Journal of Computational Mathematics
Vol.28, No.6, 2010, 837–847.

http://www.global-sci.org/jcm
doi:10.4208/jcm.3105-m2897

A FAST SIMPLEX ALGORITHM FOR LINEAR PROGRAMMING*

Pingqi Pan
Department of Mathematics, Southeast University, Nanjing 210096, China
Email: panpq@seu.edu.cn

Abstract

Recently, computational results demonstrated remarkable superiority of a so-called "largest-distance" rule and "nested pricing" rule to other major rules commonly used in practice, such as Dantzig's original rule, the steepest-edge rule and Devex rule. Our computational experiments show that the simplex algorithm using a combination of these rules turned out to be even more efficient.

Mathematics subject classification: 65K05, 90C05.
Key words: Large-scale linear programming, Simplex algorithm, Pivot rule, Nested, Largest-distance, Scaling.

1. Introduction

Consider the linear programming (LP) problem in the standard form

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b, \quad x \geq 0, \end{aligned} \tag{1.1}$$

where $A \in \mathbb{R}^{m\times n}$ $(m < n)$ and $\operatorname{rank}(A) = m$. It will be a simple matter to extend results of this paper to more general LP problems with bounds and ranges.

The pivot rule that is employed to select an index to enter the basis is crucial to computational efficiency of the simplex algorithm, since it essentially determines the number of iterations required for solving LP problems. As a result, a variety of pivot rules have been proposed and tested from time to time (for a survey, see [8] or [13]). Among them, the steepest-edge rule [4,5] and its approximation, Devex rule [6], are now accepted to be as the best, and are therefore commonly used in commercial packages, such as CPLEX [1,7].

Recently, Pan reported very encouraging computational results on the largest-distance rule [10] and the nested pricing rule [11,12] against major commonly used rules, such as Dantzig's original rule as well as the steepest-edge rule and Devex rule. Over 80 test problems, a largest-distance rule yields run times that are reduced by an average factor of 3.24, while a nested pricing rule yields run times reduced by an average factor of 5.73, compared to the Devex rule.

It has been unknown what will happen if the nested pricing rule and the largest-distance rule are put together. For this purpose, we have conducted computational tests on a combination of the two rules with the same test sets, i.e., the 48 largest Netlib problems in terms of the number of rows and columns, all of the 16 Kennington problems, and the 17 largest BPMPD problems in terms of more than 500KB in compressed form. Computational results turned out

* Received August 30, 2008 / Accepted September 15, 2009 /
Published online August 9, 2010 /

to be even more favorable than a single of them used: it outperformed the Devex rule by an average run time factor as high as 7.27.

In the remaining part of this section, we review briefly the largest-distance rule and the nested pricing rule. In Section 2, we describe the new rule. In Section 3, we report computational results and make final remarks.

Let B be the current basis and N the associated nonbasis. Without confusion, denote *basic* and *nonbasic* index sets again by B and N, respectively. The reduced costs associated with nonbasic indices may be computed by

$$\bar{c}_N = c_N - N^T \pi, \qquad B^T \pi = c_B. \tag{1.2}$$

If index set

$$J = \left\{ j \mid \bar{c}_j < 0,\ j \in N \right\} \tag{1.3}$$

is nonempty, Dantzig's original rule [2,3] selects an entering index q such that

$$\bar{c}_q = \min \left\{ \bar{c}_j \mid j \in J \right\} < 0. \tag{1.4}$$

1.1. Largest-distance rule

The determination of q is not invariant for scaling. In fact, it is seen from (1.2) and (1.4) that quantities π and $\bar{c}_N$, and hence index q are all dependent of norms of columns of the coefficient matrix A. To eliminates such dependence, the largest-distance rule [10] uses reduced costs normalized by norms of corresponding columns.

