算法第六讲——线性规划问题

乔晶整理

1 回顾

到现在为止，我们可以整理一下这门课到底讲什么。回顾一下已经学过的内容，理清思路。

我们把一个问题建模成组合优化的问题后，首先想到的是能否正面攻击它。第一个观察是，这个问题是否可以分成独立的子问题呢？如果是，基本的思路是规约（Reduction），如使用divide and Conquer 算法。如果还有最优子结构的性质，可用动态规划(dynamic programming)算法。如果进一步还有贪心选择的性质，可用贪心（Greedy）算法。当然也可以在观察到问题可以分解时，直接应用贪心算法。上节课讲过，可以用部分解的枚举树，直接做贪心算法。以上这些是我们已经学习过的内容，通过对一个问题怎样观察，观察到怎样的性质，来决定我们使用什么算法。

今天所学的内容是我们思路的一个转折点，如果我们观察到一个问题不能够分解，或者我们不愿意进行分解，这个时候怎么办呢？那就只有一种办法，逐步改进法（Improvement）。这里面包括线性规划算法（Linear programming algorithm），网络流算法（Network flow algorithm），局部搜索算法(Local Search Algorithms) 中的蒙特·卡罗方法（Monte Carlo method）LS/MC，分支限界方法(branch and bound method)，等等。

2 本章内容

我们先从实际问题讲起，然后讲这些实际问题的数学建模。这些问题包括：饮食问题，最大流问题，最小费用流，多物品流，SAT问题。接着我们考虑怎样把这些问题建模成数学的线性规划的问题，以及对于线性规划的直观认识和求解方法，包括单纯型算法，内联法，椭球法。

最后，我们介绍领域内一个新的重要的方法，平滑复杂度算法。它完美的回答了，为什么单纯型算法本身具有指数级复杂度，却往往在实际应用中可以做到在线性复杂度的时间内解决问题。

3 几个例子

3.1 第一个例子:饮食问题

问题：

Food	Energy	Protein	Calcium	Price
Oatmeal	110	4	2	3
Whole milk	160	8	285	9
Cherry pie	420	4	22	20
Pork with beans	260	14	80	19

大家考虑这样一个问题，假设市场上有四种食品，分别给出了它们所含的能量，蛋白质含量，钙离子含量，以及对应的价格。现在有一个家庭主妇，她采购一天的食物，使这些食物加起来至少要含有2000kcal 的能量，55g的蛋白质和800mg 的钙离子。请问她采用怎样的采购方案能够花费最少？

下面给出这个问题的两个解决方案：

- 方案一：10 份猪肉，花费190
- 方案二：8 份全奶+ 2 份草莓派，花费112

很显然，第二种方案比较省钱。请问，现在能不能找到一个更省钱的方案？

将这个问题形式化后，就变得简单了。我们假设买燕麦x1份，全奶x2 份，草莓派x3 份，猪肉x4 份。分别写出它们的约束条件和目标，我们的目标是花的钱最少。

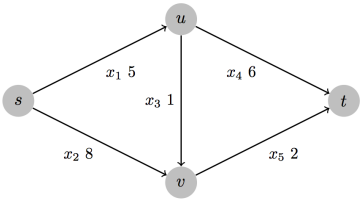
LP Formulation:

$$
\begin{array}{lrcrcrcrcl}
\min & 3x_1 & + & 9x_2 & + & 20x_3 & + & 19x_4 & & \text{money} \\
s.t. & 110x_1 & + & 160x_2 & + & 420x_3 & + & 260x_4 & \geq 2000 & \text{energy} \\
 & 4x_1 & + & 8x_2 & + & 4x_3 & + & 14x_4 & \geq 55 & \text{protein} \\
 & 2x_1 & + & 285x_2 & + & 22x_3 & + & 80x_4 & \geq 800 & \text{calcium} \\
 & x_1 & , & x_2 & , & x_3 & , & x_4 & \geq 0 &
\end{array}
$$

因为每一个约束函数以及目标函数都是线性的，所以这样的问题就是线性规划问题。

3.2 第二个例子:最大流问题

问题：



如图，假设有四个城市，s，u，v，t。连线表示，城市之间有道路。连线上的数字表示这条道路每天最多能够运送货物的吨数。比如，城市s，u之间有条路，这条路每天最多运5吨货物。

如果你是调度员，请设计一个调度方案，使得从s 到t 一天之内能够运送的货物越多越好。

现在我们形式化这个问题，将5条路的运输量分别为x1，x2，x3，x4，x5。然后写出约束条件和目标函数。

LP Formulation:

$$
\begin{array}{rrcrcrcrcrr}
\max & x_1 & + & x_2 & & & & & & & \text{output from } s \\
s.t. & x_1 & & & - & x_3 & - & x_4 & & = 0 & \text{node } u \\
 & & & x_2 & + & x_3 & & & - & x_5 = 0 & \text{node } v \\
 & & & & & & & 5 \geq & x_1 & \geq 0 & \text{edge } (s,u) \\
 & & & & & & & & \ldots & \ldots &
\end{array}
$$

注意：因为u，v是中间节点，所以必须满足运入u，v的货物必须在当天运出。另外要满足每条路的运输量不超过其最大运量。我们的目标是要从s运出的货物越多越好。

3.3 第三个例子:最小费用流问题

问题：

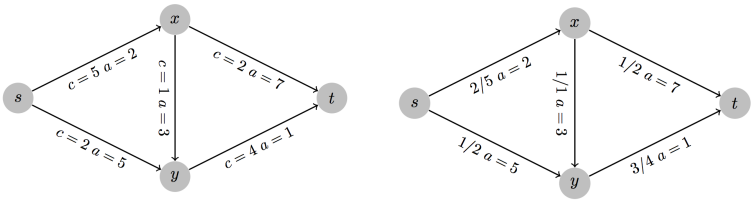


Figure 1: Cost=$2 \times 2 + 1 \times 7 + 1 \times 3 + 1 \times 5 + 3 \times 1 = 18$

这个问题比刚才那个问题多了一些条件，每条路除了有最大运量的限制，另外对每条路都给出了运输每吨货物的运价。

现在有d吨的货物要从s运到t，怎样运输能使运费最少？比如上图右边的运输方案，花费为18。

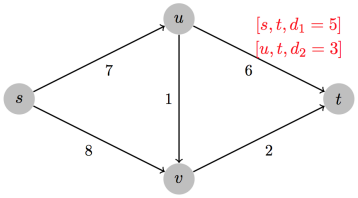
下面形式化这个问题，**LP Formulation:**

$$
\begin{array}{lrcll}
\min & \sum_{(u,v)\in E} a(u,v) f(u,v) & & & \\
s.t. & f(u,v) & \leq & C(u,v) & \text{for each } (u,v) \in E \\
 & f(u,v) & \geq & 0 & \text{for each } (u,v) \in E \\
 & \sum_{u,(u,v)\in E} f(u,v) & = & \sum_{w,(v,w)\in E} f(v,w) & \text{for each } v \in V - \{s,t\} \\
 & \sum_{v,(s,v)\in E} f(s,v) & = & d &
\end{array}
$$

