浅谈线性规划在信息学竞赛中的应用

邹逍遥

镇海中学

April 29, 2016

主要内容

论文共分为五章。

- 第一章介绍了什么是线性规划
- 第二章介绍了单纯形算法的实现
- 第三章介绍了如何用线性规划解决一类网络流问题
- 第四章介绍了如何用线性规划在一类dp问题上获得很高的正确率
- 第五章介绍了如何用线性规划求解纳什均衡

由于时间关系这里只介绍前三章。

什么是线性规划

实际生活中有很多问题都是这样的形式：它们需要最大化或者最小化一个目标；它们通常面临资源，时间等多方面的限制。假如把这些问题的目标简化成一个线性的函数，把限制表示成一些线性的等式或者不等式，那么这些问题就可以被描述成线性规划问题。

什么是线性规划

实际生活中有很多问题都是这样的形式：它们需要最大化或者最小化一个目标；它们通常面临资源，时间等多方面的限制。假如把这些问题的目标简化成一个线性的函数，把限制表示成一些线性的等式或者不等式，那么这些问题就可以被描述成线性规划问题。

正式地说，线性规划是一个需要最小化或最大化一个受限制于有限个数的约束的线性函数的问题。

什么是线性规划

实际生活中有很多问题都是这样的形式：它们需要最大化或者最小化一个目标；它们通常面临资源，时间等多方面的限制。假如把这些问题的目标简化成一个线性的函数，把限制表示成一些线性的等式或者不等式，那么这些问题就可以被描述成线性规划问题。

正式地说，线性规划是一个需要最小化或最大化一个受限制于有限个数的约束的线性函数的问题。

整数规划是线性规划的特殊形式，即要求最后的解中每个变量都是整数的线性规划。

标准型与松弛型

常用的表示线性规划的规范形式有两种：**标准型与松弛型**

标准型与松弛型

常用的表示线性规划的规范形式有两种：**标准型与松弛型**
我们一般将标准型表示成这种形式：
最大化$\mathbf{c}^{\mathrm{T}}\mathbf{x}$，满足约束

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}$$

$$x_i \geq 0$$

标准型与松弛型

常用的表示线性规划的规范形式有两种：**标准型与松弛型**
我们一般将标准型表示成这种形式：
最大化$\mathbf{c}^{\mathrm{T}}\mathbf{x}$，满足约束

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}$$
$$x_i \geq 0$$

松弛型则是这样：
最大化$\mathbf{c}^{\mathrm{T}}\mathbf{x}$，满足约束

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$
$$x_i \geq 0$$

标准型与松弛型

常用的表示线性规划的规范形式有两种：**标准型**与**松弛型**
我们一般将标准型表示成这种形式：
最大化$\mathbf{c}^{\mathrm{T}}\mathbf{x}$，满足约束

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}$$
$$x_i \geq 0$$

松弛型则是这样：
最大化$\mathbf{c}^{\mathrm{T}}\mathbf{x}$，满足约束

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$
$$x_i \geq 0$$

容易看出任何一个线性规划都能写成标准型或松弛型的形式。

单纯形算法

以下使用m表示约束个数，n表示变量个数。

单纯形算法

以下使用m表示约束个数，n表示变量个数。

假如将每一种变量的取值对应一个n维空间中的点，那么每一个不等式对应一个$n-1$维超平面，合法解的集合就对应一个n维的凸多胞体。容易发现最优解一定在顶点上。

单纯形算法

以下使用m表示约束个数，n表示变量个数。

假如将每一种变量的取值对应一个n维空间中的点，那么每一个不等式对应一个$n-1$维超平面，合法解的集合就对应一个n维的凸多胞体。容易发现最优解一定在顶点上。

单纯形算法的本质就是沿着凸多胞体的边不停地在凸多胞体的顶点之间移动。

单纯形算法

以下使用m表示约束个数，n表示变量个数。

假如将每一种变量的取值对应一个n维空间中的点，那么每一个不等式对应一个$n-1$维超平面，合法解的集合就对应一个n维的凸多胞体。容易发现最优解一定在顶点上。

单纯形算法的本质就是沿着凸多胞体的边不停地在凸多胞体的顶点之间移动。

在代码实现上，就是把一个松弛型线性规划不停地改写成等价的松弛型。每次改写需要重写所有式子，所以复杂度是$O(mn)$，总复杂度就是$O(\text{改写次数} \times mn)$。

