Journal of Computational Mathematics, Vol.18, No.6, 2000, 587–596.

PRIMAL PERTURBATION SIMPLEX ALGORITHMS FOR LINEAR PROGRAMMING*[1)]

Ping-qi Pan
(*Department of Applied Mathematics, Southeast University, Nanjing 210096, China.*)

Abstract

In this paper, we propose two new perturbation simplex variants. Solving linear programming problems without introducing artificial variables, each of the two uses the dual pivot rule to achieve primal feasibility, and then the primal pivot rule to achieve optimality. The second algorithm, a modification of the first, is designed to handle highly degenerate problems more efficiently. Some interesting results concerning merit of the perturbation are established. Numerical results from preliminary tests are also reported.

Key words: Linear programming, Perturbation, Primal simplex algorithm, Partially revised tableau.

1. Introduction

Extensive research in linear programming, such as [1,2,9,10,11, 12,13,14,19], has been to improve pivot rules to reduce the number of iterations required. Relatively less effort was made on perturbing problem data with pivot rules unaltered (for instance, the self-dual parametric method [7] and perturbation-based methods [3,5]). And, because of the papametrization, the latter do not proceed as simply as the conventional simplex algorithm itself.

Recently, Pan [17] proposes new perturbation simplex algorithms that do not have such shortcoming, and performed favorably in computational tests. These algorithms can be regarded as the *dual* ones. The purpose of this paper is to develop *primal* algorithms of such type, and hence offer the other half of the approach.

Each of the proposed algorithms proceeds in a so-called 'partially' revised tableau form. First, some perturbed version of the original problem is solved. Then, a primally feasible tableau for the original problem is recovered from the optimal tableau of the perturbed problem. If optimality remains unchanged after the recovery, the original problem is already solved; otherwise, the recovered tableau is used as a starting point to achieve optimality.

In Section 2, we develop the Phase-1 procedure based on perturbation. In Section 3, we give some interesting theoretical results concerning this strategy. In Section 4, the procedure is combined with the simplex method to form a two-Phase algorithm for

* Received November 12, 1997.

[1)] This work was supported by the National Science Foundation of China.

general purpose use. To solve highly degenerate problems more efficiently, the algorithm is then modified by applying the perturbation tactic to both phases in a cyclic manner. Finally, in Section 5, numerical results from our tests are reported, showing algorithms' promise of success.

2. The Perturbation Phase-1

We are concerned with linear programming problem in the standard form:

$$\max \quad z = cx \tag{2.1a}$$

$$\text{s.t. } Ax = b \tag{2.1b}$$

$$x \geq 0 \tag{2.1c}$$

where $A \in R^{m\times n}$ with $rank(A) = k \leq m < n$, $b \in R^m$, and c and x are row and column n-vectors, respectively.

Let an initial basis $B \in R^{m\times k}$ be given:

$$B = (a_{j_1}, \dots, a_{j_k}), \tag{2.2}$$

where a_{j_i} are the columns of A, corresponding to basic variables $x_{j_i} (i = 1, \dots, k)$. Denote by J_B the set of indices of basic variables and denote the remaining set by

$$\bar{J}_B = \{1, \dots, n\} \backslash J_B. \tag{2.3}$$

Let B^+ be the Moore-Penrose inverse of B and let c_B be the price coefficients corresponding to basic variables. The following form, termed *partially revised* simplex tableau, will be useful to our purpose:

$$\begin{array}{c|c} \bar{c} & \bar{z} \\ \hline B^+A & \bar{b} \end{array} \tag{2.4}$$

where $\bar{z}$, $\bar{b}$ and $\bar{c}$ are determined by

$$\bar{z} = c_B B^+ b \tag{2.5a}$$

$$\bar{c} = c_B B^+ A - c \tag{2.5b}$$

$$\bar{b} = B^+ b \tag{2.5c}$$

In the case in which (2.4) is both primally and dually feasible, i.e., the following two sets

$$I = \{i \mid \bar{b}_i < 0. \ i = 1, \dots, k\} \tag{2.6}$$

$$J = \{j \mid \bar{c}_j < 0, \ j \in \bar{J}_B\}, \tag{2.7}$$

are empty, the linear program is already solved, and hence we are done. Suppose that (2.4) is neither primally nor dually feasible. We perturb $\bar{c}_j (\forall j \in J)$ to some predetermined numbers $\delta_j > 0$, and consequently turn (2.4) into the following tableau:

$$\begin{array}{c|c} \bar{c}' & \bar{z} \\ \hline B^+A & \bar{b} \end{array} \tag{2.8a}$$

where

$$\bar{c}'_j = \begin{cases} \delta_j, & \forall j \in J, \\ \bar{c}_j, & \forall j \in \{1, \dots, n\} \backslash J. \end{cases} \tag{2.8b}$$

