Yugoslav Journal of Operations Research
24 (2014), Number 3, 321-332
DOI: 10.2298/YJOR140228016P

PIVOTING RULES FOR THE REVISED SIMPLEX ALGORITHM

Nikolaos PLOSKAS
Department of Applied Informatics, School of Information Sciences,
University of Macedonia, Thessaloniki, Greece
ploskas@uom.gr

Nikolaos SAMARAS
Department of Applied Informatics, School of Information Sciences,
University of Macedonia,Thessaloniki, Greece
samaras@uom.gr

Received: February 2014 / Accepted: May 2014

Abstract. Pricing is a significant step in the simplex algorithm where an improving non-basic variable is selected in order to enter the basis. This step is crucial and can dictate the total execution time. In this paper, we perform a computational study in which the pricing operation is computed with eight different pivoting rules: (i) Bland's Rule, (ii) Dantzig's Rule, (iii) Greatest Increment Method, (iv) Least Recently Considered Method, (v) Partial Pricing Rule, (vi) Queue Rule, (vii) Stack Rule, and (viii) Steepest Edge Rule; and incorporate them with the revised simplex algorithm. All pivoting rules have been implemented in MATLAB. The test sets used in the computational study are a set of randomly generated optimal sparse and dense LPs and a set of benchmark LPs (Netlib-optimal, Kennington, Netlib-infeasible).

Keywords: Linear Programming, Revised Simplex Method, Pricing, Pivoting Rules.

MSC: 90C05, 65K05.

1. INTRODUCTION

Linear Programming (LP) is the process of minimizing or maximizing a linear objective function $z = \sum_{i=1}^{n} c_i \cdot x_i$ subject to a number of linear equality and inequality constraints. Several algorithms have been proposed for solving linear programming problems (LPs). The most well-known method for solving LPs is

the simplex algorithm developed by George B. Dantzig [6]; other well-studied algorithms include the Interior Point Methods and the Exterior Point Simplex Algorithm (EPSA). The main idea of EPSA is that it moves in the exterior of the feasible region and constructs basic infeasible solutions instead of feasible solutions calculated by the simplex algorithm. A more effective approach is the Primal-Dual Exterior Point Simplex Algorithm [17]. The aforementioned algorithms can be used for solving general LPs or network optimization problems [18].
Assuming that the problem is in its general form, the linear problem can be formulated as shown in (LP.1).

$$\begin{aligned} &min && c^T x \\ &subject\ to && Ax = b \\ & && x \geq 0 \end{aligned} \qquad (LP.1)$$

where $A \in R^{m \times n}$, $(c, x) \in R^n$, $b \in R^m$, and T denotes transposition. We assume that A has full rank, $rank(A) = m$, where $(m < n)$. The simplex algorithm searches for the optimal solution by moving from one feasible solution to another, along the edges of the feasible set. The dual problem associated with the linear problem (LP.1) is shown in (DP.1).

$$\begin{aligned} &max && b^T w \\ &subject\ to && A^T w + s = c \\ & && s \geq 0 \end{aligned} \qquad (DP.1)$$

where $w \in R^m$ and $s \in R^n$. A_B is an $m \times m$ non-singular sub-matrix of A, called basic matrix or basis. The columns of A which belong to subset B are called basic and those which belong to N are called non basic. A column A_l is selected in each step in order to enter the basis and a column A_r to leave the basis. The variable l is called entering variable and is computed according to a pivoting rule in each iteration of the simplex algorithm.
The selection of the entering variable is a crucial step of the simplex algorithm and is performed according to a pivoting rule. Good pivoting rules can lead to a fast convergence to the optimal solution, while poor pivoting rules lead to worst execution times or even no solutions of the LPs. A pivoting rule is one of the main factors that will determine the number of iterations that simplex algorithm performs [14]. Hence, the pivoting rule applied for the selection of the entering variable should be designed and implemented carefully. Many pivoting rules have been proposed in the literature. Eight of these are presented and compared in this paper; namely, (i) Bland's Rule, (ii) Dantzig's Rule, (iii) Greatest Increment Method, (iv) Least Recently Considered Method, (v) Partial Pricing Rule, (vi) Queue Rule, (vii) Stack Rule, and (viii) Steepest Edge Rule. Other well-known pivoting rules include Devex [12], Modified Devex [2], Steepest Edge approximation scheme [19], Murtys Bard type scheme [15], Edmonds-Fukuda rule [8] and