单纯形算法

以下使用m表示约束个数，n表示变量个数。

假如将每一种变量的取值对应一个n维空间中的点，那么每一个不等式对应一个$n-1$维超平面，合法解的集合就对应一个n维的凸多胞体。容易发现最优解一定在顶点上。

单纯形算法的本质就是沿着凸多胞体的边不停地在凸多胞体的顶点之间移动。

在代码实现上，就是把一个松弛型线性规划不停地改写成等价的松弛型。每次改写需要重写所有式子，所以复杂度是$O(mn)$，总复杂度就是$O(\text{改写次数} \times mn)$。

代码实现非常简单，甚至比常用的网络流算法还要好写一点。

单纯形算法

具体来说，单纯形算法可以分为以下几个步骤：

单纯形算法

具体来说，单纯形算法可以分为以下几个步骤：

- 将输入的线性规划转为松弛型

单纯形算法

具体来说，单纯形算法可以分为以下几个步骤：

- 将输入的线性规划转为松弛型
- 执行初始化过程找到一个初始解，找不到返回无解

单纯形算法

具体来说，单纯形算法可以分为以下几个步骤：

- 将输入的线性规划转为松弛型
- 执行初始化过程找到一个初始解，找不到返回无解
- 执行最优化过程找到最优解或者返回无界

单纯形算法

具体来说，单纯形算法可以分为以下几个步骤：

- 将输入的线性规划转为松弛型
- 执行初始化过程找到一个初始解，找不到返回无解
- 执行最优化过程找到最优解或者返回无界

假如n个变量初始均为0是一组满足条件的解，那么可以省去初始化过程。

时间复杂度

由于凸多胞体上可能存在一条很长的路径，所以在最坏情况下，它的时间复杂度为指数级。

时间复杂度

由于凸多胞体上可能存在一条很长的路径，所以在最坏情况下，它的时间复杂度为指数级。

但是想要构造这样的一个凸多胞体需要指数级大小的权值，这在算法竞赛中就没有应用价值了。在权值大小限定为特殊情况（比如10^9范围内的整数）时单纯形算法会分析出一个更低的多项式复杂度。

时间复杂度

由于凸多胞体上可能存在一条很长的路径，所以在最坏情况下，它的时间复杂度为指数级。

但是想要构造这样的一个凸多胞体需要指数级大小的权值，这在算法竞赛中就没有应用价值了。在权值大小限定为特殊情况（比如10^9范围内的整数）时单纯形算法会分析出一个更低的多项式复杂度。

对于一般的随机数据，单纯形算法运行速度非常快，期望只需要$O(m)$次改写，所以复杂度约为$O(nm^2)$。

全幺模矩阵

全幺模矩阵（totally unimodular matrix）是指任何一个行数和列数相等的满秩子矩阵行列式的值都是1或-1的矩阵（但它自己的行列数不需要相等）。

全幺模矩阵

全幺模矩阵（totally unimodular matrix）是指任何一个行数和列数相等的满秩子矩阵行列式的值都是1或-1的矩阵（但它自己的行列数不需要相等）。

全幺模矩阵有一个非常重要的性质：假如一个线性规划$\{max\ \mathbf{c}^{\mathrm{T}}\mathbf{x}|\mathbf{Ax} \leq \mathbf{b}, x_i \geq 0\}$中的$\mathbf{A}$为全幺模矩阵，那么可以证明在执行单纯形的过程中涉及到的所有约束中的系数的值都是1,0,-1中的一个。

全幺模矩阵

全幺模矩阵（totally unimodular matrix）是指任何一个行数和列数相等的满秩子矩阵行列式的值都是1或-1的矩阵（但它自己的行列数不需要相等）。

