Solving Linear and Integer Programs

Robert E. Bixby
ILOG, Inc. and Rice University
Dual Simplex Algorithm
Some Motivation

- Dual simplex vs. primal: Dual > 2x faster
- Best algorithm of MIP
- There isn’t much in books about implementing the dual.
Dual Simplex Algorithm
(Lemke, 1954: Commercial codes ~1990)

Input: A dual feasible basis $B$ and vectors

$$X_B = A_B^{-1}b \quad \text{and} \quad D_N = c_N - A_N^T B^T c_B.$$ 

- Step 1: (Pricing) If $X_B \geq 0$, stop, $B$ is optimal; else let 

  $$i = \text{argmin}\{X_{Bk} : k \in \{1, \ldots, m\}\}.$$ 

- Step 2: (BTRAN) Solve $B^T z = e_i$. Compute $\alpha_N = -A_N^T z$.

- Step 3: (Ratio test) If $\alpha_N \leq 0$, stop, (D) is unbounded; else, let 

  $$j = \text{argmin}\{D_k/\alpha_k : \alpha_k > 0\}.$$ 

- Step 4: (FTRAN) Solve $A_B y = A_j$.

- Step 5: (Update) Set $B_i = j$. Update $X_B$ (using $y$) and $D_N$ (using $\alpha_N$)
Input: A dual feasible basis $B$ and vectors

$$X_B = A_B^{-1}b \quad \text{and} \quad D_N = c_N - A_N^T A_B^{-T} c_B.$$ 

- **Step 1:** (Pricing) If $X_B \geq 0$, stop, $B$ is optimal; else let
  
  $$i = \arg\min\{X_{Bk} : k \in \{1, \ldots, m\}\}.$$ 

- **Step 2:** (BTRAN) Solve $B^T z = e_i$. Compute $\alpha_N = -A_N^T z$.

- **Step 3:** (Ratio test) If $\alpha_N \leq 0$, stop, (D) is unbounded; else, let
  
  $$j = \arg\min\{D_k/\alpha_k : \alpha_k > 0\}.$$ 

- **Step 4:** (FTRAN) Solve $A_B y = A_j$.

- **Step 5:** (Update) Set $B_i = j$. Update $X_B$ (using $y$) and $D_N$ (using $\alpha_N$).
Implementing the Dual Simplex Algorithm
Implementation Issues for Dual Simplex

1. **Finding an initial feasible basis, or the concluding that there is none:** Phase I of simplex algorithm.

2. **Pricing:** Dual steepest edge

3. **Solving the linear systems**
   - LU factorization and factorization update
   - BTRAN and FTRAN – exploiting sparsity

4. **Numerically stable ratio test:** Bound shifting and perturbation

5. **Bound flipping:** Exploiting “boxed” variables to combine many iterations into one.
Issue 0
Preparation: Bounds on Variables

In practice, simplex algorithms need to accept LPs in the following form:

\[
\begin{align*}
\text{Minimize} & \quad c^T x \\
\text{Subject to} & \quad Ax = b \\
& \quad l \leq x \leq u
\end{align*}
\]

where \( l \) is an \( n \)-vector of lower bounds and \( u \) an \( n \)-vector of upper bounds. \( l \) is allowed to have \(-\infty\) entries and \( u \) is allowed to have \(+\infty\) entries. (Note that \((P_{BD})\) is in standard form if \( l_j = 0, u_j = +\infty \ \forall \ j \).)
A basis for $(P_{BD})$ is a triple $(B, L, U)$ where $B$ is an ordered $m$-element subset of $\{1, \ldots, n\}$ (just as before), $(B, L, U)$ is a partition of $\{1, \ldots, n\}$, $l_j > -\infty \ \forall \ j \in L$, and $u_j < +\infty \ \forall \ j \in U$. $N = L \cup U$ is the set of nonbasic variables. The associated (primal) basic solution $X$ is given by $X_L = l_L$, $X_U = u_U$ and

$$X_B = A_B^{-1}(b - A_Ll_L - A_Uu_U).$$

This solution is feasible if

$$l_B \leq X_B \leq u_B.$$

The associated dual basic solution is defined exactly as before: $D_B = 0$, $\Pi^T A_B = c_B^T$, $D_N = c_N - A_N^T \Pi$. It is dual feasible if

$$D_L \geq 0 \text{ and } D_U \leq 0.$$
(Issue 0 – Bounds on variables)  
The Full Story

