Pivot Rules for the Simplex Method

David Eigen
deigen@cs.nyu.edu

May 11, 2011

Abstract

Pivot selection, the choice of entering variable, is a crucial step in the Simplex method. Good choices can lead to a significant speedup in finding a solution to a linear program, while poor choices lead to very slow or even nonterminal progress. This report explores three widely used pivot heuristics: the Dantzig rule, Steepest-Edge, and Devex. The theoretical underpinnings of each are studied, and the different methods compared empirically.

1 Introduction

A key step in solving a linear program with the simplex method is to choose an entering variable for each pivot operation. Methods that make this selection are generally known as a pivot rules. The two major goals of a pivot rule are:

- Prevent cycling between states (in case of degeneracy)
- Enhance the speed of search by choosing good edges to traverse

While both goals are important, this report focuses on the second. Good choices can greatly speed up the number of steps to the optimal vertex. If the bookkeeping required for the choice is cheap enough, this can result in considerable performance gains.

In theory, the best pivot rule would construct the path with the shortest number of hops from start to optimum. Of course, solving this problem fully requires finding not only the optimal vertex, but also lower bounds on all path lengths to prove a
shortest path. Hence, in the best case it is harder than the linear program itself. For this reason, pivot rules rely on heuristics computed locally near the current vertex to make update decisions.

This report considers a linear program in standard form:

$$\begin{align*}
\text{minimize} & \quad c^T x \\
\text{subject to} & \quad Ax = b, \quad x \geq 0
\end{align*}$$

With dual problem

$$\begin{align*}
\text{maximize} & \quad b^T \pi \\
\text{subject to} & \quad A^T \pi + z = c, \quad z \geq 0
\end{align*}$$

In the primal, $A = (B, N)$ are the basic and non-basic variable constraint matrices. A simplex step chooses a column $A_s = N_s$ to move into the basis (using the pivot rule), and a column $A_r = B_r$ to move out of the basis. Here, I have chosen the index names $s$ for select and $r$ for remove. Selecting $s$ fixes a search direction $-B^{-1}Ne_s = -B^{-1}A_s$, and hence the index $r$ is determined to be a component $j$ at minimum distance from $x_j = 0$ in this direction, i.e. $r = \arg\min_{j \in B} x_j / [B^{-T}A_s]_j$ (if there is more than one such choice, the destination vertex is degenerate).

## 2 Dantzig Rule

A simple and effective rule developed by Dantzig [5] simply chooses the variable with most negative multiplier $z_s = \min_i z_i$ to be the entering column. This choice maximizes the rate of decrease of the objective in the search direction $p_i$:

\[ p_i \cdot \nabla_x (c^T x) = \begin{pmatrix} -B^{-1}A_i \\ e_i \end{pmatrix} \cdot c^T = -c^T B^{-1}A_i + c_i = -\pi^T BB^{-1}A_i + c_i = c_i - A_i^T \pi = z_i \]

Note that here, each $p_i$ has been constructed so that its $i$th component is 1. Thus, $z_i$ is the rate of change of $c^T x$ per unit difference of $x_i$. 

2 Dantzig Rule
3 Steepest-Edge

Steepest-Edge is a similar heuristic to the Dantzig rule, in that it chooses an entering variable with largest rate of decrease in the objective. However, it measures the rate of decrease per distance traveled along the edge of traversal, rather than per unit $x_i$. In other words, while the Dantzig rule essentially measures the rate of change as a function of $x_i$, Steepest-Edge parameterizes by arc length along the search edge. The directional derivative is then

$$\frac{p_i}{||p_i||} \cdot \nabla_x (c^T x) = \frac{z_i}{||p_i||}$$

To compute the $||p_i||$, Goldfarb and Reid [7] derived a recurrence relation between simplex iterations. Letting $\gamma_i = ||p_i||^2$, for $i \in \mathbb{N}$,

$$\bar{\gamma}_r = \gamma_s / (e_r^T p_s)^2$$

$$\bar{\gamma}_i = \gamma_i - 2 \left( \frac{e_r^T p_i}{e_r^T p_s} \right)^2 p_i^T p_s + \left( \frac{e_r^T p_i}{e_r^T p_s} \right)^2 \gamma_s, \quad i \neq s$$

Where a bar over a variable (e.g. $\bar{\gamma}_i$) indicates that it is the value for the next simplex iteration.

The usual derivation uses the Sherman-Morrison formula on $\bar{B} = B + (A_s - A_r)e_r^T$, the rank-one update that replaces column $A_r$ with $A_s$, followed by a long but straightforward calculation [6, 7, 11]. Below, however, I provide a more informal geometric motivation.