its variants [5] [22] [24] [25].
There are a few papers in the literature that have focused in the pricing step and fewer that compared pivoting rules. Forrest and Goldfarb [7] proposed several new implementations of Steepest Edge Rule and compared them with Devex variants and Dantzig's Rule over large LPs. They concluded that the Steepest Edge variants are clearly superior to Devex variants and Dantzig's Rule for solving difficult large-scale LPs. Thomadakis [20] has compared five pivoting rules: (i) Bland's Rule, (ii) Dantzig's Rule, (iii) Greatest-Increment Method, (iv) Least-Recently Considered Method, and (v) Steepest-Edge Rule. Thomadakis studied the trade-off between the number of iterations and the execution time per iteration and find that: (i) Bland's Rule requires the shortest execution time per iteration, but it usually needs many more iterations than the other methods to converge to the optimal solution, (ii) Dantzig's Rule and Least Recently Considered Method perform comparably, but the latter requires fewer iterations in cases where degenerate pivots exist, (iii) the computational cost per iteration in the Greatest Increment Method is greater than the aforementioned methods, but it usually leads to fewer iterations, and (iv) the Steepest-Edge Rule requires fewer iterations than all other pivoting rules and its computational cost is lower than Greatest Increment Method but higher than the other three methods.
To the best of our knowledge, the aforementioned are the only papers that compared some of the most widely-used pivoting rules. This paper is an extension of the work of Thomadakis [20] where eight well-known pivoting rules are compared. Thomadakis [20] has focused on the number of iterations and the execution time per iteration, while we also investigate the total execution time of the simplex algorithm relating to the pivoting rule that is used.
The structure of the paper is as follows: in Section 2, eight widely-used pivoting rules are presented. In Section 3, the computational comparison of the pivoting rules is presented over randomly generated optimal sparse and dense LPs and on a set of benchmark LPs (Netlib-optimal, Kennington, Netlib-infeasible). Finally, the conclusions of this paper are outlined in Section 4.

2. PIVOTING RULES

Eight pivoting rules are presented in this section: (i) Bland's Rule, (ii) Dantzig's Rule, (iii) Greatest Increment Method, (iv) Least Recently Considered Method, (v) Partial Pricing Rule, (vi) Queue Rule, (vii) Stack Rule, and (viii) Steepest Edge Rule. Some necessary notations should be introduced, before the presentation of the aforementioned pivoting rules. Let l be the index of the entering variable and $\overline{c_l}$ be the difference in the objective value when the non-basic variable x_l is increased by one unit and the basic variables are adjusted appropriately. Reduced cost is the amount by which the objective function on the corresponding variable must be improved before the value of the variable will be positive in the optimal solution.

2.1. Bland's Rule

According to Bland's Rule [3], the first variable among the eligible ones is selected as the entering variable. This variable is the leftmost among columns with negative relative cost coefficient. Bland's Rule avoids cycling, but it has been observed in practice that it can lead to stalling, a phenomenon where long degenerate paths are produced.

2.2. Dantzig's Rule

Dantzig's rule or Largest Coefficient Rule [6] is the first pivoting Rule that was used in the simplex algorithm. It has been widely-used in simplex implementations [1] [16]. This pivoting Rule selects the column A_l with the most negative $\overline{c_l}$. It guarantees the largest reduction in the objective value per unit of non-basic variable $\overline{c_l}$ increase. Its worst-case complexity is exponential [13]. However, Dantzig's Rule is considered as simple but powerful enough to guide simplex algorithm into short paths [20].

2.3. Greatest Increment Method

Greatest Increment Method [13] selects as entering variable the one with the largest total objective value improvement. Initially, the improvement of the objective value for each non-basic variable is calculated. Then, the variable, which offers the largest improvement in the objective value, is selected. Although this pivoting Rule can lead to fast convergence to the optimal solution, this advantage is eliminated by the additional computational cost per iteration. Finally, Gärtner [9] constructed LPs that Greatest Increment Method showed exponential complexity.

