%                x1, x2 >= 0 and are integers
% f = [-17, -12]; % take the negative for maximization problems
% A = [10  7; 1 1];
% B = [40; 5];
% lb = [0 0];
% ub = [inf inf];
% M = [1,2];
% e = 2^-24;
```

```
% [x v s] = IP(f,A,B,[],[],lb,ub,M,e)


function [x,val,status] = intprog(f,A,b,Aeq,beq,lb,ub,M,e)
options = optimset('display','off');
bound = inf; % the initial bound is set to + ve infinity
[x0,val0] = linprog(f,A,b,Aeq,beq,lb,ub,[],options);
[x,val,status,b] = rec(f,A,b,Aeq,beq,lb,ub,x0,val0,M,e,bound); % a recursive function that
processes the BB tree

function [xx,val,status,bb] = rec(f,A,b,Aeq,beq,lb,ub,x,v,M,e,bound)
options = optimset('display','off');
% x is an initial solution and v is the corressponding objective function value
% solve the corresponding LP model with the integarily constraints removed

[x0,val0,status0] = linprog(f,A,b,Aeq,beq,lb,ub,[],options);

% if the solution is not feasible or the value of the objective function is
% higher than the current bound return with the input intial solution

if status0 < = 0 | val0 > bound
   xx = x; val = v; status = status0; bb = bound;
   return;
end


% if the integer - constraint variables turned to be integers within the
% input tolerance return

ind = find( abs(x0(M) - round(x0(M)))> e );
if isempty(ind)
   status = 1;
   if val0 < bound     % this solution is better than the current solution hence replace
      x0(M) = round(x0(M));
      xx = x0;
      val = val0;
      bb = val0;
   else
      xx = x;   % return the input solution
      val = v;
      bb = bound;
   end
   return
end

% if we come here this means that the solution of the LP relaxation is
```

```
% feasible and gives a less value than the current bound but some of the
% integer - constraint variables are not integers.
% Therefore we pick the first one that is not integer and form two LP problems
% and solve them recursively by calling the same function (branching)


% first LP problem with the added constraint that Xi <= floor(Xi) , i = ind(1)

br_var = M(ind(1));
br_value = x(br_var);
if isempty(A)
    [r c] = size(Aeq);
else
    [r c] = size(A);
end
A1 = [A ; zeros(1,c)];
A1(end,br_var) = 1;
b1 = [b;floor(br_value)];

% second LP problem with the added constraint that Xi >= ceil(Xi) , i = ind(1)

A2 = [A ;zeros(1,c)];
A2(end,br_var) = -1;
b2 = [b; -ceil(br_value)];


% solve the first LP problem

[x1,val1,status1,bound1] = rec(f,A1,b1,Aeq,beq,lb,ub,x0,val0,M,e,bound);
status = status1;
if status1 > 0 & bound1 < bound % if the solution was successfull and gives
a better bound
   xx = x1;
   val = val1;
   bound = bound1;
   bb = bound1;
else
   xx = x0;
   val = val0;
   bb = bound;
end

% solve the second LP problem

[x2,val2,status2,bound2] = rec(f,A2,b2,Aeq,beq,lb,ub,x0,val0,M,e,bound);

if status2 > 0 & bound2 < bound % if the solution was successfull and gives
```

```
a better bound.
    status = status2;
    xx = x2;
    val = val2;
    bb = bound2;
end
```

例 5-11

$$\min z = -5x_1 - x_2$$

使得

$$\begin{cases} 3x_1 + x_2 \geqslant 9 \\ x_1 + x_2 \geqslant 5 \\ x_1 + 8x_2 \geqslant 8 \\ x_1, x_2 \geqslant 0 \text{ 且为整数} \end{cases}$$

解：在 MATLAB 输入命令

```
f = [ - 5, - 1];
A = [3,1;1,1;1,8];
b = [9;5;8];
lb = [0 0];
ub = [inf, inf];
M = [1,2];
e = 2^ - 24;
[x,v,s] = intprog(f,A,b,[],[],lb,ub,M,e)
```

得到

