DM545

Linear and Integer Programming

Lecture 7
Revised Simplex Method

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

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.

$$
\begin{aligned}
\max \ & \sum_{j=1}^{n} c_j x_j \\
& \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad i = 1..m \\
& x_j \geq \ 0 \quad j = 1..n
\end{aligned}
\qquad
\begin{aligned}
\max \ & \mathbf{c}^T \mathbf{x} \\
A\mathbf{x} &= \mathbf{b} \\
\mathbf{x} &\geq \mathbf{0} \\
A &\in \mathbb{R}^{m \times (n+m)} \\
\mathbf{c} \in \mathbb{R}^{(n+m)}&, \mathbf{b} \in \mathbb{R}^m, \mathbf{x} \in \mathbb{R}^{n+m}
\end{aligned}
\qquad
\max\{\mathbf{c}^T \mathbf{x} \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}
$$

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

For each basic feasible solution:

- $B = \{1 \ldots m\}$ basis
- $N = \{m+1 \ldots m+n\}$
- $A_B = [\mathbf{a}_1 \ldots \mathbf{a}_m]$ basis matrix
- $A_N = [\mathbf{a}_{m+1} \ldots \mathbf{a}_{m+n}]$
- $\mathbf{x}_N = 0$
- $\mathbf{x}_B \geq 0$

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

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

Theorem

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

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

for the objective function:

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

Substituting for $\mathbf{x}_B$ from above:

$$\begin{aligned} z &= \mathbf{c}_B^T (A_B^{-1}\mathbf{b} - A_B^{-1} A_N \mathbf{x}_N) + \mathbf{c}_N^T \mathbf{x}_N = \\ &= \mathbf{c}_B^T A_B^{-1}\mathbf{b} + (\mathbf{c}_N^T - \mathbf{c}_B^T A_B^{-1} A_N)\mathbf{x}_N \end{aligned}$$

Collecting together:

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

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

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

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:
  $\mathbf{c}_N^T - \mathbf{c}_B^T A_B^{-1} A_N$
  1. find $\mathbf{y}^T = \mathbf{c}_B^T A_B^{-1}$ (by solving $\mathbf{y}^T A_B = \mathbf{c}_B^T$, the latter can be done more efficiently)
  2. calculate $\mathbf{c}_N^T - \mathbf{y}^T A_N$

Step 1:

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

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

Step 2:

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

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

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

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

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

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

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

Step 3:

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

$2 - \theta \geq 0 \implies \theta \leq 2 \rightsquigarrow x_4$ 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 $\mathbf{y}^T A_B = \mathbf{c}_B^T$ and $A_B\mathbf{d} = \mathbf{a}$ 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

1. Revised Simplex Method

2. Efficiency Issues

Solving the two Systems of Equations

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

Recall how the inverse is computed:

For a 2×2 matrix the matrix inverse is

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

For a 3×3 matrix the matrix inverse is

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

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

Eta Factorization of the Basis

Let $A_B = B$, kth iteration
B_k be the matrix with col p differing from B_{k-1}
Column p is the $\mathbf{a}$ column appearing in $B_{k-1}\mathbf{d} = \mathbf{a}$ 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 $\mathbf{y}^T B_{k-1} = \mathbf{c}_B^T$ and $B_{k-1}\mathbf{d} = \mathbf{a}$, 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 $B_1 = E_1, B_2 = E_1E_2 \cdots$:

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

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

LU factorization

Worth to consider also the case of $B_0 \neq I$:

$$B_k = B_0 E_1 E_2 \dots E_k \qquad \text{eta factorization}$$

$$((((\mathbf{y}^T B_0)E_1)E_2)\cdots)E_k = \mathbf{c}_B^T$$

$$(B_0(E_1 \cdots E_k \mathbf{d})) = \mathbf{a}$$

We need an LU factorization of B_0

LU Factorization

To solve the system $A\mathbf{x} = \mathbf{b}$ by Gaussian Elimination we put the A matrix in row echelon form by means of elementary row operations. Each row operation corresponds to multiply left and right side by a lower triangular matrix L and a permuation matrix P. Hence, the method:

$$\begin{aligned} A\mathbf{x} &= \mathbf{b} \\ L_1 P_1 A\mathbf{x} &= L_1 P_1 \mathbf{b} \\ L_2 P_2 L_1 P_1 A\mathbf{x} &= L_2 P_2 L_1 P_1 \mathbf{b} \\ &\vdots \\ L_m P_m \ldots L_2 P_2 L_1 P_1 A\mathbf{x} &= L_m P_m \ldots L_2 P_2 L_1 P_1 \mathbf{b} \end{aligned}$$

thus

$$U = L_m P_m \ldots L_2 P_2 L_1 P_1 A \qquad \text{triangular factorization of } A$$

where U is an upper triangular matrix whose entries in the diagonal are ones. (if A is nonsingular such triangularization is unique)

[see numerical example in Va sc 8.1]

We can compute the triangular factorization of B_0 before the initial iterations of the simplex:

$$L_m P_m \dots L_2 P_2 L_1 P_1 B_0 = U$$

We can then rewrite U as

$$U = U_m U_{m-1} \dots, U_1$$

Hence, for $B_k = B_0 E_1 E_2 \dots E_k$:

$$L_m P_m \dots L_2 P_2 L_1 P_1 B_k = U_m U_{m-1} \dots U_1 E_1 E_2 \cdots E_k$$

Then $\mathbf{y}^T B_k = \mathbf{c}_B^T$ can be solved by first solving:

$$((((\mathbf{y}^T U_m) U_{m-1}) \cdots) E_k = \mathbf{c}_B^T$$

and then replacing

$$\mathbf{y}^T \text{ by } ((\mathbf{y}^T L_m P_m) \cdots) L_1 P_1$$

$$B_k = \underbrace{(L_m P_m \cdots L_1 P_1)}_{L}{}^{-1} \underbrace{U_m \cdots E_k}_{U}$$

$$\mathbf{y} L^{-1} U = \mathbf{c}$$

$$\mathbf{w} U = \mathbf{c}$$

$$\mathbf{w} = \mathbf{y} L^{-1} \implies \mathbf{y} = L\mathbf{w}$$

- Solving $\mathbf{y}^T B_k = \mathbf{c}_B^T$ also called backward transformation (BTRAN)
- Solving $B_k \mathbf{d} = \mathbf{a}$ 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
- Same for the triangular eta matrices L_j, U_j
- while for P_j just two indices are needed

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 $\mathbf{y}^T = \mathbf{c}_B^T A_B^{-1}$ and $A_B \mathbf{d} = \mathbf{a}$ are resolved

Efficient Implementations

- Dual simplex with steepest descent
- 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).