全幺模矩阵有一个非常重要的性质：假如一个线性规划$\{max\ \mathbf{c}^{\mathrm{T}}\mathbf{x}|\mathbf{Ax} \leq \mathbf{b}, x_i \geq 0\}$中的$\mathbf{A}$为全幺模矩阵，那么可以证明在执行单纯形的过程中涉及到的所有约束中的系数的值都是1,0,-1中的一个。

也就是说，每个变量限制为实数和限制为0或1解出来得到的答案是一样的，所以直接执行线性规划就能得到相同的整数规划问题的正确答案。

全幺模矩阵

最大流问题和最小费用最大流问题都可以写成全幺模线性规划的形式。

全幺模矩阵

最大流问题和最小费用最大流问题都可以写成全幺模线性规划的形式。

有一些网络流题本来没有比较显然的建图，需要通过先列出整数规划再经过转对偶、差分等转化最后将整数规划问题转化成网络流解决。

全幺模矩阵

最大流问题和最小费用最大流问题都可以写成全幺模线性规划的形式。

有一些网络流题本来没有比较显然的建图，需要通过先列出整数规划再经过转对偶、差分等转化最后将整数规划问题转化成网络流解决。

但是能够使用这种方法写成网络流问题的，写成整数规划后矩阵都是全幺模的。这时候线性规划就是解决这类问题的一种通用的解法，不需要再对问题模型进行深入的分析。

全幺模矩阵

最大流问题和最小费用最大流问题都可以写成全幺模线性规划的形式。

有一些网络流题本来没有比较显然的建图，需要通过先列出整数规划再经过转对偶、差分等转化最后将整数规划问题转化成网络流解决。

但是能够使用这种方法写成网络流问题的，写成整数规划后矩阵都是全幺模的。这时候线性规划就是解决这类问题的一种通用的解法，不需要再对问题模型进行深入的分析。

同时由于这种矩阵的特殊性，可以加上一些常数优化提升几十倍的速度：可以直接使用整数存储系数；改写的式子的系数一定是1，所以可以省去除法那一步；0在矩阵中占的比重很大，可以用链表优化去除那些值为0的项进行常数优化从而大大提升速度。常数优化之后运行速度就和其他费用流算法效率差不多了。

线性规划对偶性

线性规划对偶性（linear-programming duality）是指对于任何一个线性规划$\{\max\ \mathbf{c}^{\mathrm{T}}\mathbf{x}|\mathbf{A}\mathbf{x} \leq \mathbf{b}, x_i \geq 0\}$，它的最优解与$\{\min\ \mathbf{b}^{\mathrm{T}}\mathbf{x}|\mathbf{A}^{\mathrm{T}}\mathbf{x} \leq \mathbf{c}, x_i \geq 0\}$的最优解相同。

线性规划对偶性

线性规划对偶性（linear-programming duality）是指对于任何一个线性规划$\{\max\ \mathbf{c}^{\mathrm{T}}\mathbf{x}|\mathbf{A}\mathbf{x} \leq \mathbf{b}, x_i \geq 0\}$，它的最优解与$\{\min\ \mathbf{b}^{\mathrm{T}}\mathbf{x}|\mathbf{A}^{\mathrm{T}}\mathbf{x} \leq \mathbf{c}, x_i \geq 0\}$的最优解相同。

将最大流问题和最小割问题分别使用线性规划表示出来以后它们就互为对偶问题，所以最大流最小割定理可以使用线性规划对偶性证明。

线性规划对偶性

线性规划对偶性（linear-programming duality）是指对于任何一个线性规划$\{\max\ \mathbf{c}^{\mathrm{T}}\mathbf{x}|\mathbf{A}\mathbf{x} \leq \mathbf{b}, x_i \geq 0\}$，它的最优解与$\{\min\ \mathbf{b}^{\mathrm{T}}\mathbf{x}|\mathbf{A}^{\mathrm{T}}\mathbf{x} \leq \mathbf{c}, x_i \geq 0\}$的最优解相同。