Intuitively, the recurrence can be seen by considering that the next edge directions are

$$\bar{p}_i = p_i - \frac{e_r^T p_i}{e_r^T p_s} p_s$$

in which case equation (2) is simply $\bar{p}_i^T \bar{p}_i$ expanded out.

A picture illustrating this edge update in a special case with three variables and one constraint is shown in Figure 1. Since the LP is in standard form, the constraint
Figure 1: In this simple 3-variable case, the new edge $\overline{p}_i$ to vertex $v_i$ is the difference between the edge $p_i$ from the current vertex, and the edge $p_s$ of the simplex step.

plane makes a triangle in the region $x_i \geq 0$, and the vertices are its intersections with the coordinate axes. The old direction $p_i$ corresponds to the edge between a vertex $v_i$ and the current vertex, while the new direction $\overline{p}_i$ corresponds to the edge between this same $v_i$ and the vertex reached after the simplex step.

In higher dimensions, the vertex $v_i$ pointed to by $p_i$ will generally not remain the same after the update. However, the difference $\overline{p}_i = p_i - \frac{c_i}{c_i^T p_i} p_s$ still corresponds to an edge to a vertex in the new reachable set.

Note that the $r$-th component of $\overline{p}_i$ is indeed zero, as required by the simplex method: if $\overline{p}_i$ is chosen as the entering variable following the current choice, then that step must keep $x_r = 0$, since by then $x_r$ has transitioned out of the basis.

Computing equation (2) by explicitly finding the directions $p_i$ is computationally too expensive for a heuristic; indeed, it is no better than finding the largest decrease in $c^T x$ among all neighboring vertices. However, the recurrence can be rephrased by re-expanding $I_B p_i = -B^{-1}A_i$ for $i \in N$, where $I_B p_i$ is the projection of $p_i$ to only the basic components. Then (2) becomes
\[ \bar{\gamma}_i = \gamma_i - 2 \left( \frac{(B^{-T}e_r)^T A_i}{(B^{-T}e_s)^T A_s} \right) A^T B^{-T} B^{-1} A_s + \left( \frac{(B^{-T}e_r)^T A_i}{(B^{-T}e_s)^T A_s} \right)^2 \gamma_s \]  

\[ = \gamma_i - 2 \left( \frac{\sigma^T A_i}{\sigma^T A_s} \right) w^T A_i + \left( \frac{\sigma^T A_i}{\sigma^T A_s} \right)^2 \gamma_s, \quad i \neq s \]  

By solving \( B^T \sigma = e_r \) and \( BB^T w = A_s \), we can compute (4) using the dot products of these vectors with \( A_i \) and \( A_s \). [6, 7]

4 Devex

Devex is an approximation to steepest-edge, developed by Harris [8]. Historically, it was developed before the recurrence (4) was derived. Both methods are in use today, since either may be more efficient depending on the problem.

Devex maintains approximations \( u_i \approx \|p_i\|, \quad i \in N \). Periodically, a reference frame \( F \) is fixed to be the current non-basic index set, and initial \( u_i \) are set to 1. In each simplex iteration, the norm estimates are updated by

\[ \bar{u}_r = \max \left( 1, \frac{\|p_s\|}{e_r^T p_s} \right) \]  

\[ \bar{u}_i = \max \left( u_i, \frac{e_r^T p_i}{e_r^T p_s} \|p_s\| \right), \quad i \neq r \]

where the norms are taken only over the components in \( F \).

Equation (6) approximates the length \( \|p_i - \frac{e_r^T p_i}{e_r^T p_s} p_s\| \approx \max(\|p_i\|, \|\frac{e_r^T p_i}{e_r^T p_s} p_s\|) \). Equation (5) truncates small estimates of the traversed edge to 1, which can happen if the direction \( p_s \) is roughly orthogonal to the reference frame. In this case, simply guessing an estimate of 1 generally leads to better performance [8].

In the tableau architecture used by Harris [8], the slopes \( \frac{e_r^T p_i}{e_r^T p_s} \) correspond to the coefficients for column operations already performed on \( N \), so the method requires almost no additional computation. However, only the coefficients for dimensions in
$F \cap N$ are known for free, so the norms are approximated using only these dimensions. Thus, the framework loses its coverage of the space as $N$ evolves.

Two features of note are that the length estimates $u_i$ can only increase, and that the estimate of the leaving variable $u_r$ may be poor if $\|p_i\|_{c[p]}$ is small and truncation occurs. As discussed above, the effective reference frame loses dimensions over time, so the latter may happen frequently once many pivots have been performed. For these reasons, the framework $F$ is periodically reset to the current nonbasic indices $N$, and the norms $u_i$ set back to 1.

5 lp_solve Implementation

I reviewed the implementation of pivot methods in the lp_solve codebase [1], which I used for computational experiments described in Section 6.

lp_solve provides four pivot methods itself:

- First-column
- Dantzig
- Steepest-Edge
- Devex-like steepest-edge approximation

In addition to these, I implemented a second devex-like approximation, described towards the end of this section.

