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.

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

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

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

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

For each basic feasible solution:

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

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

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

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

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

for the objective function:

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

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

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

Collecting together:

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

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

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

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:

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

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

Step 2:

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

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

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

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

$$
\begin{aligned}
\boldsymbol{x}_B &= \boldsymbol{x}_B^* - A_B^{-1} A_N \boldsymbol{x}_N \\
\boldsymbol{x}_B &= \boldsymbol{x}_B^* - \boldsymbol{d}\theta
\end{aligned}
$$

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

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

Step 3:

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

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

$A_B x = b$ 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 $\boldsymbol{a}$ column appearing in $B_{k-1}\boldsymbol{d} = \boldsymbol{a}$ 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, which is set equal to $\boldsymbol{d}$

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

No matter how we solve $\boldsymbol{y}^T B_{k-1} = \boldsymbol{c}_B^T$ and $B_{k-1}\boldsymbol{d} = \boldsymbol{a}$, 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 $B_1 = E_1, B_2 = E_1E_2 \cdots$:

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

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

$$(E_1(E_2\cdots E_k\boldsymbol{d})) = \boldsymbol{a}, \qquad E_1\boldsymbol{u} = \boldsymbol{a},\ E_2\boldsymbol{v} = \boldsymbol{u},\ E_3\boldsymbol{w} = \boldsymbol{v},\ E_4\boldsymbol{d} = \boldsymbol{w}$$

Exercise

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

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

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

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 $u_1 = 16$. The next syestem is:

$$\boldsymbol{v}^T \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 3 \\ 0 & 0 & 1 \end{bmatrix} = [16, 2, 3]$$

From the first column we get $v_1 = 16$, from the second column $v_2 = 2$ from the last column $v_1 + 3v_2 + v_3 = 3$, which yields $v_3 = -19$. The next:

$$\boldsymbol{w}^T \begin{bmatrix} 2 & 0 & 0 \\ 1 & 1 & 0 \\ 4 & 0 & 1 \end{bmatrix} = [16, 2, -19]$$

...

- Solving $\boldsymbol{y}^T B_k = \boldsymbol{c}_B^T$ also called backward transformation (BTRAN)
- Solving $B_k \boldsymbol{d} = \boldsymbol{a}$ 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 $\boldsymbol{y}^T A_B = \boldsymbol{c}_B^T$ and $A_B \boldsymbol{d} = \boldsymbol{a}$ 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).

Further topics in LP

- Ellipsoid method: cannot compete in practice but polynomial time (Khachyian, 1979)
- Interior point algorithm(s) (Karmarkar, 1984) competitive with simplex and polynomial in some versions
  - iterate through points interior to the feasibility region
  - because of patents reasons, also known as barrier algorithm
  - one single iteration is computationally more intensive than the simplex
  - particularly competitive in presence of many constraints (eg, for $m = 10,000$ may need less than 100 iterations)
  - bad for post-optimality analysis $\rightsquigarrow$ crossover algorithm to convert a sol of barrier method into a basic feasible solutions for the simplex
  - usually fastest for large, difficult models but numerically sensitive.
- Lagrangian relaxation
- Column generation
- Decomposition methods:
  - Dantzig Wolfe decomposition
  - Benders decomposition

Interior Point Algorithm

1. Start at an interior point of the feasible region
2. Move in a direction that improves the objective function value at the fastest possible rate
3. Transform the feasible region to place the current point at the center of it

How Large Problems Can We Solve?

Very large model

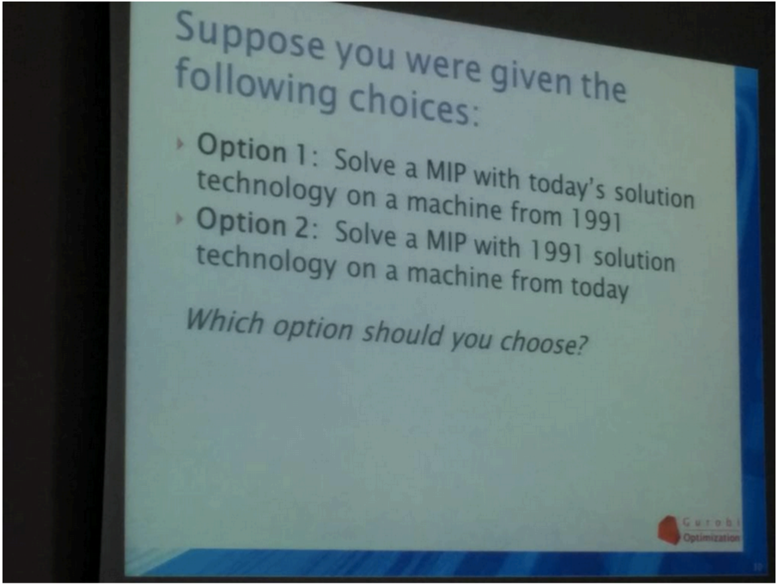
	Rows	Columns	Nonzeros
Original size	5034171	7365337	25596099
After presolve	1296075	2910559	10339042

Solution times were as follows:

Very large model—solution times

	Algorithm		
Version	Barrier	Dual	Primal
CPLEX 5.0	8642.6	350000.0	71039.7
CPLEX 7.1	5642.6	6413.1	1880.0

Source: Bixby, 2002


Suppose you were given the following choices:
Option 1: Solve a MIP with today's solution technology on a machine from 1991
Option 2: Solve a MIP with 1991 solution technology on a machine from today
Which option should you choose?
Gurobi Optimization