Solving Linear and Integer Programs

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

- Dual simplex vs. primal (2002): Dual 2.7x 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  
$$i = \arg \min \{X_{Bk} : k \in \{1, \ldots, m\}\}.$$  

☑ **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 $\alpha_N$)
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 \( \alpha_N \)).
Implementing the Dual Simplex Algorithm
Implementation Issues for Dual Simplex

1. Finding an initial feasible basis, or the concluding that there is none
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:

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

where \( l \) is an \( n \)-vector of lower bounds and \( u \) an \( n \)-vector of upper bounds. In general, \( l \) is allowed to have \(-\infty\) entries and \( u \) is allowed to have \(+\infty\) entries. (Note that \((P_{BD})\) is in standard form if \( l_j = 0, u_j = +\infty \forall j \).) Assuming all upper and lower bounds are finite, the corresponding dual is:

$$\begin{align*}
\text{Maximize} & \quad b^T \pi + l^T r - u^T s \\
\text{Subject to} & \quad A^T \pi + r - s = c \\
& \quad \pi \text{ free, } r \geq 0, s \geq 0
\end{align*} \quad (D_{BD})$$
A basis for \((P_{BD})\) is a triple \((B,L,U)\) where \(B\) is an ordered \(m\)-element subset of \(\{1,\ldots,n\}\) (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 primal feasible if
\[
l_B \leq X_B \leq u_B.
\]

The associated dual basic variables are \(\pi, \ r_L, \) and \(s_U\) with values:
\[
\Pi = A_B^{-T}c_B, \ R_L = c_L - A_L^T\Pi, \) and \(S_U = -c_U + A_U^T\Pi. \) It is dual feasible if
\[
R_L \geq 0 \ and \ S_U \geq 0.
\]
(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

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

- **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 \(j \not\in B\), pick some finite bound and place \(j\) in \(L\) or \(U\), as appropriate.
(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

  \[ K = \{ j : D_j < 0 \} \text{ and } \overline{K} = \{ j : D_j \geq 0 \} \]

  Then our problem becomes

  \[ \text{Maximize } \kappa^T d \]

  \[ \text{Subject to } A^T \pi + d = c \]

  \[ d_K \leq 0, \ d_{\overline{K}} \geq 0 \]

- Apply dual simplex, and whenever \( d_j \) for \( j \in K \) becomes 0, move it to \( \overline{K} \).
Imagine performing the ratio test to determine which $d_j$ will leave the basis given some $d_{Bi}$ is entering:

Case 1: $d_j < 0$ and hits 0  
Case 2: $d_j > 0$ and hits 0

Consider Case 2. Then a further increase in $d_{Bi}$ will make $d_j < 0$.

This can be handled by updating K. But is it desirable?

Update formula for “reduced cost”:

$$new\_reduced\_cost = old\_reduced\_cost \pm y_i$$

If the reduced cost does not change sign, we have a cheap update and can continue the step.

Note that this also improves numerical stability
The textbook rule: Choose the largest primal violation is TERRIBLE: For a problem in standard form

\[ j = \arg\min\{X_{Bi} : i = 1, \ldots, m\} \]

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 \( \eta_i = \|e_i^T A_B^{-1}\|_2 \), and gave an efficient update.
(Issue 2 – Pricing)
Dual Steepest Edge

- **Idea:** Compute the rate of change of the objective per unit movement along the “corresponding” edge of the polyhedron of feasible solutions.

- **Setup**
  - Assume the problem is in standard form with a dual basic feasible solution specified by a basis $B$.
  - $d_{Bi} =$ entering variable
    - $X_{Bi} < 0$
    - $d_{Bi} = \theta > 0 \Rightarrow \Delta \text{objective} = - \theta X_{Bi} > 0$
(Issue 2 – Pricing)
Dual Steepest Edte

- Old solution vector:
  \[ d_B = 0 \quad d_N = D_N \quad \pi = \Pi \]

- New solution vector:
  \[ d_B = \theta e_i \quad d_N = D_N - \theta \alpha_N \quad \pi = \Pi - \theta z \]
where \( \alpha_N = -A_N^T z \) and \( A_B^T z = e_i \).

- Hence the change in the solution vector for \( \theta = 1 \) is given by
  \[ \Delta d_B = e_i \quad \Delta d_N = - \Delta \alpha_N \quad \Delta \pi = z \]
(Issue 2 – Pricing)
Dual Steepest Edge

- Hence the change in the solution vector for $\theta=1$ is given by

$$\Delta d_B = e_i \quad \Delta d_N = -\Delta \alpha_N \quad \Delta \pi = z$$

And so the rate of change of the objective per unit movement along the edge is given by

$$x_{Bi} / \sqrt{e_i^T e_i + \alpha_N^T \alpha_N + z^T z}$$

- Goldfarb and Forrest observation: Projection onto the space of the $\pi$ variables gives equally good iteration counts and is much simpler to compute

$$x_{Bi} / \sqrt{z^T z} = x_{Bi} / \| e_i^T A_B^{-1} \|_2$$
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.