2.4. Least Recently Considered Method

According to Least Recently Considered Method, in the first iteration of the simplex algorithm, the incoming variable l is selected according to Bland's Rule, that is the leftmost among columns with negative relative cost coefficient. In the next iterations, Least Recently Considered Method [23] starts searching for the first eligible variable with index greater than l. If $l = n$ then Least Recently Considered Method starts searching from the first column again. This pivoting Rule prevents stalling and it performs fairly well in practice [20]. However, its worst-case complexity has not been proved yet.

2.5. Partial Pricing Rule

Partial Pricing methods are variants of the standard pivoting rules that take only a part of non-basic variables into account. In static partial pricing, non-basic variables are divided into equal segments with predefined size and the pricing operation is carried out segment by segment, until a reduced cost is found. In dynamical partial pricing, the segments' size is determined dynamically during the execution of the algorithm. In the computational study presented in Section 3, we have implemented Partial Pricing Rule as a variant of Dantzig's Rule with static partial pricing.

2.6. *Queue Rule*

Queue is a FIFO (First-In-First-Out) data structure, where the first element added to the queue is the first one to be removed. In this pivoting Rule, two queues are initially constructed; the first one stores the indices of the basic variables, while the other the indices of the non-basic variables. The entering and leaving variables are selected from the front of the corresponding queue. The variable, which is extracted from the front of the queue that stores the basic variables, is inserted to the end of the queue that stores the non-basic variables. Respectively, the variable, which is extracted from the front of the queue that stores the non-basic variables, is inserted to the end of the queue that stores the basic variables.

2.7. *Stack Rule*

Stack is a LIFO (Last-In-First-Out) data structure, where the last element added to the stack is the first one to be removed. In the stack Rule, the entering and leaving variables are selected from the top of the corresponding stack. The variable, which is extracted from the top of the stack that stores the basic variables, is inserted to the top of the stack that stores the non-basic variables. Respectively, the variable, which is extracted from the top of the stack that stores the non-basic variables, is inserted to the end of the stack that stores the basic variables.

2.8. *Steepest Edge Rule*

Steepest Edge Rule or All-Variable Gradient Method [11] selects as entering variable the variable with the most objective value reduction per unit distance. This pivoting Rule can lead to fast convergence to the optimal solution. However, this advantage is debatable due to the additional computational cost. Approximate methods have been proposed in order to improve the computational efficiency of this method [19] [21].

3. COMPUTATIONAL RESULTS

Computational studies have been widely-used in order to examine the practical efficiency of an algorithm or even compare algorithms. In this section, we present a computational study of the aforementioned pivoting rules. The computational comparison has been performed on a quad-processor Inter Core i7 3.4 GHz with 32 Gbyte of main memory and 8 cores. The revised simplex method and the pivoting rules have been implemented using MATLAB Professional R2013a. MATLAB is a powerful programming environment and is especially designed for matrix computations.

The test sets used in the computational study are a set of randomly generated optimal sparse and dense LPs and a set of benchmark LPs (Netlib-optimal, Kennington, Netlib-infeasible) [10] [4]. Table 1 presents some useful information about the second test bed, which was used in the computational study. The first column includes the name of the problem, the second the number of constraints, the third the number of variables, the fourth the nonzero elements of matrix A and the fifth the optimal objective value. The test bed includes 40 optimal and 5 infeasible LPs from Netlib and 3 Kennington LPs that do not have ranges and bounds sections in their mps files.

