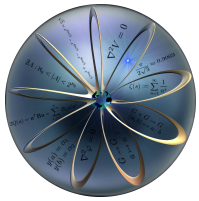

planetmath.org

Math for the people, by the people.

simplex algorithm

Canonical name	SimplexAlgorithm
Date of creation	2013-03-22 13:35:21
Last modified on	2013-03-22 13:35:21
Owner	Mathprof (13753)
Last modified by	Mathprof (13753)
Numerical id	22
Author	Mathprof (13753)
Entry type	Algorithm
Classification	msc 90C05
Synonym	simplex method

The simplex algorithm is used as part of the simplex method (due to George B. Dantzig) to solve linear programming problems. The algorithm is applied to a linear programming problem that is in canonical form.

A *canonical system* of equations has an ordered subset of variables (called the *basis*) such that for each i, the i^{th} basic variable has a unit coefficient in the i^{th} equation and zero coefficient in the other equations.

As an example $x_1, \ldots, x_r$ are basic variables in the following system of r equations:

$$\begin{array}{rcl} x_1 & + & a_{1,r+1}x_{r+1} + \cdots + a_{1,n}x_n = b_1 \\ x_2 & + & a_{2,r+1}x_{r+1} + \cdots + a_{2,n}x_n = b_2 \\ \cdots & & \\ x_r & + & a_{r,r+1}x_{r+1} + \cdots + a_{r,n}x_n = b_r \end{array}$$

The simplex algorithm is used as one phase of the simplex method.

Suppose that we have a canonical system with basic variables $x_1, \ldots, x_m, -z$ and we seek to find nonnegative x_i $i = 1, \ldots, n$ such that z is minimal. That is, we have

$$\begin{aligned} x_i + \sum_{j=m+1}^{n} a_{ij}x_j &= b_i \quad i = 1, \ldots, m \\ -z + \sum_{j=m+1}^{n} c_j x_j &= -z_o \end{aligned}$$

where a_{ij}, b_j, c_j, z_o are constants, and $b_j \geq 0, \quad j = 1, \ldots, m$.

Notice that if we set $x_{m+1} = 0, \ldots, x_n = 0$ we will have a feasible solution with $z = z_o$.. Hence, any optimal solution will have $z \leq z_o$. The algorithm can now be described as follows:

Step 1. Set $N = \{m+1, \ldots, n\}$ and $B = \{1, \ldots, m\}$. Put $c_j = 0$ for $j \in B$.

Step 2. If there an index $j \in N$ such that $c_j < 0$ then choose $s \in N$ such that

$$c_s = \min_{j \in N} c_j$$

else stop. The solution is given by $x_i = 0$ for $i \in N$ and $x_i = b_i$ for $i \in B$, $z = z_o$.

Step 3. If $a_{is} \leq 0$ for all i then stop. The value of z has no lower bound. Else, let $\frac{b_r}{a_{rs}} = \min_{a_{is}>0} \frac{b_i}{a_{is}}$. If there is more than one choice for r it does not matter which one is chosen *unless* $b_i = 0$. This is the so-called degenerate case. In this case, one can choose uniformly at random from among those i for which $b_i = 0$.

Step 4. (Pivot on a_{rs}). Multiply the r^{th} equation by $\frac{1}{a_{rs}}$ and for each $i = 1, \ldots, m$, $i \neq r$ replace equation i by the sum of equation i and the (replaced) equation r multiplied by $-a_{is}$. Replace the equation for z by the sum of the equation for z and the (replaced) equation r multiplied by $-c_s$. Note: The replacement operations of course change the coefficients a_{ij} and c_j. As the algorithm proceeds it is of course necessary to use the changed coefficients.

Step 5. (Update B and N) Put s into B and r into N and remove s from N and r from B. Go to step 2.

There are examples where the algorithm does not terminate in a finite number of steps; but if there is non-degeneracy at each iteration, the algorithm will terminate in a finite number of steps.