Chers Dudes,

Le capitaine Darkman a dit : je suis le Tarzan de Panama-City qui a vu l'ours.
Jesusman a dit : je suis le gourou incontournable du grand tout (cosmos) qui a vu Flatman.
PectoBoy a dit : je suis le gros lard d'une tonne qui a ecrasé ta voiture en hurlant : je suis un demme[1].
Moloch-man a dit : Je suis le litre de colle qui a vu le Tarzan de Panama-City.
Un biceps a dit : plus de plâtre, camarades !
EUGENIO RODRIGUEZ a dit : je suis le Chuck Norris de Porto-Harico qui va investir le Séminaire Poulpor pour y causer de :

ÜBER DAS SIMPLEXÜS.

Date: Lundi 22 avril 2002
Heure: 16h15
Lieu: Salle 623
Tenue du conférencier: Croisement pénible entre Jean-Claude Dus et Mitch Buchanan.

Laborieuses populations du Cap bienvenues.
Laborieuses culations du popac s'abstenir.

M.Baillif (wunder karma)
D.Cimasoni (cerebral moustachu)

"J'entend du bruit je crois qu'il vient, Azazel ne saurait être plus cruel."

T.G.

[1] demme : n.m. Variété de Thermoflex fort répandue au Sud-Tyrol.

ÜBER SIMPLEXUS ALGORITHMUS

EUGENIO "ZIG ZIG FRAÜLEIN" RODRIGUEZ

Abstract

This is a short explanation of the simplex algorithm designed by DANTZIG [Da]. It is based on books by CHVÁTAL [Ch] and DE WERRA [We].

1 Stating the problem

Our goal is to solve problems of the type:

$$\begin{array}{rlr} \text{maximise} & c_1x_1 + \cdots + c_nx_n & \\ \text{subject to} & a_{i1}x_1 + \cdots + a_{in}x_n \leq b_i & \text{for } i = 1, \ldots, m \\ & x_j \geq 0 & \text{for } j = 1, \ldots, n \end{array}$$

Remark that we can transform any problem of linear optimization under linear constraints to the form above; it is often called the *canonical form.*

(1) Minimizing $c_1x_1 + \cdots + c_nx_n$ is equivalent to maximizing $-c_1x_1 - \cdots - c_nx_n$.

(2) $\alpha_1x_1 + \cdots + \alpha_nx_n \geq \beta$ can be written as $-\alpha_1x_1 - \cdots - \alpha_nx_n \leq -\beta$.

(3) $\alpha_1x_1 + \cdots + \alpha_nx_n = \beta$ can be written as $\alpha_1x_1 + \cdots + \alpha_nx_n \leq \beta$ and $-\alpha_1x_1 - \cdots - \alpha_nx_n \leq -\beta$.

The first step towards the resolution is to add *slack variables*

$$x_{n+i} = b_i - a_{i1}x_1 - \cdots - a_{in}x_n \ .$$

Now write

$$
\begin{aligned}
A &= \begin{pmatrix} a_{11} & \dots & a_{1n} & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{1m} & \dots & a_{mn} & 0 & \dots & 1 \end{pmatrix} \\
b &= \big(b_1 \dots b_m\big) \\
c &= \big(c_1 \dots c_n \, 0 \dots 0\big) \\
x &= \big(x_1 \dots x_{n+m}\big)
\end{aligned}
$$

Let $Ax = b$ and $z = cx$, the original problem is equivalent to the problem

$$
\begin{aligned}
\text{maximize} \quad & z \\
\text{subject to} \quad & x \geq 0
\end{aligned}
$$

We make the following remarks and definitions.

(1) A is a $m \times (n+m)$ matrix of maximal rank.

(2) The set $\mathcal{P} := \{x \in \mathbb{R}^{(m+m)} \mid Ax = b \text{ and } x \geq 0\}$ is a convex polyhedron. It may be either empty or unbounded.

(3) Define a *factet* of $\mathcal{P}$ as one of the m hyperplanes $A_j x = b_j$, for $j = 1, \dots, m$.

(4) Define an *edge* as an intersection of two facets.

(5) Define a *vertex* as a point $x \in \mathcal{P}$ that is uniquely determined as an intersection of some factets.

(6) Recall from calculus that if a linear function defined on a convex polyhedron admits a finite extrema, then it must be on one of it's vertices; see for example [We, Chapter 3].

2 Basic feasible solutions

Forget about the main problem and let's do some linear algebra. Consider A a $m \times n$ matrix, $n \geq m$, of maximal rank. Define a *basis* of A as a subset $\mathcal{B} \subset \{1, \dots, n\}$ such that:

(1) $|\mathcal{B}| = m$.

(2) The matrix $A_{\mathcal{B}}$ defined by the column-vectors $A_{b_1}, \dots, A_{b_m}$ has a non-null determinant.

