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:

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

With dual problem

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

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. $r = \text{argmin}_{j \in B} x_j / [B^{-T} A_s]_j$ (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:

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

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

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 $i \in N$,

$$\begin{aligned} \bar{\gamma}_r &= \gamma_s/(e_r^T p_s)^2 & (1) \\ \bar{\gamma}_i &= \gamma_i - 2\left(\frac{e_r^T p_i}{e_r^T p_s}\right) p_i^T p_s + \left(\frac{e_r^T p_i}{e_r^T p_s}\right)^2 \gamma_s, \quad i \neq s & (2) \end{aligned}$$

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

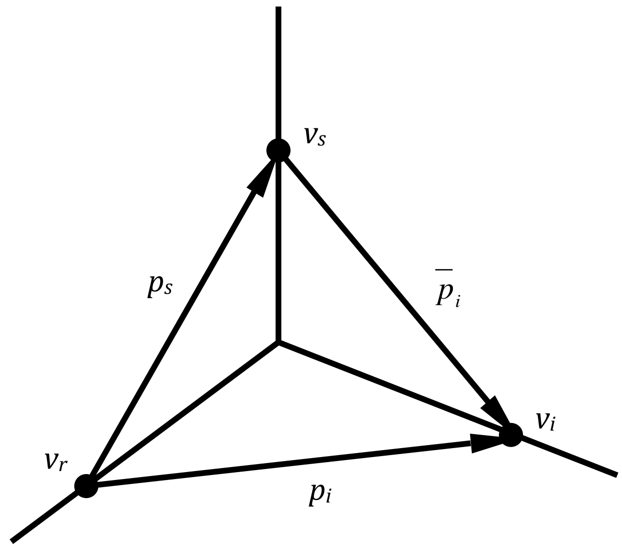

Figure 1: In this simple 3-variable case, the new edge $\bar{p}_i$ to vertex v_i is the difference between the edge p_i from the current vertex, and the edge p_s of the simplex step.

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 $\bar{p}_i$ 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 $\bar{p}_i = p_i - \frac{e_r^T p_i}{e_r^T p_s} p_s$ still corresponds to an edge to a vertex in the new reachable set.

Note that the r-th component of $\bar{p}_i$ is indeed zero, as required by the simplex method: if $\bar{p}_i$ 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

$$\bar{\gamma}_i = \gamma_i - 2\left(\frac{(B^{-T}e_r)^T A_i}{(B^{-T}e_r)^T A_s}\right) A_i^T B^{-T} B^{-1} A_s + \left(\frac{(B^{-T}e_r)^T A_i}{(B^{-T}e_r)^T A_s}\right)^2 \gamma_s \quad (3)$$

$$= \gamma_i - 2\left(\frac{\sigma^T A_i}{\sigma^T A_s}\right) w^T A_i + \left(\frac{\sigma^T A_i}{\sigma^T A_s}\right)^2 \gamma_s, \quad i \neq s \quad (4)$$

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 $u_i \approx ||p_i||$, $i \in N$. 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

$$\bar{u}_r = \max\left(1, \quad \frac{||p_s||}{|e_r^T p_s|}\right) \quad (5)$$

$$\bar{u}_i = \max\left(u_i, \quad \left|\frac{e_r^T p_i}{e_r^T p_s}\right| ||p_s||\right), \quad i \neq r \quad (6)$$

where the norms are taken only over the components in F.

Equation (6) approximates the length $||p_i - \frac{e_r^T p_j}{e_r^T p_s} p_s|| \approx \max(||p_i||, ||\frac{e_r^T p_j}{e_r^T p_s} p_s||)$. 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 $\frac{||p_s||}{|e_r^T p_s|}$ 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

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

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

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

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.

