大学数学实验

第8讲

优化方法II　线性规划(续)

无约束优化

优化问题的一般形式

优化问题三要素：决策变量；目标函数；约束条件

$$
\begin{aligned}
\min \quad & f(x) \\
s.t. \quad & h_i(x) = 0,\ i = 1, \dots, m \\
& g_j(x) \le 0,\ j = 1, \dots, l \\
& x \in D \subseteq \Re^n
\end{aligned}
$$

目标函数

约束条件

决策变量

求解线性规划(LP)的基本原理

基本模型

$$\max(\text{or } \min)\ z = c^T x,\ x \in R^n$$

$$s.t. \qquad Ax = b,\ x \geq 0$$

$$c \in R^n,\ A \in R^{m\times n},\ b \in R^m$$

单纯形算法的基本思想

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

$$s.t.\begin{cases} -x_1 + x_2 + x_3 = 2 \\ x_1 - 2x_2 + x_4 = 2 \\ 3x_1 + 2x_2 + x_5 = 14 \end{cases}$$

$$x_j \ge 0; j = 1,2,3,4,5$$

用当前的非基变量表示目标函数，判断当前的基本可行解是否为最优；
如果不是最优，选定一个可使目标值下降的非基变量入基，其余非基变量不变，
使入基的非基变量增大，第1个变为0的基变量出基。

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

$$s.t.\begin{cases} -x_1 + x_2 + x_3 = 2 \\ x_1 - 2x_2 + x_4 = 2 \\ 3x_1 + 2x_2 + x_5 = 14 \end{cases}$$

$$x_j \ge 0; j = 1,2,3,4,5$$

单纯形算法的基本思想

$$\min z = -7x_2 + 3x_4 - 6$$

$$\begin{cases} 0 - x_2 + x_3 + x_4 = 4 \\ x_1 - 2x_2 + x_4 = 2 \\ 0 + x_2 - \dfrac{3}{8}x_4 + \dfrac{1}{8}x_5 = 1 \end{cases}$$

$$\min z = \frac{3}{8}x_4 + \frac{7}{8}x_5 - 13$$

$$\begin{cases} 0 + x_3 + \dfrac{5}{8}x_4 \quad \dfrac{1}{8}x_5 = 5 \\ x_1 + \dfrac{1}{4}x_4 \quad \dfrac{1}{4}x_5 = 4 \\ 0 + x_2 - \dfrac{3}{8}x_4 + \dfrac{1}{8}x_5 = 1 \end{cases}$$

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

$$s.t.\begin{cases} -x_1 + x_2 + x_3 = 2 \\ x_1 - 2x_2 + x_4 = 2 \\ 3x_1 + 2x_2 + x_5 = 14 \end{cases}$$

$$x_j \geq 0;\ j = 1,2,3,4,5$$

$$\min z = \frac{3}{8}x_4 + \frac{7}{8}x_5 - 13$$

$$\begin{cases} 0 + x_3 + \frac{5}{8}x_4 \quad \frac{1}{8}x_5 = 5 \\ x_1 + \frac{1}{4}x_4 \quad \frac{1}{4}x_5 = 4 \\ 0 + x_2 - \frac{3}{8}x_4 + \frac{1}{8}x_5 = 1 \end{cases}$$

单纯形算法的矩阵表示

$$\begin{aligned} \min \quad & z = C^T X \\ s.t. \quad & AX = b \\ & X \geq 0 \end{aligned}$$

Z	$-C^T$	0
	A	b

$$\begin{aligned} \min \quad & z = C_B^T X_B + C_N^T X_N \\ s.t. \quad & BX_B + NX_N = b \\ & X_B, X_N \geq 0 \end{aligned}$$

Z	$-C_B^T$	$-C_N^T$	0
	B	N	b

$$\begin{aligned} \min \quad & z = C_B^T X_B + C_N^T X_N \\ s.t. \quad & X_B = B^{-1}b - B^{-1}NX_N \\ & X_B, X_N \geq 0 \end{aligned}$$

	X_B	X_N	RHS
Z	$-C_B^T$	$-C_N^T$	0
X_B	I	$B^{-1}N$	$B^{-1}b$

标准单纯形表，检验数

$$\begin{aligned}
\min \quad & z = C_B^T B^{-1} b - (C_B^T B^{-1} N - C_N^T) X_N \\
s.t. \quad & X_B = B^{-1} b - B^{-1} N X_N \\
& X_B, X_N \ge 0
\end{aligned}$$

z	0^T	$C_B^T B^{-1} N - C_N^T$	$C_B^T B^{-1} b$
X_B	I	$B^{-1} N$	$B^{-1} b$

线性规划的敏感性分析

（当C, b有小的扰动时，对解的影响）

z	$-C_B^T$	$-C_N^T$	0
	B	N	b

	X_B	X_N	RHS
z	$-C_B^T$	$-C_N^T$	0
X_B	I	$B^{-1}N$	$B^{-1}b$

z	0^T	$C_B^T B^{-1} N - C_N^T$	$C_B^T B^{-1} b$
X_B	I	$B^{-1}N$	$B^{-1}b$

线性规划的对偶问题

对偶问题 (D)

$$\boxed{\begin{aligned}&\max\ \ b^T y\\ &s.t.\ \ A^T y \le c\end{aligned}}$$

$$\boxed{\begin{aligned}&\min\ \ z = c^T x\\ &s.t.\ \ Ax = b,\ x \ge 0\end{aligned}}$$

原(始)问题 (P)

定理：原问题和对偶问题互为对偶问题，即：
对偶问题的对偶问题就是原问题。

对偶定理: 如果x是原问题的可行解（原可行解），y 是对偶问题的可行解（对偶可行解），则 $b^T y \le c^T x$

- 若x和 y还满足$b^T y = c^T x$, 则分别是(P)和(D)的最优解
- 若原问题（P）无下界，则对偶问题（D）不可行
- 若对偶问题（D）无上界，则原问题（P）不可行

对偶问题的经济学解释：影子价格

1、定义 影子价格是最优配置下资源的理想价格

2、含义 由于$f^* = c^T x^* = b^T y^* = y_1^* b_1 + y_2^* b_2 + \cdots + y_m^* b_m$

考虑在最优解处,右端项b_i的微小变动对目标函数值的影响.

假设$b_1, b_2, \cdots, b_m$是变化的，则$\dfrac{\partial f^*}{\partial b_1} = y_1^*, \dfrac{\partial f^*}{\partial b_2} = y_2^*, \cdots, \dfrac{\partial f^*}{\partial b_m} = y_m^*$

y_i^*可以理解成当资源b_i变化1单位时，

极小化(L)问题的目标函数值的变化量

影子价格

z	0^T	$C_B^T B^{-1} N - C_N^T$	$C_B^T B^{-1} b$
X_B	I	$B^{-1}N$	$B^{-1}b$

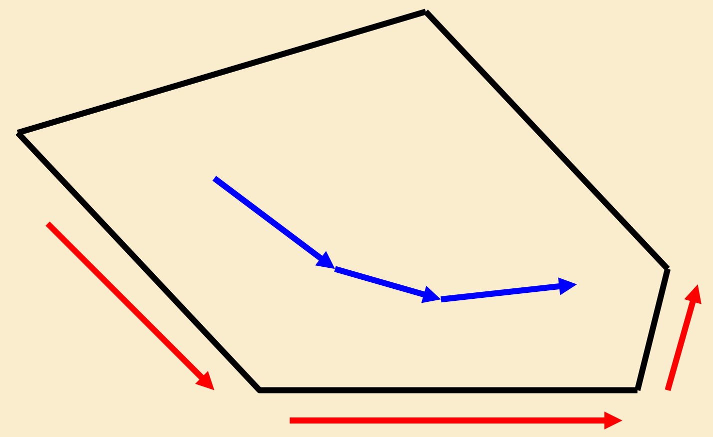
$$f^* = c^T x^* = b^T y^* = y_1^* b_1 + y_2^* b_2 + \cdots + y_m^* b_m$$

$B^{-1}b > 0$，对充分小的需求增量Δb，$B^{-1}(b + \Delta b) > 0$仍为最优解，此时相应的最优费用变化为$C_B^T B^{-1} \Delta b$对偶变量$y^* = (C_B^T B^{-1})^T$被称为边际价格或影子价格y^*_i可以看成最优解时，为了第i种需求提供一个单位需求的边际费用，即当达到最优时，为了满足附加的需求，必须向顾客索取的最小单位价格。

LP其他算法

内点算法(Interior point method)

- 1980年代人们提出的一类新的算法—内点算法
- 也是迭代法，但不再从可行域的一个顶点转换到另一个顶点，而是直接从可行域的内部逼近最优解。



MATLAB 求解 LP

$$\min\ z = c^T x$$

$$s.t.\ \ A_1 x \le b_1, A_2 x = b_2, v_1 \le x \le v_2$$

```
[x,fval,exitflag,output,lambda] =
linprog(c,A1,b1,A2,b2,v1,v2,x0,opt)
```

输入:

x0~初始解(缺省时为0)

opt ~ MATLAB控制参数

中间所缺参数项补[]

Exp07.m

输出:

lambda ~ Lagrange乘子，维数等于约束个数，非零分量对应于起作用约束

- lambda.ineqlin: 对应 $A_1x\le b_1$
- lambda. eqlin : 对应 $A_2x = b_2$
- lambda. lower : 对应 $v_1\le x$
- lambda. upper : 对应 $x \le v_2$

```
% 线性规划例1
% max   z=3*x1+x2
% s.t.-x1+x2    <=2      <== x1-x2>=-2
%          x1-2*x2 <=2
%       3*x1+2*x2 <=14
%          x1,x2>=0

c=-[3,1];A=[-1,1;1,-2;3,2];b=[2,2,14];
v1=[0 0];

[x,f,exitflag,output,lag]=linprog(c,A,b,[],[],v1)
zz=-f
z=-c*x
```

$$\min\ z = c^T x$$

$$s.t.\ \ A_1 x \le b_1, A_2 x = b_2, v_1 \le x \le v_2$$

exitflag

1	Function converged to a solution x.
0	Number of iterations exceeded options.MaxIterations.
-2	No feasible point was found.
-3	Problem is unbounded.
-4	NaN value was encountered during execution of the algorithm.
-5	Both primal and dual problems are infeasible.
-7	Search direction became too small. No further progress could be made.

lambda—Lagrange乘子

lower	Lower bounds corresponding to lb
upper	Upper bounds corresponding to ub
ineqlin	Linear inequalities corresponding to A and b
eqlin	Linear equalities corresponding to Aeq and beq

实例1: 食谱问题

$$\min \; z = c^T x$$
$$s.t. \quad Ax \geq b$$
$$x \geq 0$$

$\boldsymbol{x} = (x_1, x_2, x_3, x_4, x_5)^T$

$\boldsymbol{c} = (10, 15, 5, 60, 8)^T$

$\boldsymbol{b} = (50, 4000, 1000)^T$

$$A = \begin{pmatrix} 0.3 & 1.2 & 0.7 & 3.5 & 5.5 \\ 73 & 96 & 20253 & 890 & 279 \\ 9.6 & 7 & 19 & 57 & 22 \end{pmatrix}$$

x=(0， 0， 49.3827， 0, 2.8058)， 最优值z0 =269.36：

每天吃**49.3827**个胡萝卜和**2.8058**个鸡蛋，成本**269.36**美分

- 维生素**A**的需求增加**1**单位，是否改变食谱？成本增加多少？

  lag.ineqlin =(0.4714;0; 0.2458)　不改变；不增加

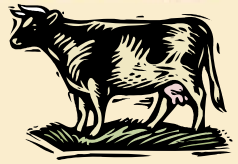
- 胡萝卜价格增**1**美分，是否改变食谱？成本增加多少？

  用**MATLAB**重新求解　　不改变；成本增加**49.38**

```
% 食谱问题
clc; clear all; c=[10,15,5,60,8];
%c=[10,15,6,60,8];  % 5==>6;
A=[ 0.3  1.2  0.7     3.5  5.5
     73  96  20253  890  279
    9.6   7   19     57  22];
b=[50,4000,1000]; v=[0 0 0 0 0];
[x,z0,ef,out,lag]=linprog(c,-A,-b,[],[],v)
lag.ineqlin

% 胡罗卜价格变化
c=[10,15,6,60,8];
A=[ 0.3  1.2  0.7     3.5  5.5
     73  96  20253  890  279
    9.6   7   19     57  22];
b=[50,4000,1000]; v=[0 0 0 0 0];
[x,z0,ef,out,lag]=linprog(c,-A,-b,[],[],v)
```