In Tables 2 - 7, the following abbreviations are used: (i) Bland's Rule - BR, (ii) Dantzig's Rule - DR, (iii) Greatest Increment method - GIM, (iv) Least Recently Considered method - LRCM, (v) Partial Pricing Rule - PPR, (vi) Queue Rule - QR, (vii) Stack Rule - SR, and (viii) Steepest Edge Rule - SER. For each instance we averaged times over 10 runs. All times are measured in seconds. A limit of $70,000$ iterations was set that explains why there are no measurements for some pivoting rules on specific instances. Finally, the objective value calculated using each pivoting rule was accurate with a precision of 8 decimal digits. Table 2 presents the results from the total execution time of the revised simplex algorithm combined with the aforementioned pivoting rules over the benchmark LPs (Netlib-optimal, Kennington, Netlib-infeasible), while Table 3 the iterations needed.
From the following results, we observe that only Dantzig's Rule has solved all instances, while Bland's Rule solved 45 out of 48 instances, Greatest Increment method solved 46 out of 48, Least Recently Considered method solved 45 out of 48, partial pricing solved 45 out of 48, Queue's Rule solved 41 out of 48, Stacks' Rule solved 43 out of 48, and Steepest Edge Rule solved 46 out of 48. Furthermore, Dantzig's Rule requires the shortest execution time both on average and on almost all instances. On the other hand, Steepest Edge Rule has the worst execution time both on average and on almost all instances. Despite its computational cost, Steepest Edge Rule needs the fewest number of iterations than all the other pivoting rules, while Bland's Rule is by far the worst pivoting rule in terms of the number of iterations.

Table 4 presents the results from the total execution time of the revised simplex algorithm combined with the aforementioned pivoting rules over the randomly generated sparse LPs with density 10%, while Table 5 presents the iterations needed. From the following results, we observe that all pivoting rules have solved all instances. Again, Dantzig's Rule requires the shortest execution time both on average and on all instances. On the other hand, Steepest Edge Rule has the worst execution time both on average and on all instances. Despite its computational cost, Steepest Edge Rule needs the fewest number of iterations than all the other pivoting rules, while Bland's Rule is the worst pivoting rule in terms of the number of iterations.

Table 6 presents the results from the total execution time of the revised simplex algorithm combined with the aforementioned pivoting rules over the randomly generated dense LPs, while Table 7 presents the iterations needed. From the following results, we extract the same results as in the benchmark LPs and the sparse LPs.

4. CONCLUSIONS

The selection of the entering variable is a crucial step in the revised simplex algorithm and should be carefully designed in order to economize this operation.

Table 1: Statistics of the Netlib set (Optimal, Kennington and Infeasible LPs)

Name	Constraints	Variables	Non-zeros A	Objective value
25FV47	822	1,571	11,127	5.50E+03
ADLITTLE	57	97	465	2.25E+05
AFIRO	28	32	88	-4.65E+02
AGG	489	163	2,541	-3.60E+07
AGG2	517	302	4,515	-2.02E+07
AGG3	517	302	4,531	1.03E+07
BANDM	306	472	2,659	-1.59E+02
BEACONFD	174	262	3,476	3.36E+04
BLEND	75	83	521	-3.08E+01
BNL1	644	1,175	6,129	1.98E+03
BNL2	2,325	3,489	16,124	1.81E+03
BRANDY	221	249	2,150	1.52E+03
CRE_A	3,517	4,067	19,054	2.36E+07
CRE_C	3,069	3,678	16,922	2.53E+07
DEGEN2	445	534	4,449	-1.44E+03
E226	224	282	2,767	-1.88E+01
FFFFF800	525	854	6,235	5.56E+05
ISRAEL	175	142	2,358	-8.97E+05
ITEST2	10	4	17	Infeasible
ITEST6	12	8	23	Infeasible
KLEIN1	55	54	696	Infeasible
KLEIN2	478	54	4,585	Infeasible
KLEIN3	995	88	12,107	Infeasible
LOTFI	154	308	1,086	-2.53E+01
OSA-07	1,119	23,949	167,643	5.36E+05
SC50A	51	48	131	-6.46E+01
SC50B	51	48	119	-7.00E+01
SC105	106	103	281	-5.22E+01
SC205	206	203	552	-5.22E+01
SCAGR7	130	140	553	-2.33E+06
SCFXM1	331	457	2,612	1.84E+04
SCFXM2	661	914	5,229	3.67E+04
SCFXM3	991	1,371	7,846	5.49E+04
SCORPION	389	358	1,708	1.88E+03
SCRS8	491	1,169	4,029	9.04E+02
SCTAP1	301	480	2,052	1.41E+03
SCTAP2	1,091	1,880	8,124	1.72E+03
SCTAP3	1,481	2,480	10,734	1.42E+03
SHARE1B	118	225	1,182	-7.66E+04
SHARE2B	97	79	730	-4.16E+02
SHIP04L	403	2,118	8,450	1.79E+06
SHIP04S	403	1,458	5,810	1.80E+06
SHIP08L	779	4,283	17,085	1.91E+06
SHIP08S	779	2,387	9,501	1.92E+06
SHIP12L	1,152	5,427	21,597	1.47E+06
SHIP12S	1,152	2,763	10,941	1.49E+06
STOCFOR1	118	111	474	-4.11E+04
STOCFOR2	2,158	2,031	9,492	-3.90E+04