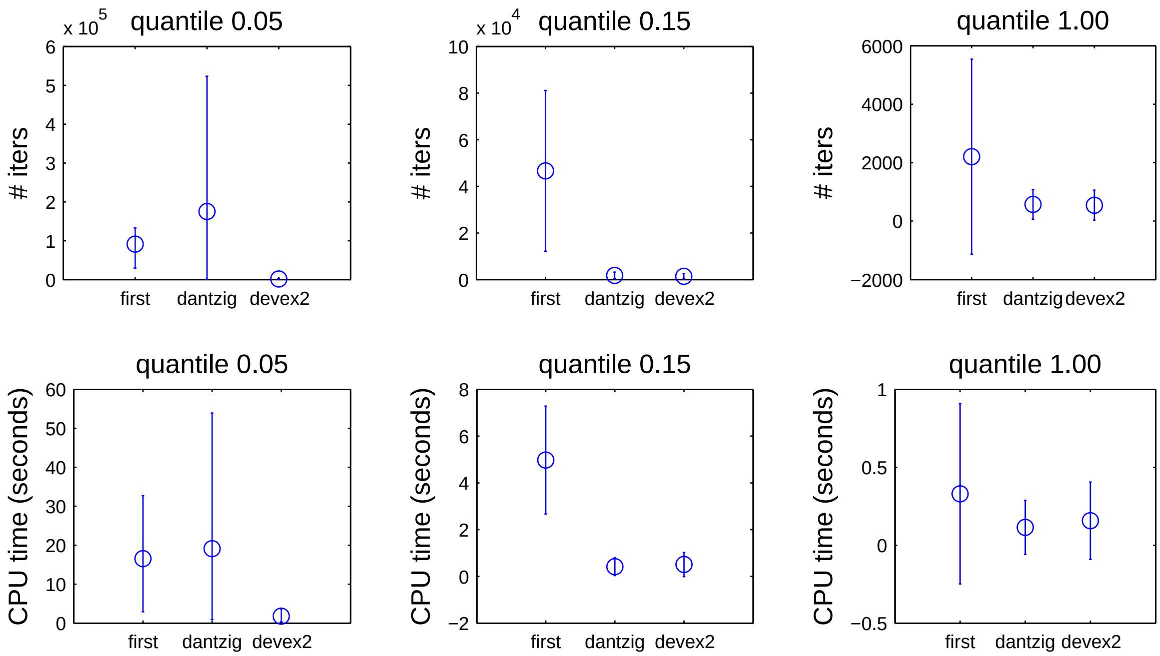

Figure 2: Comparison between first-column selection and descent heuristics among problems able to be solved by the first-column method.

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

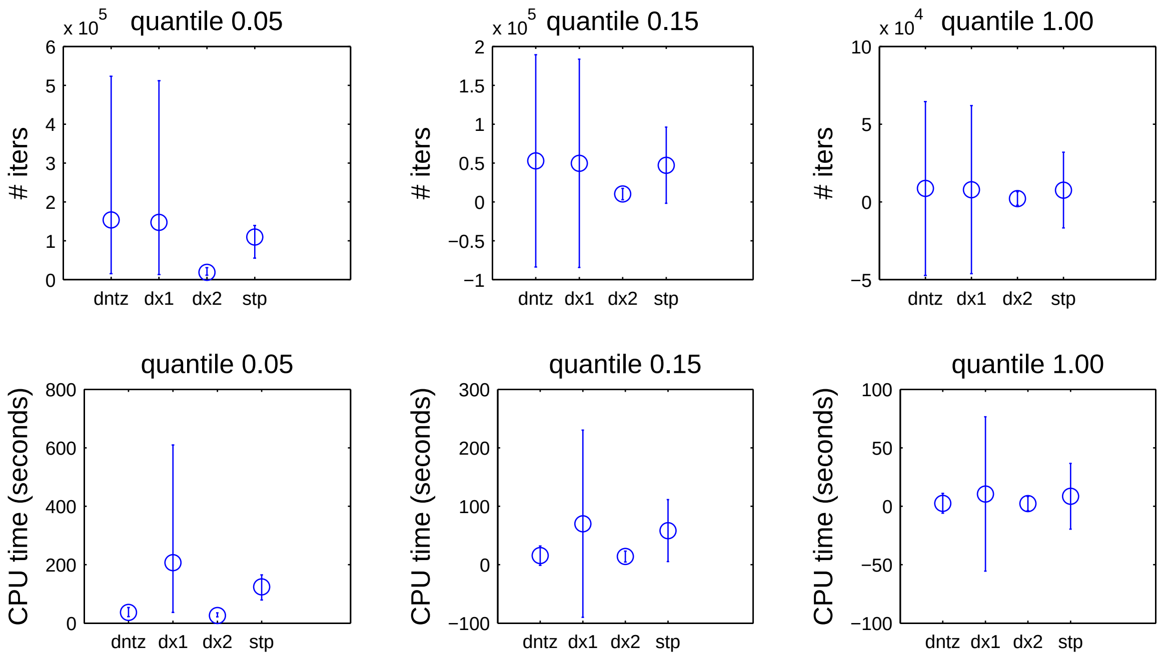

Figure 3: Comparison between pivot methods. Each point is an average among problems successfully solved by each single pivot method. Problems were broken down by quantile rank within each individual method.