注意：$a(u,v)$表示u到v之间的运输成本，$f(u,v)$表示u 到v 之间运送货物的吨数，$C(u,v)$ 表示u到v之间允许的最大运货量。

3.4 第四个例子:多物品流问题

问题：



四个城市的交通网络，要从s到t运送5吨货物d1，同时要从u到t运送3 吨货物d2。这个问题与之前不同的是，要同时运送两种货物。

LP Formulation:

$$
\begin{array}{lrcll}
\max & 0 & & & \\
s.t. & \sum_{i=1}^{k} f_i(u,v) & \leq & c(u,v) & \text{for each } (u,v) \\
& f_i(u,v) & \geq & 0 & \text{for each } i,(u,v) \\
& \sum_{u,(u,v)\in E} f_i(u,v) & = & \sum_{w,(v,w)\in E} f_i(v,w) & \text{for each } i, v \in V - \{s_i, t_i\} \\
& \sum_{v,(s_i,v)\in E} f_i(s_i,v) & = & d_i & \text{for each } i
\end{array}
$$

注意：目标函数为$max0$，这个看起来有些奇怪，其实这个问题只需要解出满足这些约束条件的解就可以了。

讲到这里我们就可以回答为什么这堂课是我们思维的一个转折点了。我们以前遇到一个问题可以分解，我们就很高兴，有很多的办法去解决，但是上面这个问题是不容易分解的。值得指出的是，线性规划是到目前为止唯一的一个多项式算法。也就是说，我们或许可以写出一个多物品的贪心算法，但它的时间复杂度不是多项式时间的。

3.5 第五个例子:多物品流问题

SAT问题值得讲，因为它是一个整数线性规划问题。

问题：

INPUT:

A set of m conjunction normal formula (CNF) clauses over n Boolean variables $x_1, x_2, ..., x_n$

OUTPUT:

Whether all clauses can be satisfied by an `TRUE/FALSE` assignment of the n variables.

- A SAT instance:

$$
\begin{array}{rcl}
\Phi & = & (x_1 \vee \neg x_2 \vee x_3) \quad \wedge \\
& & (\neg x_1 \vee x_2 \vee \neg x_3) \quad \wedge \\
& & (x_1 \vee x_2 \vee \neg x_3)
\end{array}
$$

- An assignment to make all clauses `TRUE`:

$$x_1 = \texttt{TRUE}, x_2 = \texttt{TRUE}, x_3 = \texttt{TRUE}$$

注意：每个x_i都为一个布尔变量，取值只能是TRUE或者FALSE两者之一。用∨连起来的式子叫做子式，这些子式连接成一个和取范式。能不能找

到一种组合，使得所有的子式都为TRUE？这个是可以做到的，比如我们取$x_1 = \texttt{TRUE}, x_2 = \texttt{TRUE}, x_3 = \texttt{TRUE}$, 就可以使得三个子句都为TRUE。

对于这个问题，诸位怎样求解呢？先看能不能分解？答案是肯定的，我们下堂课介绍分解的方法。我们先来看不分解的方法，把这个问题写成下面的形式：

LP Formulation:

$$
\begin{array}{rrrrl}
\max & c_1+ & c_2+ & c_3 & \\
s.t. & x_1+ & (1-x_2)+ & x_3 & \geq c_1 \\
& (1-x_1)+ & x_2+ & (1-x_3) & \geq c_2 \\
& x_1+ & x_2+ & (1-x_3) & \geq c_3 \\
& x_1, & x_2, & x_3 & = 0/1 \\
& c_1, & c_2, & c_3 & = 0/1
\end{array}
$$

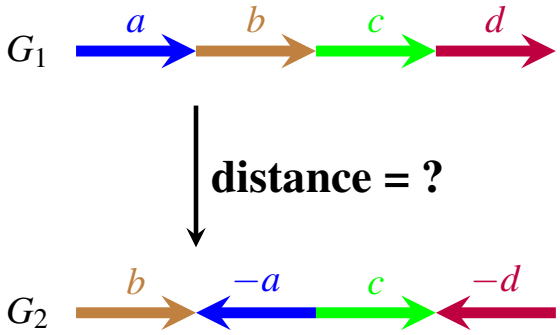
注意：第一行的$max c_1 + c_2 + c_3$表明，$c_1 c_2 c_3$均取1。所以对于这个例子来说，当且仅当$c_1 + c_2 + c_3 = 3$时才成立。

在这个问题中每个x_i只能取0或1两个值，这样的问题叫做整数线性规划。

3.6 补充例子:基因组重排问题

问题：

- The minimum number of operations to transform G_1 into G_2

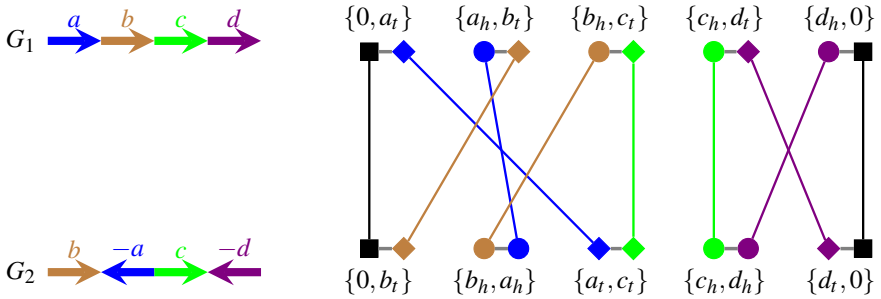


Operations: reverse a fragment of the genome;

假设人的基因组G_1上有a, b, c, d这样四个基因，老鼠的基因组G_2上a, d这两个基因是反过来的，问基因组G_1翻转几次可以得到基因组G_2？

补充： 如果诸位熟悉计算机历史的话，这个问题是否曾经见过？这是Bill Gates退学前和老师一起研究的一个翻煎饼的问题。（这里有一个Bill 的故事和翻煎饼的说明）

形式化模型： 我们给这个问题建立一个邻接图形式化的模型。

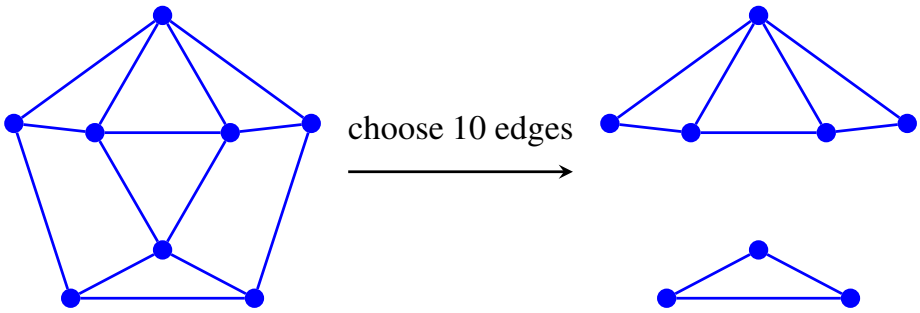


- 可以证明：DCJ distance = (#adjacencies) − (#cycles).
- 在这个例子中，DCJ distance = 3
- 所以在这个图中，因为adjacencies为固定值，所以求最小的DCJ distance的问题就转换为求图中有多少个环的问题。

问题描述：

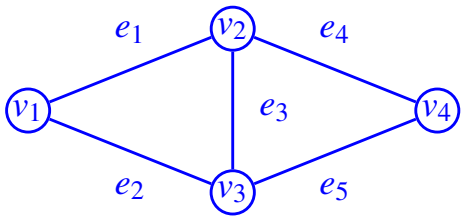
Problem: given an undirected graph $G = (V, E)$, to choose k edges and remove others, such that the number of connected components in the remaining graph is maximized.