实例2: 奶制品生产销售计划



$$Max \quad z = 12x_1 + 8x_2 + 22x_3 + 16x_4 - 1.5x_5 - 1.5x_6$$

$$\frac{x_1 + x_5}{3} + \frac{x_2 + x_6}{4} \le 50 \quad \Rightarrow 4x_1 + 3x_2 + 4x_5 + 3x_6 \le 600$$

$$4(x_1 + x_5) + 2(x_2 + x_6) + 2x_5 + 2x_6 \le 480 \quad \Rightarrow 2x_1 + x_2 + 3x_5 + 2x_6 \le 240$$

$$x_1 + x_5 \le 100$$

$$x_3 - 0.8x_5 = 0$$

$$x_4 - 0.75x_6 = 0$$

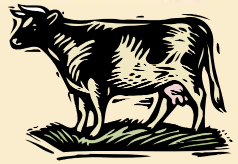
$$x_1, x_2, x_3, x_4, x_5, x_6 \ge 0$$

```
c=[12 8 22 16 -1.5 -1.5];
A1=[4 3 0 0 4 3;2 1 0 0 3 2;1 0 0 0 1 0];
b1=[600 240 100];
A2=[0 0 1 0 -0.8 0;0 0 0 1 0 -0.75];
b2=[0 0];
v1=[0 0 0 0 0 0];
[x,z0,ef,out,lag]=linprog(-c,A1,b1,A2,b2,v1)
```

x=(0,168,19.2,0,24,0) ; z = -z0 =1730.4; **Exp07.m**

lag.ineqlin =(1.58;3.26; 0.00) ; …

实例2: 奶制品生产销售计划



$$Max\quad z = 12x_1 + 8x_2 + 22x_3 + 16x_4 - 1.5x_5 - 1.5x_6$$

$$\frac{x_1 + x_5}{3} + \frac{x_2 + x_6}{4} \le 50 \quad \Rightarrow 4x_1 + 3x_2 + 4x_5 + 3x_6 \le 600$$

$$\Downarrow$$

601

$$4(x_1 + x_5) + 2(x_2 + x_6) + 2x_5 + 2x_6 \le 480$$

$$x_1 + x_5 \le 100$$

$$x_3 = 0.8x_5$$

$$x_4 = 0.75x_6$$

$$x_1, x_2, x_3, x_4, x_5, x_6 \ge 0$$

z1=1731.98

z1-z = 1731.98-1730.4=1.58

z1=lag.ineqlin(1)

“影子价格”

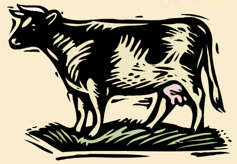
x=(0,168,19.2,0,24,0) ; z = -z0 =1730.4
lag.ineqlin =(1.58;3.26; 0.00) ; …

z1*12=1.58*12= 18.96>15

- **15元可增加1桶牛奶，应否投资？**

应该投资！

实例2: 奶制品生产销售计划



$$Max \quad z = 12x_1 + 8x_2 + 22x_3 + 16x_4 - 1.5x_5 - 1.5x_6$$

$$\frac{x_1 + x_5}{3} + \frac{x_2 + x_6}{4} \le 50 \quad \Rightarrow 4x_1 + 3x_2 \qquad + 4x_5 + 3x_6 \le 600$$

$$4(x_1 + x_5) + 2(x_2 + x_6) + 2x_5 + 2x_6 \le 480 \Rightarrow 2x_1 + x_2 \qquad + 3x_5 + 2x_6 \le 240$$

$$x_1 + x_5 \le 100$$

$$x_3 = 0.8x_5$$

$$x_4 = 0.75x_6$$

$$x_1, x_2, x_3, x_4, x_5, x_6 \ge 0$$

x=(0,168,19.2,0,24,0) ; z = -z0 =1730.4
lag.ineqlin =(1.58;3.26; 0.00) ; …

lag.ineqlin(2)=3.26，所以1小时劳动时间的影子价格应为3.26/2=1.63，即单位劳动时间增加的利润是1.63(元)

• 聘用临时工人增加劳动时间，工资最多每小时几元？

实例2: 奶制品生产销售计划

$$Max \quad z = 12x_1 + 8x_2 + 22x_3 + 16x_4 - 1.5x_5 - 1.5x_6$$

$$\frac{x_1 + x_5}{3} + \frac{x_2 + x_6}{4} \le 50$$

$$4(x_1 + x_5) + 2(x_2 + x_6) + 2x_5 + 2x_6 \le 480$$

$$x_1 + x_5 \le 100$$

$$x_3 = 0.8x_5$$

$$x_4 = 0.75x_6$$

$$x_1, x_2, x_3, x_4, x_5, x_6 \ge 0$$

若每公斤**B1**的获利下降**10%**，应将目标函数中$\mathbf{x_3}$的系数改为**19.8**，重新计算发现最优解和最优值均发生了变化

若**B2**的获利向上波动**10%**，原计划也不再是最优的

x=(0,168,19.2,0,24,0) ; z = -z0 =1730.4
lag.ineqlin =(1.58;3.26; 0.00) ;

MATLAB没有给出这种敏感性分析的结果

- **B$_1$**，**B$_2$**的获利经常有**10%**的波动，对计划有无影响？

布置实验内容

实验目的

•了解线性规划问题

•理解单纯形算法的基本思想

•掌握用MATLAB优化工具箱求解线性规划问题；

• 练习建立实际问题的线性规划模型。

实验内容

课程作业

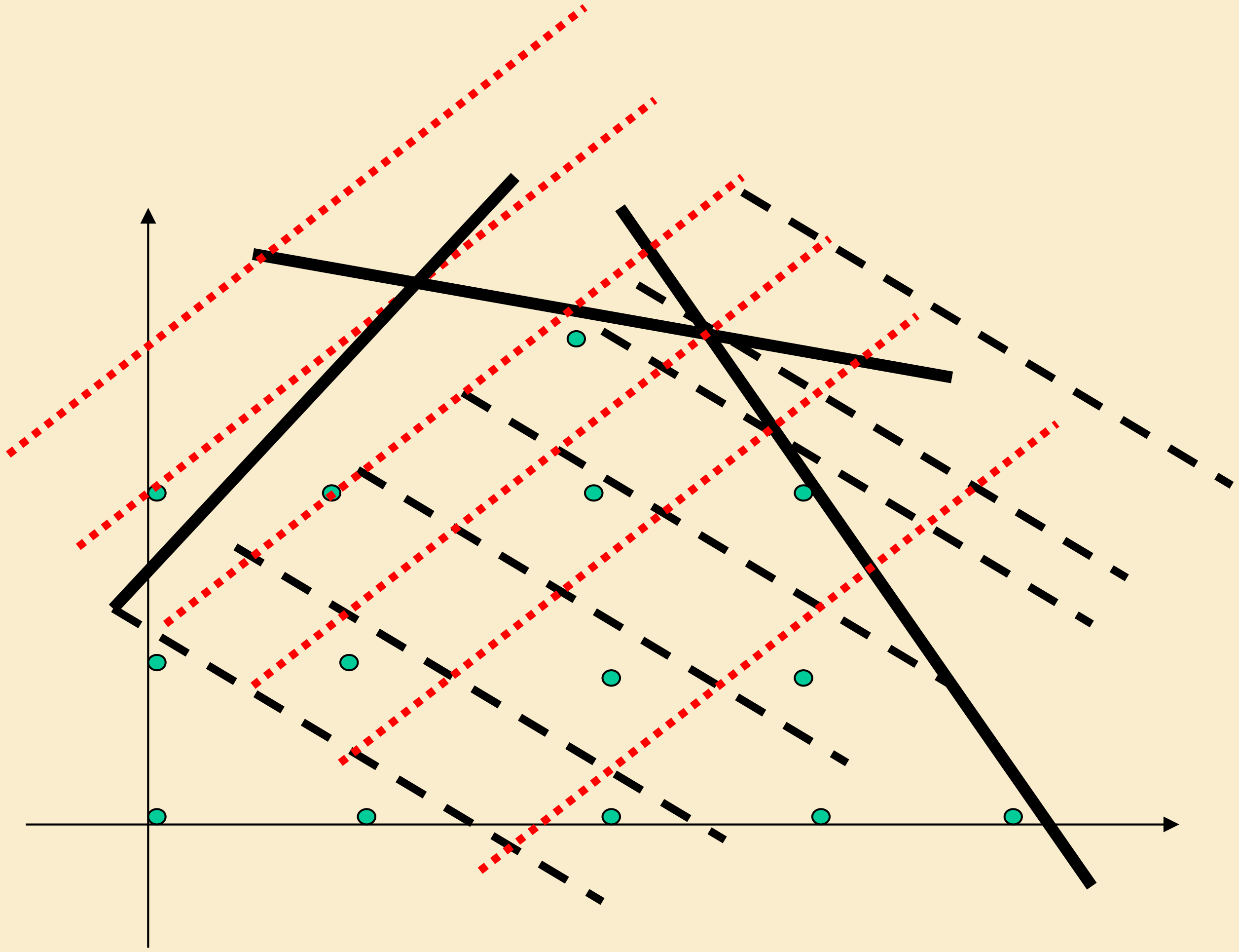
整数线性规划的分支定界算法

$$\min c^{\mathrm{T}} x$$

$$s.t.\begin{cases} Ax = b \\ x \geq 0, x\text{为整数} \end{cases}$$

1. **分枝定界**
2. **割平面法**

整数线性规划的困难性



与线性规划的关系

整数规划

$$\min c^{\mathrm{T}} x$$
$$s.t.\begin{cases} Ax = b \\ x \ge 0, x\text{为整数} \end{cases}$$

放松的线性规划

$$\min c^{\mathrm{T}} x$$
$$s.t.\begin{cases} Ax = b \\ x \ge 0 \end{cases}$$

可行解是放松问题的可行解

最优值大于等于放松问题的最优值

分枝定界算法思想

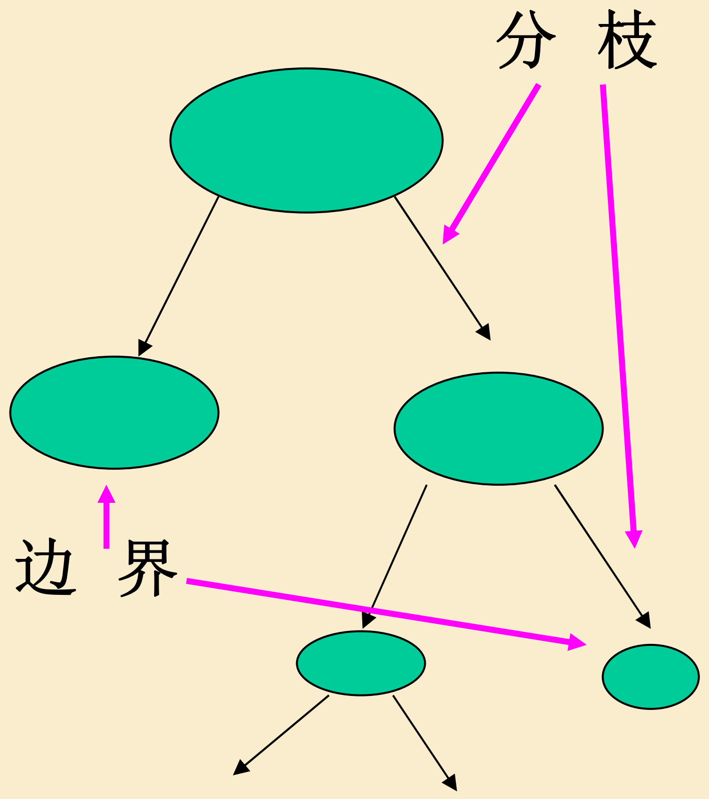
➢枚举

求解放松问题

最优值比界坏 ⇒ 舍　弃

最优解为整数
最优值比界好

最优解为非整
数最优值比界好 ⇒ 分　支



用分枝定界法求解整数规划问题（图解法演示）

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

$$\begin{cases} x_1 - x_2 \ge -2 \\ 5x_1 + 6x_2 \le 30 \\ x_1 \le 4 \\ x_1, x_2 \ge 0 \text{且全为整数} \end{cases}$$