Consider a basis $\mathcal{B}$, reorder the variables according to $\mathcal{B}$ and $\mathcal{N} := \{1, \dots, m\} \setminus \mathcal{B}$,

$$
\begin{aligned}
x &= (x_{\mathcal{B}} \, x_{\mathcal{N}}) \\
A &= (A_{\mathcal{B}} \, A_{\mathcal{N}})
\end{aligned}
$$

and write $A_{\mathcal{B}}x_{\mathcal{B}} + A_{\mathcal{N}}x_{\mathcal{N}} = b$. This is equivalent to

$$x_{\mathcal{B}} = A_{\mathcal{B}}^{-1}b - A_{\mathcal{B}}^{-1}A_{\mathcal{N}}x_{\mathcal{N}}.$$

We naturally set $x_{\mathcal{N}} = 0$ and call $x_{\mathcal{B}} = A_{\mathcal{B}}^{-1}b$ a *basic solution.* If $A_{\mathcal{B}}^{-1}b \geq 0$ we say that the basic solution is *feasible.* We call the variables of $x_{\mathcal{B}}$ *basic variables* and the other ones *non-basic.*

PROPOSITION 0.1. *A point x is a vertex of $\mathcal{P}$ if and only if x is a basic feasible solution.*

Proof. See [Ch] or [We]. □

We now have a theoritical algorithm for finding the optimal solution of our problem: *go through all the basic feasible solutions and evaluate the linear function cx on each one.* The number of basic solutions is at most $\binom{n}{m}$, so this algorithm is finite. But it is certainly not very optimized.

3 How the simplex works

By definition of the matrix A in section 1, we have an obvious candidate for a first basic solution: $\mathcal{B} = \{n+1, \ldots, n+m\}$. The matrix $A_{\mathcal{B}}$ corresponds to identity matrix and $A_{\mathcal{N}}$ is the matrix with elements a_{ij}. The relation $x_{\mathcal{B}} = A_{\mathcal{B}}^{-1}b - A_{\mathcal{B}}^{-1}A_{\mathcal{N}}x_{\mathcal{N}}$ yields in this case:

$$\begin{array}{ccccccccc}
x_{n+1} & = & b_1 & - & a_{11}x_1 & - & \ldots & - & a_{1n}x_n \\
\vdots & & \vdots & & & & & & \vdots \\
x_{n+m} & = & b_m & - & a_{m1}x_1 & - & \ldots & - & a_{mn}x_n \\
\hline
z & = & 0 & + & c_1x_1 & + & \ldots & + & c_nx_n
\end{array}$$

We call such a set of equations a *dictionnary.* Suppose that $b_i \geq 0$; this means that b is a basic feasible solution and that the polyhedron $\mathcal{P}$ is not empty. This is not true in general, but we will see in section 4 what can be done when it is not the case.

The idea of the simplex algorithm is to travel from one basic solution to another but without visiting them all. We are only interested in those which will increase the value of z. This is done in the following procedure.

Look at the last line:

$$z = 0 + c_1x_1 + \cdots + c_nx_n$$

Since $\{x_1, \ldots, x_n\}$ are non-basic variables we have set their values to 0. If $c_j \leq 0$ for all j, we have reached a maximum. Since the x_j must be non-negatif we can't decrease their value, and by trying to increase them, we would only decrease the value of z. So let's suppose that there exists j_0 such that $c_{j_0} > 0$.

We decide to increase the value of x_{j_0} from 0 to a certain value t. We keep the values of the other non-basic variables at 0. Can we increase x_{j_0} indefinitly ? To answer this question let's look at the column j_0 of the dictionnary.

$$
\begin{array}{rcccl}
x_{n+1} & = & b_1 & - & a_{1j_0} x_{j_0} \\
\vdots & & & & \vdots \\
x_{n+m} & = & b_m & - & a_{mj_0} x_{j_0}
\end{array}
$$

Remember that we must satisfy the constraints $x_{n+i} \geq 0$. If $a_{ij_0} \leq 0$ for all i, then we can increase x_{j_0} as much as we want. This means that $\mathcal{P}$ is unbounded, hence there is no finite maximum.

So suppose that the set $I := \{i \mid a_{ij_0} > 0\}$ is not empty. Since $x_{n+i} \geq 0$, we have for all $i \in I$

$$\frac{b_i}{a_{ij_0}} \geq x_{j_0}$$

Denote $i_0 := \min_{i \in I}\{\frac{b_i}{a_{ij_0}}\}$. This answers our question: increase x_{j_0} upto $\frac{b_{i_0}}{a_{i_0 j_0}}$. Note that the actual value of x_{m+i_0} is 0.

What we actually did in this manipulation is to travel along an edge of $\mathcal{P}$ from x_{j_0} to x_{m+i_0}. We now *pivot* around the *pivot-element* $a_{i_0 j_0}$: write x_{j_0} in function of x_{n+i_0} and the other basic variables and substitute throughout all the equations. This means that x_{j_0} becomes basic and x_{m+i_0} non-basic. We have actually made a change of basis.

What follows is straightforward. We do the same procedure with the new values $\tilde{c}_j$, over and over, until they are all strictly negatif. But are we sure that this procedure is finite?