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

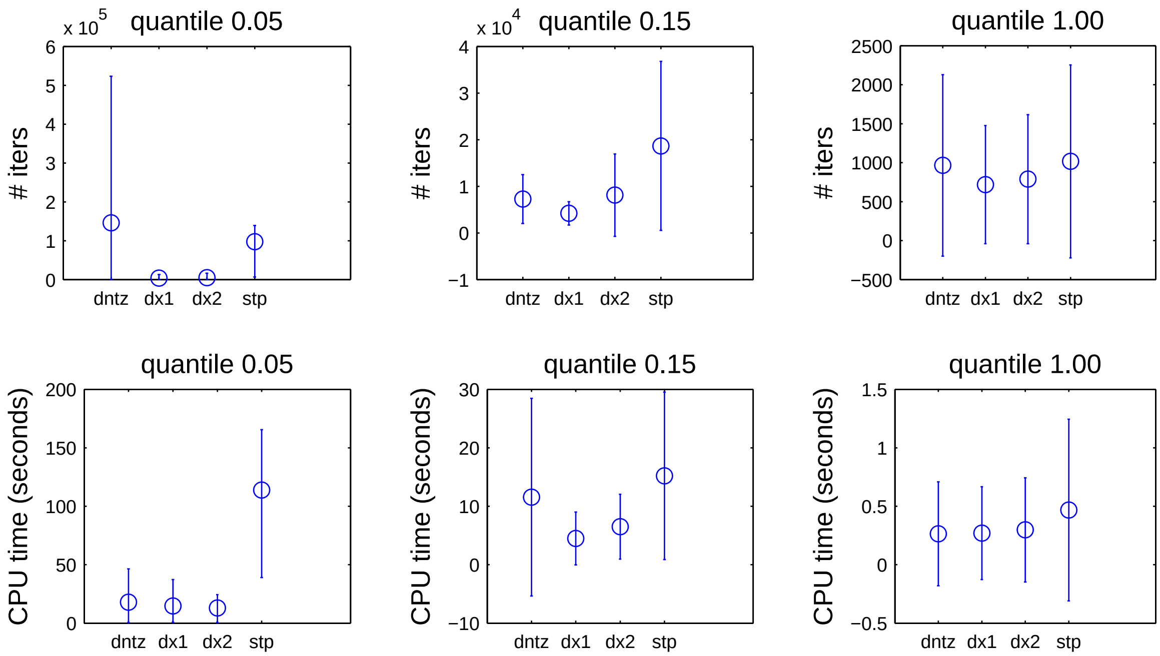

Figure 4: Comparison between pivot methods. Each point is an average among problems successfully solved by all pivot methods. Problems were broken down by quantile rank according to the sum across all methods.

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

[1] Michel Berkelaar, Kjell Eikland, and Peter Notebaert. lp_solve [computer software]. http://sourceforge.net/projects/lpsolve/.

[2] Robert E. Bixby. Solving real-world linear programs: A decade and more of progress. *Operations Research*, Jan-Feb 2002.

[3] Robert G. Bland. Finite pivoting rules for simplex method. *Mathematics of Operations Research*, 1977.

[4] Harlan Crowder and J. M. Hattingh. Partially normalized pivot selection in linear programming. *Mathematical Programming*, 1975.

[5] George B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, 1963.

[6] John J. Forrest and Donald Goldfarb. Steepest-edge simplex algorithms for linear programming. *Mathematical Programming*, 1992.

[7] D. Goldfarb and J. K. Reid. A practicable stepest-edge simplex algorithm. *Mathematical Programming*, 1977.

[8] Paula M. J. Harris. Pivot selection methods of the devex lp code. *Mathematical Programming*, 1975.