(**Formulate this problem as an ILP**.)



假定要从一个图中选出一些边，使得连通的点越多越好。

例子： 比如我们要选出3条边，使得连通的点越多越好。



- 用x_i表示第i条边是否被选择，选则为1，不选为0。

$$x_1 + x_2 + x_3 + x_4 + x_5 = 3$$

- 用y_i来表示第i个顶点的标签。

$$1 \le y_1 \le 1$$
$$1 \le y_2 \le 2$$
$$1 \le y_3 \le 3$$
$$1 \le y_4 \le 4$$

- 如果一条边被选择，它的两个顶点需要有相同的标签。

$$y_1 \le y_2 + 1 \cdot (1 - x_1);\ y_2 \le y_1 + 2 \cdot (1 - x_1) \quad \text{(for } e_1\text{)}$$
$$y_1 \le y_3 + 1 \cdot (1 - x_2);\ y_3 \le y_1 + 3 \cdot (1 - x_2) \quad \text{(for } e_2\text{)}$$
$$y_2 \le y_3 + 2 \cdot (1 - x_3);\ y_3 \le y_2 + 3 \cdot (1 - x_3) \quad \text{(for } e_3\text{)}$$
$$y_2 \le y_4 + 2 \cdot (1 - x_4);\ y_4 \le y_2 + 4 \cdot (1 - x_4) \quad \text{(for } e_4\text{)}$$
$$y_3 \le y_4 + 3 \cdot (1 - x_5);\ y_4 \le y_3 + 4 \cdot (1 - x_5) \quad \text{(for } e_5\text{)}$$

- 同样的连通单元中的顶点有相同的标签。
- Since all vertices have distinct upper bounds, in each connected component, at most one vertex can reach its upper bound. Thus, we can use the number of vertices whose upper bound is reached, to count the number of connected components.
- ¡2-¿ We use a binary variable z_j to indicate whether the label of v_j reaches

its upper bound:

$$1 \cdot z_1 \leq y_1$$
$$2 \cdot z_2 \leq y_2$$
$$3 \cdot z_3 \leq y_3$$
$$4 \cdot z_4 \leq y_4$$

We can verify that, $z_j = 1$ only if $y_j = j$, i.e., the label of v_j reaches its upper bound.

- The objective function of the ILP formulation can be set to maximize the number of vertices whose upper bound can be reached:

$$\max z_1 + z_2 + z_3 + z_4$$

4 动态规划问题的历史回顾

看了这些问题之后，大家一定会有感觉，我们生活中的很多问题，都可以写成一个最优化的问题，不论我们想要最优化的是目标函数，还是问题的约束条件，它们都是一些线性组合。这是一类很常见的问题，我们简单看一下这类问题的历史。

4.1 问题提出：

1946年，在美国空军的指挥部，George B. Dantzig作为数学顾问，他的同事问了他一个问题关于部队部署的问题：怎样快速调度训练计划和物流。在前电子计算机的时代，人们是用穿孔卡和一些模拟的设备来制造机械式的计算机器。”program”这个词用来表示规划和调度，而不是现在所说的编程。在1947年，Dantzig 把这个问题抽象成了一个数学问题。

4.2 有关算法和分析：

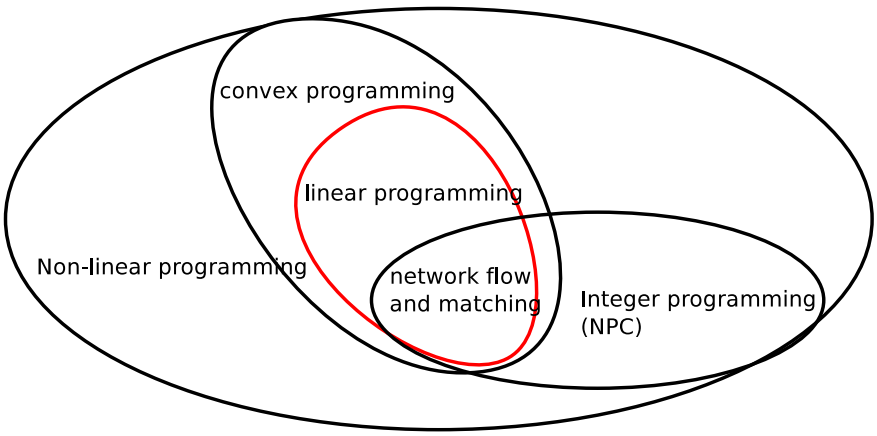
1949 年，Dantzig又提出了单纯形算法。大家回忆一下我们第一堂课就讲过的，解决一个问题的三个步骤。首先我们要确定要解决的实际的问题是什么，然后将它用公式描述为一个数学问题，最后提出算法。Dantzig 的这个故事向我们清晰的展示了这三个步骤，首先在1946年他的同事向他提出了这个问题，1947年他为这个问题建立了数学模型，然后在1949 年，他提出了解决这个问题的单纯形算法（simplex algorithm）。

人们发现单纯形算法太好了，运行的飞快，所有的问题都解决的非常好。所以人们一直以为这个算法是多项式时间复杂度的算法，然而很不幸，在1971年，Klee 和Minty 提出了第一个反例，说明单纯型算法不是多项式时间复杂度。这下大家产生了怀疑，线性规划问题应该是一个难以分解（NP hard）的问题，因为单纯型算法的成功，很多人都在用它。典型的例子是，1975 年L. V. Kantorovich 和T. C. Koopmans将线性规划的方法用于经济学领域的资源分配问题上，得到了诺贝尔经济学奖。

在1971年后，虽然人们怀疑线性规划问题是NP完全问题，但是在1979年，峰回路转，苏联科学家L. G. Khanchian提出了多项式时间复杂度的椭球算法，这是一个可以分解的问题。椭球法理论上非常漂亮，常常用于其他领域，但是实际效果却很慢，不实用。1984年，另一个苏联科学家N. Karmarkar提出了内联法，是多项式时间复杂度的算法，同时运行起来也非常快。

但是这个问题还是没有完全解决，人不不知道为什么单纯型算法理论上是指数型复杂度，但实际运行时却常常是多项式时间复杂度。直到2001 年，D. Spielman和S. Teng 提出了平滑型复杂度理论（smoothed complexity），完美地证明了这个问题。