4 Avoiding some technical problems

The algorithm above describes a way to find entering and leaving variables, how to pivot and when to stop. Here we discuss 2 problems that may occure. The first one concerns initialization. What do we do if the vector b is not a feasible solution ? May the polyhedron still be non-empty ? The second one concerns iterations. Are sure to finish one day ? Couldn't we go on iterating on a sort of cycle?

4.1 Initialization

The problem of initialization arises when the vector b does not satisfy $b_i \geq 0$. There are two cases to consider:

(1) The polyhedron $\mathcal{P}$ is empty; we must be able to show it.

(2) $\mathcal{P}$ is not empty; in this case we must be able to exhibit a basic feasible solution.

The idea is to consider an auxilary problem:

$$\begin{array}{lll} \text{minimize} & x_0 & \\ \text{subject to} & \sum_{j=1}^{n} a_{ij} - x_0 \leq b_i & i = 1, \ldots, m \\ & x_j \geq 0 & j = 0, \ldots, n \end{array}$$

We easily have an initial feasible solution: set $x_1, \ldots, x_n = 0$ and take x_0 large enough so that $\sum_{j=1}^{n} a_{ij} - x_0 \leq b_i$ for $i = 1, \ldots, m$. Writing it in terms of dictionnaries is straight forward. We then solve the problem with the simplex algorithm. It is easy to see that the original problem has a feasible solution if and only if the auxilary problem has a minimum of 0. The last dictionnary then yields a feasible solution for the original problem.

4.2 Cycling

To explain cycling, consider the following example from [Ch]. Suppose our problem leads us to the dictionnary:

$$\begin{array}{ccccccccc} x_4 & = & 1 & & & & & - & 2x_3 \\ x_5 & = & 3 & - & 2x_1 & + & 4x_2 & - & 6x_3 \\ x_6 & = & 2 & + & x_1 & - & 3x_2 & - & 4x_3 \\ \hline z & = & & & 2x_1 & - & x_2 & + & 8x_3 \end{array}$$

Choose now x_3 as the leaving variable. We then notice that as the entering variable we can choose either x_4, x_5 or x_6. All of them limit the increase of x_3 to $\frac{1}{2}$. Choose arbitrarily x_4. After pivoting we obtain:

$$\begin{array}{ccccccccc} x_3 & = & \frac{1}{2} & & & & & - & \frac{1}{2}x_4 \\ x_5 & = & & - & 2x_1 & + & 4x_2 & - & 3x_4 \\ x_6 & = & & & x_1 & - & 3x_2 & - & 2x_4 \\ \hline z & = & 4 & + & 2x_1 & - & x_2 & + & 4x_4 \end{array}$$

Notice that the basic feasible solution some basic variables with a 0 value. Such a solution is called *degenerate*, and may cause some problems as seen in the next iteration. Choose x_1 and x_5 as pivot.

$$\begin{array}{ccccccccc} x_3 & = & \frac{1}{2} & & & & & - & \frac{1}{2}x_4 \\ x_1 & = & & - & \frac{1}{2}x_5 & + & 2x_2 & + & \frac{3}{2}x_4 \\ x_6 & = & & - & \frac{1}{2}x_5 & - & x_2 & + & \frac{7}{2}x_4 \\ \hline z & = & 4 & - & x_5 & + & 3x_2 & - & x_4 \end{array}$$

The problem that arises is that the value of z has not increased. We naturally call such an iteration *degenerate*. If you try to continue you will notice that the next iteration is also degenerate but not the one after. In most cases (*most cases* means that you randomly choose your variables) degenerated iterations are not harmfull: after some iterations you get back a non-degenerated basic solution. But a atypical phenomena called *cycling* may happen, a secuence of iterations leads back to the initial one. In this case the simplex algorithm goes on iterating forever (we assume here that the algorithm is performed by a computer that chooses the entering and leaving variables in a determined way). A manner of avoiding this problem is to choose the entering and leaving variables in a *good* way. This is called *Bland's rule* [Bl]:

Among the possible candiates for entering or leaving choose the one with the smallest subscript.

BLAND proved that the simplex always terminates when choosing the variables with this rule. What is usually done in computer implementations is to do normal iterations (choosing the first variable that comes) until it starts degenerating. If after a certain number of iterations we are still cycling then use Bland's rule to get out of the cycle and continue with normal iterations. This is usually faster since cycling is a rare case.

References

[Bl] Bland R.G., *New finite pivoting rules for the simplex method*, Mathematics of Operations Research 2: 103-107, (1977).

[Ch] Vašek Chvátal, *Linear Programming*, W. H. Freeman Company, (1983).

[Da] Dantzig G. B., *Maximization of a linear function of variables subject to linear inequalities*, Activity analysis of production and allocation, Cowles Commission monograph n° 13, J. Wiley, New-York, (1951), Chap. 21.

[We] de Werra Dominique, *Eléments de programmation linéaire avec applications aux graphes*, Presses polytechniques romandes, (1990).