```
x =
     3
     0
v =
   - 15.0000
s =
     1
```

故最小值为 15，此时 $x_1 = 3, x_2 = 0$。

例 5-12

$$\max z = x_1 + x_2$$

使得

$$\begin{cases} x_1 + \dfrac{9}{14}x_2 \leqslant \dfrac{51}{14} \\ -2x_1 + x_2 \leqslant \dfrac{1}{3} \\ x_1, x_2 \geqslant 0 \\ x_1, x_2 \text{ 取整数} \end{cases}$$

解：当求最大值时，先添加负号，相当于求

$$\min z = -x_1 - x_2$$

这样即可化为求解的标准型。

在 MATLAB 中输入下列命令：

```
f = [ - 1, - 1];
A = [1,9/14; - 2,1];
b = [51/14;1/3];
lb = [0 0];
ub = [inf, inf];
M = [1,2];
e = 2^ - 24;
[x,v,s] = intprog(f,A,b,[],[],lb,ub,M,e)
```

结果为

```
x =
     3
     1
v =
    - 4.0000
s =
     1
```

故最大值为 4。

5.9 整数规划的解法

0-1 型整数规划是一种特殊的整数规划，若含有 n 个变量，则可以产生 2^n 个可能的变量组合。当 n 较大时，采用完全枚举法解题几乎是不可能的。已有的求解 0-1 型整数规划的方法一般都属于隐枚举法。

在 2^n 个可能的变量组合中，往往只有一部分是可行解。只要发现某个变量组合不满足其中一个约束条件时，就不必再去检验其他约束条件是否可行。对于可行解，其目标函数值也有优劣之分。若已发现一个可行解，则根据它的目标函数值可以产生一个过滤条件

(filtering constraint),即对于目标函数值比它差的变量组合就不必再去检验它的可行性。在以后的求解过程中,每次发现比原来更好的可行解,即以此替换原来的过滤条件。上述做法都可以减少运算次数,使最优解能较快地被发现。

例 5-13 求解 0-1 整数规划

$$\max z = 3x_1 - 2x_2 + 5x_3$$

使得

$$\begin{cases} x_1 + 2x_2 - x_3 \leqslant 2 \\ x_1 + 4x_2 + x_3 \leqslant 4 \\ x_1 + x_2 \leqslant 3 \\ 4x_2 + x_3 \leqslant 6 \\ x_1, x_2, x_3 = 0 \text{ 或 } 1 \end{cases} \tag{5-12}$$

解:求解过程可以用表表示(见表 5-7)。

表 5-7 例 5-13 的求解过程

x_1, x_2, x_3	z 值	约束条件 a b c d	过滤条件
(0,0,0)	0	√ √ √ √	$z \geqslant 0$
(0,0,1)	5	√ √ √ √	$z \geqslant 5$
(0,1,0)	−2		
(0,1,1)	3		
(1,0,0)	3		
(1,0,1)	8	√ √ √ √	$z \geqslant 8$
(1,1,0)	1		
(1,1,1)	6		

所以,最优解$(x_1, x_2, x_3)^{\mathrm{T}} = (1,0,1)^{\mathrm{T}}$,$\max z = 8$。

采用上述算法,实际只做了 20 次运算。

为了进一步减少运算量,常按目标函数中各变量系数的大小顺序重新排列各变量,以使最优解有可能较早出现。对于最大化问题,可按由小到大的顺序排列;对于最小化问题,则相反。为此例 5-13 可写成下列形式:

$$\max z = 5x_3 + 3x_1 - 2x_2$$

使得

$$\begin{cases} -x_3 + x_1 + 2x_2 \leqslant 2 \\ x_3 + x_1 + 4x_2 \leqslant 4 \\ x_1 + x_2 \leqslant 3 \\ x_3 + 4x_2 \leqslant 6 \\ x_3, x_1, x_2 = 0 \text{ 或 } 1 \end{cases} \tag{5-13}$$

求解时先令排在前面的变量取值为1,如本例中可取$(x_3,x_1,x_2)=(1,0,0)$,若不满足约束条件时,可调整取值为(0,1,0);若仍不满足约束条件,可退为取值(0,0,1)等,依此类推。据此改写后模型的求解过程可见表5-8。

表5-8 改写后模型的求解过程

(x_3,x_1,x_2)	z值	约束条件 a b c d	过滤条件
(0,0,0)	0	√ √ √ √	$z\geqslant 0$
(1,0,0)	5	√ √ √ √	$z\geqslant 5$
(1,1,0)	8	√ √ √ √	$z\geqslant 8$

从目标函数方程看到,z值已不可能再增大,$(x_3,x_1,x_2)=(1,1,0)$即为本例的最优解。

采取这样的形式用上述方法解此例,可以很大程度减少运算次数。一般问题的规模越大,这样做的好处就越明显。

5.10 0-1整数规划的MATLAB实现

对于混合整数规划问题,MATLAB Optimization Toolbox中给出了INTLINPROG函数来解决。可以看到,INTLINPROG可以解决以下(混合)整数线性规划问题。