- Modify simplex algorithm
  - Only the “Pricing” and “Ratio Test” steps must be changed substantially.
  - The complicated part is the ratio test
- Reference: See Chvátal for the primal
Issue 1
The Initial Feasible Basis – Phase I

- Two parts to the solution
  1. Finding some initial basis (probably not feasible)
  2. Modified simplex algorithm to find a feasible basis

(Issue 1 – Initial feasible basis)

Initial Basis

- Primal and dual bases are the same. We begin in the context of the primal. Consider

\[
\begin{align*}
\text{Minimize} & \quad c^T x \\
\text{Subject to} & \quad Ax = b \quad (P_{BD}) \\
& \quad l \leq x \leq u
\end{align*}
\]

- **Assumption:** Every variable has some finite bound.

- **Trick:** Add artificial variables \( x_{n+1}, \ldots, x_{n+m} \):

\[
Ax + I \begin{pmatrix}
  x_{n+1} \\
  \vdots \\
  x_{n+m}
\end{pmatrix} = b
\]

where \( l_j = u_j = 0 \) for \( j = n+1, \ldots, n+m \).

- **Initial basis:** \( B = (n+1, \ldots, n+m) \) and for each \( j \notin B \), pick some finite bound and place \( j \) in \( L \) or \( U \), as appropriate.

- **Free Variable Refinement:** Make free variables non-basic at value 0. This leads to a notion of a superbasis, where non-basic variables can be between their bounds.
(Issue 1 – Initial feasible basis)
Solving the Phase I

- If the initial basis is not dual feasible, we consider the problem:

$$\text{Maximize } \sum (d_j : d_j < 0)$$
$$\text{Subject to } A^T \pi + d = c$$

- This problem is “locally linear”: Define $\kappa \in \mathbb{R}^n$ by $\kappa_j = 1$ if $D_j < 0$, and 0 otherwise. Let

$$K = \{j : D_j < 0\} \text{ and } K^c = \{j : D_j \geq 0\}$$

Then our problem becomes

$$\text{Maximize } \kappa^T d$$
$$\text{Subject to } A^T \pi + d = c$$
$$d_K \leq 0, \quad d_{K^c} \geq 0$$

- Apply dual simplex, and whenever $d_j$ for $j \in K$ becomes 0, move it to $K$. 
Solving Phase I: An Interesting Computation

- Suppose $d_{Bi}$ is the entering variable. Then $X_{Bi} < 0$ where $X_B$ is obtained using the following formula:

$$X_B = A_B^{-1} A_N \kappa$$

- Suppose now that $d_j$ is determined to be the leaving variable. Then in terms of the phase I objective, this means $\kappa_j$ is replace by $\kappa_j + \varepsilon e_j$, where $\varepsilon \in \{0, +1, -1\}$. It can then be shown that

$$x_{Bi} = X_{Bi} + \varepsilon \alpha_j$$

- **Conclusion:** If $x_{Bi} < 0$, then the current iteration can continue without the necessity of changing the basis.

- **Advantages**
  - Multiple iterations are combined into one.
  - $x_{Bi}$ will tend not to change sign precisely when $\alpha_j$ is small. Thus this procedure tends to avoid unstable pivots.
The textbook rule is TERRIBLE: For a problem in standard form, select the entering variable using the formula
\[ j = \arg\min\{X_{Bi} : i = 1, \ldots, m\} \]

Geometry is wrong: Maximizes rate of change relative to axis; better to do relative to edge.

Goldfard and Forrest 1992 suggested the following steepest-edge alternative
\[ j = \arg\min\{X_{Bi} / \eta_i : i = 1, \ldots, m\} \]
where \( \eta_i = \|e_i^TA_B^{-1}\|_2 \), and gave an efficient update.

Note that there are two ingredients in the success of Dual SE:
- Significantly reduced iteration counts
- The fact that there is a very efficient update for \( \eta_i \)
Example: Pricing
Model: dfl001

Pricing: Greatest infeasibility
Dual simplex - Optimal: Objective = 1.1266396047e+07
Solution time = 1339.86 sec. Iterations = 771647 (0)

Pricing: Goldfarb-Forrest steepest-edge
Dual simplex - Optimal: Objective = 1.1266396047e+07
Solution time = 24.48 sec. Iterations = 18898 (0)
**Issue 3**

**Solving FTRAN, BTRAN**


- **Updating the Factorization:** Forrest-Tomlin update is the method of choice. See Chvátal Chapter 24.
  - There are multiple, individually relatively minor tweaks that collectively have a significant effect on update efficiency.