It can easily be verified that tableau (2.8) actually corresponds to the following perturbation of (2.1):

$$\max \ z = \hat{c}x, \tag{2.9a}$$

$$\text{s.t. } Ax = b \tag{2.9b}$$

$$x \geq 0, \tag{2.9c}$$

where

$$\hat{c}_j = \begin{cases} c_B B^+ a_j - \delta_j, & \forall j \in J, \\ c_j, & \forall j \in \{1, \ldots, n\} \backslash J. \end{cases} \tag{2.9d}$$

Clearly, tableau (2.8) is dually feasible, and hence can be used to get the dual simple method started to solve program (2.9).

Any dual rule is now applicable for pivoting. The classical one is to select a pivot-row index r such that

$$\bar{b}_r = \min\{\bar{b}_i \mid i = 1, \ldots, k\} < 0 \tag{2.10}$$

and the pivot-column s such that

$$\bar{c}'_s/(B^+ a_s)_r = \max\{\bar{c}'_j/(B^+ a_j)_r \mid j \in J'\}, \tag{2.11a}$$

where

$$J' = \{j \mid (B^+ a_j)_r < 0, \ j \in \bar{J}_B\}. \tag{2.11b}$$

The Moore-Penrose inverse of the new basis can be determined by the following formula: (See [15] for more detail)

$$\tilde{B}^+ := B^+ + \frac{(e_r - B^+ a_s) e_r^T B^+}{(B^+ a_s)_r}, \tag{2.12}$$

where e_r is the identity k-vector with the r-th component 1, and the corresponding new partially revised tableau is then:

$$\begin{array}{c|c} \tilde{c}' & \tilde{z}' \\ \hline \tilde{B}^+ A & \tilde{b} \end{array} \tag{2.13}$$

where $\tilde{z}, \tilde{c}$ and $\tilde{b}$ may be computed from their predecessors:

$$\tilde{z}' = \bar{z} - (\bar{b}_r/\bar{a}_{rs})\bar{c}'_s \tag{2.14a}$$

$$\tilde{c}'_j = \bar{c}'_j - (\bar{a}_{rj}/\bar{a}_{rs})\bar{c}'_s \qquad j = 1, \ldots, n \tag{2.14b}$$

$$\tilde{b}_i = \begin{cases} \bar{b}_r/\bar{a}_{rs}, & \text{for} \quad i = r \\ \bar{b}_i - (\bar{b}_r/\bar{a}_{rs})\bar{a}_{is} & \text{for} \quad i = 1, \ldots, k; i \neq r \end{cases} \tag{2.14c}$$

Thus, we completed a single iteration, and thereafter we start anew.

Now suppose that termination occurs with an optimal tableau, say (2.13), for program (2.9); or more precisely, it holds that $\tilde{c}' \geq 0$, $\tilde{b} \geq 0$. Then, a *feasible* tableau for the original linear program (2.1), can be recovered from (2.13) by simply replacing $\tilde{z}'$ and $\tilde{c}'$ with

$$\tilde{z} = c_{\tilde{B}} \tilde{B}^+ b \tag{2.15}$$

$$\tilde{c} = c_{\tilde{B}} \tilde{B}^+ A - c \tag{2.16}$$

since such replacement does not affect the right-hand side at all. If, it happens that $\tilde{c} \geq 0$ holds, desirably, the recovered tableau (or the basis $\tilde{B}$) is already optimal to the original program; otherwise, it can be used to initiate Phase-2 of the simplex method.

3. Analysis of the Perturbation

By making an analysis of the perturbation, in this section we attack the interesting question below:

What conditions can guarantee that the tableau, restored from (2.13), is optimal to the original program (2.1)?

Denote by (2.9)' the linear program, resulting from (2.9) by replacing (2.9d) with any given $\hat{c}$.