```
function [x, fval, exitflag, output] = intlinprog(f, intcon, A, b, Aeq, beq, lb, ub, x0, options)
% INTLINPROG Mixed integer linear programming.
%
%    X = INTLINPROG(f,intcon,A,b) attempts to solve problems of the form
%
%            min f'*x     subject to:  A*x  <= b
%             x                       Aeq*x = beq
%                                     lb <= x <= ub
%                                     x(i) integer, where i is in the index
%                                     vector intcon (integer constraints)
```

函数INTLINPROG能够求

$$\min \boldsymbol{f}^{\mathrm{T}}x$$

满足条件

$$\begin{cases} x(\text{在 intcon 中的位置上的值})\text{是整数} \\ \boldsymbol{A}x \leqslant \boldsymbol{b} \\ \mathbf{Aeq} * x = \mathbf{beq} \\ \mathbf{lb} \leqslant x \leqslant \mathbf{ub} \end{cases}$$

在要求 0-1 整数规划问题时，可令 intcon 包含所有 x 的元素，并且 **lb** 和 **ub** 分别是全为 0、1 的向量即可。当然，该函数对于所有的（混合）整数线性规划都是适用的，应用十分广泛。

更多该函数的详细代码可以在 intlinprog.m 中查阅，不在此赘述。

下面是一个使用 INTLINPROG 解 0-1 整数规划的例子。

例 5-14

$$\min f(x)=x_1+2x_2+3x_3+x_4+x_5$$

使得

$$\begin{cases}2x_1+3x_2+5x_3+4x_4+7x_5\geqslant 8\\ x_1+2x_2+4x_3+2x_4+2x_5\geqslant 5\\ x_1,x_2,x_3,x_4,x_5=0\text{ 或 }1\end{cases}$$

在 MATLAB 中输入下列命令：

```
f = [1;2;3;1;1];
intcon = 1:5;
A = [ - 2, - 3, - 5, - 4, - 7; - 1, - 2, - 4, - 2, - 2];
b = [ - 8; - 5];
lb = [0,0,0,0,0];
ub = [1,1,1,1,1];
[x,fval] = intlinprog(f,intcon,A,b,[],[],lb,ub)
```

得到

```
x =
     1
     0
     0
     1
     1
fval =

     3
```

5.11 整数规划解决旅行商问题的 MATLAB 实例

这个例子将展示如何使用二进制整数规划来解决经典的旅行商问题。这个问题涉及通过一组城市（200 个城市）找到最短的封闭旅游路径。

其中包含要解决的问题如下：对于所有不同的城市站点，生成所有可能的行程；计算每次旅行的距离；最小化成本函数即每次旅行距离之和。

将问题公式化，设置二进制的决策变量，并与每个行程相关联：1 表示在行程中的路径，0 表示不在行程中的路径。为了确保行程经过每个站点，设置线性限制条件使得每个站点链接两条路径，即一条离开一条到达的路径。

(1) 生成停止条件。

将这组城市的连线粗略地看作多边形，在其中设置随机停止：

```
load('usborder.mat','x','y','xx','yy');
rng(3,'twister') % Makes a plot with stops in Maine & Florida, and is reproducible
nStops = 200; % You can use any number, but the problem size scales as N^2
stopsLon = zeros(nStops,1); % Allocate x-coordinates of nStops
stopsLat = stopsLon; % Allocate y-coordinates
n = 1;
while (n <= nStops)
   xp = rand*1.5;
   yp = rand;
   if inpolygon(xp,yp,x,y) % Test if inside the border
      stopsLon(n) = xp;
      stopsLat(n) = yp;
      n = n+1;
   end
end
```

(2) 计算两个站点之间的距离。

生成所有的路径：

```
idxs = nchoosek(1:nStops,2);
```

因为有 200 个站点，所以一共有 19900 条路径，即有 19900 个二进制变量(# variables = 200 choose 2)。

计算所有的旅行距离，假设地球是平的，以便使用毕达哥拉斯法则。