4.3 NLP, Convex Programming, LP, Network flow, 和ILP：



我们发现生活中的很多问题都能够描述成一个优化的问题，诸位以后研究生生涯中碰到的问题可能80%的情况是优化问题，所以最外面这个大圈，就可以覆盖大家研究问题80%的情况。这里面有一类凸的规划是非常特殊的，目标函数是凸的，可行域也是凸的，这就是凸规划问题。凸规划问题里面又有一个线性规划问题，目标函数和约束都是线性组合。还有另一类问题是整数型规划

问题，我们学过的网络流和匹配这两类问题是整数型规划问题，但是它们又很特殊，网络流问题中假设$0 \le x_i \le 1$，但是线性规划问题x_i只有0或1两种取值。

4.4 求解LP问题的工具：

现在有很多求解线性规划的软件包，典型的一个包是GLPK，还有一个是Gurobi，是目前运行最快的软件包。下面给大家看两个例子：

55:25-第一部分结束，演示

5 线性规划问题的求解

前面介绍了建模的方法，下面介绍求解的过程。

线性规划问题的形式有：一般形式，标准形式和松弛形式。

5.1 一般形式线性规划问题

- General form: mixture of linear inequalities and equalities

$$\begin{array}{lrcrcccrcll} \min & c_1x_1 & + & c_2x_2 & + & ... & + & c_nx_n & & & \\ s.t. & a_{i1}x_1 & + & a_{i2}x_2 & + & ... & + & a_{in}x_n & \ge & b_i & i \in M \\ & a_{j1}x_1 & + & a_{j2}x_2 & + & ... & + & a_{jn}x_n & = & b_j & j \in \overline{M} \\ & & & & & & & x_i & \ge & 0 & i \in N \end{array}$$

5.2 标准形式线性规划问题

- Standard form: linear inequalities;

$$\begin{array}{lrcrcccrcll} \min & c_1x_1 & + & c_2x_2 & + & ... & + & c_nx_n & & & \\ s.t. & a_{11}x_1 & + & a_{12}x_2 & + & ... & + & a_{1n}x_n & \le & b_1 & \\ & a_{21}x_1 & + & a_{22}x_2 & + & ... & + & a_{2n}x_n & \le & b_2 & \\ & ... & & ... & & ... & & ... & & & \\ & a_{m1}x_1 & + & a_{m2}x_2 & + & ... & + & a_{mn}x_n & \le & b_m & \\ & & & & & & & x_i & \ge & 0 & \text{for } \forall i \end{array}$$

- Standard form in matrix language:

$$\begin{array}{lr} \min & \mathbf{c^T x} \\ s.t. & \mathbf{Ax} \le \mathbf{b} \\ & \mathbf{x} \ge \mathbf{0} \end{array}$$

- Here $\mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}$, $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$,

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

注意：在一般形式的线性规划问题中，约束条件函数可能有等式和不等式的形式，我们希望把这些约束条件统一写成标准的方程形式，比如都写成用$\leq$表示的方程，并且满足$x_i \geq 0$。我们做如下变换：

一般形式转换为标准形式：

- Transformations:

1. **Variables:** a free variable $\Rightarrow$ two non-negativeive variables;
   x_i may or may not be positive $\Rightarrow$ replacing x_i with $x_i' - x_i''$
   and adding constraints: $x_i' \geq 0; x_i'' \geq 0$

2. **Constraints:** an equality $\Rightarrow$ two inequalities;
   $a_{j1}x_1 + a_{j2}x_2 + ... + a_{jn}x_n = b_j \Rightarrow$
   $a_{j1}x_1 + a_{j2}x_2 + ... + a_{jn}x_n \geq b_j$
   $a_{j1}x_1 + a_{j2}x_2 + ... + a_{jn}x_n \leq b_j$

5.3 松弛形式线性规划问题

- Slack form: linear equality;

$$\begin{array}{lrcrcccrcl} \min & c_1x_1 & + & c_2x_2 & + & ... & + & c_nx_n & & \\ s.t. & a_{11}x_1 & + & a_{12}x_2 & + & ... & + & a_{1n}x_n & = & b_1 \\ & a_{21}x_1 & + & a_{22}x_2 & + & ... & + & a_{2n}x_n & = & b_2 \\ & ... & & ... & & ... & & ... & & \\ & a_{m1}x_1 & + & a_{m2}x_2 & + & ... & + & a_{mn}x_n & = & b_m \\ & & & & & & & x_i & \geq & 0 \quad \text{for } \forall i \end{array}$$

- Slack form in matrix language:

$$\begin{aligned} \min \quad & \mathbf{c^T x} \\ s.t. \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{aligned}$$

注意：松弛形式也是一种比较常见的形式，所有的约束条件方程都采用等式的形式。假如一开始我们写出的目标函数是$a_{j1}x_1 + a_{j2}x_2 + ... + a_{jn}x_n \leq b_j$的形式，可以引入一个整数$s$，使之变为等式形式，$s$称为松弛变量。

- Transformations:

1. **Variables:** changing "inequality on partial solution $(x_1, ..., x_n)$" to "equality on full solution $(s, x_1, ..., x_n)$" by introducing a slack variable s.
   $a_{j1}x_1 + a_{j2}x_2 + ... + a_{jn}x_n \leq b_j \Rightarrow$
   $a_{j1}x_1 + a_{j2}x_2 + ... + a_{jn}x_n + s = b_j$

2. **Constraint:** $s \geq 0$. (s is called a slack variable)

例子：标准形式VS 松弛形式

- Standard form:

$$\begin{aligned} -x_3 + 2x_4 &\leq 2 \\ 3x_3 - 2x_4 &\leq 6 \\ x_3 \ , \ x_4 &\geq 0 \end{aligned}$$

- Slack form:

$$\begin{aligned} x_1 \qquad - x_3 + 2x_4 &= 2 \\ x_2 + 3x_3 - 2x_4 &= 6 \\ x_1 \ , \ x_2 \ , \ x_3 \ , \ x_4 &\geq 0 \end{aligned}$$

注意：大家不要小看这种变换，把不等式的约束变为等式的约束，是线性规划中常常要考虑的问题。

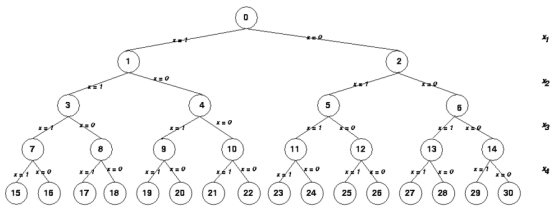
5.4 求解方法

下面我们讨论怎样求解。

$$
\begin{array}{lrrrl}
\max & c_1+ & c_2+ & c_3 & \\
s.t. & x_1+ & (1-x_2)+ & x_3 & \geq c_1 \\
& (1-x_1)+ & x_2+ & (1-x_3) & \geq c_2 \\
& x_1+ & x_2+ & (1-x_3) & \geq c_3 \\
& x_1, & x_2, & x_3 & = 0/1 \\
& c_1, & c_2, & c_3 & = 0/1
\end{array}
$$

方法一： 首先观察解的形式，$\mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}](x_i = o/1)$，一定可以画出一个部分解的枚举树，叶子节点即为所求的完整解。如下:

- Solution: $X = [x_1, x_2, ..., x_n]$, $x_i = 0/1$
- Partial solution enumeration tree:

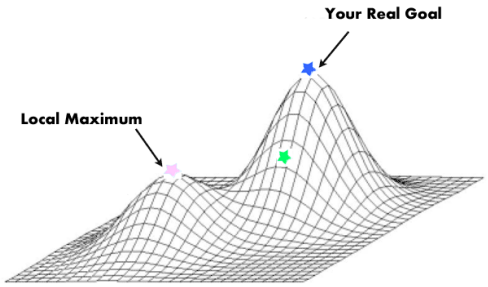


- We will talk about "intelligent enumeration", say branch-and-bound, and backtracking, later.

