DM545/DM871

Linear and Integer Programming

Lecture 7

Revised Simplex Method

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

1. Revised Simplex Method

2. Efficiency Issues

Motivation

Complexity of single pivot operation in standard simplex:

- entering variable $O(n)$
- leaving variable $O(m)$
- updating the tableau $O(mn)$

Problems with this:

- Time: we are doing operations that are not actually needed
  Space: we need to store the whole tableau: $O(mn)$ floating point numbers
- Most problems have sparse matrices (many zeros)
  sparse matrices are typically handled efficiently
  the standard simplex has the 'Fill in' effect: sparse matrices are lost
- accumulation of Floating Point Errors over the iterations

Outline

Revised Simplex Method

Several ways to improve wrt pitfalls in the previous slide, requires matrix description of the simplex.

$$\max \sum_{j=1}^{n} c_j x_j$$
$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \;\; i = 1..m$$
$$x_j \geq 0 \;\; j = 1..n$$

$$\max c^T x$$
$$Ax = b$$
$$x \geq 0$$
$$A \in \mathbb{R}^{m \times (n+m)}$$
$$c \in \mathbb{R}^{(n+m)}, b \in \mathbb{R}^m, x \in \mathbb{R}^{n+m}$$

$$\max\{c^T x \mid Ax = b, x \geq 0\}$$

At each iteration the simplex moves from a basic feasible solution to another.

For each basic feasible solution:

- $B = \{1 \ldots m\}$ basis
- $N = \{m+1 \ldots m+n\}$
- $A_B = [a_1 \ldots a_m]$ basis matrix
- $A_N = [a_{m+1} \ldots a_{m+n}]$
- $x_N = 0$
- $x_B \geq 0$

$$\left[\begin{array}{cc:c:c} A_N & A_B & 0 & \mathrm{b} \\ \hdashline \mathrm{c}_N^T & \mathrm{c}_B^T & 1 & 0 \end{array}\right]$$

$$A\mathrm{x} = A_N\mathrm{x}_N + A_B\mathrm{x}_B = \mathrm{b}$$
$$A_B\mathrm{x}_B = \mathrm{b} - A_N\mathrm{x}_N$$

Basic feasible solution $\iff$ A_B is non-singular

$$\mathrm{x}_B = A_B^{-1}\mathrm{b} - A_B^{-1}A_N\mathrm{x}_N$$

for the objective function:

$$z = \mathrm{c}^T \mathrm{x} = \mathrm{c}_B^T \mathrm{x}_B + \mathrm{c}_N^T \mathrm{x}_N$$

Substituting for x_B from above:

$$z = \mathrm{c}_B^T (A_B^{-1}\mathrm{b} - A_B^{-1} A_N \mathrm{x}_N) + \mathrm{c}_N^T \mathrm{x}_N =$$
$$= \mathrm{c}_B^T A_B^{-1}\mathrm{b} + (\mathrm{c}_N^T - \mathrm{c}_B^T A_B^{-1} A_N)\mathrm{x}_N$$

Collecting together:

$$\mathrm{x}_B = A_B^{-1}\mathrm{b} - A_B^{-1} A_N \mathrm{x}_N$$
$$z = \mathrm{c}_B^T A_B^{-1}\mathrm{b} + (\mathrm{c}_N^T - \mathrm{c}_B^T \underbrace{A_B^{-1} A_N}_{\bar{A}})\mathrm{x}_N$$

In tableau form, for a basic feasible solution corresponding to B we have:

$A_B^{-1}A_N$	I	0	$A_B^{-1}\mathrm{b}$
$\mathrm{c}_N^T - \mathrm{c}_B^T A_B^{-1} A_N$	0	1	$-\mathrm{c}_B^T A_B^{-1}\mathrm{b}$

We do not need to compute all elements of $\bar{A}$

Example

$$\begin{aligned} \max \quad & x_1 + x_2 \\ & -x_1 + x_2 \leq 1 \\ & x_1 \leq 3 \\ & x_2 \leq 2 \\ & x_1, x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \max \quad & x_1 + x_2 \\ & -x_1 + x_2 + x_3 = 1 \\ & x_1 + x_4 = 3 \\ & x_2 + x_5 = 2 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