1.2. Nested pricing rule

At each iteration, the nested pricing rule [11, 12] gives indices in a subset of N priority to become basic. Pricing is first conducted on it to determine a reduced cost by some criterion. If one is found significantly negative, then the associated index is selected to enter B. If not, the same is done with the remaining set; if no such one is found, optimality is declared.

2. Nested Largest-Distance Rule

To make further progress, we combine the largest-distance rule and the nest pricing rule as follows.

Rule 2.1. Let an optimality tolerance $\epsilon > 0$ be given. Set $J = N$ initially.

1. If

$$\widehat{J} \triangleq \left\{ j \;\mid\; \bar{c}_j / \|a_j\| < -\epsilon,\ j \in J \right\} \tag{2.1}$$

is nonempty, go to step 4; else,

2. If

$$\widehat{J} \triangleq \left\{ j \;\mid\; \bar{c}_j / \|a_j\| < -\epsilon,\ j \in N \backslash J \right\} \tag{2.2}$$

is nonempty, go to step 4; else

3. Stop and declare optimality.
4. Determine an entering index q such that

$$q = \arg\min \left\{ \bar{c}_j / \|a_j\| \mid j \in \widehat{J} \right\}, \tag{2.3}$$

and set $J = \widehat{J} \backslash q$ for the next iteration.

It is noted that full pricing is performed only at the initial iteration and at iterations where Step 2 is carried out (when set $N \backslash J$ is touched). Following a full pricing, in general, many iterations perform pricing only on J, each of which is a proper subset of its predecessor. Therefore, Rule 2.1 falls to the partial pricing category.

It is favorable to implement the normalization of columns of A in a scaling preprocess. If this is so, Rule 2.1 becomes one with $\|aj\| = 1$, being just the nested pricing rule.

3. Computational Experiments

Our computational experiments with Rule 2.1 turned out to be very favorable. In this section, we report obtained results, giving an insight into the interesting behavior of the new rule, and make final remarks.

3.1. Test codes

The following three codes were tested and compared against one another:

- Devex: uses the Devex rule.
- NLD1: uses Rule 2.1 with the 2-norm.
- NLD2: uses Rule 2.1 with the ∞-norm.

To have the competitions fair and easy, all the three codes were implemented within Minos 5.51 [9] by only changing its rule. Code Devex resulted from Minos 5.51 by replacing its rule by the Devex rule. Codes NLD1 and NLD2 yielded by inserting a few lines for relevant column normalization in Subroutine m2scla of module M20amat, and using the nested pricing rule.

Compiled using Visual Fortran 5.0, the three codes were run under a Windows XP system Home Edition Version 2002 on an IBM PC with an Intel(R) Pentium(R) processor 1.86GHz, 1.00GB of 1.86GHz memory, and about 16 digits of precision. All reported CPU times were measured in seconds with utility routine CPU TIME, excluding the time spent on preprocessing and scaling.

The codes used the default options, except for the following: Rows 200000; Columns 300000; Elements 5000000; Iterations 4000000; Scale yes; Solution no; Log frequency 0; Print level 0.

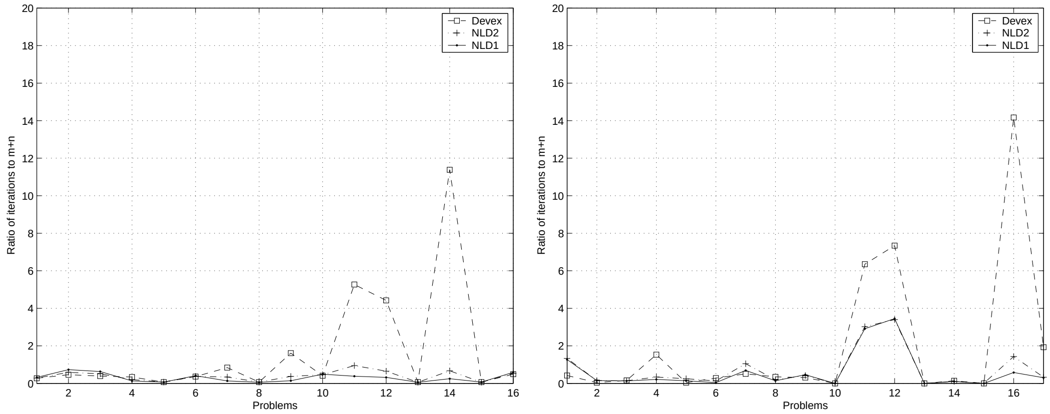
We order test problems by their sizes in terms of $m + n$, where m and n are the numbers of rows and columns of the constraint matrix, excluding slack variables.