```
dist = hypot(stopsLat(idxs(:,1)) - stopsLat(idxs(:,2)), ...
           stopsLon(idxs(:,1)) - stopsLon(idxs(:,2)));
lendist = length(dist);
```

(3) 创建图形并绘制地图。

绘图表示问题(如图 5-10 所示)，其中点为站点，边为路径。

```
G = graph(idxs(:,1),idxs(:,2));
figure
hGraph = plot(G,'XData',stopsLon,'YData',stopsLat,'LineStyle','none','NodeLabel',{});
```

```
hold on
 % Draw the outside border
plot(x,y,'r-')
hold off
```

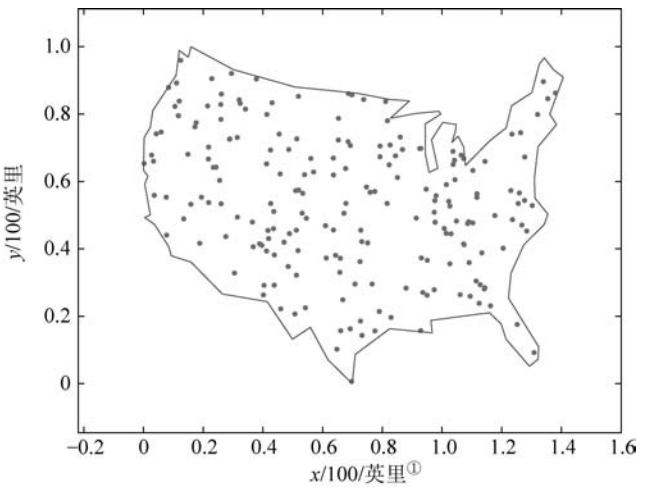

图 5-10　旅行商问题城市地图

(4) 创建约束条件。

创建线性约束条件,即每个站点都有两个关联的路径,因为每个站点必须有一个到达路径和一个离开站点的路径。

```
Aeq = spalloc(nStops,length(idxs),nStops*(nStops-1)); % Allocate a sparse matrix
for ii = 1:nStops
   whichIdxs = (idxs == ii); % Find the trips that include stop ii
   whichIdxs = sparse(sum(whichIdxs,2)); % Include trips where ii is at either end
   Aeq(ii,:) = whichIdxs'; % Include in the constraint matrix
end
beq = 2*ones(nStops,1);
```

增加二进制限制,将 intcon 参数设置为决策变量的个数,每个变量下界为 0,上界为 1。

```
intcon = 1:lendist;
lb = zeros(lendist,1);
ub = ones(lendist,1);
```

(5) 用 intlinprog 进行优化,将结果可视化。

创建一个用来求解路径的新图。为此,在某些值不是整数的情况下对解决方案进行舍

① 1 英里=1.609 千米。

入，并将结果值转换为逻辑值。

```
opts = optimoptions('intlinprog','Display','off');
[x_tsp,costopt,exitflag,output] = intlinprog(dist,intcon,[],[],Aeq,beq,lb,ub,opts);
x_tsp = logical(round(x_tsp));
Gsol = graph(idxs(x_tsp,1),idxs(x_tsp,2));
hold on
highlight(hGraph,Gsol,'LineStyle','- ')
title('Solution with Subtours')
```

如图 5-11 所示，该解决方案有几个子环路。目前为止指定的约束并不能阻止这些子环路的发生。为了防止任何可能的子环路发生，需要大量的不等式约束。

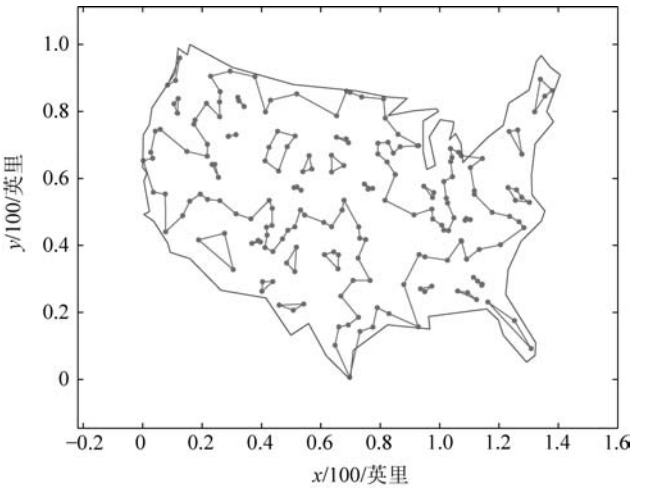

图 5-11　旅行商最佳旅游线路

(6) 避免子环路的限制条件。

由于无法添加所有的子环路的约束，这里采用迭代方法。检测当前解决方案中的子环路，然后添加不等式约束以防止这些特定的子环路发生。这样就可以在几个迭代中找到一个合适的路径。

消除带有不等式约束的子函数。一个这样做的例子是，如果在一个子环路中有 5 个点，那么有 5 条线连接这些点来创建子环路。通过实现一个不等式约束来消除子环路，即这 5 个点之间必须有小于或等于 4 条线。

更重要的是，找到这 5 个点之间的所有直线，并增加约束条件，使这些直线不超过 4 条。如果一个解决方案中存在 5 条或更多的行，那么该解决方案将具有一个子环路(具有 n 个节点和 n 条边的图总是包含一个循环)，如图 5-12 所示。

加入线性不等式约束以消除子环路，并重复调用，直到剩下一个环路。