- **Further exploiting sparsity:** This is the main recent development.
We must solve two linear systems per iteration:

\[
\begin{align*}
A_B y &= A_j \\
A_B^T z &= e_i
\end{align*}
\]

where

\[
\begin{align*}
A_B &= \text{basis matrix} \quad \text{(very sparse)} \\
A_j &= \text{entering column} \quad \text{(very sparse)} \\
e_i &= \text{unit vector} \quad \text{(very sparse)}
\end{align*}
\]

\[\Rightarrow y \text{ an } z \text{ are typically very sparse}\]

**Example:** Model pla85900 (from TSP)

- Constraints: 85900
- Variables: 144185
- Average $|y|$: 15.5
\[ A_B = \begin{bmatrix} U \\ L \end{bmatrix} \]

**Triangular solve:** \[ Lw = A_j \quad (A_By = L(Uy) = A_j) \]

**Graph structure:** Define an acyclic digraph \( D = (\{1, \ldots, m\}, E) \) where \( (i, j) \in E \iff l_{ij} \neq 0 \) and \( i \neq j \).

**Solving using \( D \):** Let \( X = \{ i \in V : A_{ij} \neq 0 \} \). Compute \( X = \{ j \in V : \exists \text{ a directed path from } j \text{ to } X \} \). \( X \) can be computed in time linear in \( |E(X)| + |X| \).
**PDS Models**


<table>
<thead>
<tr>
<th>MODEL</th>
<th>ROWS</th>
<th>CPLEX1.0</th>
<th>CPLEX5.0</th>
<th>CPLEX8.0</th>
<th>SPEEDUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>pds02</td>
<td>2953</td>
<td>1988</td>
<td>1997</td>
<td>2002</td>
<td>1.0→8.0</td>
</tr>
<tr>
<td>pds06</td>
<td>9881</td>
<td>0.4</td>
<td>0.1</td>
<td>0.1</td>
<td>4.0</td>
</tr>
<tr>
<td>pds10</td>
<td>16558</td>
<td>26.4</td>
<td>2.4</td>
<td>0.9</td>
<td>29.3</td>
</tr>
<tr>
<td>pds20</td>
<td>33874</td>
<td>16558</td>
<td>208.9</td>
<td>13.0</td>
<td>80.3</td>
</tr>
<tr>
<td>pds30</td>
<td>49944</td>
<td>5268.8</td>
<td>232.6</td>
<td>20.9</td>
<td>247.3</td>
</tr>
<tr>
<td>pds40</td>
<td>66844</td>
<td>15891.9</td>
<td>1154.9</td>
<td>39.1</td>
<td>406.4</td>
</tr>
<tr>
<td>pds50</td>
<td>83060</td>
<td>58920.3</td>
<td>2816.8</td>
<td>79.3</td>
<td>743.0</td>
</tr>
<tr>
<td>pds60</td>
<td>99431</td>
<td>122195.9</td>
<td>8510.9</td>
<td>114.6</td>
<td>1066.3</td>
</tr>
<tr>
<td>pds70</td>
<td>114944</td>
<td>205798.3</td>
<td>7442.6</td>
<td>160.5</td>
<td>1282.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Primal Simplex</th>
<th>Dual Simplex</th>
<th>Dual Simplex</th>
</tr>
</thead>
</table>

20
Not just faster -- Growth with size: Quadratic *then* & Linear *now*!
Issue 4
Ratio Test and Finiteness

The “standard form” dual problem is

Maximize \( b^T \pi \)
Subject to \( A^T \pi + d = c \)
\[ d \geq 0 \]

Feasibility means \( d \geq 0 \)

However, in practice this condition is replaced by

\[ d \geq -\varepsilon e \]

where \( e^T= (1, \ldots, 1) \) and \( \varepsilon = 10^{-6} \). Reason: Degeneracy.