- **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)

<table>
<thead>
<tr>
<th>Constraints</th>
<th>85900</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables</td>
<td>144185</td>
</tr>
<tr>
<td>Average $</td>
<td>y</td>
</tr>
</tbody>
</table>
Triangular solve: \( Lw = A_j \) \( (A_Bw = L(Uy) = A_j) \)

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></td>
<td></td>
<td>1988</td>
<td>1997</td>
<td>2002</td>
<td>1.0 → 8.0</td>
</tr>
<tr>
<td><em>pds02</em></td>
<td>2953</td>
<td>0.4</td>
<td>0.1</td>
<td>0.1</td>
<td>4.0</td>
</tr>
<tr>
<td><em>pds06</em></td>
<td>9881</td>
<td>26.4</td>
<td>2.4</td>
<td>0.9</td>
<td>29.3</td>
</tr>
<tr>
<td><em>pds10</em></td>
<td>16558</td>
<td>208.9</td>
<td>13.0</td>
<td>2.6</td>
<td>80.3</td>
</tr>
<tr>
<td><em>pds20</em></td>
<td>33874</td>
<td>5268.8</td>
<td>232.6</td>
<td>20.9</td>
<td>247.3</td>
</tr>
<tr>
<td><em>pds30</em></td>
<td>49944</td>
<td>15891.9</td>
<td>1154.9</td>
<td>39.1</td>
<td>406.4</td>
</tr>
<tr>
<td><em>pds40</em></td>
<td>66844</td>
<td>58920.3</td>
<td>2816.8</td>
<td>79.3</td>
<td>743.0</td>
</tr>
<tr>
<td><em>pds50</em></td>
<td>83060</td>
<td>122195.9</td>
<td>8510.9</td>
<td>114.6</td>
<td>1066.3</td>
</tr>
<tr>
<td><em>pds60</em></td>
<td>99431</td>
<td>205798.3</td>
<td>7442.6</td>
<td>160.5</td>
<td>1282.2</td>
</tr>
<tr>
<td><em>pds70</em></td>
<td>114944</td>
<td>335292.1</td>
<td>21120.4</td>
<td>197.8</td>
<td>1695.1</td>
</tr>
</tbody>
</table>

Primal Simplex    Dual Simplex    Dual Simplex
Not just faster -- Growth with size: Quadratic *then* & Linear *now*!

![Graph showing quadratic and linear growth over time periods ranging from PDS02 to PDS70.](chart.png)
The “standard form” dual problem is

\[ \text{Maximize} \quad b^T \pi \]
\[ \text{Subject to} \quad A^T \pi + d = c \]
\[ d \geq 0 \]

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 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

$$\theta^{\varepsilon}_{\text{max}} = \min\{(D_j + \varepsilon) / \alpha_j : \alpha_j > 0\} > \theta_{\text{max}}$$
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)
Gurobi Optimizer version 2.0.0
Copyright (c) 2009, Gurobi Optimization, Inc.

Read MPS format model from file cont1.mps.bz2
cont1: 160792 Rows, 80795 Columns, 440387 NonZeros
Optimize a model with 160792 Rows, 80795 Columns and 440387 NonZeros

Presolve removed 40397 rows and 40397 columns
Presolve time: 0.31 sec.
Presolved: 120395 Rows, 40398 Columns, 359593 Nonzeros