Initial tableau

x1	x2	x3	x4	x5	−z	b
−1	1	1	0	0	0	1
1	0	0	1	0	0	3
0	1	0	0	1	0	2
1	1	0	0	0	1	0

After two iterations

x1	x2	x3	x4	x5	−z	b
1	0	−1	0	1	0	1
0	1	0	0	1	0	2
0	0	1	1	−1	0	2
0	0	1	0	−2	1	3

Basic variables x_1, x_2, x_4. Non basic: x_3, x_5. From the initial tableau:

$$A_B = \begin{bmatrix} -1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad A_N = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \quad x_B = \begin{bmatrix} x_1 \\ x_2 \\ x_4 \end{bmatrix} \quad x_N = \begin{bmatrix} x_3 \\ x_5 \end{bmatrix}$$

$$c_B^T = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \quad c_N^T = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

- **Entering variable**:
  in std. we look at tableau, in revised we need to compute: $c_N^T - c_B^T A_B^{-1} A_N$
  1. find $y^T = c_B^T A_B^{-1}$ (by solving $y^T A_B = c_B^T$, the latter can be done more efficiently)
  2. calculate $c_N^T - y^T A_N$

Step 1:

$$\begin{bmatrix} y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} -1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \qquad \mathrm{y}^T A_B = \mathrm{c}_B^T$$

$$\begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 2 \end{bmatrix} \qquad \mathrm{c}_B^T A_B^{-1} = \mathrm{y}^T$$

Step 2:

$$\begin{bmatrix} 0 & 0 \end{bmatrix} - \begin{bmatrix} -1 & 0 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -2 \end{bmatrix} \qquad \mathrm{c}_N^T - \mathrm{y}^T A_N$$

(Note that they can be computed individually: $c_j - \mathrm{y}^T \mathrm{a}_j > 0$)
Let's take the first we encounter x_3

- **Leaving variable**
  we increase variable by largest feasible amount θ

$$\begin{aligned} &\text{R1: } x_1 - x_3 + x_5 = 1 \\ &\text{R2: } x_2 + 0x_3 + x_5 = 2 \\ &\text{R3: } -x_3 + x_4 - x_5 = 2 \end{aligned} \qquad \begin{aligned} x_1 &= 1 + x_3 \geq 0 \\ x_2 &= 2 \geq 0 \\ x_4 &= 2 - x_3 \geq 0 \end{aligned}$$

$\mathrm{x}_B = \mathrm{x}_B^* - A_B^{-1} A_N \mathrm{x}_N$
$\mathrm{x}_B = \mathrm{x}_B^* - \mathrm{d}\theta$

d is the column of $A_B^{-1} A_N$ that corresponds to the entering variable, ie, $\mathrm{d} = A_B^{-1}\mathrm{a}$ where a is the entering column

3. Find θ such that x_B stays positive:
   Find $\mathrm{d} = A_B^{-1}\mathrm{a}$ (by solving $A_B\mathrm{d} = \mathrm{a}$)

Step 3:

$$\begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \implies \mathrm{d} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \implies \mathrm{x}_B = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix} - \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \theta \geq 0$$

$2 - \theta \geq 0 \implies \theta \leq 2 \rightsquigarrow x_4$ leaves

- So far we have done computations, but now we save the pivoting update. The update of A_B is done by replacing the leaving column by the entering column

$$x_B^* = \begin{bmatrix} x_1 - d_1\theta \\ x_2 - d_2\theta \\ \theta \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 2 \end{bmatrix} \qquad A_B = \begin{bmatrix} -1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

- Many implementations depending on how $\mathrm{y}^T A_B = \mathrm{c}_B^T$ and $A_B \mathrm{d} = \mathrm{a}$ are solved. They are in fact solved from scratch.

- many operations saved especially if many variables!

- special ways to call the matrix A from memory

- better control over numerical issues since A_B^{-1} can be recomputed.

Outline

Solving the two Systems of Equations

$A_B x = b$ solved without computing A_B^{-1}
(costly and likely to introduce numerical inaccuracy)

Recall how the inverse is computed:

For a 2×2 matrix

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

the matrix inverse is

$$A^{-1} = \frac{1}{|A|} \begin{bmatrix} d & -c \\ -b & a \end{bmatrix}^T = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

For a 3×3 matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

the matrix inverse is

$$A^{-1} = \frac{1}{|A|} \begin{bmatrix} + \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & - \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} & + \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \\ - \begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix} & + \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & - \begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix} \\ + \begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} & - \begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix} & + \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{bmatrix}^T$$