记为（*IP*）

解：首先去掉整数约束，变成一般线性规划问题

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

$$\begin{cases} x_1 - x_2 \ge -2 \\ 5x_1 + 6x_2 \le 30 \\ x_1 \le 4 \\ x_1, x_2 \ge 0 \end{cases}$$

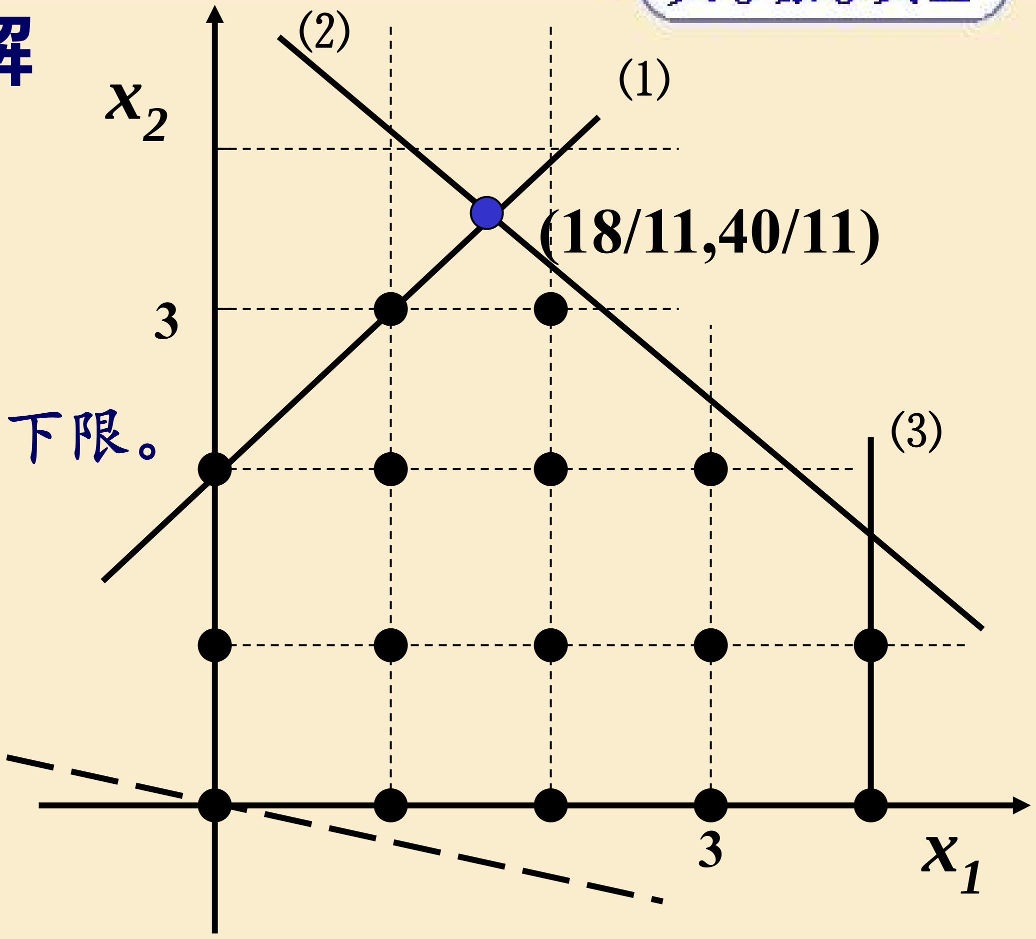
记为（*LP*）

求（*LP*）的最优解

$x_1 = 18/11$， $x_2 = 40/11$

$Z^{(0)} = -218/11 \approx (-19.8)$

即 Z 也是（IP）最小值的下限。



对于$x_1 = 18/11 \approx 1.64$，

取值$x_1 \leqslant 1$， $x_1 \geqslant 2$

对于$x_2 = 40/11 \approx 3.64$，取值$x_2 \leqslant 3$ ， $x_2 \geqslant 4$

先将（LP）划分为（LP1）和（LP2），取$x_1 \leqslant 1$， $x_1 \geqslant 2$

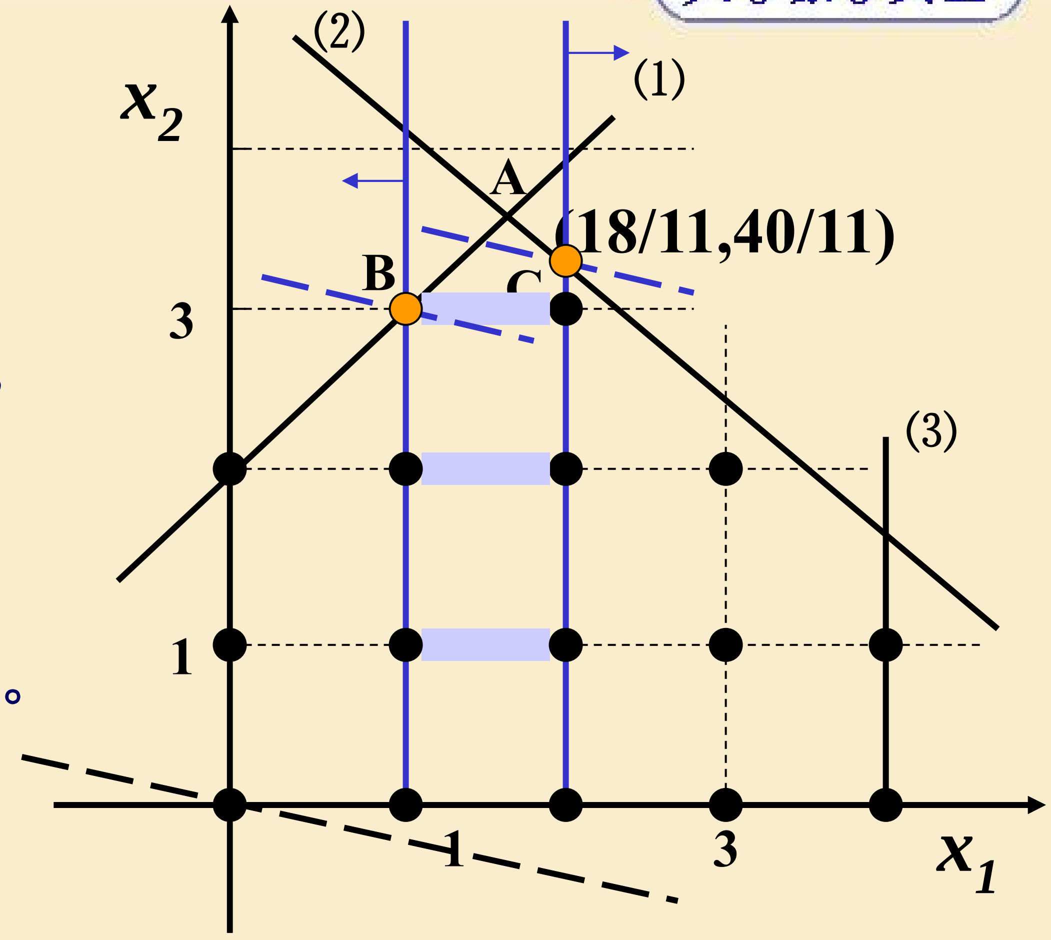
拆分的两个问题分别为:

$$\min\ Z = -x_1 - 5x_2 \qquad\qquad \min\ Z = -x_1 - 5x_2$$

$$(IP1)\begin{cases} x_1 - \ \ x_2 \geq -2 \\ 5x_1 + 6x_2 \leq 30 \\ x_1 \qquad\quad \leq 4 \\ x_1 \qquad\quad \leq 1 \\ x_1, x_2 \geq 0\text{且为整数} \end{cases} \qquad (IP2)\begin{cases} x_1 - \ \ x_2 \geq -2 \\ 5x_1 + 6x_2 \leq 30 \\ x_1 \qquad\quad \leq 4 \\ x_1 \qquad\quad \geq 2 \\ x_1, x_2 \geq 0\text{且为整数} \end{cases}$$

现在只要求出（*LP1*）和（*LP2*）的最优解即可。

先求（LP1），
B 点取得最优解。
$x_1=1$， $x_2=3$， $Z^{(1)}=-16$
找到整数解，问题已探明，
此枝停止计算。



同理求（LP2），如图所示。

在C 点取得最优解。

即$x_1=2$， $x_2=10/3$，

$Z^{(2)}=-56/3\approx-18.7$

因为 $Z_2<Z_1=-16$， 所以原问题有比（－16）更小的最优解，

但 x_2 不是整数，故利用 $4\geq 10/3\geq 3$ 加入条件。

加入条件： $x_2 \le 3, x_2 \ge 4$ 有如下拆分：

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

$$(IP3)\begin{cases} x_1 - x_2 \ge -2 \\ 5x_1 + 6x_2 \le 30 \\ x_1 \le 4 \\ x_1 \ge 2 \\ x_2 \le 3 \\ x_1, x_2 \ge 0\text{且为整数} \end{cases}$$

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

$$(IP4)\begin{cases} x_1 - x_2 \ge -2 \\ 5x_1 + 6x_2 \le 30 \\ x_1 \le 4 \\ x_1 \ge 2 \\ x_2 \ge 4 \\ x_1, x_2 \ge 0\text{且为整数} \end{cases}$$