注意：我们的目标就是从8个完整解中找出最好的解。另外除了使用枚举的方法，贪心法，回溯法，分支限界法均可以使用。所以，我们的解题思路是，只要解的形式为$\mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}](x_i = o/1)$，就可以使用各种方法来找最优解。

扩展：如果$x_i \subset [0, 1]$，我们可以用分支限界的方法。分支限界实际上就是枚举，只不过要枚举的聪明一点，要进行大幅度的剪支，加快运行速度。这种方法会在后续学习中讨论。

方法二： 我们换一种思路，考虑完全解的情况。我们不对问题进行分类，也不对解进行分类，每次都考虑完整的解。画出解空间如下，



每个格子点都是一个完整解，如果两个格子点之间X的距离如果非常小，表示对一个解进行很小的改变就可以到达另一个解。

5.5 线性规划问题的直观认识

我们讨论第二种求解方法，每次都考虑完整解的情况。考察一个线性规划如下，

$$\begin{aligned} \min \quad & \mathbf{c^T x} \\ s.t. \quad & \mathbf{Ax = b} \\ & \mathbf{x \geq 0} \end{aligned}$$

大家观察一下会发现，我们已经学过如何求解$\mathbf{Ax = b}$，但没有学过优化$\mathbf{c^T x}$，另外还要考虑$\mathbf{x \geq 0}$这个约束条件。下面我们分别来讨论。

5.6 约束条件$\mathbf{x \geq 0}$的影响：

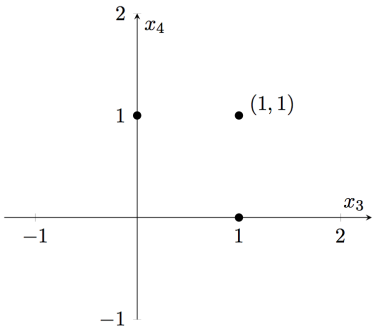
5.6.1 回顾线性方程$\mathbf{Ax = b}$ 的求解：

$$\begin{aligned} x_1 \quad\quad - \quad x_3 \quad + \quad 2x_4 \quad = \quad 2 \\ x_2 \quad + \quad 3x_3 \quad - \quad 2x_4 \quad = \quad 6 \\ 2x_1 \quad + \quad x_2 \quad + \quad x_3 \quad + \quad 2x_4 \quad = \quad 10 \end{aligned}$$

通过高斯消元法，我们得到：

$$\begin{aligned} x_1 \quad\quad - \quad x_3 \quad + \quad 2x_4 \quad = \quad 2 \\ x_2 \quad + \quad 3x_3 \quad - \quad 2x_4 \quad = \quad 6 \end{aligned}$$

4个未知量，两个方程，我们只能确定两个独立变量，得到的解平面上的任意一个点都是这个方程组的完整解。如下图：



5.6.2 约束条件$\mathbf{x} \geq \mathbf{0}$ 的影响：

下面我们考虑对这个线性方程加一个约束条件，

- 例如：$\mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}$

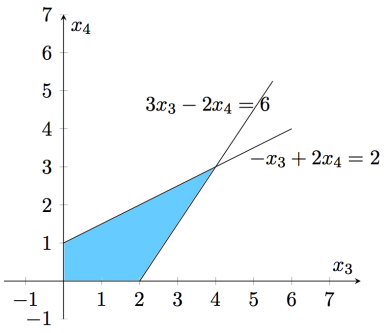
$$\begin{array}{rcrcrcrcl} x_1 & & & - & x_3 & + & 2x_4 & = & 2 \\ & & x_2 & + & 3x_3 & - & 2x_4 & = & 6 \\ 2x_1 & + & x_2 & + & x_3 & + & 2x_4 & = & 10 \\ x_1 & , & x_2 & , & x_3 & , & x_4 & \geq & 0 \end{array}$$

- 通过高斯消元法，我们得到：

$$\begin{array}{rcrcrcrcl} x_1 & & & - & x_3 & + & 2x_4 & = & 2 \\ & & x_2 & + & 3x_3 & - & 2x_4 & = & 6 \\ x_1 & , & x_2 & , & x_3 & , & x_4 & \geq & 0 \end{array}$$

- 得到一个线性不等式组：

$$\begin{array}{rcrcrcl} & & - & x_3 & + & 2x_4 & \leq & 2 \\ & & & 3x_3 & - & 2x_4 & \leq & 6 \\ & & & x_3 & , & x_4 & \geq & 0 \end{array}$$



对于这个线性不等式组，分别画出$x_3+2x_4=2$和$3x_3-2x_4=6$之后，就可以确定出它的解域如上图的阴影区域。与刚才的平面解域不同，这个解域为多胞形。

5.6.3 多胞形⇔ 可行域

定理： **Any polytope** $\mathbf{P}\subset\mathbf{R}^{n-m}$ corresponds to the feasible region of a linear program $\mathbf{Ax}=\mathbf{b},\mathbf{x}\geq\mathbf{0}$ (denoted as $F=\{\mathbf{x}:\mathbf{Ax}=\mathbf{b},\mathbf{x}\geq\mathbf{0}\}$), and vice versa.