Eta Factorization of the Basis

Let $B := A_B$, kth iteration
B_k be the matrix with col p differing from B_{k-1}
Column p is the a column appearing in $B_{k-1}\mathrm{d} = \mathrm{a}$ solved at 3)
Hence:

$$B_k = B_{k-1}E_k$$

E_k is the eta matrix differing from id. matrix in only one column

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

No matter how we solve $\mathrm{y}^T B_{k-1} = \mathrm{c}_B^T$ and $B_{k-1}\mathrm{d} = \mathrm{a}$, their update always relays on $B_k = B_{k-1}E_k$ with E_k available.
Plus when initial basis by slack variable $B_0 = I$ and $B_1 = E_1, B_2 = E_1E_2\cdots$:

$$B_k = E_1E_2\ldots E_k \qquad \text{eta factorization}$$

$$((((\mathrm{y}^T E_1)E_2)E_3)\cdots)E_k = \mathrm{c}_B^T, \qquad \mathrm{u}^T E_4 = \mathrm{c}_B^T,\ \mathrm{v}^T E_3 = \mathrm{u}^T,\ \mathrm{w}^T E_2 = \mathrm{v}^T,\ \mathrm{y}^T E_1 = \mathrm{w}^T$$
$$(E_1(E_2\cdots E_k\mathrm{d})) = \mathrm{a}, \qquad E_1\mathrm{u} = \mathrm{a},\ E_2\mathrm{v} = \mathrm{u},\ E_3\mathrm{w} = \mathrm{v},\ E_4\mathrm{d} = \mathrm{w}$$

Exercise

Solve the systems $y^T E_1 E_2 E_3 E_4 = [1\ 2\ 3]$ and $E_1 E_2 E_3 E_4 d = [1\ 2\ 3]^T$ with

$$E_1 = \begin{bmatrix} 1 & 3 & 0 \\ 0 & 0.5 & 0 \\ 0 & 4 & 1 \end{bmatrix} \qquad E_2 = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 1 & 0 \\ 4 & 0 & 1 \end{bmatrix} \qquad E_3 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} \qquad E_4 = \begin{bmatrix} -0.5 & 0 & 0 \\ 3 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

We use backward transformation and solve the sequence of linear systems:

$$\mathrm{u}^T E_4 = [1\ 2\ 3], \quad \mathrm{v}^T E_3 = \mathrm{u}^T, \quad \mathrm{w}^T E_2 = \mathrm{v}^T, \quad \mathrm{y}^T E_1 = \mathrm{w}^T$$

$$\mathrm{u}^T \begin{bmatrix} -0.5 & 0 & 0 \\ 3 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = [1, 2, 3]$$

Since the eta matrices have always one 1 in two columns then the solution can be read up easily. From the third column we find $u_3 = 3$. From the second column, we find $u_2 = 2$. Substituting in the first column, we find $-0.5u_1 + 3 * 2 + 1 * 3 = 1$, which yields $u_1 = 18$. The next syestem is:

$$\mathrm{v}^T \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} = [18, 2, 3]$$

From the first column we get $v_1 = 18$, from the second column $v_2 = 2$ from the last column $v_3 = 3/24$. The next:

$$\mathrm{w} = \begin{bmatrix} 2 & 0 & 0 \\ 1 & 1 & 0 \\ 4 & 0 & 1 \end{bmatrix} = [18, 2, 3/24]$$

...

- Solving $y^T B_k = c_B^T$ also called backward transformation (BTRAN)
- Solving $B_k d = a$ also called forward transformation (FTRAN)

- E_i matrices can be stored by only storing the column and the position
- If sparse columns then can be stored in compact mode, ie only nonzero values and their indices

More on LP

- Tableau method is unstable: computational errors may accumulate. Revised method has a natural control mechanism: we can recompute A_B^{-1} at any time
- Commercial and freeware solvers differ from the way the systems $\mathrm{y}^T A_B = \mathrm{c}_B^T$ and $A_B \mathrm{d} = \mathrm{a}$ are resolved

Efficient Implementations

- Dual simplex with steepest descent (largest increase)
- Linear Algebra:
  - Dynamic LU-factorization using Markowitz threshold pivoting (Suhl and Suhl, 1990)
  - sparse linear systems: Typically these systems take as input a vector with a very small number of nonzero entries and output a vector with only a few additional nonzeros.
- Presolve, ie problem reductions: removal of redundant constraints, fixed variables, and other extraneous model elements.
- dealing with degeneracy, stalling (long sequences of degenerate pivots), and cycling:
  - bound-shifting (Paula Harris, 1974)
  - Hybrid Pricing (variable selection): start with partial pricing, then switch to devex (approximate steepest-edge, Harris, 1974)
- A model that might have taken a year to solve 10 years ago can now solve in less than 30 seconds (Bixby, 2002).