只要求出（*LP3*）和（*LP4*）的最优解即可。

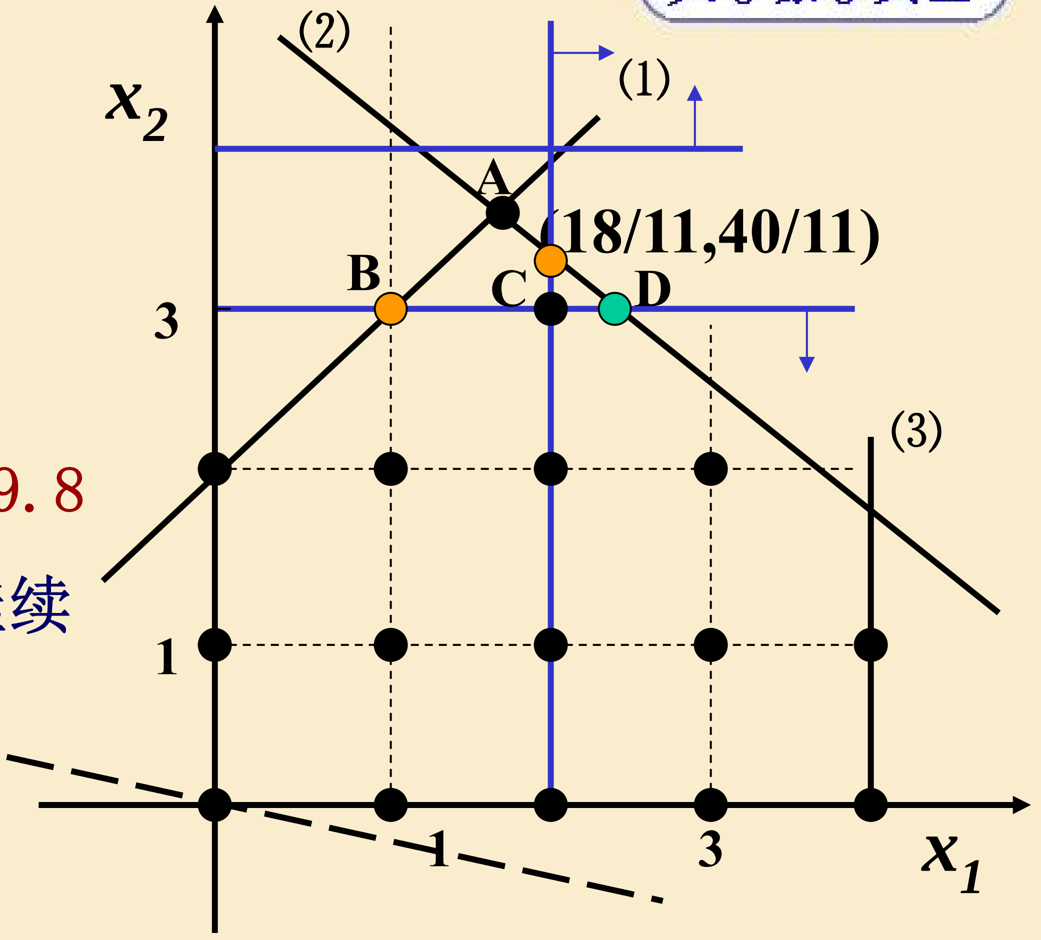
先求（LP3）

此时D 在点取得最优解。

即 $x_1=12/5\approx2.4$, $x_2=3$,

$Z^{(3)}=-87/5\approx-17.4<Z\approx-19.8$

但$x_1=12/5$不是整数，可继续分枝，即 $3\leqslant x_1\leqslant2$。



求（*LP4*），如图所示。
无可行解，不再分枝。

在（LP3）的基础上继续分枝,加入条件 3≤x_1≤2

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

$$(IP5)\begin{cases} x_1-\ \ x_2\ge -2\\ 5x_1+6x_2\le 30\\ x_1\qquad\quad \le 4\\ x_1\qquad\quad \ge 2\\ x_2\qquad\quad \le 3\\ x_1\qquad\quad \le 2\\ x_1,x_2\ge 0\text{且为整数}\end{cases}$$

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

$$(IP6)\begin{cases} x_1-\ \ x_2\ge -2\\ 5x_1+6x_2\le 30\\ x_1\qquad\quad \le 4\\ x_1\qquad\quad \ge 2\\ x_2\qquad\quad \le 3\\ x_1\qquad\quad \ge 3\\ x_1,x_2\ge 0\text{且为整数}\end{cases}$$

只要求出（*LP5*）和（*LP6*）的最优解即可。

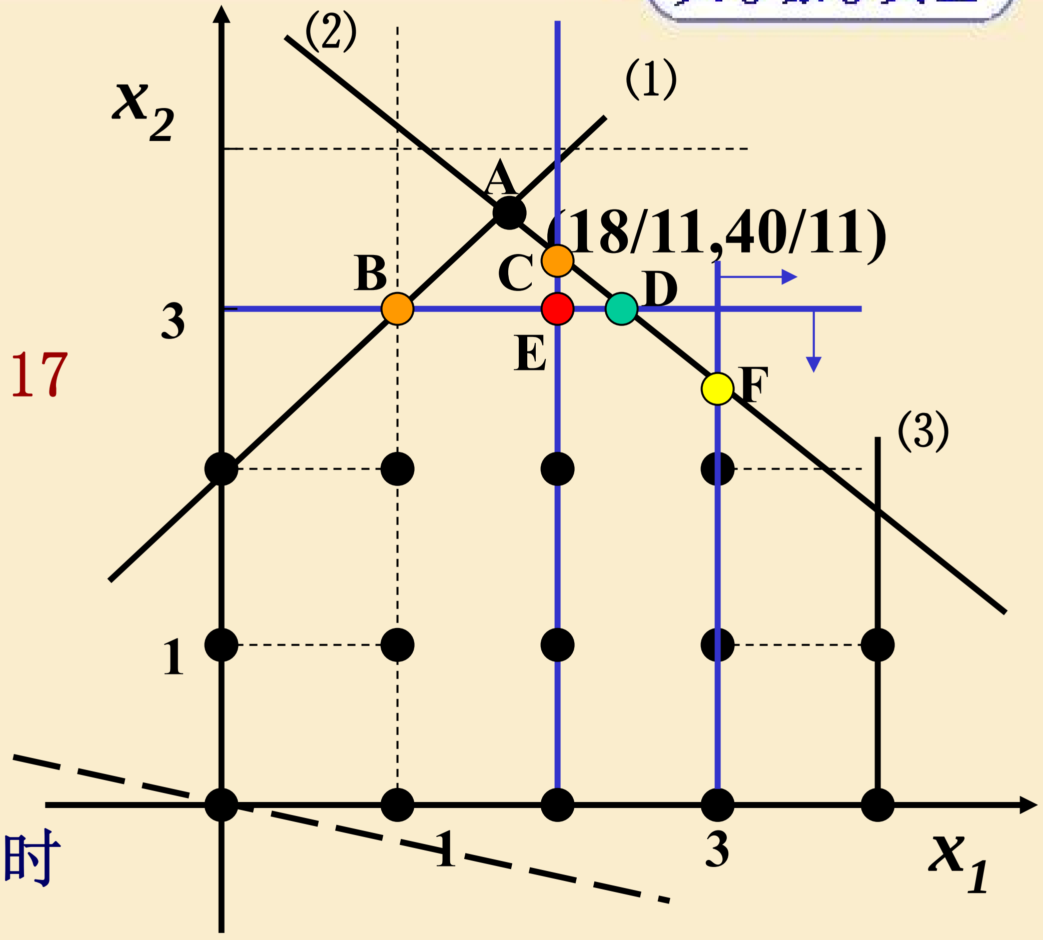
先求（LP5）

此时E 在点取得最优解。

即 $x_1=2$， $x_2=3$， $Z^{(5)}=-17$

找到整数解，问题已探明，

此枝停止计算。



求（LP6），如图所示。此时F在点取得最优解。

$x_1=3$， $x_2=2.5$，

$Z^{(6)}=-31/2\approx-15.5 > Z^{(5)}$

如对 $Z^{(6)}$ 继续分解，其最小值也不会低于 - 15.5 ，问题探明,剪枝。

至此，原问题（IP）的最优解为：

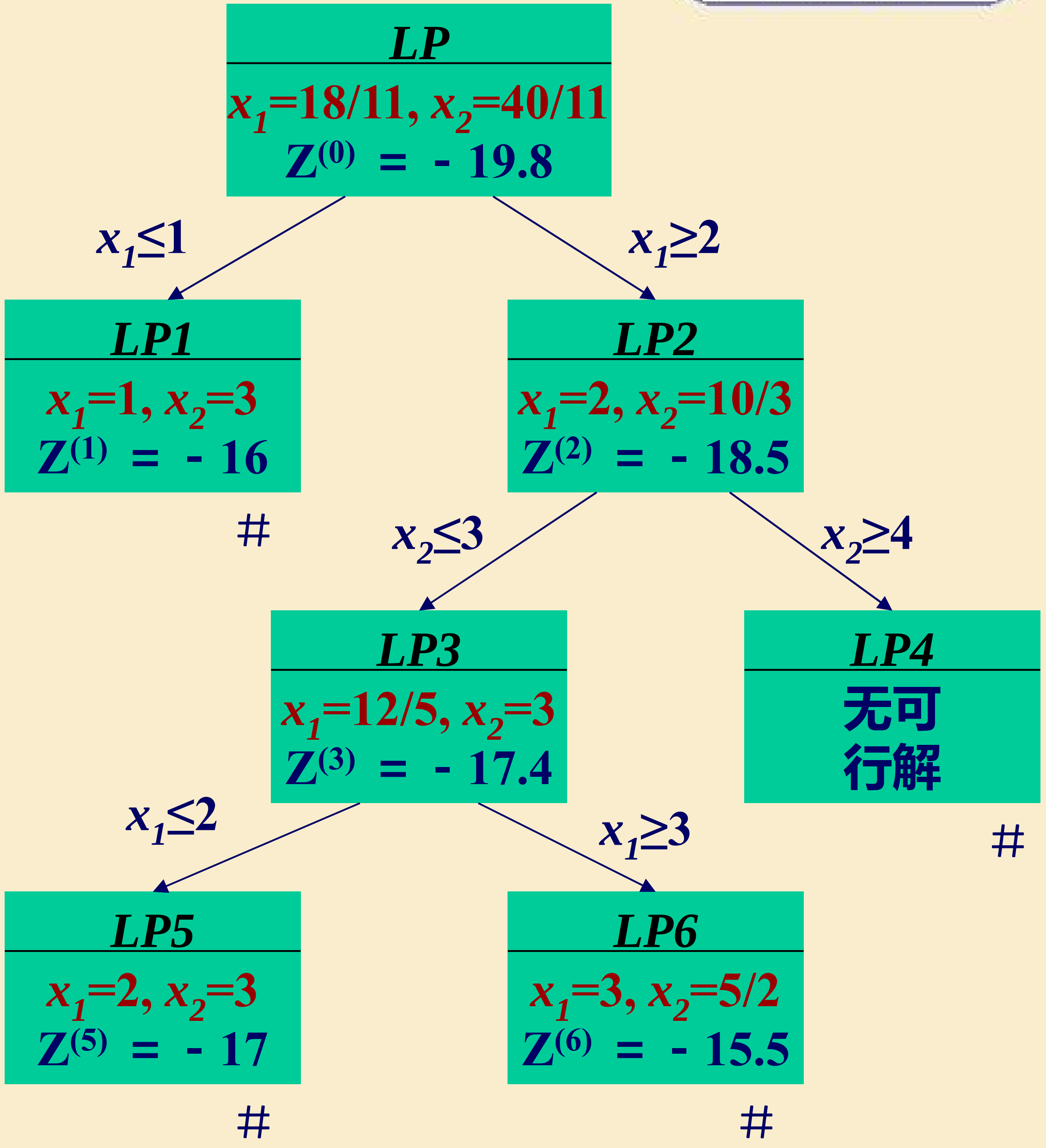
$x_1=2$,

$x_2=3$,

最优值为

$Z^* = Z^{(5)} = -17$

以上的求解过程可以用一个树形图表示如右：



旅行商问题（TSP）的动态规划算法

旅行商问题/货郎（担）问题(Traveling Salesman Problem)

一名推销员准备前往若干城市推销产品. 如何为他(她)设计一条最短的旅行路线(从驻地出发，经过每个城市恰好一次，最后返回驻地)?