Table 3.1: Statistics for 48 Netlib problems

Problem	Devex		NLD2		NLD1	
	Iters	Time	Iters	Time	Iters	Time
SCRS8	386	0.1	598	0.1	530	0.1
GFRD-PNC	608	0.2	689	0.2	744	0.2
BNL1	934	0.4	1736	0.5	1643	0.5
SHIP04S	150	0.1	229	0.1	138	0.1
PEROLD	2200	1.1	3873	1.4	4071	1.5
MAROS	1442	0.7	1933	0.8	2023	0.8
FIT1P	554	0.3	901	0.4	999	0.4
MODSZK1	644	0.2	1002	0.3	1116	0.3
SHELL	252	0.1	275	0.1	256	0.1
SCFXM3	927	0.5	1483	0.6	1410	0.6
25FV47	3728	2.3	6301	3.0	5981	2.6
SHIP04L	229	0.1	350	0.1	215	0.1
QAP8	5921	6.1	11161	10.7	12286	11.1
WOOD1P	610	0.9	1166	0.9	1144	0.8
PILOT.JA	3379	2.3	5312	2.8	5573	2.8
SCTAP2	731	0.4	698	0.3	747	0.3
GANGES	590	0.3	726	0.3	781	0.3
PILOTNOV	1310	0.9	2587	1.4	1986	1.0
SCSD8	1900	0.8	2161	0.6	1890	0.4
SHIP08S	235	0.2	252	0.1	280	0.1
SIERRA	1160	0.6	1148	0.5	918	0.4
DEGEN3	3599	3.5	4064	3.5	9364	6.7
PILOT.WE	2090	1.2	5278	2.0	5348	2.0
NESM	3035	1.5	4932	1.6	4158	1.2
SHIP12S	376	0.3	401	0.2	411	0.2
SCTAP3	835	0.6	898	0.5	747	0.4
STOCFOR2	1400	1.1	3770	2.7	3768	2.6
CZPROB	1043	0.6	1531	0.6	1387	0.5
CYCLE	2194	2.1	2371	1.7	2214	1.5
SHIP08L	446	0.3	597	0.3	612	0.3
PILOT	8884	21.2	23962	38.5	21436	32.3
BNL2	3788	4.3	5015	4.2	5428	4.4
SHIP12L	818	0.7	844	0.5	947	0.5
D6CUBE	21235	23.6	14972	7.3	13957	6.1
D6CUBE2	20824	23.2	13343	6.6	15219	6.7
PILOT87	10028	55.9	30023	105.6	24986	80.8
D2Q06C	11456	17.8	20997	23.1	21654	22.7
GREENBEA	13201	18.3	12224	12.3	10411	10.1
WOODW	3098	3.7	3690	2.1	3630	1.9
TRUSS	9552	12.6	14445	8.9	12859	7.3
FIT2D	10296	20.5	21683	13.2	17035	6.2
QAP12	77356	696.7	170669	1464.4	185205	1505.8
80BAU3B	8069	9.6	13221	8.7	13276	8.3
MAROS-R7	2720	18.0	6547	21.4	7449	22.3
FIT2P	9171	25.6	13585	33.4	14945	34.7
DFL001	453186	1682.6	132770	376.6	151391	397.7
STOCFOR3	12220	78.3	33154	176.8	31668	166.7
Total	718810	2742.4	599567	2341.6	624236	2354.4
QAP15	596500	20272.6	625953	23410.4	807019	28660.5