The First-column method simply selects the first column with negative multiplier.

lp_solve’s “devex” implementation is actually a devex-like approximation of steepest-edge. Instead of using the max approximation for vector sum and periodically fixing a reference frame as described in Section 4, it instead calculates the slope terms using the method described in Section 3, by finding $B^T \sigma = e_r$, and using the approximate recurrence

$$\bar{\gamma}_i \geq \gamma_i + \left( \frac{\sigma^T A_i}{\sigma^T A_s} \right)^2 \gamma_s, \quad i \neq r$$

The reference frame $F$ is therefore always the full space. The resulting weights are arguably better estimates than in the original devex, but not much cheaper than
steepest edge: the only difference is that \( w = B^{-T}B^{-1}A_s \) need not be computed. As in the original devex, the weights are reset if any one becomes larger than a threshold.

To better compare against the original devex description, I wrote a second devex-like approximation that uses the recurrence

\[
\tilde{\gamma}_i \approx \max \left( \gamma_i, \left( \frac{\sigma^T A_i}{\sigma^T A_s} \right)^2 \gamma_s \right), \quad i \neq r
\]

The reference frame \( F \) in this case is still the whole space.

6 Experiments

Using the lp_solve codebase [1], I compared different pivot methods using the netlib dataset [10]. I ran each netlib problem using the primal simplex algorithm for both phases 1 and 2.

Each netlib problem was solved using each of the five pivot methods described in Section 5. The maximum number of iterations allowed was capped at 2 million. Runs that did not find a solution in this amount of time were recorded as having failed. Other types of failures also occurred at times, particularly on very degenerate problems, for which the solver sometimes diverged due to errors.

For each successfully solved problem, I recorded the number of iterations performed as well as CPU time. For each of these measurements, problems were broken down into three groups by percentile rank: those in the hardest 5%, next 10%, and easiest 85%. Mean and standard deviation were computed within each group. The results are shown in Section 7. For the 5% quantile, the minimum and maximum were used instead of standard deviation, because only three problems were in the group.

This analysis was performed twice: once including only those problems successfully solved by all pivot methods, and once treating each pivot method individually.
7 Results

Table 1 shows iteration count and CPU time for each netlib model. Omitted are models solved by all methods in under 500 iterations.

First-column selection takes far longer to complete than any of the descent heuristics; in fact, it fails to solve many of the problems. This clearly demonstrates the importance of good pivot selection. Furthermore, out of the models where first-column was successful, the descent-based selection methods perform better, as shown in Figure 2.

Figures 3 and 4 show results excluding the first-column method. Methods considered in these figures are Dantzig, Devex with sum approximation (“devex1”), Devex with max approximation (“devex2”), and Steepest-Edge. Figure 3 shows the average number of iterations and CPU time for problems successfully solved by each method. Figure 4 shows the same measurements, but only among the problems successfully solved by all methods. Error bars indicate ±1 standard deviation, except in the 5% quantile group, where they are minimum and maximum values, as explained in
Figure 3: Comparison between pivot methods. Each point is an average among problems successfully solved by each single pivot method. Problems were broken down by quantile rank within each individual method.

Section 6.

In the easiest problems, the pivot algorithm seems not to matter very much: in both Figures 3 and 4, there is little difference within the 100% quantile group.

In harder problems, devex2, the max-value length approximation, appears to be best overall, followed by devex1, which uses the sum approximation. As expected, dantzig fairs worse (see Figure 4), though it does perform relatively well.

Surprisingly, steepest-edge takes more iterations on average than the devex-like approximations. A reason for this may be that numerical errors can make some edge lengths unreliable, particularly in degenerate problems, when there are many zero-length edges. These can lead to division by small values when computing the slopes. The devex-like algorithm in lp_solve is able to reset these approximations if any get too large, while this safeguard is not done for steepest-edge. Even so, there are some problems where steepest-edge can be better, such as “maros”.

In addition, the failures in Table 1 often happen in different problems for different
methods. This indicates that it may be advantageous to try several methods when solving a problem, since any may work best depending on the structure.

8 Conclusion

Pivot selection is an important component of the Simplex method, as illustrated by these experiments. Widely-used heuristics such as Devex and Steepest-Edge make choices by approximating the speeds of descent along the edges, which are used as indicators of the decrease in objective at the next vertex.

Even for easy problems, these descent heuristics provide a significant speedup over arbitrary column choices. Still harder problems require better selections to make good progress. The exact method that works best is problem-dependent, though devex-style steepest-edge approximations tend to work well in most cases.
References


Table 1: Results for netlib models. Problems that were successfully solved by all methods in under 500 iterations are omitted.