在空间$\mathbf{R}^{n-m}$中，任何一个多胞形都对应着$\mathbf{Ax}=\mathbf{b},\mathbf{x}\geq\mathbf{0}$的一个可行域$F=\{\mathbf{x}:\mathbf{Ax}=\mathbf{b},\mathbf{x}\geq\mathbf{0}\}$)，反之亦然然。

也就是说，等式约束条件：$x_2+3x_3-2x_4$=6，等价为不等式约束条件$3x_3-2x_4\leq$6。

下面我们证明一下这个定理。

证明：

注意：变量数为n，约束条件有m个。

Proof: feasible region ⇒ polytope：

- Consider a **feasible full solution x** of the following LP:

$$\begin{array}{rcrcrcrcl}
a_{11}x_1 & + & a_{12}x_2 & + & \ldots & + & a_{1n}x_n & = & b_1\\
a_{21}x_1 & + & a_{22}x_2 & + & \ldots & + & a_{2n}x_n & = & b_2\\
 & & & & \ldots & & & & \\
a_{m1}x_1 & + & a_{m2}x_2 & + & \ldots & + & a_{mn}x_n & = & b_m\\
x_1 & , & x_2 & , & \ldots & , & x_n & \geq & 0
\end{array}$$

- 一定可以使用高斯消元法化简为:

$$\begin{array}{rcrcrcrcrcl}
x_1 & & & & + & a'_{1,m+1}x_{m+1} & + & \ldots & + & a'_{1n}x_n & = & b'_1\\
 & x_2 & & & + & a'_{2,m+1}x_{m+1} & + & \ldots & + & a'_{2n}x_n & = & b'_2\\
 & \ldots & & & & \ldots & & & & & & \\
 & & x_m & + & a'_{m,m+1}x_{m+1} & + & \ldots & + & a'_{mn}x_n & = & b'_m\\
x_1 & , x_2 , & x_m & , & x_{m+1} & , & \ldots & , & x_n & \geq & 0
\end{array}$$

- By **removing positive variables** $x_1,x_2,\ldots,x_m$, we have the following

linear inequalities:

$$\begin{array}{rcccccl} a'_{1,m+1}x_{m+1} & + & ... & + & a'_{1n}x_n & \leq & b'_1 \\ a'_{2,m+1}x_{m+1} & + & ... & + & a'_{2n}x_n & \leq & b'_2 \\ & & ... & & & & \\ a'_{m,m+1}x_{m+1} & + & ... & + & a'_{mn}x_n & \leq & b'_m \\ x_{m+1} & , & ... & , & x_n & \geq & 0 \end{array}$$

- Define a polytope $\mathbf{P} \subset \mathbf{R}^{n-m}$ as the intersection of m half-spaces:

  $HS_j : a'_{j,m+1}x_{m+1} + ... + a'_{jn}x_n \leq b'_j$, $1 \leq j \leq m$. (by $x_j \geq 0$)

- Thus, **any feasible full solution** $\mathbf{x} = (x_1, x_2, ..., x_n) \Rightarrow$ **partial solution** $\mathbf{x_N} = (x_{m+1}, ..., x_n) \in \mathbf{P}$.

Proof: polytope ⇒ feasible region：

注意：反过来，我们加入一个松弛变量s_j，将一个多胞形变成了$AX = B$的形式。

- Basic idea: changing **inequality** to **equality** through introducing **slack variables.**

  – Suppose P is the intersection of m half-spaces (inequalities), say:
  $HS_j : a_{j1}x_1 + a_{j2}x_2 + ... + a_{jn}x_n \leq b_j$ $(1 \leq j \leq m)$

  – Introducing a non-negative slack variable s_j to each inequality, we have:
  $a_{j1}x_1 + a_{j2}x_2 + ... + a_{jn}x_n + s_j = b_j$ $(s_j \geq 0)$

- Thus we change

$$\begin{array}{rcccccl} a_{11}x_1 & + & a_{12}x_2 & + & ... & + & a_{1n}x_n & \leq & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & ... & + & a_{2n}x_n & \leq & b_2 \\ & & & & ... & & & & \\ a_{m1}x_1 & + & a_{m2}x_2 & + & ... & + & a_{mn}x_n & \leq & b_m \\ x_1 & , & x_2 & , & ... & , & x_n & \geq & 0 \end{array}$$

into

$$
\begin{array}{cccccccccccc}
s_1 & & & + & a_{1,1}x_1 & + & ... & + & a_{1n}x_n & = & b_1 \\
 & s_2 & & + & a_{2,1}x_1 & + & ... & + & a_{2n}x_n & = & b_2 \\
 & ... & & & ... & & & & & & \\
 & & s_m & + & a_{m,1}x_1 & + & ... & + & a_{mn}x_n & = & b_m \\
s_1 & , & s_2 & , & s_m & , & x_1 & , & ... & , & x_n & \geq & 0
\end{array}
$$

- Thus, a **partial solution** $(x_1, x_2, ..., x_n) \in \mathbf{P} \Rightarrow$ a **feasible full solution** $(s_1, s_2, ..., s_m, x_1, x_2, ..., x_n) \geq 0$.

我们引入一些记号、常用的称谓。

记号：

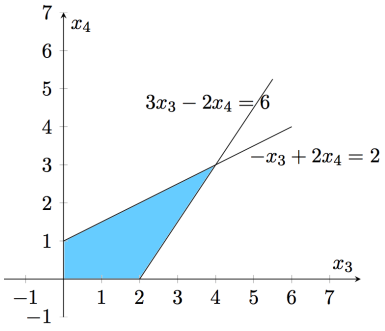
- 超平面: $\{\mathbf{x} : a_1x_1 + a_2x_2 + ... + a_nx_n = b\}$ (线性等式约束)
- 半空间: $\{\mathbf{x} : a_1x_1 + a_2x_2 + ... + a_nx_n \leq b\}$ (线性不等式约束)
- 多面体: 一些半空间的组合;
- 多胞形: 有界非空的多面体;

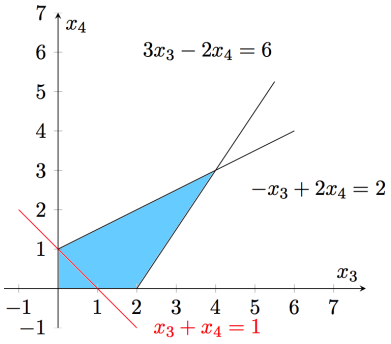
5.7 约束条件$\min \mathbf{c^T x}$的影响：

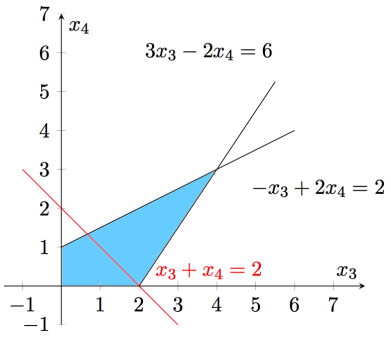
我们来看第二个差异，我们想要求下面这个问题，

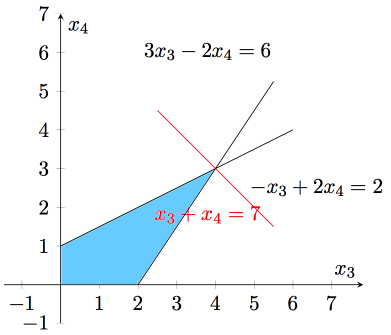
$$
\begin{array}{cccccc}
\max & x_3 & + & x_4 & & \\
s.t. & -x_3 & + & 2x_4 & \leq & 2 \\
 & 3x_3 & - & 2x_4 & \leq & 6 \\
 & x_3 & , & x_4 & \geq & 0
\end{array}
$$