In 1972 Paula Harris proposed suggested exploiting this fact to improve numerical stability.
**Motivation:** Feasibility \( \Rightarrow \) step length \( \theta \) satisfies

\[
D_N - \theta \alpha_N \geq 0
\]

However, the bigger the step length, the bigger the change in the objective. So, we choose

\[
\theta_{\text{max}} = \min\{D_j/\alpha_j : \alpha_j > 0\}
\]

Using \( \varepsilon \), we have

\[
\theta_{\varepsilon \text{ max}} = \min\{(D_j + \varepsilon)/\alpha_j : \alpha_j > 0\} > \theta_{\text{max}}
\]
(Issue 4 – Ratio test & finiteness)

- **Advantages**
  - Numerical stability – $\alpha_{jenter} = \text{“pivot element”}$
  - Degeneracy – Reduces # of 0-length steps

- **Disadvantage**
  - $D_{jenter} < 0 \Rightarrow$ objective goes in wrong direction

- **Solution: BOUND SHIFTING**
  - If $D_{jenter} < 0$, we replace the lower bound on $d_{jenter}$ by something less than its current value.
  - Note that this shift changes the problem and must be removed: 5% of cases, this produces dual infeasibility $\Rightarrow$ process is iterated.
Example: Bound-Shifting Removal

Problem 'pilot87.sav.gz' read. Reduced LP has 1809 rows, 4414 columns, and 70191 nonzeros.

Iteration log . . .
Iteration:  1  Scaled dual infeas =  0.697540
Iteration:  733  Scaled dual infeas =  0.000404
Iteration:  790  Dual objective =  -185.892207
...
Iteration:  16326  Dual objective =  302.786794
Removing shift (3452).
Iteration:  16417  Scaled dual infeas =  0.207796
Iteration:  16711  Scaled dual infeas =  0.000021
Iteration:  16726  Dual objective =  296.758656
Elapsed time = 104.36 sec. (17000 iterations).
Iteration:  17072  Dual objective =  300.965492
...
Iteration:  17805  Dual objective =  301.706409
Removing shift (76).
Iteration:  17919  Scaled dual infeas =  0.000060
Iteration:  17948  Dual objective =  301.708660
Elapsed time = 114.42 sec. (18000 iterations).
Removing shift (10).
Iteration:  18029  Scaled dual infeas =  0.000050
Iteration:  18039  Dual objective =  301.710058
Removing shift (1).
Dual simplex - Optimal: Objective = 3.0171034733e+002
Solution time = 116.44 sec. Iterations = 18095 (1137)
(Issue 4 – Ratio test & finiteness)

**Finiteness:** Bound shifting is closely related to the “perturbation” method employed in CPLEX if no progress is being made in the objective.

“**No progress**” \( \Rightarrow \)

\[
d_j \geq -\varepsilon \quad j = 1, ..., n
\]

is replaced by

\[
d_j \geq -\varepsilon - \varepsilon_j \quad j = 1, ..., n,
\]

where \( \varepsilon_j \) is random uniform on \([0, \varepsilon]\).
Issue 5
Bound Flipping

- A basis is given by a triple (B,L,U)
  - L = non-basics at lower bound: Feasibility $D_L \geq 0$
  - U = non-basics at upper bound: Feasibility $D_U \leq 0$

- Ratio test: Suppose $X_{Bi}$ is the leaving variable, and the step length is blocked by some variable $d_j, j \in L$, that is about to become negative and such that $u_j < +\infty$:
  - Flipping means: Move $j$ from $L$ to $U$.
  - Check: Do an update to see if $X_{Bi}$ is still favorable (just as we did in Phase I!)

- Can combine many iterations into a single iteration.
Example: Bound Flipping

Problem 'fit2d.sav.gz' read.
Initializing dual steep norms . . .

Iteration log . . .
Iteration:  1  Dual objective =  -80412.550000
Perturbation started.
Iteration:  203  Dual objective =  -80412.550000
Iteration:  1313  Dual objective =  -80412.548666
Iteration:  2372  Dual objective =  -77028.548350
Iteration:  3413  Dual objective =  -71980.245530
Iteration:  4316  Dual objective =  -70657.605570
Iteration:  5151  Dual objective =  -68994.477061
Iteration:  5820  Dual objective =  -68472.659371
Removing perturbation.

Dual simplex - Optimal: Objective =  -6.8464293294e+004
Solution time =  18.74 sec.  Iterations = 5932 (0)

Problem 'fit2d.sav.gz' read.
Initializing dual steep norms . . .

Iteration log . . .
Iteration:  1  Dual objective =  -77037.550000

Dual simplex - Optimal: Objective =  -6.8464293294e+004
Solution time =  1.88 sec.  Iterations = 201 (0)