Table 2: Total Execution Time of Benchmark LPs (Netlib-optimal, Kennington, Netlib-infeasible)

Name	BR	DR	GIM	LRCM	PPR	QR	SR	SER
25FV47	-	63.26	-	-	-	-	-	6,504.45
ADLITTLE	0.03	0.06	0.02	0.04	0.02	0.03	0.03	0.04
AFIRO	0.01	0.004	0.005	0.01	0.01	0.01	0.01	0.004
AGG	0.06	0.05	0.1	0.07	0.07	0.07	0.07	0.2
AGG2	0.14	0.11	0.41	0.11	0.11	0.13	0.15	1.24
AGG3	0.22	0.13	0.49	0.26	0.22	0.2	0.29	1.39
BANDM	1.33	0.48	0.8	1.28	1.08	-	1.98	1.88
BEACONFD	0.02	0.02	0.02	0.02	0.03	0.02	0.02	0.02
BLEND	0.05	0.03	0.07	0.77	0.06	2.06	0.09	0.04
BNL1	181.4	20.92	30.2	95.43	53.2	-	-	264.11
BNL2	-	211.51	-	-	-	-	-	-
BRANDY	1.69	0.16	0.34	0.51	0.67	0.39	0.78	0.56
CRE_A	1,205.65	100.33	4,156.39	3,109.87	145.63	287.45	210.48	5,567.89
CRE_C	830.45	84.59	255.67	325.41	224.2	320.35	165.39	2,801.39
DEGEN2	15.89	2.48	5.01	39.64	15.67	-	9.2	16.86
E226	1.31	0.25	0.95	0.69	0.76	-	0.86	2.21
FFFFF800	4.04	0.49	3.31	1.39	1.98	-	2.48	15.9
ISRAEL	0.12	0.12	0.16	0.17	0.14	0.15	0.19	0.41
ITEST2	0.002	0.002	0.002	0.002	0.002	0.002	0.002	0.002
ITEST6	0.002	0.002	0.003	0.002	0.003	0.003	0.003	0.003
KLEIN1	0.02	0.03	0.07	0.05	0.04	0.06	0.05	0.05
KLEIN2	2.25	0.45	0.46	1.42	1.07	0.77	1.63	1.29
KLEIN3	24.05	6.8	3.68	17.75	19.33	7.34	71.12	15.22
LOTFI	0.27	0.12	0.39	0.16	0.2	0.25	0.23	0.75
OSA-07	8.95	6.31	14.07	3.86	22.11	14.11	9.54	-
SC50A	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
SC50B	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
SC105	0.03	0.02	0.03	0.03	0.03	0.03	0.04	0.04
SC205	0.1	0.11	0.16	0.15	0.11	0.16	0.16	0.3
SCAGR7	0.1	0.05	0.11	0.1	0.08	0.1	0.12	0.08
SCFXM1	1.73	0.4	1.43	0.82	1.32	0.84	2.03	2.92
SCFXM2	17.81	2.54	7.86	8.01	11.29	12.34	9.87	15.64
SCFXM3	45.65	7.98	48.13	28.67	33.16	29.65	-	80.67
SCORPION	0.28	0.24	0.3	0.26	0.27	0.27	-	0.41
SCRS8	-	1.3	4.83	-	-	3.73	1.62	29.4
SCTAP1	0.5	0.16	0.77	0.37	0.49	0.34	0.48	2.77
SCTAP2	5.28	3.6	38.04	6.92	7.08	5.29	6.97	58.28
SCTAP3	9.64	6.2	92.85	10.5	17.1	10.72	13.7	335.03
SHARE1B	0.58	0.09	0.22	0.26	0.32	0.2	0.41	0.37
SHARE2B	0.07	0.03	0.06	0.05	0.06	0.06	0.06	0.06
SHIP04L	0.89	0.85	3.22	0.95	0.99	1.44	1.41	2.43
SHIP04S	0.33	0.32	1.19	0.38	0.37	0.54	0.48	1.06
SHIP08L	6.39	4.01	13.14	4.65	5.86	7.99	6.25	15.79
SHIP08S	1.21	0.59	2.36	0.68	0.78	1.26	0.99	5.91
SHIP12L	9.84	9.14	33.19	9.76	11.08	13.79	12	39.77
SHIP12S	1.35	1.13	5.54	1.38	1.42	2.21	1.75	5.32
STOCFOR1	0.07	0.03	0.06	0.06	0.06	0.05	0.08	0.05
STOCFOR2	140.14	34.04	81.46	84.34	98.13	85.65	120.56	130.13
AVERAGE	**56**	**11.91**	**104.51**	**83.49**	**15.04**	**19.76**	**15.2**	**346.14**