Table 3.2: Ratio for 48 Netlib problems

Problem	m	n	Devex/NLD2		Devex/NLD1		NLD2/NLD1	
			Iters	Time	Iters	Time	Iters	Time
SCRS8	491	1169	0.65	1.00	0.73	1.08	1.13	1.08
GFRD-PNC	617	1092	0.88	1.06	0.82	0.89	0.93	0.84
BNL1	644	1175	0.54	0.75	0.57	0.81	1.06	1.08
SHIP04S	403	1458	0.66	1.00	1.09	1.33	1.66	1.33
PEROLD	626	1376	0.57	0.76	0.54	0.72	0.95	0.95
MAROS	847	1443	0.75	0.95	0.71	0.97	0.96	1.03
FIT1P	628	1677	0.61	0.68	0.55	0.68	0.90	1.00
MODSZK1	688	1620	0.64	0.81	0.58	0.79	0.90	0.96
SHELL	537	1775	0.92	1.00	0.98	1.00	1.07	1.00
SCFXM3	991	1371	0.63	0.85	0.66	0.91	1.05	1.07
25FV47	822	1571	0.59	0.79	0.62	0.89	1.05	1.13
SHIP04L	403	2118	0.65	1.00	1.07	1.22	1.63	1.22
QAP8	913	1632	0.53	0.57	0.48	0.55	0.91	0.97
WOOD1P	245	2594	0.52	1.03	0.53	1.11	1.02	1.08
PILOT.JA	941	1988	0.64	0.82	0.61	0.82	0.95	1.00
SCTAP2	1091	1880	1.05	1.39	0.98	1.30	0.93	0.93
GANGES	1310	1681	0.81	0.97	0.76	0.91	0.93	0.94
PILOTNOV	976	2172	0.51	0.70	0.66	0.93	1.30	1.33
SCSD8	398	2750	0.88	1.53	1.01	1.87	1.14	1.22
SHIP08S	779	2387	0.93	1.23	0.84	1.23	0.90	1.00
SIERRA	1228	2036	1.01	1.33	1.26	1.64	1.25	1.23
DEGEN3	1504	1818	0.89	1.01	0.38	0.52	0.43	0.52
PILOT.WE	723	2789	0.40	0.61	0.39	0.61	0.99	1.00
NESM	663	2923	0.62	0.98	0.73	1.27	1.19	1.30
SHIP12S	1152	2763	0.94	1.14	0.91	1.14	0.98	1.00
SCTAP3	1481	2480	0.93	1.21	1.12	1.41	1.20	1.17
STOCFOR2	2158	2031	0.37	0.41	0.37	0.42	1.00	1.02
CZPROB	930	3523	0.68	1.03	0.75	1.17	1.10	1.13
CYCLE	1904	2857	0.93	1.27	0.99	1.41	1.07	1.11
SHIP08L	779	4283	0.75	1.11	0.73	1.15	0.98	1.04
PILOT	1442	3652	0.37	0.55	0.41	0.66	1.12	1.19
BNL2	2325	3489	0.76	1.02	0.70	0.98	0.92	0.96
SHIP12L	1152	5427	0.97	1.33	0.86	1.44	0.89	1.08
D6CUBE	416	6184	1.42	3.23	1.52	3.88	1.07	1.20
D6CUBE2	416	6184	1.56	3.53	1.37	3.44	0.88	0.97
PILOT87	2031	4883	0.33	0.53	0.40	0.69	1.20	1.31
D2Q06C	2172	5167	0.55	0.77	0.53	0.78	0.97	1.02
GREENBEA	2393	5405	1.08	1.50	1.27	1.82	1.17	1.22
WOODW	1099	8405	0.84	1.77	0.85	1.96	1.02	1.11
TRUSS	1001	8806	0.66	1.40	0.74	1.72	1.12	1.23
FIT2D	26	10500	0.47	1.56	0.60	3.32	1.27	2.13
QAP12	3193	8856	0.45	0.48	0.42	0.46	0.92	0.97
80BAU3B	2263	9799	0.61	1.11	0.61	1.16	1.00	1.05
MAROS-R7	3137	9408	0.42	0.84	0.37	0.80	0.88	0.96
FIT2P	3001	13525	0.68	0.77	0.61	0.74	0.91	0.96
DFL001	6072	12230	3.41	4.47	2.99	4.23	0.88	0.95
STOCFOR3	16676	15695	0.37	0.44	0.39	0.47	1.05	1.06
Average	1610	4256	1.20	1.17	1.15	1.16	0.96	0.99
QAP15	6331	22275	0.95	0.87	0.74	0.71	0.78	0.82