考虑 **n+1** 个城市，从任意一个城市出发，下一站有 **n** 种可能的选择，在下一站有**n-1**种可能，**……** 直到返回出发城市，一共有 **n**！ 种可能的走法。

J. Soc. INDUST. APPL. MATH.， Vol. 10, No. 11962
A dynamic programming approach to sequencing problelms
Michael Held, Richard Karp

旅行商问题的动态规划算法

用$f_k(v_i, V)$表示从v_i点出发，经过V中的点各一次，最后回到v_0点的最短路程，V是一个顶点集合，$|V|=k$，d_{ij}是v_i到v_j的弧长，则

$$\begin{cases} f_k(v_i,V)=\min\limits_{v_j\in V}\{d_{ij}+f_{k-1}(v_j,V\setminus\{v_j\})\},k=1,2,\cdots,n \\ f_0(v_i,\varnothing)=d_{i0} \end{cases}$$

$$D=\begin{pmatrix} 0 & 8 & 5 & 6 \\ 6 & 0 & 8 & 5 \\ 7 & 9 & 0 & 5 \\ 9 & 7 & 8 & 0 \end{pmatrix}$$

$$\mathbf{d_{01}=8, d_{12}=8, d_{32}=8}$$

$$\begin{cases} f_k(v_i,V)=\min\limits_{v_j\in V}\{d_{ij}+f_{k-1}(v_j,V\setminus\{v_j\})\}, k=1,2,\cdots,n \\ f_0(v_i,\varnothing)=d_{i0} \end{cases}$$

$$f_3\left(v_0,\{v_1,v_2,v_3\}\right)=\min\begin{Bmatrix} d_{01}+f_2\left(v_1,\{v_2,v_3\}\right) \\ d_{02}+f_2\left(v_2,\{v_1,v_3\}\right) \\ d_{03}+f_2\left(v_3,\{v_1,v_2\}\right) \end{Bmatrix}$$

$$D=\begin{pmatrix} 0 & 8 & 5 & 6 \\ 6 & 0 & 8 & 5 \\ 7 & 9 & 0 & 5 \\ 9 & 7 & 8 & 0 \end{pmatrix}$$

$$f_2\left(v_1,\{v_2,v_3\}\right)=\min\left\{d_{12}+f_1\left(v_2,\{v_3\}\right),d_{13}+f_1\left(v_3,\{v_2\}\right)\right\} \quad v_0\to v_1\to v_2,v_3\,?\to v_0$$

$$f_2\left(v_2,\{v_1,v_3\}\right)=\min\left\{d_{21}+f_1\left(v_1,\{v_3\}\right),d_{23}+f_1\left(v_3,\{v_1\}\right)\right\} \quad v_0\to v_2\to v_1,v_3\,?\to v_0$$

$$f_2\left(v_3,\{v_1,v_2\}\right)=\min\left\{d_{31}+f_1\left(v_1,\{v_2\}\right),d_{32}+f_1\left(v_2,\{v_1\}\right)\right\} \quad v_0\to v_3\to v_1,v_2\,?\to v_0$$

$$f_3\left(v_0,\{v_1,v_2,v_3\}\right)=\min\begin{Bmatrix} d_{01}+f_2\left(v_1,\{v_2,v_3\}\right) \\ d_{02}+f_2\left(v_2,\{v_1,v_3\}\right) \\ d_{03}+f_2\left(v_3,\{v_1,v_2\}\right) \end{Bmatrix}$$

$$D=\begin{pmatrix} 0 & 8 & 5 & 6 \\ 6 & 0 & 8 & 5 \\ 7 & 9 & 0 & 5 \\ 9 & 7 & 8 & 0 \end{pmatrix}$$

$$f_2\left(v_1,\{v_2,v_3\}\right)=\min\left\{d_{12}+f_1\left(v_2,\{v_3\}\right),d_{13}+f_1\left(v_3,\{v_2\}\right)\right\}$$

$$f_2\left(v_2,\{v_1,v_3\}\right)=\min\left\{d_{21}+f_1\left(v_1,\{v_3\}\right),d_{23}+f_1\left(v_3,\{v_1\}\right)\right\}$$

$$f_2\left(v_3,\{v_1,v_2\}\right)=\min\left\{d_{31}+f_1\left(v_1,\{v_2\}\right),d_{32}+f_1\left(v_2,\{v_1\}\right)\right\}$$

$$f_1\left(v_1,\{v_2\}\right)=d_{12}+d_{20}=15,\quad f_1\left(v_1,\{v_3\}\right)=d_{13}+d_{30}=14$$

$$f_1\left(v_2,\{v_1\}\right)=d_{21}+d_{10}=15,\quad f_1\left(v_2,\{v_3\}\right)=d_{23}+d_{30}=14$$

$$f_1\left(v_3,\{v_1\}\right)=d_{31}+d_{10}=13,\quad f_1\left(v_3,\{v_2\}\right)=d_{32}+d_{20}=15$$

$$\begin{tikzcd} v_0 \arrow[r] & v_2 \arrow[r] & v_3 \arrow[d] \\ & v_0 & v_1 \arrow[l] \end{tikzcd}$$

$$f_2\left(v_1,\{v_2,v_3\}\right)=\min\left\{d_{12}+f_1\left(v_2,\{v_3\}\right),d_{13}+f_1\left(v_3,\{v_2\}\right)\right\}=\min\left\{8+f_1\left(v_2,\{v_3\}\right),5+f_1\left(v_3,\{v_2\}\right)\right\}=20$$

$$f_2\left(v_2,\{v_1,v_3\}\right)=\min\left\{d_{21}+f_1\left(v_1,\{v_3\}\right),d_{23}+f_1\left(v_3,\{v_1\}\right)\right\}=\min\left\{9+f_1\left(v_1,\{v_3\}\right),5+f_1\left(v_3,\{v_1\}\right)\right\}=18$$

$$f_2\left(v_3,\{v_1,v_2\}\right)=\min\left\{d_{31}+f_1\left(v_1,\{v_2\}\right),d_{32}+f_1\left(v_2,\{v_1\}\right)\right\}=\min\left\{7+f_1\left(v_1,\{v_2\}\right),8+f_1\left(v_2,\{v_1\}\right)\right\}=22$$

$$\begin{cases} f_k(v_i,V)=\min\limits_{v_j\in V}\{d_{ij}+f_{k-1}(v_j,V\setminus\{v_j\})\}, k=1,2,\cdots,n \\ f_0(v_i,\varnothing)=d_{i0} \end{cases}$$

$f_k\left(v_i,V_k\right)\to\left(n-k\right)C_n^k$， $f_k\left(v_i,V_k\right)$:k个子状态的比较

总计算量： $\sum\limits_{k=0}^{n-1}k\left(n-k\right)C_n^k\to O\left(n^2 2^n\right)$

$$f_2\left(v_1,\left\{v_3,v_4\right\}\right)=\min\left\{d_{13}+f_1\left\{v_3,v_4\right\},d_{14}+f_1\left\{v_4,v_3\right\}\right\}$$
$$f_2\left(v_2,\left\{v_3,v_4\right\}\right)=\min\left\{d_{23}+f_1\left\{v_3,v_4\right\},d_{24}+f_1\left\{v_4,v_3\right\}\right\}$$

$n=30$时， $n!<\left(\frac{n}{e}\right)^n<10^{30}$， $n^2\cdot 2^n<10^3\cdot\left(10^3\right)^3=10^{12}$

如果用一台每秒计算10^{12}的计算机,10^{30}计算量需要$\dfrac{10^{18}}{31\,536\,000}\approx 3\cdot 10^{10}$年

计算复杂性的概念

算法复杂性： 多项式时间算法， 指数时间算法

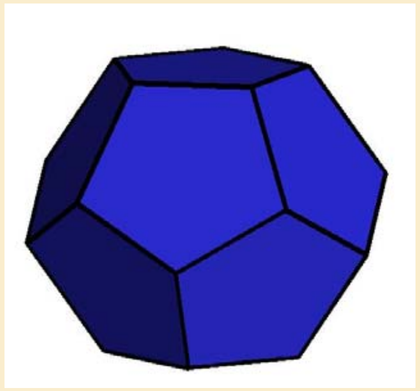
问题复杂性：容易的（多项式时间可解问题）
难的（找不到多项式时间算法）

NP完全问题

NP与**P**， **NP=P**？

Complexity Analysis

Why The Simplex Method Usually Takes Polynomial Time



Remarkable Algorithms and Heuristics

Work well in practice, but

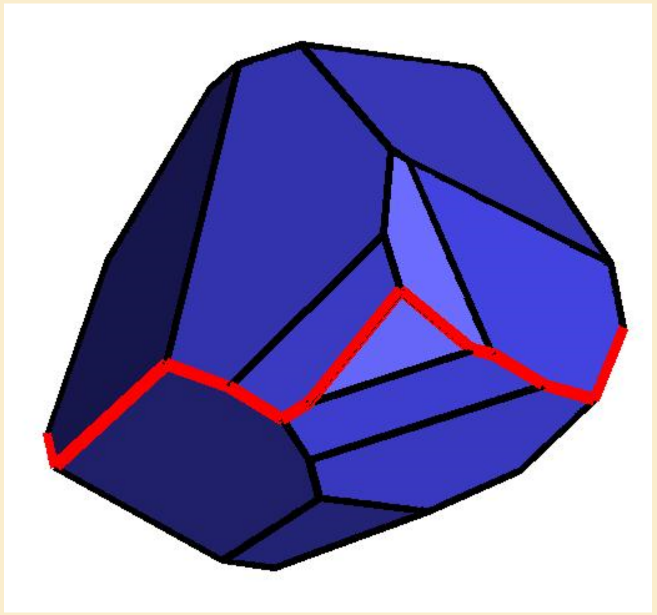
Worst case: bad,
exponential,
contrived(artifical).

Average case: good,
polynomial,
meaningful?

Smoothed analysis

Classical Example: Simplex Method for Linear Programming

$$\begin{array}{ll} \max & z^T x \\ \text{s.t.} & A x \le y \end{array}$$



- Worst-Case: exponential
- Average-Case: polynomial
- Widely used in practice

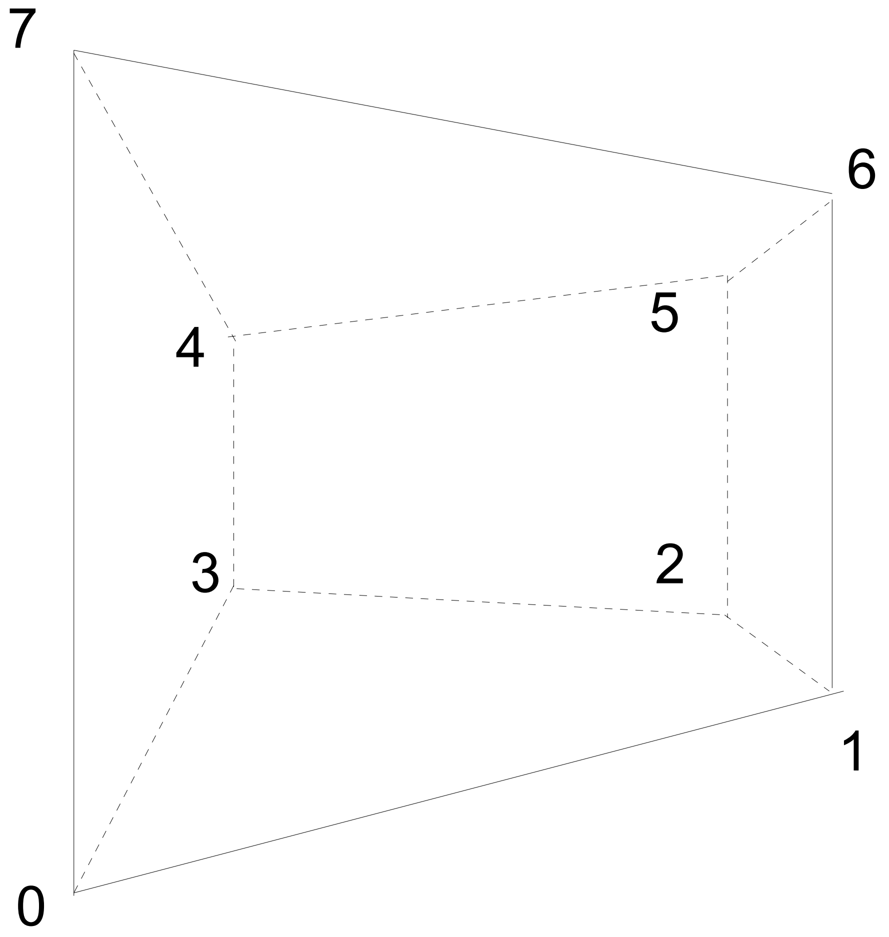
Worst case:

Klee and Minty Problem(1972)

$$\max \; e_n^T x$$

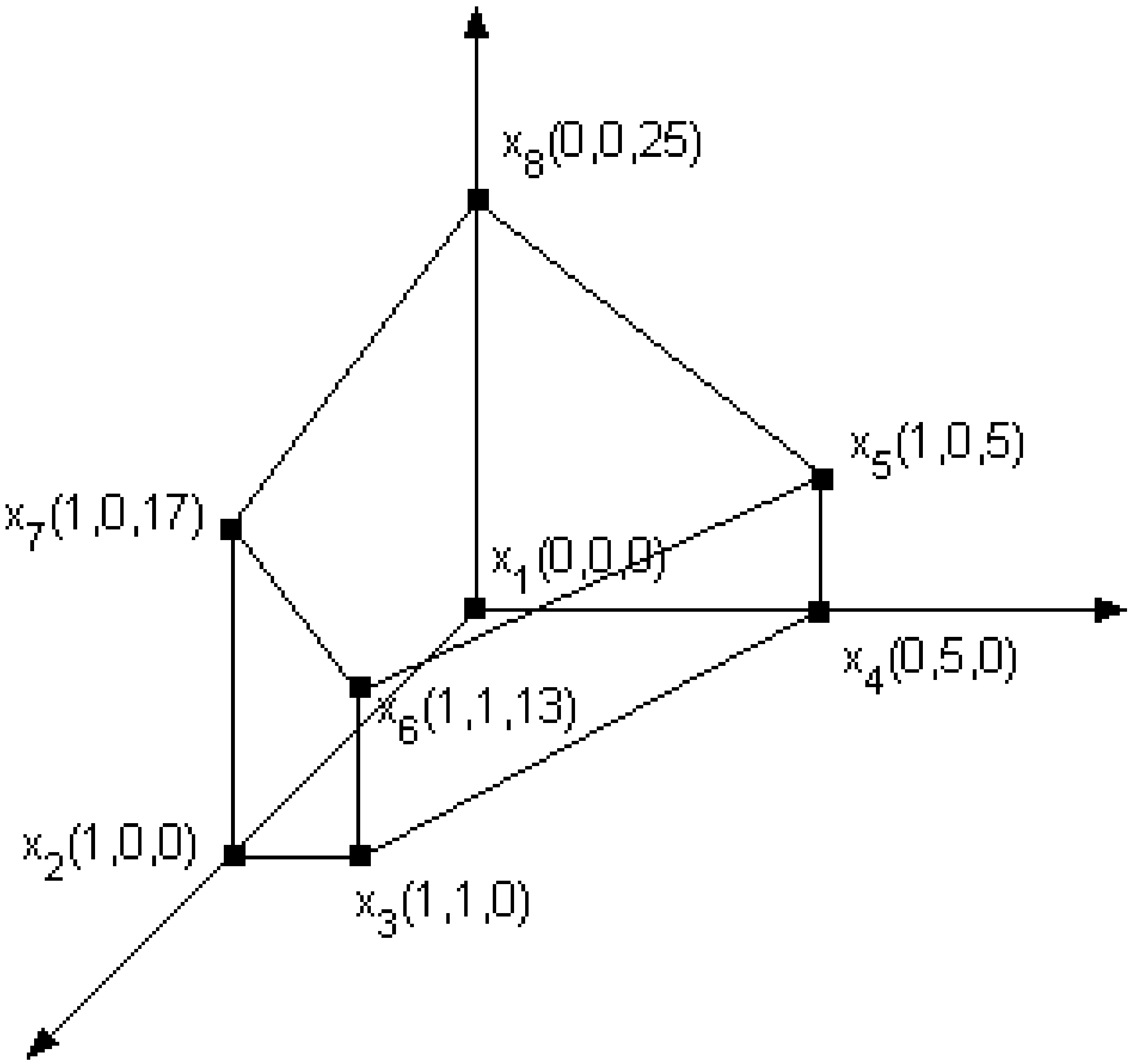
$$\text{s.t} \quad \begin{array}{ll} x_1 \geq 0 & x_1 \leq 1 \\ x_2 \geq \varepsilon x_1 & x_2 \leq 1 - \varepsilon x_1 \\ x_3 \geq \varepsilon x_2 & x_3 \leq 1 - \varepsilon x_2 \\ \vdots & \vdots \\ x_n \geq \varepsilon x_{n-1} & x_n \leq 1 - \varepsilon x_{n-1} \end{array}$$

$$0 < \varepsilon < 1/2$$


7
6
5
4
3
2
1
0

Example (n=3,a=5)

$$
\begin{array}{lrcl}
\max & 4x_1 + 2x_2 + x_3 & & \\
\text{s.t.} & x_1 & \le & 1 \\
 & 4x_1 + x_2 & \le & 5 \\
 & 8x_1 + 2x_2 + x_3 & \le & 25 \\
 & x_1, x_2, x_3 & \ge & 0
\end{array}
$$



$x_8(0,0,25)$
$x_5(1,0,5)$
$x_7(1,0,17)$
$x_1(0,0,0)$
$x_4(0,5,0)$
$x_6(1,1,13)$
$x_2(1,0,0)$
$x_3(1,1,0)$

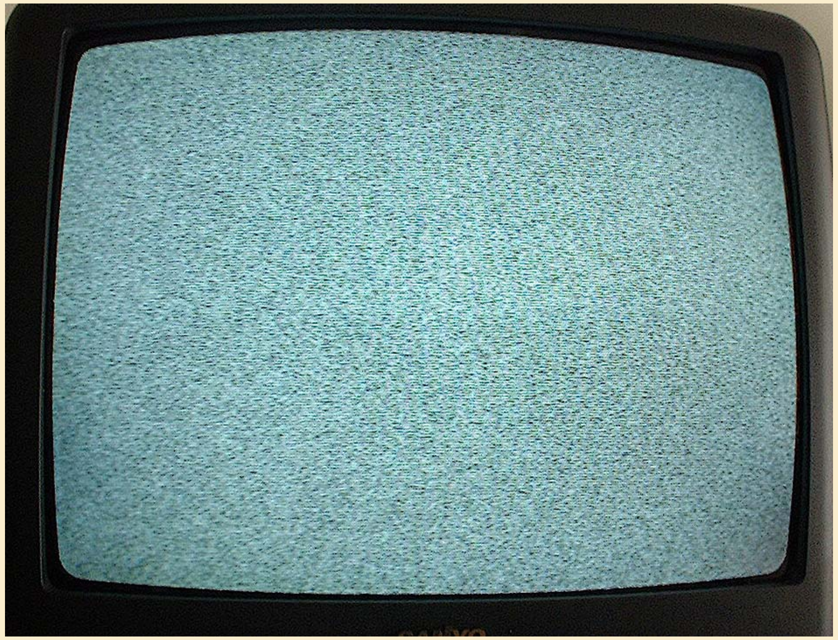
History of Linear Programming

- Simplex Method (Dantzig, ‘47)
- Exponential Worst-Case (Klee-Minty ‘72)
- Avg-Case Analysis (Borgwardt ‘77, Smale ‘82, Haimovich, Adler, Megiddo, Shamir, Karp, Todd)
- Ellipsoid Method (Khaciyan, ‘79)
- Interior-Point Method (Karmarkar, ‘84)
- Randomized Simplex Method ($m^{O(\sqrt{d})}$) _

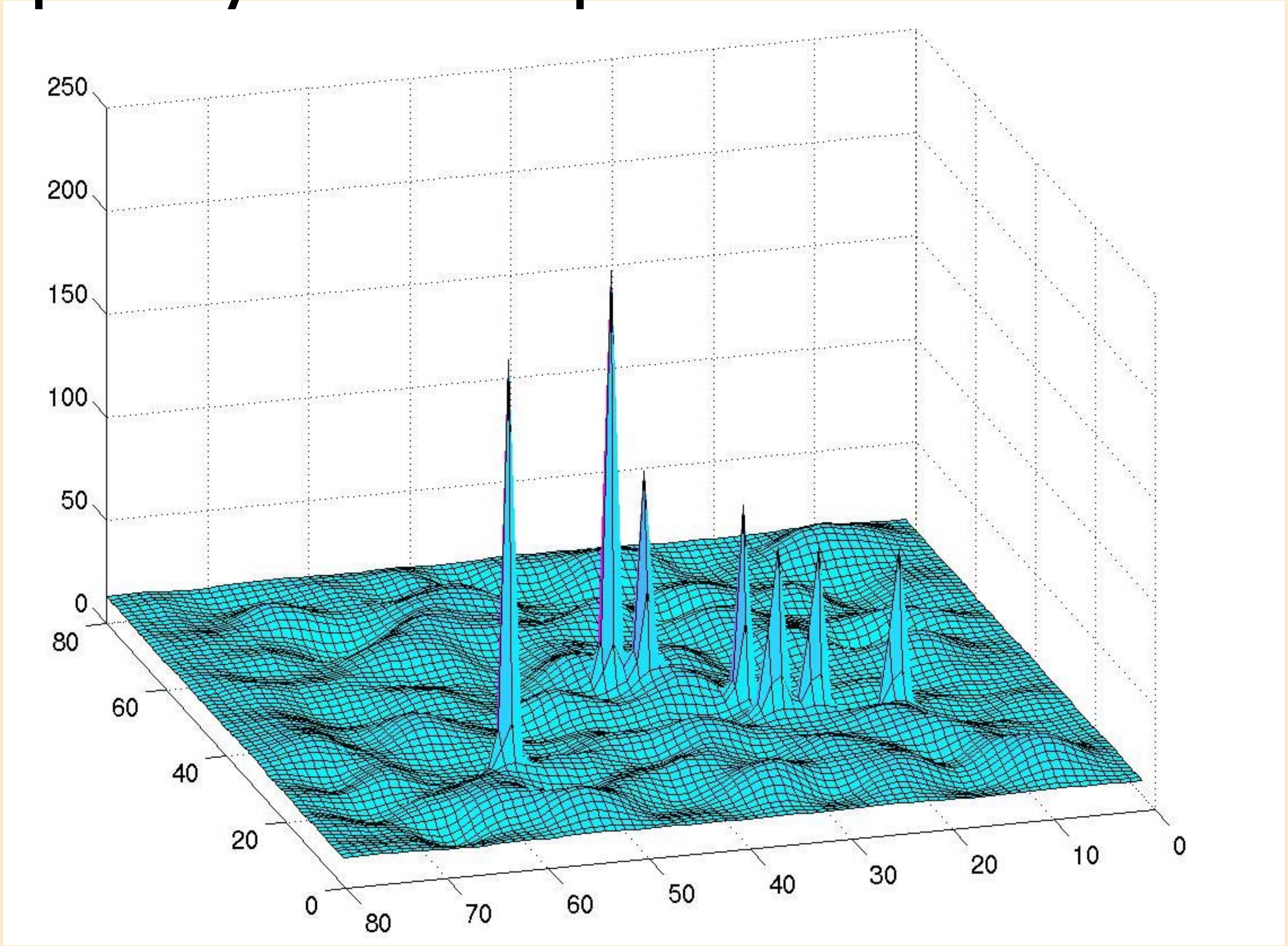
  (Kalai ‘92, Matousek-Sharir-Welzl ‘92)
- Smoothed Analysis

  (Spielman,Teng ‘00)