Table 3: Number of Iterations of Benchmark LPs (Netlib-optimal, Kennington, Netlib-infeasible)

Name	BR	DR	GIM	LRCM	PPR	QR	SR	SER
25FV47	-	6,522	-	-	-	-	-	1,510
ADLITTLE	198	118	82	239	209	172	205	100
AFIRO	24	11	9	18	20	18	20	10
AGG	106	73	69	124	111	104	108	69
AGG2	184	141	127	167	132	157	172	122
AGG3	300	153	145	386	293	234	359	134
BANDM	1,654	544	279	1,561	1,346	-	2,467	252
BEACONFD	40	22	22	30	23	31	31	22
BLEND	170	102	104	3,069	198	7,280	271	60
BNL1	36,078	4,447	1,643	25,096	12,673	-	-	866
BNL2	-	8,517	-	-	-	-	-	-
BRANDY	4,401	360	275	1,368	1,876	999	2,165	330
CRE_A	64,132	5,487	3,098	14,856	4,867	13,985	12,045	1,801
CRE_C	52,134	5,126	2,854	21,098	14,378	18,654	11,345	1,530
DEGEN2	7,415	844	569	19,392	6,678	-	4,014	441
E226	2,687	522	285	1,585	1,664	-	1,673	235
FFFFF800	6,499	457	403	1,890	2,517	-	3,523	253
ISRAEL	371	363	150	492	416	394	505	141
ITEST2	4	4	4	4	4	4	4	4
ITEST6	4	4	4	4	4	4	4	4
KLEIN1	90	190	102	276	182	267	225	117
KLEIN2	2,515	554	231	1,623	1,245	832	1,972	466
KLEIN3	7,805	2,411	602	6,114	6,840	2,408	24,398	1,263
LOTFI	708	351	191	431	527	570	577	144
OSA-07	6,405	1,059	860	3,546	7,011	4,015	3,330	-
SC50A	33	30	28	33	30	29	38	27
SC50B	29	31	30	34	31	31	34	30
SC105	73	66	53	69	81	70	106	56
SC205	141	139	124	221	155	211	235	115
SCAGR7	241	87	87	236	164	207	268	73
SCFXM1	1,756	403	388	1,025	1,486	836	2,007	286
SCFXM2	5,378	786	1,198	3,101	4,013	4,421	2,689	541
SCFXM3	7,745	1,227	2,453	4,854	4,578	4,801	-	789
SCORPION	155	112	117	157	141	154	-	111
SCRS8	-	658	348	-	-	2,639	1,155	373
SCTAP1	814	284	284	614	675	491	684	167
SCTAP2	2,283	1,132	1,378	2,653	2,348	2,300	2,329	333
SCTAP3	2,501	2,862	1,733	2,582	3,426	2,683	2,972	619
SHARE1B	1,860	200	92	762	955	493	1,134	162
SHARE2B	294	104	100	220	276	212	249	77
SHIP04L	368	227	241	500	326	508	572	211
SHIP04S	243	208	167	412	240	337	282	152
SHIP08L	3,788	469	443	1,479	2,130	2,093	1,317	377
SHIP08S	1,696	237	237	522	457	774	571	224
SHIP12L	1,795	733	731	1,751	1,668	1,892	1,460	633
SHIP12S	929	378	432	990	665	1,032	758	324
STOCFOR1	235	69	70	206	200	131	212	70
STOCFOR2	11,034	1,617	2,013	6,010	5,890	5,541	9,145	1,386
AVERAGE	**5,273.67**	**1,050.85**	**540.33**	**2,928.89**	**2,069.98**	**2,000.34**	**2,270.47**	**369.78**