(a) (b)

Fig. 3.1. Comparison of Normalized iterations for (a):16 Kennington problems, (b):17 BPMPD problems.

3.2. Results for 48 Netlib problems

The first set of test problems included the 48 largest Netlib problems. Numerical results obtained are listed in Table 3.1, where total iterations and time required for solving each problem are listed in columns labeled Iters and Time under Devex, NLD2 and NLD1. As the data are heavily dominated by QAP15, results associated with it are listed at the bottom line, separately. So, the sums listed in the second bottom line are for the other 47 problems.

Table 3.2 serves as an overall comparison between the three codes. It is seen that Devex

Table 3.3: Statistics for 16 Kennington problems

Problem	Devex		NLD2		NLD1	
	Iters	Time	Iters	Time	Iters	Time
KEN-07	1657	1.5	1718	1.3	1926	1.5
CRE-C	3113	4.1	4001	4.0	4938	4.7
CRE-A	2986	4.4	3830	4.4	4778	5.2
PDS-02	3550	4.6	2099	1.9	1319	1.1
OSA-07	1757	6.0	1788	2.2	1726	2.4
KEN-11	12814	77.4	13361	63.0	14889	72.1
PDS-06	32424	170.1	13100	41.3	4891	13.4
OSA-14	3730	27.1	3818	9.3	3342	10.6
PDS-10	105469	988.1	24179	137.8	9297	45.4
KEN-13	29196	355.6	33074	342.3	34774	362.7
CRE-D	415765	4141.5	75142	349.0	29683	95.2
CRE-B	363018	3833.4	53747	242.2	26028	88.2
OSA-30	7048	96.3	6640	28.8	6007	38.4
PDS-20	1589107	33356.3	93045	1220.4	34775	404.4
OSA-60	16188	505.9	14962	158.4	16163	217.9
KEN-18	127756	6818.1	137975	6699.0	152141	7374.1
Total	2715578	50390.5	482479	9305.2	346677	8737.5

Table 3.4: Ratio for 16 Kennington problems

Problem	m	n	Devex/NLD2		Devex/NLD1		NLD2/NLD1	
			Iters	Time	Iters	Time	Iters	Time
KEN-07	2427	3602	0.96	1.16	0.86	0.99	0.89	0.86
CRE-C	3069	3678	0.78	1.01	0.63	0.86	0.81	0.85
CRE-A	3517	4067	0.78	1.00	0.62	0.84	0.80	0.84
PDS-02	2954	7535	1.69	2.48	2.69	4.04	1.59	1.63
OSA-07	1119	23949	0.98	2.70	1.02	2.47	1.04	0.91
KEN-11	14695	21349	0.96	1.23	0.86	1.07	0.90	0.87
PDS-06	9882	28655	2.48	4.12	6.63	12.71	2.68	3.09
OSA-14	2338	52460	0.98	2.92	1.12	2.55	1.14	0.87
PDS-10	16559	48763	4.36	7.17	11.34	21.76	2.60	3.03
KEN-13	28633	42659	0.88	1.04	0.84	0.98	0.95	0.94
CRE-D	8927	69980	5.53	11.87	14.01	43.49	2.53	3.66
CRE-B	9649	72447	6.75	15.83	13.95	43.48	2.06	2.75
OSA-30	4351	100024	1.06	3.35	1.17	2.51	1.11	0.75
PDS-20	33875	105728	17.08	27.33	45.70	82.48	2.68	3.02
OSA-60	10281	232966	1.08	3.19	1.00	2.32	0.93	0.73
KEN-18	105128	154699	0.93	1.02	0.84	0.92	0.91	0.91
Average	16087	60785	5.63	5.42	7.83	5.77	1.39	1.06