[9] T. L. Magnanti and J. B. Orlin. Parametric linear programming and anti-cycling pivoting rules. Technical Report 1730-85, Sloan School of Management, MIT, Oct 1985.

[10] netlib. http://www.netlib.org/lp/data/.

[11] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 1999.

[12] Ping-Qi Pan. A largest-distance pivot rule for the simplex algorithm. *European Journal of Operational Research*, March 2007.

model	first	dantzig	devex1	devex2	steepest
25fv47	-	7749	3150	3604	6804
80bau3b	-	13405	8797	8034	10100
bandm	6174	654	541	559	652
bnl1	17959	1644	1194	1223	1499
bnl2	33215	4349	3775	4137	5223
boeing1	3739	719	673	577	724
boeing2	546	183	155	162	182
bore3d	2352	181	177	178	310
brandy	12276	461	322	360	657
capri	669	515	422	460	439
cycle	8239	853	1872	1205	14849
czprob	27504	3139	1741	1819	2305
d2q06c	-	59867	12817	16446	131152
d6cube	-	-	511807	-	-
degen2	29407	523081	1577	1714	6381
degen3	-	-	-	16506	-
e226	2528	640	337	411	626
etamacro	1754	650	566	672	635
fffff800	4228	717	781	679	830
finnis	816	826	580	541	586
fit1d	26835	1222	1059	1121	1242
fit1p	-	3062	1151	1623	2905
fit2d	-	11542	7949	9190	-
fit2p	-	-	15295	-	23693
forplan	87766	256	233	294	456
ganges	3018	1776	1563	1622	1643
gfrd-pnc	2525	872	775	830	870
greenbea	-	15061	7143	8766	-
greenbeb	-	11006	5654	7368	11208
grow15	12440	901	884	1169	856
grow22	21344	1458	1647	881	2701
grow7	905	293	343	352	330
lotfi	707	270	192	224	217
maros-r7	-	3993	4049	4267	4133
maros	-	2516	1489	30556	3187
modszk1	-	1395	914	902	6832
perold	-	4387	2097	2412	2602
pilot.ja	-	4648	2838	3041	3959
pilot	-	15727	6897	7725	42581
pilot.we	-	5857	2920	2903	4271
pilot4	-	1271	1189	1302	1292
pilot87	-	-	48324	10959	10673
pilotnov	-	2065	1507	1767	2085
scagr25	2240	637	703	671	755
scfxm1	2398	389	456	447	444
scfxm2	7352	901	858	870	973
scfxm3	9981	1401	1352	1388	1430
scrs8	3862	982	723	805	971
scsd1	132900	232	157	228	203
scsd6	-	6153	7856	677	-
scsd8	-	5950	-	1755	5224
sctap2	1236	1871	1144	1739	1886
sctap3	1508	2356	1689	2477	2717
seba	479	659	733	632	718
share1b	1042	299	249	210	281
shell	1123	829	764	822	825
ship04l	922	601	455	649	587
ship04s	601	503	425	520	540
ship08l	2110	1129	774	1057	1131
ship08s	1083	861	672	843	834
ship12l	3003	1732	1123	1585	1634
ship12s	1609	1285	1005	1237	1190
sierra	897	993	888	978	926
stair	15108	639	573	581	673
standmps	469	587	323	555	556
stocfor2	110478	2244	2173	2352	3544
tuff	111644	526	546	419	113062
wood1p	-	613	573	581	139127
woodw	-	2419	1907	2190	55371