对这个不等式，我们已经不怕它了，只要添加变量，就可以把不等式变为等式。画出它的可行域如下：









在可行域的所有点中，我们要挑一个满足约束，并使得$max x_3 + x_4$的点。从上面的图中，可以很直观的看出，只要找顶点就可以了，不用考虑内部点。

6 改进法求线性规划问题

经过上面这些准备工作之后，接下来我们来看怎样正式的求解线性规划。

改进策略就三句话：

第一步，随便找一个初始的解；第二步，从这个初始点开始往一个方向稍微改进一点；第三步，判断改进是否足够好，若足够好，则返回。

大家注意，如果大家在学运筹学，或者数值计算方法的话，那些里面的方法全都是这个套路。比如牛顿法，梯度下降法等等，都不考虑是否可分的问题，都是通过这样一种改进初始解的方法做。

我们过去的思路是，拿到一个问题，我们先琢磨这个问题好不好分。现在拿到这个问题，我们先不考虑怎么分（讲完线性规划后会讲怎么分）。

下面这个例子说明这个套路。

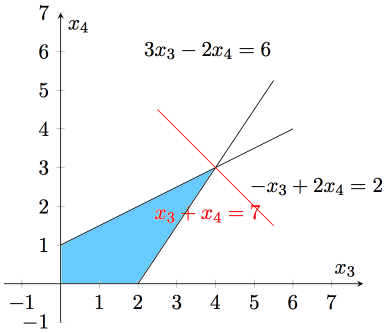
6.1 例子：

3. 怎样改进？

4. 什么时候结束？

我们来一一回答。

6.2 为什么我们只用考虑多胞形的顶点就够了呢？

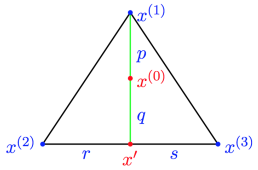


直观地看，我们发现最优解一定在顶点上，不需要考虑内部点，问题得到了极大的简化。

定理： There exists a vertex in $\mathbf{P}$ that takes the optimal value.

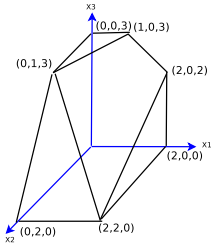
证明：

- Since $\mathbf{P}$ is a bounded close set, $\mathbf{c^T x}$ reaches its optimum in $\mathbf{P}$.
- Denote the optimal solution as $\mathbf{x^{(0)}}$. We will show there is a vertex at least as good as $\mathbf{x^{(0)}}$. Why?
  - $\mathbf{x^{(0)}}$ can be represented as the convex combination of vertices of $\mathbf{P}$, i.e. $\mathbf{x^{(0)}} = \lambda_1 \mathbf{x^{(1)}} + \lambda_2 \mathbf{x^{(2)}} + ... + \lambda_k \mathbf{x^{(k)}}$, where $\lambda_i \geq 0, \lambda_1 + ... + \lambda_k = 1$. (See Appendix for details.)
  - Thus $\mathbf{c^T x}^{(0)} = \lambda_1 \mathbf{c^T x}^{(1)} + \lambda_2 \mathbf{c^T x}^{(2)} + ... + \lambda_k \mathbf{c^T x}^{(k)}$
  - Let $x^{(i)}$ be the vertex with the minimal objective value $\mathbf{c^T x}^{(i)}$;
  - $\mathbf{c^T x}^{(0)} = \lambda_1 \mathbf{c^T x}^{(1)} + \lambda_2 \mathbf{c^T x}^{(2)} + ... + \lambda_k \mathbf{c^T x}^{(k)} \geq \mathbf{c^T x}^{(i)}$.
- Thus, vertex $\mathbf{x}^{(i)}$ is also an optimal solution since $\mathbf{c^T x}^{(i)} \leq \mathbf{c^T x}^{(0)}$



- Suppose $\mathbf{x}^{(0)}$ is an optimal solution.
- Connecting $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(1)}$ with a line. Suppose the line intersects line segment $(\mathbf{x}^{(2)}, \mathbf{x}^{(3)})$ at point $\mathbf{x}'$.
- We have $\mathbf{x}^{(0)} = \lambda_1 \mathbf{x}^{(1)} + (1-\lambda_1)\mathbf{x}'$, where $\lambda_1 = \frac{p}{p+q}$.
- We also have $\mathbf{x}' = \lambda_2 \mathbf{x}^{(2)} + (1-\lambda_2)\mathbf{x}^{(3)}$, where $\lambda_2 = \frac{r}{r+s}$.
- Thus, we have $\mathbf{x}^{(0)} = \lambda_1 \mathbf{x}^{(1)} + (1-\lambda_1)\lambda_2 \mathbf{x}^{(2)} + (1-\lambda_1)(1-\lambda_2)\mathbf{x}^{(3)}$.
- Suppose $\mathbf{c^T x^{(1)}}$ is the minimum of $\mathbf{c^T x^{(1)}}, \mathbf{c^T x^{(2)}}, \mathbf{c^T x^{(3)}}$.
- Notice that $\lambda_1 + (1-\lambda_1)\lambda_2 + (1-\lambda_1)(1-\lambda_2) = 1$.
- We have: $\mathbf{c^T x^{(1)}} \leq \mathbf{c^T x^{(0)}}$. Thus, a vertex $\mathbf{x}^{(1)}$ is found not worse than $\mathbf{x}^{(0)}$.

6.3 怎样选择初始点？



考虑一个三维的多胞体，我们已经知道，对于我们所要求解的问题，只用考虑顶点就够了，内部点不用考虑。

我们人通过看图很容易得到这些顶点，那么，怎样写程序找到多胞体的顶点？

定理： A vertex of $\mathbf{P}$ corresponds to a basis of matrix $\mathbf{A}$.

一个多边形的顶点就对应着一个矩阵的基，这下就好做了，只要给一个矩阵求基就行了。

例子：

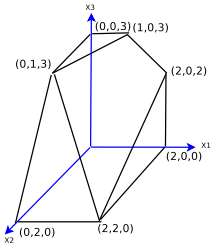
$$
\begin{array}{lrcrcrcl}
\min & -x_1 & - & 14x_2 & - & 6x_3 & & \\
s.t. & x_1 & + & x_2 & + & x_3 & \leq & 4 \\
 & x_1 & & & & & \leq & 2 \\
 & & & & & x_3 & \leq & 3 \\
 & & & 3x_2 & + & x_3 & \leq & 6 \\
 & x_1 & , & x_2 & , & x_3 & \geq & 0
\end{array}
$$

证明：

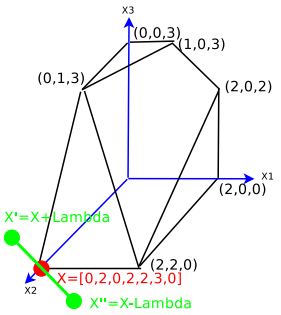
Part 1: Vertex ⇒ basic feasible solution