Table 3.5: Statistics for set 3 of 17 BPMPD problems

Problem	Devex		NLD2		NLD1	
	Iters	Time	Iters	Time	Iters	Time
RAT7A	5226	93.6	16665	209.5	15910	187.4
NSCT1	1607	27.6	5834	44.6	5527	39.7
NSCT2	6266	88.1	5168	36.8	4833	33.8
ROUTING	68504	606.5	15342	88.3	9370	50.3
DBIR1	1848	45.8	11583	84.3	6570	46.4
DBIR2	13031	248.8	5824	44.9	1907	14.6
T0331-4L	24435	249.7	49932	329.1	32673	168.6
NEMSEMM2	17117	118.1	9267	27.4	7085	18.2
SOUTHERN	16774	234.6	22600	305.8	25153	338.4
RADIO.PR	2	1.4	2	1.4	2	1.4
WORLD.MD	427776	7403.1	203648	2760.5	196317	2600.9
WORLD	501004	8747.2	232371	3316.9	235491	3252.8
RADIO.DL	3	1.0	3	1.0	3	1.0
NEMSEMM1	10574	168.8	7675	34.9	7797	27.2
NW14	394	9.5	736	6.0	588	4.6
LPL1	2337341	60073.4	235314	3955.3	94994	1417.1
DBIC1	437658	18252.6	74285	1714.9	68402	1272.2
Total	3869560	96369.8	896249	12961.6	712622	9474.5

Table 3.6: Ratio for 17 BPMPD problems

Problem	m	n	Devex/NLD2		Devex/NLD1		NLD2/NLD1	
			Iters	Time	Iters	Time	Iters	Time
RAT7A	3137	9408	0.31	0.45	0.33	0.50	1.05	1.12
NSCT1	22902	14981	0.28	0.62	0.29	0.70	1.06	1.12
NSCT2	23004	14981	1.21	2.39	1.30	2.61	1.07	1.09
ROUTING	20895	23923	4.47	6.87	7.31	12.05	1.64	1.75
DBIR1	18805	27355	0.16	0.54	0.28	0.99	1.76	1.82
DBIR2	18907	27355	2.24	5.54	6.83	17.07	3.05	3.08
T0331-4L	665	46915	0.49	0.76	0.75	1.48	1.53	1.95
NEMSEMM2	6944	42133	1.85	4.30	2.42	6.48	1.31	1.51
SOUTHERN	18739	35421	0.74	0.77	0.67	0.69	0.90	0.90
RADIO.PR	58867	8052	1.00	1.01	1.00	1.00	1.00	0.99
WORLD.MD	35665	31728	2.10	2.68	2.18	2.85	1.04	1.06
WORLD	35511	32734	2.16	2.64	2.13	2.69	0.99	1.02
RADIO.DL	8053	66918	1.00	1.00	1.00	1.00	1.00	1.00
NEMSEMM1	3946	71413	1.38	4.84	1.36	6.20	0.98	1.28
NW14	74	123409	0.54	1.59	0.67	2.06	1.25	1.30
LPL1	39952	125000	9.93	15.19	24.61	42.39	2.48	2.79
DBIC1	43201	183235	5.89	10.64	6.40	14.35	1.09	1.35
Average	21133	52056	4.32	7.43	5.43	10.17	1.26	1.37

outperformed both NLD2 and NLD1 with QAP15 alone. But the situation is contrary for the 47 problems as a whole. It is seen that ratios of Devex to NLD2 and NLD1 total iterations are 1.20 and 1.15 while ratios of Devex to NLD2 and NLD1 average run time are 1.17 and 1.16, respectively. The differences of performance between ND1 and ND2 are small.