model	first	dantzig	devex1	devex2	steepest
25fv47	-	2.17s	1.45s	1.69s	3.72s
80bau3b	-	10.06s	10.36s	10.24s	14.96s
bandm	0.4s	0.07s	0.08s	0.09s	0.12s
bnl1	2.53s	0.34s	0.38s	0.39s	0.55s
bnl2	14.02s	2.46s	3.43s	3.7s	5.33s
boeing1	0.28s	0.07s	0.09s	0.08s	0.11s
boeing2	0.02s	0.01s	0.01s	0.01s	0.01s
bore3d	0.12s	0.01s	0.02s	0.02s	0.03s
brandy	0.67s	0.03s	0.03s	0.04s	0.08s
capri	0.04s	0.04s	0.05s	0.05s	0.05s
cycle	3.39s	0.55s	1.5s	0.96s	12.13s
czprob	5.95s	0.97s	0.99s	1.01s	1.52s
d2q06c	-	46.41s	17.51s	23.43s	157.67s
d6cube	-	-	609.84s	-	-
degen2	2.92s	53.92s	0.3s	0.31s	1.3s
degen3	-	-	-	10.73s	-
e226	0.15s	0.05s	0.04s	0.05s	0.08s
etamacro	0.15s	0.07s	0.09s	0.1s	0.11s
fffff800	0.61s	0.12s	0.19s	0.17s	0.24s
finnis	0.09s	0.1s	0.1s	0.09s	0.12s
fit1d	2.61s	0.13s	0.2s	0.21s	0.31s
fit1p	-	0.8s	0.44s	0.63s	1.34s
fit2d	-	23.78s	17.58s	21.92s	-
fit2p	-	-	36.48s	-	79.39s
forplan	5.26s	0.03s	0.04s	0.05s	0.07s
ganges	0.59s	0.48s	0.58s	0.59s	0.68s
gfrd-pnc	0.27s	0.12s	0.15s	0.15s	0.19s
greenbea	-	11.06s	9.21s	10.77s	-
greenbeb	-	8.48s	7.68s	9.46s	17.81s
grow15	1.4s	0.14s	0.19s	0.27s	0.15s
grow22	3.35s	0.32s	0.5s	0.11s	0.4s
grow7	0.05s	0.02s	0.04s	0.04s	0.05s
lotfi	0.03s	0.02s	0.02s	0.02s	0.02s
maros-r7	-	22.27s	37.31s	24.44s	39.05s
maros	-	0.63s	0.59s	14.91s	1.42s
modszk1	-	0.32s	0.33s	0.33s	2.27s
perold	-	1.03s	0.82s	0.94s	1.21s
pilot.ja	-	1.64s	1.65s	1.72s	2.76s
pilot	-	16.34s	11.97s	12.39s	48.7s
pilot.we	-	1.94s	1.6s	1.57s	2.76s
pilot4	-	0.27s	0.36s	0.38s	0.45s
pilot87	-	-	145.86s	35.15s	46.3s
pilotnov	9.74s	0.82s	0.97s	1.07s	1.48s
scagr25	0.19s	0.07s	0.1s	0.1s	0.13s
scfxm1	0.17s	0.04s	0.06s	0.06s	0.07s
scfxm2	0.92s	0.17s	0.23s	0.23s	0.3s
scfxm3	1.88s	0.38s	0.51s	0.52s	0.64s
scrs8	0.44s	0.16s	0.18s	0.21s	0.28s
scsd1	6.91s	0.03s	0.03s	0.04s	0.04s
scsd6	-	0.71s	1.7s	0.16s	-
scsd8	-	1.58s	-	0.81s	2.82s
sctap2	0.33s	0.56s	0.53s	0.76s	0.99s
sctap3	0.53s	0.92s	1s	1.4s	1.87s
seba	0.08s	0.09s	0.14s	0.13s	0.16s
share1b	0.03s	0.01s	0.02s	0.01s	0.02s
shell	0.19s	0.17s	0.23s	0.24s	0.29s
ship04l	0.17s	0.15s	0.18s	0.22s	0.24s
ship04s	0.1s	0.09s	0.13s	0.14s	0.17s
ship08l	0.67s	0.47s	0.52s	0.66s	0.86s
ship08s	0.27s	0.24s	0.3s	0.34s	0.4s
ship12l	1.28s	0.94s	0.99s	1.31s	1.63s
ship12s	0.5s	0.44s	0.52s	0.6s	0.7s
sierra	0.28s	0.32s	0.4s	0.42s	0.48s
stair	1.6s	0.11s	0.13s	0.12s	0.18s
standmps	0.08s	0.09s	0.08s	0.13s	0.15s
stocfor2	32.74s	0.93s	1.22s	1.34s	2.3s
tuff	8.69s	0.07s	0.1s	0.07s	19.96s
wood1p	-	0.47s	0.67s	0.64s	165.51s
woodw	-	2.35s	3.1s	3.37s	93.12s

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