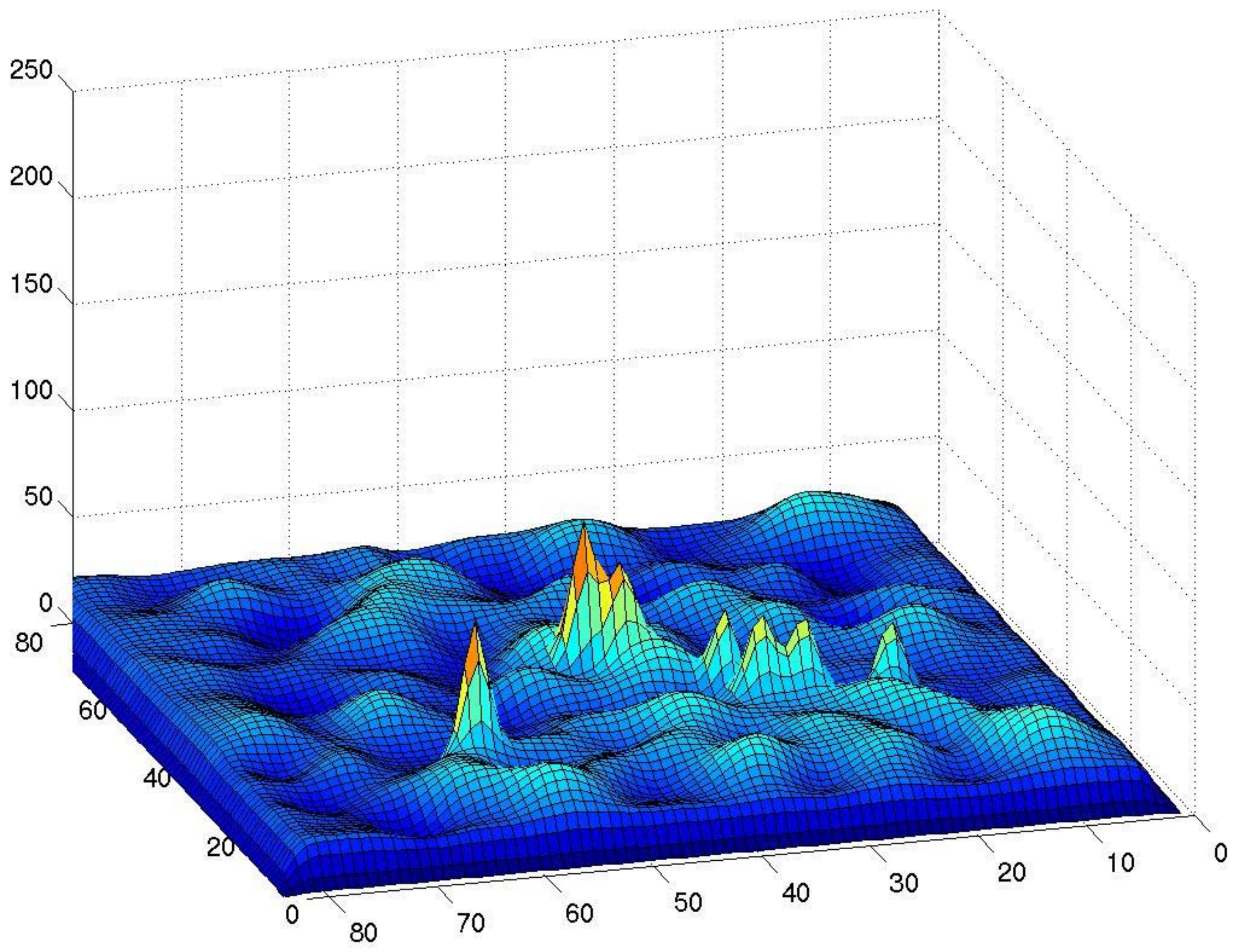
Random is not typical

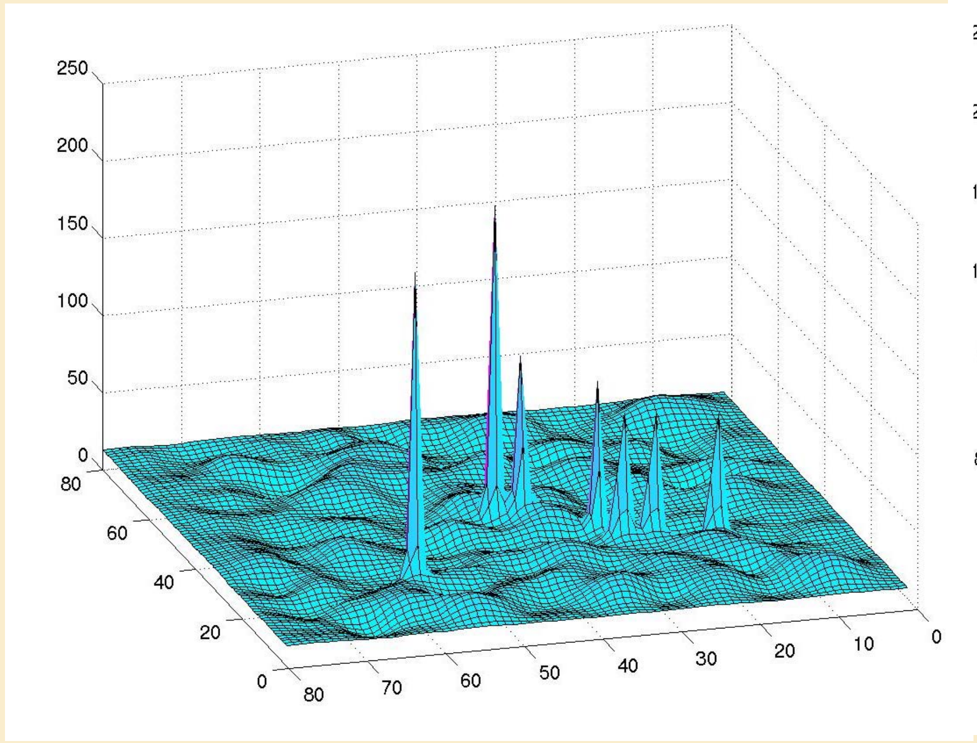


Complexity Landscape

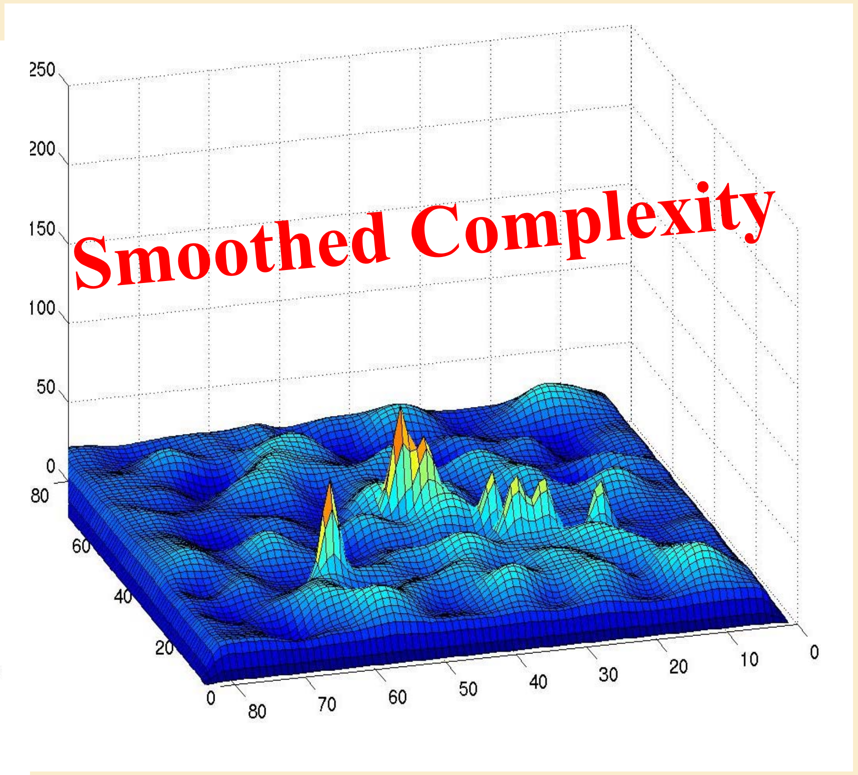


Smoothed Complexity Landscape





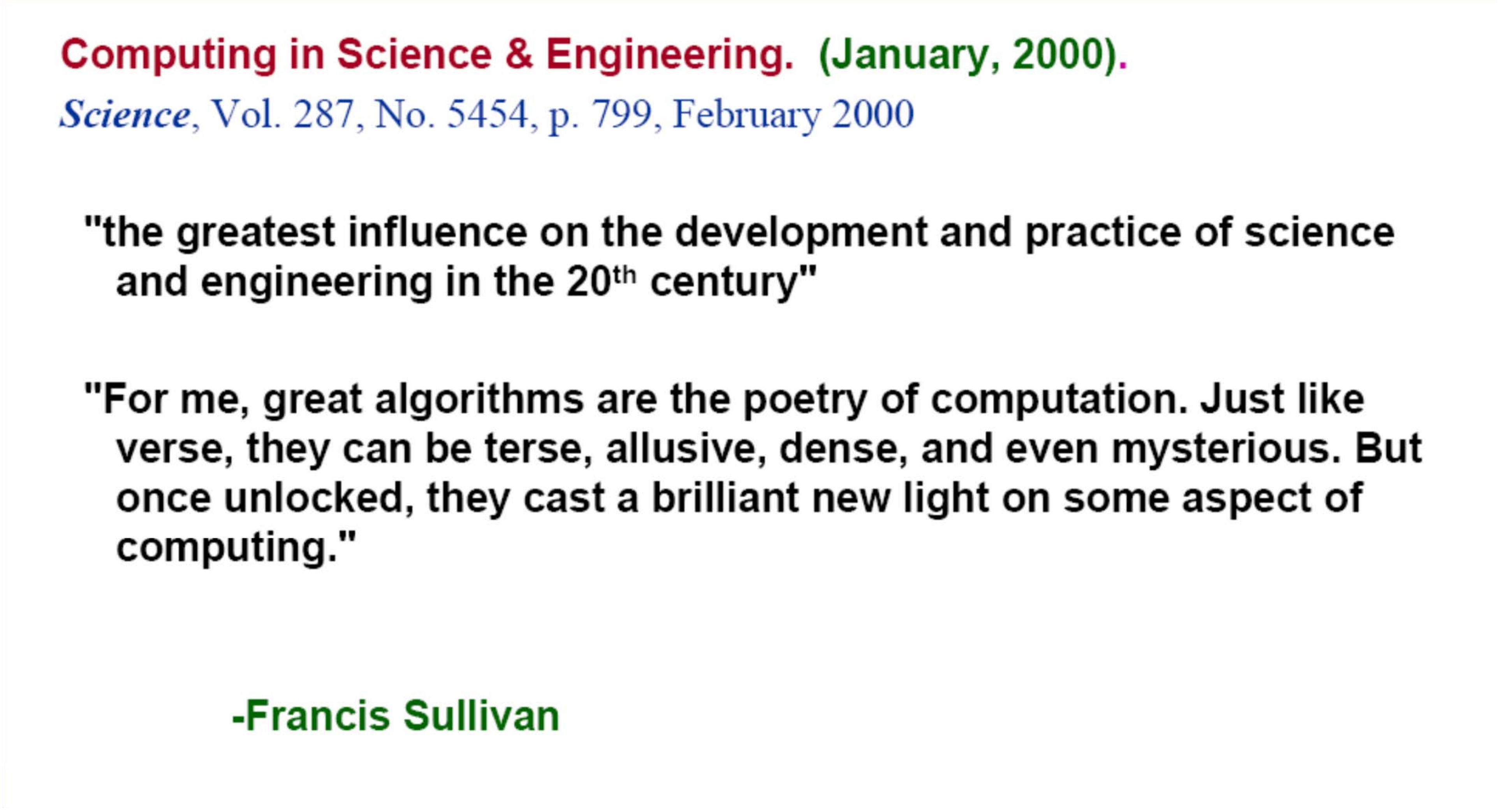
250
200
150
100
50
0
80
60
40
20
0
80
70
60
50
40
30
20
10
0



Smoothed Complexity
250
200
150
100
50
0
80
60
40
20
0
80
70
60
50
40
30
20
10
0

20世纪十大算法

- SIAM NEWS， **May 16, 2000 ， Barry A Cipra**



Computing in Science & Engineering. (January, 2000).

Science, Vol. 287, No. 5454, p. 799, February 2000

"the greatest influence on the development and practice of science and engineering in the 20th century"

"For me, great algorithms are the poetry of computation. Just like verse, they can be terse, allusive, dense, and even mysterious. But once unlocked, they cast a brilliant new light on some aspect of computing."