3.3. Results for 16 Kennington problems

The second test set includes all of the 16 Kennington problems [14]. Associated numerical results are listed in Table 3.3 and compared in Table 3.4. From the bottom row of the latter, it is seen that ratios of Devex to NLD2 and NDL1 total iterations are 5.63 and 7.83, and those of Devex to NLD2 and NLD1 average time are 5.42 and 5.77, respectively. So, the two new codes outperformed Devex unambiguously. NLD2 is actually faster than Devex with all the 16

Table 3.7: Summary for statistics

Problem	Devex		NLD2		NLD1	
	Iters	Time	Iters	Time	Iters	Time
Netlib(47)	718810	2742.4	599567	2341.6	624236	2354.4
Kennington(16)	2715578	50390.5	482479	9305.2	346677	8737.5
BPMPD(17)	3869560	96369.8	896249	12961.6	712622	9474.5
Total(80)	7303948	149502.7	1978295	24608.4	1683535	20566.4

Table 3.8: Ratio Summary

Problem	Devex/NLD2		Devex/NLD1		NLD2/NLD1	
	Iters	Time	Iters	Time	Iters	Time
Netlib(47)	1.20	1.17	1.15	1.16	0.96	0.99
Kenningt(16)	5.63	5.42	7.83	5.77	1.39	1.06
BPMPD(17)	4.32	7.43	5.43	10.17	1.26	1.37
Average(80)	3.69	6.08	4.34	7.27	1.18	1.20

Table 3.9: Normalized iteration counts

(a) for 16 Kennington problems

Prob.	Devex	NLD2	NLD1
1	0.27	0.28	0.32
2	0.46	0.59	0.73
3	0.39	0.51	0.63
4	0.34	0.20	0.13
5	0.07	0.07	0.07
6	0.36	0.37	0.41
7	0.84	0.34	0.13
8	0.07	0.07	0.06
9	1.61	0.37	0.14
10	0.41	0.46	0.49
11	5.27	0.95	0.38
12	4.42	0.65	0.32
13	0.07	0.06	0.06
14	11.38	0.67	0.25
15	0.07	0.06	0.07
16	0.49	0.53	0.59
Ave.	2.21	0.39	0.28

(b) for 17 BPMPD problems

Prob.	Devex	NLD2	NLD1
1	0.42	1.33	1.27
2	0.04	0.15	0.15
3	0.16	0.14	0.13
4	1.53	0.34	0.21
5	0.04	0.25	0.14
6	0.28	0.13	0.04
7	0.51	1.05	0.69
8	0.35	0.19	0.14
9	0.31	0.42	0.46
10	0.00	0.00	0.00
11	6.35	3.02	2.91
12	7.34	3.40	3.45
13	0.00	0.00	0.00
14	0.14	0.10	0.10
15	0.00	0.01	0.00
16	14.17	1.43	0.58
17	1.93	0.33	0.30
Ave.	3.11	0.72	0.57

problems except for CRE-A. As for the new codes, NLD1 is superior to NLD2 with this set with iterations and time ratio 1.39 and 1.06, repectively.