将最大流问题和最小割问题分别使用线性规划表示出来以后它们就互为对偶问题，所以最大流最小割定理可以使用线性规划对偶性证明。

在大部分的题目中，原问题和对偶问题中通常会有一个能直接找到初始解不需要写初始化从而降低码量和常数。

一类特殊整数规划问题

对于一个整数规划$\{\max\ \mathbf{c}^{\mathrm{T}}\mathbf{x}|\mathbf{A}\mathbf{x} \leq \mathbf{b}, x_i \geq 0\}$（max可以是min，$\leq$可以是$\geq$），假如矩阵$\mathbf{A}$中只存在0和1并且每一列（或每一行）的1都是连续的，比如

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

一类特殊整数规划问题

对于一个整数规划$\{\max\ \mathbf{c}^{\mathrm{T}}\mathbf{x}|\mathbf{A}\mathbf{x} \leq \mathbf{b}, x_i \geq 0\}$（max可以是min，$\leq$可以是$\geq$），假如矩阵$\mathbf{A}$中只存在0和1并且每一列（或每一行）的1都是连续的，比如

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

那么这类问题可以是可以用最小费用最大流解决的。

一类特殊整数规划问题

对于一个整数规划$\{\max\ \mathbf{c}^{\mathrm{T}}\mathbf{x}|\mathbf{A}\mathbf{x} \leq \mathbf{b}, x_i \geq 0\}$（max可以是min，$\leq$可以是$\geq$），假如矩阵$\mathbf{A}$中只存在0和1并且每一列（或每一行）的1都是连续的，比如

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

那么这类问题可以是可以用最小费用最大流解决的。

比如NOI2008 志愿者招募、ZJOI2013 Day1 防守战线、ONTAK2010 vacation都属于这类问题。

费用流解法

首先，假如这个矩阵是每一行的1连续，就把它转成对偶形式变成每一列1连续。

费用流解法

首先，假如这个矩阵是每一行的1连续，就把它转成对偶形式变成每一列1连续。

然后对于所有不等式进行差分，然后矩阵的每一列就最多只存在一个1和一个-1。

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 1 & -1 & 0 \\ -1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

费用流解法

首先，假如这个矩阵是每一行的1连续，就把它转成对偶形式变成每一列1连续。

然后对于所有不等式进行差分，然后矩阵的每一列就最多只存在一个1和一个-1。

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 1 & -1 & 0 \\ -1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

那么将每条不等式视作每个点的流量平衡，每个变量视作一条边，就能建出最小费用最大流模型从而解决。

由于这样的矩阵一定是全幺模的，建出整数规划后写一个单纯形就能直接求出最优方案。

线性规划解法

由于这样的矩阵一定是全幺模的，建出整数规划后写一个单纯形就能直接求出最优方案。

只不过为了省去初始化，需要根据不等式的方向是$\geq$还是$\leq$选择是否需要转对偶。

线性规划解法

由于这样的矩阵一定是全幺模的，建出整数规划后写一个单纯形就能直接求出最优方案。

只不过为了省去初始化，需要根据不等式的方向是$\geq$还是$\leq$选择是否需要转对偶。

线性规划在这类矩阵上运行速度非常快，随机数据下期望次数远远不到m次。

对比

在解决这类网络流问题上，单纯形算法在思考和码代码的时间上占有非常大的优势，但是在时间效率和空间占用上有一些劣势。

对比

在解决这类网络流问题上，单纯形算法在思考和码代码的时间上占有非常大的优势，但是在时间效率和空间占用上有一些劣势。

不过大多数网络流算法在随机数据中的运行速度也是远远低于理论复杂度上界，同时也没有复杂度接近的暴力，所以通常表现在题目中就是0.1秒和0.01秒的差别，影响并不是很大。

对比

在解决这类网络流问题上，单纯形算法在思考和码代码的时间上占有非常大的优势，但是在时间效率和空间占用上有一些劣势。

不过大多数网络流算法在随机数据中的运行速度也是远远低于理论复杂度上界，同时也没有复杂度接近的暴力，所以通常表现在题目中就是0.1秒和0.01秒的差别，影响并不是很大。

总的来说，单纯形算法提供了一个新的解决网络流问题的方式，和其他网络流算法相比各有优劣，但也不失为一种好的选择。

感谢大家的聆听

感谢大家的聆听。