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Objective</th>
<th>Primal Inf.</th>
<th>Dual Inf.</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>handle free variables</td>
<td></td>
<td>0.0000000e+00</td>
<td>0s</td>
</tr>
<tr>
<td>17434</td>
<td>1.6725221e-02</td>
<td>6.416129e+01</td>
<td>0.0000000e+00</td>
<td>5s</td>
</tr>
<tr>
<td>20749</td>
<td>1.6929624e-02</td>
<td>4.681255e+00</td>
<td>0.0000000e+00</td>
<td>10s</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32371</td>
<td>2.2108293e-02</td>
<td>9.527316e+00</td>
<td>0.0000000e+00</td>
<td>101s</td>
</tr>
<tr>
<td>32953</td>
<td>2.2381550e-02</td>
<td>3.618798e+01</td>
<td>0.0000000e+00</td>
<td>110s</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37997</td>
<td>2.5924066e-02</td>
<td>1.204414e+02</td>
<td>0.0000000e+00</td>
<td>200s</td>
</tr>
<tr>
<td>38579</td>
<td>2.6442899e-02</td>
<td>6.255491e+01</td>
<td>0.0000000e+00</td>
<td>212s</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42853</td>
<td>3.0820162e-02</td>
<td>7.662419e+01</td>
<td>0.0000000e+00</td>
<td>300s</td>
</tr>
<tr>
<td>43400</td>
<td>3.1467196e-02</td>
<td>8.031314e+01</td>
<td>0.0000000e+00</td>
<td>311s</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46184</td>
<td>3.456856e-02</td>
<td>7.302474e+01</td>
<td>0.0000000e+00</td>
<td>372s</td>
</tr>
<tr>
<td>46822</td>
<td>3.6248845e-02</td>
<td>1.600513e-01</td>
<td>0.0000000e+00</td>
<td>386s</td>
</tr>
<tr>
<td>46994</td>
<td>3.6272914e-02</td>
<td>0.0000000e+00</td>
<td>0.0000000e+00</td>
<td>390s</td>
</tr>
<tr>
<td>47222</td>
<td>1.4881820e-02</td>
<td>0.0000000e+00</td>
<td>3.185893e+00</td>
<td>400s</td>
</tr>
<tr>
<td>47415</td>
<td>1.4864227e-02</td>
<td>0.0000000e+00</td>
<td>4.802191e+01</td>
<td>406s</td>
</tr>
<tr>
<td>47830</td>
<td>1.4649598e-02</td>
<td>0.0000000e+00</td>
<td>7.049439e-01</td>
<td>420s</td>
</tr>
<tr>
<td>48267</td>
<td>1.4450227e-02</td>
<td>0.0000000e+00</td>
<td>7.578008e+00</td>
<td>431s</td>
</tr>
<tr>
<td>48815</td>
<td>1.2095665e-02</td>
<td>0.0000000e+00</td>
<td>6.917880e-01</td>
<td>444s</td>
</tr>
<tr>
<td>49144</td>
<td>1.0459793e-02</td>
<td>0.0000000e+00</td>
<td>6.762116e-01</td>
<td>452s</td>
</tr>
<tr>
<td>49241</td>
<td>8.7824861e-03</td>
<td>0.0000000e+00</td>
<td>0.0000000e+00</td>
<td>471s</td>
</tr>
</tbody>
</table>

Switched to Primal

Solved in 49241 iterations and 471.05 seconds
Optimal objective 8.78248612e-03
Finiteness: Bound shifting is closely related to the “perturbation” method employed 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 \([0, \varepsilon]\).

Implementation detail: For a basis \( B \), we initially perturb only the bounds on the variables in \( d_N \). Bound perturbations are then introduced for other \( d_j \) variables when \( j \) enters \( N \) for the first time.
Issue 5
Bound Flipping

- If the current basis is not optimal, then there is a $B_i$ such that
  - Case 1: $X_{Bi} < l_{Bi}$, or
  - Case 2: $X_{Bi} > u_{Bi}$.

- Consider Case 2 (Case 1 is similar). Then the corresponding dual move is to consider increasing the dual non-basic variable $s_{Bi}$ to some $θ > 0$, leaving all other non-basics at 0. The resulting values of the basic variables are given by
  - $R_L^θ = R_L - θ \alpha_L \geq 0$
  - $S_U^θ = S_U + θ \alpha_U \geq 0$

- The maximum step length is then given by
  
  $$θ_{max} = \min\{θ_{max}^r, θ_{max}^s\}$$

  where
  
  $$θ_{max}^r = \min\{r_j/α_j: α_j > 0, j \in L\} \text{ and } θ_{max}^s = \min\{-s_j/α_j: α_j < 0, j \in U\}$$
Now suppose that $\theta_{\text{max}} = r_j/\alpha_j$, $\alpha_j > 0$. Then the normal simplex step would be to remove $r_j$ from the basis and replace it by $s_{Bi}$.

However, instead of doing this, we consider replacing $r_j$ by $s_j$ in the basis, which is possible if $u_j < +\infty$. Since $r_j$ and $s_j$ are dual slacks in the same constraint, and have opposite signs, $r_j$ becoming negative translates to $s_j$ becoming positive, and preserves dual feasibility.

That is, we consider setting $L \leftarrow L \backslash \{j\}$ and $U \leftarrow U \cup \{j\}$. This is a good idea $\iff$ the $\text{updated}_X Bi > u_{Bi}$. But it is easy to show that

$$\text{updated}_X Bi = X_{Bi} + \alpha_j (l_j - u_j) < X_{Bi} \quad \text{(since $\alpha_j > 0$)}.$$

So it is easy to determine whether this “flipping” is desirable: If $\text{updated}_X Bi > u_{Bi}$. In this case we obtain a cheap basis update and can continue with the ratio test.
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)