- We will first show that **any vertex** of the polytope corresponds to a basis of the matrix $\mathbf{A}$.
- An example (slack form):

$$
\begin{array}{lrcrcrcrcrcrcrcl}
\min & -x_1 & - & 14x_2 & - & 6x_3 & & & & & & & & & \\
s.t. & x_1 & + & x_2 & + & x_3 & + & x_4 & & & & & & = & 4 \\
 & x_1 & & & & & & & + & x_5 & & & & = & 2 \\
 & & & & & x_3 & & & & & + & x_6 & & = & 3 \\
 & & & 3x_2 & + & x_3 & & & & & & & + & x_7 & = & 6 \\
 & x_1 & , & x_2 & , & x_3 & , & x_4 & , & x_5 & , & x_6 & , & x_7 & \geq & 0
\end{array}
$$



矩阵描述：



$$
\mathbf{x} = \begin{bmatrix} 0 & 2 & 0 & 2 & 2 & 3 & 0 \end{bmatrix}^T
$$

$$
\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}
$$

补充：关于矩阵的描述，推荐大家看一本书，*Linear Algebra and Its Applications*（*David C. Lay*）。

- Take the vertex $(x_1, x_2, x_3) = (0, 2, 0)$ as an example. The corresponding full solution is $(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (0, 2, 0, 2, 2, 3, 0)$.
- We will show that the column vectors corresponding to **non-zero** x_i, i.e. $\{\mathbf{a_2}, \mathbf{a_4}, \mathbf{a_5}, \mathbf{a_6}\}$, are linearly independent.

- Suppose $\exists(\lambda_2, \lambda_4, \lambda_5, \lambda_6) \neq 0$ such that $\lambda_2\mathbf{a_2} + \lambda_4\mathbf{a_4} + \lambda_5\mathbf{a_5} + \lambda_6\mathbf{a_6} = 0$, i.e. $\mathbf{A}\lambda = \mathbf{0}$, where $\lambda = [0, \lambda_2, 0, \lambda_4, \lambda_5, \lambda_6, 0]$.
- Then we can construct **two other points: $\mathbf{x}' = \mathbf{x} + \theta\lambda$ and $\mathbf{x}'' = \mathbf{x} - \theta\lambda$.**
- It is easy to deduce that both $\mathbf{x}'$ and $\mathbf{x}''$ lie inside P since:
  - $\mathbf{Ax}' = \mathbf{Ax} + \mathbf{0} = \mathbf{b}$ and $\mathbf{Ax}'' = \mathbf{Ax} - \mathbf{0} = \mathbf{b}$
  - In addition, we can guarantee $\mathbf{x}' \geq \mathbf{0}$ and $\mathbf{x}'' \geq \mathbf{0}$ via setting θ to be sufficiently small since $\mathbf{x} \geq \mathbf{0}$, and $\lambda_1 = \lambda_3 = \lambda 7 = 0$.
- Contradiction: it is impossible for a vertex to be center of two inner points of $\mathbf{P}$.

现在我们已经知道顶点可以表示成基，我们来把过去的东西用矩阵的语言表示。对于任何一个顶点，都对应一个基B，我们把非基的列都叫做N。于是有，

- For a vertex $\mathbf{x}$ of the polytope, a basis $\mathbf{B}$ can be derived via extracting the column vectors corresponding to non-zero x_i. The non-basis column vectors are denoted as $\mathbf{N}$.
- Then the original LP can be represented as:
- Here, $\mathbf{x}$ is decomposed as $\mathbf{x} = \begin{bmatrix} \mathbf{x_B} \\ \mathbf{x_N} \end{bmatrix}$. Then we have $\mathbf{x_N} = \mathbf{0}$, and $\mathbf{x_B} = \mathbf{B^{-1}b}$ (Reason: $\mathbf{Ax} = \mathbf{b}$, i.e. $\mathbf{Bx_B} + \mathbf{Nx_N} = \mathbf{b}$)
- The corresponding objective value is $\mathbf{c^T x} = \mathbf{c_B^T x_B} + \mathbf{c_N^T x_N} = \mathbf{c_B^T B^{-1} b}$.

一个例子：

- For a vertex $\mathbf{x} = \begin{bmatrix} 0 & 2 & 0 & 2 & 2 & 3 & 0 \end{bmatrix}^T$, the columns corresponding to non-zero x_i are extracted to form a basis $\mathbf{B} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 3 & 0 & 0 & 0 \end{bmatrix}$.
- Let's decompose $\mathbf{x} = \begin{bmatrix} 0 & 2 & 0 & 2 & 2 & 3 & 0 \end{bmatrix}^T$ accordingly into $\mathbf{x_B} = [2\ 2\ 2\ 3]^T$ and $\mathbf{x_N} = [0\ 0\ 0]^T$.

- It is easy to verify that $\mathbf{x_B} = \mathbf{B^{-1}b}$. In this example, $\mathbf{b} = \begin{bmatrix} 4 \\ 2 \\ 3 \\ 6 \end{bmatrix}$.

Part 2: Basic feasible solution ⇒ vertex

- Given a basis $\mathbf{B}$ of matrix $\mathbf{A}$, we call $\mathbf{x} = \begin{bmatrix} \mathbf{B^{-1}b} \\ \mathbf{0} \end{bmatrix}$ a **basic solution respect to B**.
- If we further have $\mathbf{x_B} = \mathbf{B^{-1}b} \geq \mathbf{0}$, $\mathbf{x}$ is called a **basic feasible solution respect to B**.
- We will show that a **basic feasible solution x respect to B** is a vertex of the polytope $\mathbf{P}$.

证明：

- It suffices to show that $\mathbf{x}$ cannot be represented as a convex combination of any two points in $\mathbf{P}$.
- By contradiction, suppose there are two different points $\mathbf{x^{(1)}}$ and $\mathbf{x^{(2)}}$ in $\mathbf{P}$ such that $\mathbf{x} = \lambda_1\mathbf{x^{(1)}} + \lambda_2\mathbf{x^{(2)}}$, where $0 < \lambda_1, \lambda_2 < 1$.
- Note that $\lambda_1\mathbf{x_N^{(1)}} + \lambda_2\mathbf{x_N^{(2)}} = \mathbf{x_N} = 0$.
- So $\mathbf{x_N^{(1)}} = \mathbf{x_N^{(2)}} = 0$.
- We have $\mathbf{x_B^{(1)}} = \mathbf{x_B^{(2)}} = \mathbf{B^{-1}b} = \mathbf{x_B}$ (by $\mathbf{Ax^{(1)}} = \mathbf{0}$ and $\mathbf{Ax^{(2)}} = \mathbf{0}$). A contradiction.

一个例子：

- For matrix $\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$, and $\mathbf{b} = \begin{bmatrix} 4 \\ 2 \\ 3 \\ 6 \end{bmatrix}$, we first calculate a basis of $\mathbf{A}$ as $\mathbf{B} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 3 & 0 & 0 & 0 \end{bmatrix}$.

- The **basic feasible solution x respect to B** is $\mathbf{x} = \begin{bmatrix} \mathbf{B^{-1}b} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} 0 & 2 & 0 & 2 & 2 & 3 & 0 \end{bmatrix}^T$.
- It is easy to verify that $(x_1, x_2, x_3) = (0, 2, 0)$ is a vertex of the polytope $\mathbf{P}$.

6.4 结束条件是什么?

下节课内容