-Francis Sullivan

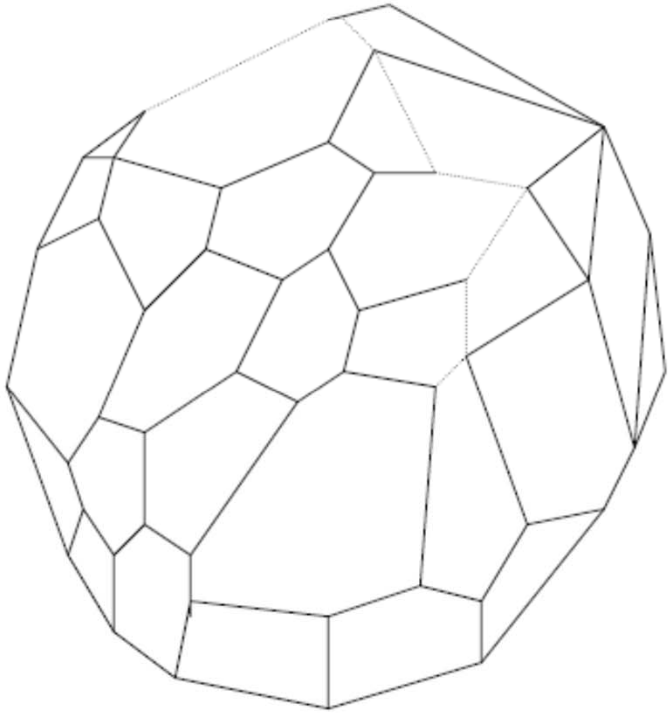
1. Monte Carlo

1. Metropolis Algorithm/ Monte Carlo method (Francis Sullivan, Neumann, Ulam, Metropolis, 1946). Through the use of random processes, this algorithm offers an efficient way to stumble toward answers to problems that are too complicated to solve exactly. Indeed Monte Carlo method is the only practical choice for evaluating problems of high dimensions.
   - Approximate solutions to numerical problems with too many degrees of freedom.
   - Approximate solutions to combinatorial optimization problems.
   - Generation of random numbers.

2. 单纯形算法

2. Simplex Method for Linear Programming (Dantzig 1947).
An elegant solution to a common problem in planning and decision-making: $\max\{cx : Ax \le b, x \ge 0\}$.

- One of most successful algorithms of all time.
- Dominates world of industry.



3. 线性方程组与特征值的Krylov子空间迭代

3. Krylov Subspace Iteration Method (Hestenes, Stiefel, Lanczos, 1950).
A technique for rapidly solving $Ax = b$ where A is a huge $n \times n$ matrix.

Subspace Iteration Method

```
Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$
for i = 1,2,...
    Solve $Mz^{(i-1)} = r^{(i-1)}$
    $\rho_{i-1} = r^{(i-1)T} z^{(i-1)}$
    if i = 1
        $p^{(1)} = z^{(0)}$
    else
        $\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$
        $p^{(i)} = z^{(i-1)} + \beta_{i-1} p^{(i-1)}$
    endif
    $q^{(i)} = Ap^{(i)}$
    $\alpha_i = \rho_{i-1} / p^{(i)T} q^{(i)}$
    $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$
    $r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$
    Check convergence; continue if necessary
end
```

Preconditioned Conjugate Gradient

4. 矩阵分解方法

4. Decompositional Approach to Matrix Computations (Householder, 1951). A suite of technique for numerical linear algebra that led to efficient matrix packages.

- Factor matrices into triangular, diagonal, orthogonal, tri-diagonal, and other forms.
- Analysis of rounding errors.
- Applications to least squares, eigenvalues, solving systems of linear equations.
- LINPACK, EISPACK.

5. John Backus leads a team at IBM in developing the **Fortran optimizing compiler**

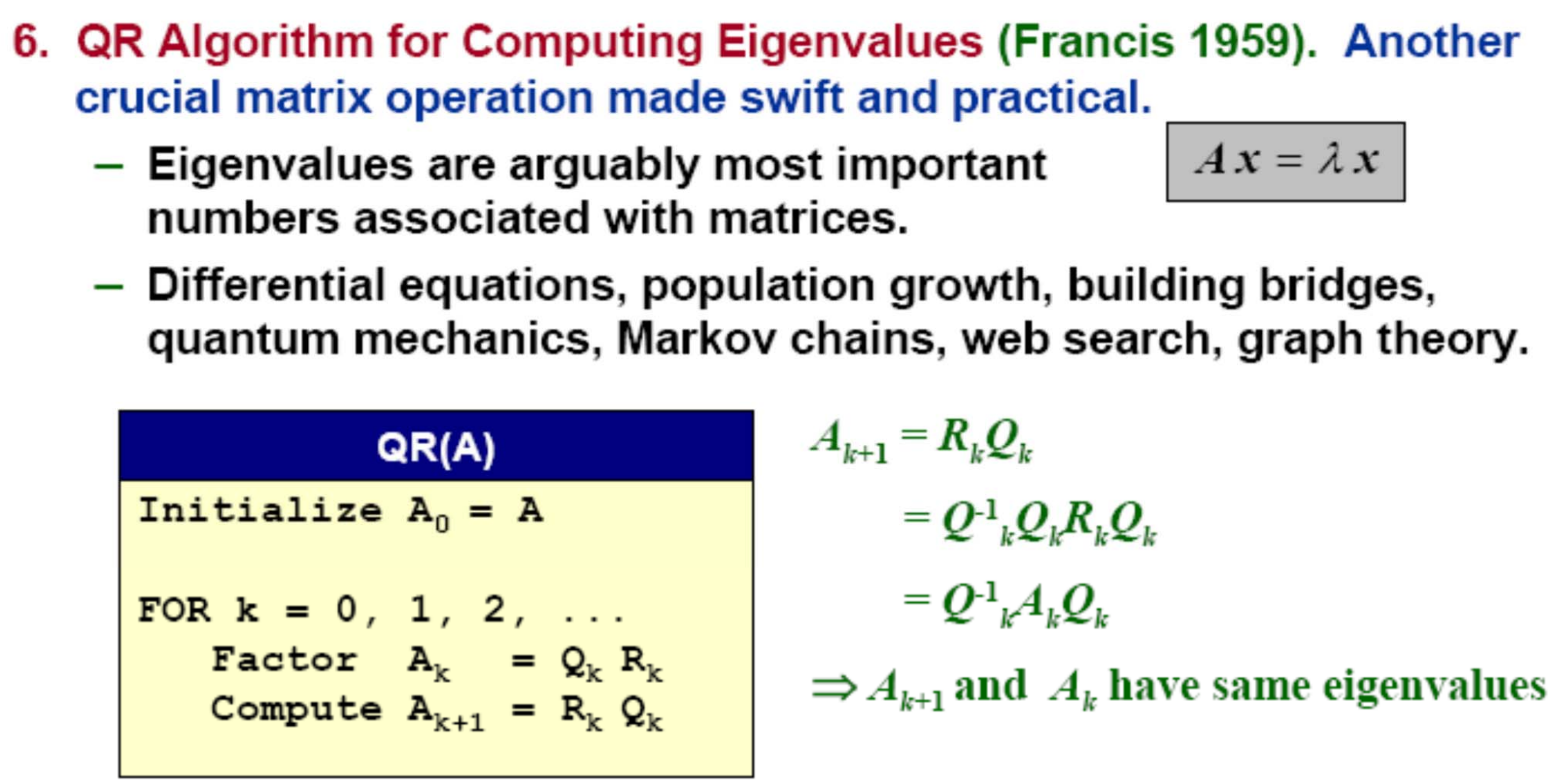
5. Fortran Optimizing Compiler (John Warner Backus, 1957). Turns high-level code into efficient computer-readable code.

- Among single most important events in history of computing: scientists could program computer without learning assembly.
- the inventor for the popular metalanguage BNF
- He is the 1977 recipient of the ACM Turing Award

Fortran Code

```
500    C = 0.0
C   *** START LOOP ***
       DO 540 I=L,K
          F = S*RV1(I)
          RV1(I) = C*RV1(I)
          IF (ABS(F).LE.EPS) GO TO 550
          G = W(I)
          H = SQRT(F*F+G*G)
          W(I) = H
          C = G/H
          S = -F/H
510    CONTINUE
```

6. 矩阵特征值的QR方法



6. QR Algorithm for Computing Eigenvalues (Francis 1959). Another crucial matrix operation made swift and practical.

- Eigenvalues are arguably most important numbers associated with matrices.
- Differential equations, population growth, building bridges, quantum mechanics, Markov chains, web search, graph theory.

$\boxed{Ax = \lambda x}$

QR(A)

```
Initialize $A_0$ = A

FOR k = 0, 1, 2, ...
    Factor   $A_k$   = $Q_k$ $R_k$
    Compute  $A_{k+1}$ = $R_k$ $Q_k$
```

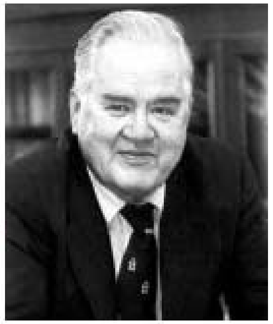
$$
\begin{aligned}
A_{k+1} &= R_k Q_k \\
&= Q^{-1}{}_k Q_k R_k Q_k \\
&= Q^{-1}{}_k A_k Q_k
\end{aligned}
$$

$\Rightarrow A_{k+1}$ and A_k have same eigenvalues

7. 快速排序； 8. FFT

7. Quicksort (Hoare, 1962). Given n items over a totally order universe, rearrange them in increasing order.
   - $O(n \log n)$ instead of $O(n^2)$.
     - » Efficient handling of large databases.
   - He is the 1980 recipient of the ACM Turing Award
     - » For his fundamental contributions to the definition and design of programming languages.

8. Fast Fourier Transform (Cooley, Tukey 1965). Perhaps the most ubiquitous algorithm in use today, it breaks down waveforms (like sound) into periodic components.
   - $O(n \log n)$ instead of $O(n^2)$ by Runge- König (1924).
     - » foundation of signal processing
     - » CD players, JPEG, analyzing astronomical data, etc.



1. 对于如下线性规划问题（有 $3n$ 个决策变量 (x,r,s) 和 $2n$ 个约束）.

$$
\begin{aligned}
&\min(-x_n)\\
&s.t.\quad 4x_1-4r_1=1,\\
&\qquad x_1+s_1=1,\\
&\qquad 4x_j-x_{j-1}-4r_j=0,\quad j=2,3,\cdots,n,\\
&\qquad 4x_j+x_{j-1}+4s_j=0,\quad j=2,3,\cdots,n,\\
&\qquad x_j,s_j,r_j\ge 0,\qquad j=1,3,\cdots,n,
\end{aligned}
$$

请分别对 n 的不同取值（如 $n=2,10,50,\cdots$）求解上述线性规划问题，算法使用 linprog 默认的算法（dual-simplex 算法）观察计算时间。

2. 某牧场主知道，对于一匹普通的马，每周最低的营养需求为：40 磅蛋白质，20 磅碳水化合物，45 磅粗饲料。这些影响成分是从不同饲料中得到的，饲料及其价格在下表中列出。建立数学模型，确定如何以最低的成本满足马的最低营养需求，并分析当蛋白质最低需求量增加 2 磅或碳水化合物需求增加 1 磅时，最低成本的变化。

——	蛋白质/磅	碳水化合物/磅	粗饲料/磅	价格/元
干草/捆	0.5	2.0	5.0	1.80
燕麦片/袋	1.0	4.0	2.0	3.50
饲料块/块	2.0	0.5	1.0	0.40
高蛋白浓缩料/袋	6.0	1.0	2.5	1.00
每匹马的需求/周	40.0	20.0	45.0	——

3. 用分支定界算法求解下面整数线性规划问题

$$\min z = 7x_1 + 9x_2$$

$$\begin{aligned} s.t.\ & -x_1 + 3x_2 \ge 6, \\ & 7x_1 + x_2 \ge 35 \\ & x_1, x_2 \in Z^+ \end{aligned}$$

4. 用动态规划方法求解下面旅行商问题。

$$D = \begin{bmatrix} 0 & 4 & 2 & 3 \\ 7 & 0 & 3 & 5 \\ 9 & 1 & 0 & 4 \\ 5 & 6 & 2 & 0 \end{bmatrix}$$