Table 4: Total Execution Time of Sparse LPs with Density 10%

Size	**BR**	**DR**	**GIM**	**LRCM**	**PPR**	**QR**	**SR**	**SER**
1000x1000	8.56	1.85	30.90	10.63	3.86	4.43	2.34	62.75
1500x1500	10.08	3.08	34.47	38.41	7.40	11.33	11.22	75.42
2000x2000	311.45	52.53	242.92	152.15	42.57	173.61	72.08	815.51
2500x2500	229.24	28.50	653.84	279.66	94.94	57.88	81.96	841.23
3000x3000	122.71	32.19	288.41	254.47	61.78	100.66	49.81	2,283.19
AVERAGE	136.41	23.63	250.10	147.07	42.11	69.58	43.48	815.62

Table 5: Number of Iterations of Sparse LPs with Density 10%

Size	**BR**	**DR**	**GIM**	**LRCM**	**PPR**	**QR**	**SR**	**SER**
1000x1000	1,161	140	196	417	353	406	553	91
1500x1500	566	163	79	686	255	282	764	30
2000x2000	6,122	784	1,081	1,203	2,443	3,665	4,993	222
2500x2500	1,424	922	480	2,992	2,092	571	3,068	122
3000x3000	1,599	422	314	2,371	508	1,405	609	312
AVERAGE	2,174.60	486.26	430.29	1,533.86	1,130.26	1,265.85	1,997.38	155.33

Table 6: Total Execution Time of Dense LPs

Size	**BR**	**DR**	**GIM**	**LRCM**	**PPR**	**QR**	**SR**	**SER**
1000x1000	29.55	5.74	63.34	48.48	8.95	13.42	9.58	160.64
1500x1500	56.03	12.00	157.17	114.46	22.56	34.55	22.56	305.47
2000x2000	719.45	134.48	1,217.01	1,030.08	169.44	423.60	200.37	4,053.10
2500x2500	797.75	132.52	1,981.13	836.18	190.82	261.06	266.36	3,238.72
3000x3000	624.58	157.72	1,312.25	1,402.15	313.87	400.61	227.12	4,575.52
AVERAGE	445.47	88.49	946.18	686.27	141.13	226.65	145.20	2,466.69

Table 7: Number of Iterations of Dense LPs

Size	**BR**	**DR**	**GIM**	**LRCM**	**PPR**	**QR**	**SR**	**SER**
1000x1000	4,132	628	477	2,091	1,287	1,262	2,342	182
1500x1500	2,254	523	282	2,359	962	968	1,857	129
2000x2000	18,428	3,537	4,421	7,357	9,797	8,595	12,132	983
2500x2500	6,496	2,158	1,446	11,998	4,079	2,827	9,582	433
3000x3000	8,172	1,816	1,707	11,877	2,797	4,849	3,033	453
AVERAGE	7,896.34	1,732.40	1,666.77	7,136.41	3,784.49	3,700.09	5,789.05	435.97

Eight well known pivoting rules have been reviewed and compared in this paper. The computational study over a set of randomly generated optimal sparse and dense LPs and a set of benchmark LPs (Netlib-optimal, Kennington, Netlib-infeasible) showed that only Dantzig's Rule solved all instances. Furthermore, Dantzig's Rule performs better than the other pivoting rules in terms of execution time both on randomly generated and benchmark LPs. Finally, the Steepest Edge Rule requires the fewest number of iterations, but it has the worst execution time compared to the other pivoting rules.