3.4. Results for 17 BPMPD problems

The third test set consists of the 17 largest BPMPD problems, in terms of more than 500KB in compressed form [15]. Associated numerical results are listed in Table 3.5 and compared in Table 3.6. From the bottom row of the latter it is seen that the two new codes performed even more successfully than with the first two test sets. While ratios of Devex to NLD2 and NDL1 total iterations are 4.32 and 5.43, the ratios of Devex to NLD2 and NLD1 average time are as high as 7.43 and 10.17. In fact, either NLD2 or NLD1 is faster than Devex with 12 out of the 17 problems.

3.5. Summary of results

Table 3.7 offers a summary of statistics over all the 80 test problems (excluding QAP15) and Table 3.8 lists the associated ratios. From the bottom row of the latter, it is seen that iterations ratios of Devex to NLD2 and NLD1 are 3.69 and 4.34, while time ratios are as high as 6.08 and 7.27. It is noted, on the other hand, that NLD1 is superior to NLD2 with iterations and time ratios 1.18 and 1.20, respectively.

Following Forrest and Goldfarb [4], we list in Table 3.9 the ratio of the number of iterations required by each of the three codes to the sum of the number of rows and columns for each of the Kennington and BPMPD problems. It would be reasonable to regard a code amenable to a problem when such a normalized number of iterations is less than one. From Table 3.9, it is seen that both NLD1 and NLD2 are superior to Devex, but NLD1 is the best. For an overview of codes' performance against one another, see plots of the normalized iteration counts in Figure 3.1.

Based on our computational experiences, including those reported in [10–12], we feel safe to conclude that the simplex algorithm using the nested largest-distance rule is very fast for solving large-scale sparse LP problems, relative to major commonly used simplex algorithms.

Acknowledgments. The author would like thank Professor Michael. Saunders for kindly providing us the MINOS 5.51 package. The research is supported by National Natural Science Foundation of China under the Projects 10871043 and 70971136.

References

[1] R.E. Bixby, Solving real-world linear programs: A decade and more of progress, *Oper. Res.*, **50**:1 (2002), 3-15.

[2] G.B. Dantzig, A. Orden and P. Wolfe, The generalized simplex method for minimizing a linear form under linear inequality restraints, *Pac. J. Math.*, **5** (1955), 183-195.

[3] G.B. Dantzig, Linear Programming and Extensions, Princeton University Press, Princeton, NJ, 1963.

[4] J.J.H. Forrest and D. Goldfarb, Steepest-edge simplex algorithms for linear programming, *Math. Program.*, **57** (1992), 341-374

[5] D. Goldfarb and J. Reid, A practicable steepest edge simplex algorithm, *Math. Program.*, **12** (1977), 361-371.

[6] P.M.J. Harris, Pivot selection methods of the Devex LP code, *Math. Program.*, **5** (1973), 1-28.

[7] ILOG CPLEX: `http://www.ilog.com/ products/cplex` High Performance Software of Mathematical Programming.

[8] I. Maros, Computational Techniques of the Simplex Method, International Series in Operations Research and Management, Vol. 61, Kluwer Academic Publishers, Boston, 2003.

[9] B.A. Murtagh and M. A. Saunders, *MINOS 5.5 User's Guide*, Technical Report SOL 83-20R, Dept. of Operations Research, Stanford University, Stanford, 1998.

[10] P.-Q. Pan, A largest-distance pivot rule for the Simplex Algorithm, *Eur. J. Oper. Res.*, **187** (2008), 393-402.

[11] P.-Q. Pan, Effcient nested pricing in the simplex algorithm, *Oper. Res. Lett.*, **36** (2008), 309-313.

[12] P.-Q. Pan, An empirical evaluation of pivot rules in the simplex algorithm, http://www.optimization-online.org/DB FILE/2007/03/1602.pdf

[13] T. Terlaky and S. Zhang, Pivot rules for linear programming: A survey on recent theoretical developments, *Ann. Oper. Res.*, **46** (1993), 2023-233.

[14] `http://www-fp.mcs.anl.gov/otc/Guide/TestProblems/LPtest`

[15] http://www.sztaki.hu/m̃eszaros/bpmpd/