```
tourIdxs = conncomp(Gsol);
numtours = max(tourIdxs); % number of subtours
fprintf('# of subtours: %d\n',numtours);

A = spalloc(0,lendist,0); % Allocate a sparse linear inequality constraint matrix
b = [];
while numtours > 1 % Repeat until there is just one subtour
    % Add the subtour constraints
   b = [b;zeros(numtours,1)]; % allocate b
   A = [A;spalloc(numtours,lendist,nStops)]; % A guess at how many nonzeros to allocate
   for ii = 1:numtours
      rowIdx = size(A,1) + 1; % Counter for indexing
      subTourIdx = find(tourIdxs == ii); % Extract the current subtour
%       The next lines find all of the variables associated with the
%       particular subtour, then add an inequality constraint to prohibit
%       that subtour and all subtours that use those stops.
      variations = nchoosek(1:length(subTourIdx),2);
      for jj = 1:length(variations)
         whichVar = (sum(idxs == subTourIdx(variations(jj,1)),2)) & ...
                    (sum(idxs == subTourIdx(variations(jj,2)),2));
         A(rowIdx,whichVar) = 1;
      end
      b(rowIdx) = length(subTourIdx) - 1; % One less trip than subtour stops
   end

    % Try to optimize again
   [x_tsp,costopt,exitflag,output] = intlinprog(dist,intcon,A,b,Aeq,beq,lb,ub,opts);
   x_tsp = logical(round(x_tsp));
   Gsol = graph(idxs(x_tsp,1),idxs(x_tsp,2));

    % Visualize result
   hGraph.LineStyle = 'none'; % Remove the previous highlighted path
   highlight(hGraph,Gsol,'LineStyle','-')
   drawnow

    % How many subtours this time?
   tourIdxs = conncomp(Gsol);
   numtours = max(tourIdxs); % number of subtours
   fprintf('# of subtours: %d\n',numtours)
end

title('Solution with Subtours Eliminated');
hold off
```

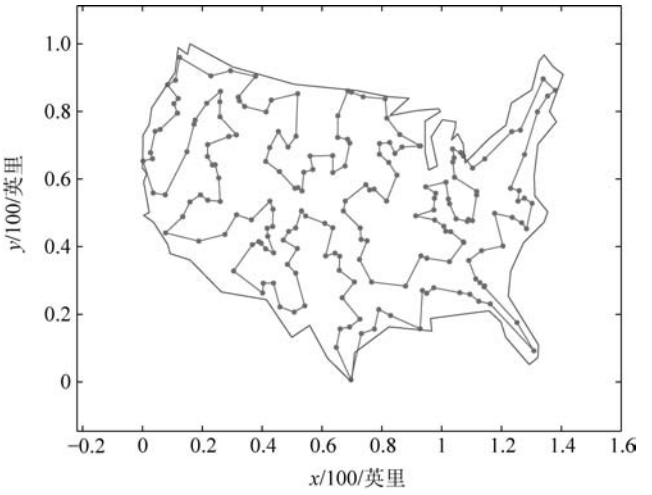

图 5-12　旅行商避免子环路后的最佳旅游线路