References

[1] Bazaraa, M.S., Jarvis, J.J., Sherali, H.D., Linear Programming and Network Flows. John Wiley & Sons, Inc., 1990.

[2] Benichou, M., Gautier, J., Hentges, G., Ribiere, G., "The efficient solution of large-scale linear programming problems", *Mathematical Programming*, 13(1977) 280-322.

[3] Bland, R.G., "New finite pivoting rules for the simplex method", *Mathematics of Operations Research*, 2(2)(1977) 103-107.

[4] Carolan, W.J., Hill, J.E., Kennington, J.L., Niemi, S., Wichmann, S.J., "An Empirical Evaluation of the KORBX Algorithms for Military Airlift Applications", *Operations Research*, 38(2)(1990) 240-248.

[5] Clausen, J., "A note on Edmonds-Fukudas pivoting rule for the simplex method", *European Journal of Operational Research*, 29 (1987) 378-383.

[6] Dantzig, G.B., Linear programming and extensions, Princeton University Press, New Jersey, 1963.

[7] Forrest, J.J., Goldfarb, D., "Steepest-edge simplex algorithms for linear programming", *Mathematical programming*, 57(1-3)(1992) 341-374.

[8] Fukuda, K., Oriented matroid programming, Ph.D. Thesis, Waterloo University, Waterloo, Ontario, Canada, 1982.

[9] Gärtner, B., Randomized optimization by Simplex-type methods, PhD thesis, Freien Universität, Berlin, 1995.

[10] Gay, D.M., "Electronic mail distribution of linear programming test problems", *Mathematical Programming*, Society COAL Newsletter 13(1985) 10-12.

[11] Goldfarb, D., Reid, J.K., "A Practicable Steepest-Edge Simplex Algorithm", *Mathematical Programming*, 12(3)(1977) 361-371.

[12] Harris, P.M.J., "Pivot selection methods for the Devex LP code", *Mathematical Programming*, 5 (1973) 1-28.

[13] Klee, V., Minty, G.J., How good is the simplex algorithm, in: Shisha, O. (ed.) Inequalities - III. Academic Press, New York and London, 1972.

[14] Maros, I., Khaliq, M.H., "Advances in design and implementation of optimization software", *European Journal of Operational Research*, 140(2) (1999) 322-337.

[15] Murty, K.G., "A note on a Bard type scheme for solving the complementarity problem",*Opsearch*, 11 (1974) 123-130.

[16] Papadimitriou, C.H., Steiglitz, K., Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, Inc., 1982.

[17] Samaras, N., Sifaleras, A., Triantafyllidis, C., "A primal - dual exterior point algorithm for linear programming problems", *Yugoslav Journal of Operations Research*, 19 (1)(2009) 123-132.

[18] Sifaleras, A., "Minimum cost network flows: Problems, algorithms, and software", *Yugoslav Journal of Operations Research*, 23 (1)(2013) 3-17.

[19] Świetanowski, A., "A new steepest edge approximation for the simplex method for linear programming", *Computational Optimization and Application*, 10 (3)(1998) 271-281.

[20] Thomadakis, M.E., Implementation and Evaluation of Primal and Dual Simplex Methods with Different Pivot-Selection Techniques in the LPBench Environment, A Research Report. Texas A&M University, 1994.

[21] Vanderbei, R.J., Linear Programming: Foundations and Extensions (2nd ed.), Kluwer Academic Publishers, Whitmore, 2001.
[22] Wang, Z., "A modified version of the Edmonds-Fukuda algorithm for LP problems in the general form", *Asia-Pacific Journal of Operational Research*, 8(1)(1991)55-61.
[23] Zadeh, N., What is the worst case behavior of the simplex algorithm? Technical report, Stanford University, 1980.
[24] Zhang, S., On anti-cycling pivoting rules for the simplex method, Operations Research Letters 10(4)(1991) 189-192.
[25] Ziegler, G.M., Linear programming in oriented matroids, Technical Report No. 195, University of Augsburg, Germany, 1990.