Lemma 3.1. *Suppose that program (2.1) has an optimal solution which is also optimal to the (2.9)'. If any optimal tableau (2.13) of (2.9)' is dually non-degenerate, the tableau restored from the (2.13) gives the optimal solution to program (2.1).*

Proof. In this case dual non-degeneracy of the (2.13) ensures the uniqueness of the optimal solution of the (2.9)'; on the other hand, that the basic feasible solution given by the (2.13) is not an optimal solution of (2.1) implies that there exist at least two optimal solutions to (2.9)', leading to a contradiction. Therefore, the tableau restored from the (2.13) gives the optimal solution of (2.1).

Nevertheless, the preceding does not say whether or not the restored tableau is optimal, in the sense of satisfaction of the sufficient optimality condition. We have the following in this respect:

Lemma 3.2. *Suppose that program (2.1) has an optimal basis which is also optimal to (2.9)'. If any optimal tableau (2.13) of (2.9)' is both primally and dually non-degenerate, the tableau restored from the (2.13) is optimal to (2.1).*

Proof. It is well known that the primal and dual non-degeneracy of the (2.13) guarantees uniqueness of the optimal basis of (2.9)'. So, the optimal basis $\tilde{B}$, related to the (2.13), of (2.9)' is also optimal for program (2.1) because otherwise there would exist at least two optimal bases for (2.9)', leading to a contradiction again. Thus, the tableau restored from the (2.13) is the optimal tableau of (2.1).

By the theory of linear programming, if there exists the unique, primally non-degenerate optimal solution to the original program (2.1), then the optimal solution must be also dually non-degenerate. In this case, if $\|\hat{c} - c\|$ is small enough, the associated optimal basis of (2.1) is also optimal to (2.9)', according to a classical sensitivity analysis; consequently, by Lemma 3.2, the tableau restored from the (2.13) is optimal to (2.1) desirably, if (2.13) is both primally and dually non-degenerate. Unfortunately, for the (2.9) itself, setting $\hat{c}$ by (2.9d) makes $\|\hat{c} - c\|$ essentially uncontrollable, and usually not small. However, we shall show that the according tableau restored can still be optimal, in certain circumstances.

Lemma 3.3. *Let B_* be an optimal basis of the original program (2.1) and let $J \cap J_{B_*}$ be empty. Then, B_* is also an optimal basis for program (2.9).*

Proof. It is seen that, corresponding to the same basis B_*, the optimal tableau of (2.1) and the tableau of (2.9) are the same except for those components of the two relative price rows which are with indices $j \in J$. For the optimal tableau of (2.1), these components are

$$\sigma_j \equiv c_{B_*} B_*^+ a_j - c_j \geq 0, \qquad \forall j \in J. \tag{3.1}$$

By (2.9d) and emptiness of $J \cap J_{B_*}$, implying $\hat{c}_{B*} = c_{B*}$, the corresponding components of the tableau of program (2.9) are

$$\begin{aligned} \hat{c}_{B_*}(B_*)^+ a_j - \hat{c}_j &= c_{B_*} B_*^+ a_j - (c_B B^+ a_j - \delta_j) \\ &= (c_{B_*} B_*^+ a_j - c_j) - (c_B B^+ a_j - c_j) + \delta_j \\ &= \sigma_j - \bar{c}_j + \delta > \sigma_j \geq 0, \qquad \forall j \in J, \end{aligned} \tag{3.2}$$

where inequality yields from combining $\delta > 0$, (2.5b) and (2.7). Therefore, the B_* is also an optimal basis for program (2.9).

Theorem 3.4. *Under the same assumption of Lemma 3.3, the tableau restored from (2.13) gives an optimal solution to the original program if (2.13) is dually non-degenerate, and even an optimal tableau of it if, in addition, (2.13) is also primally non-degenerate.*

Proof. According to Lemma 3.3, there exists an optimal basis, and hence optimal solution of (2.1) which are optimal for (2.9). So, if (2.13) is dually non-degenerate, the tableau restored from (2.13) gives the optimal solution to (2.1), by Lemma 3.1; if, in addition, (2.13) is primally non-degenerate, the restored tableau itself is also optimal to (2.1), by Lemma 3.2.

The preceding says that if only those components of the relative price row of (2.4) are changed, for which the associated variables are also non-basic for the optimal tableau, then such changes do not interfere with our purpose, assuming both primal and dual non-degeneracy. Moreover, the first half of the theorem can also be proved if '(2.13) is dually non-degenerate' is replaced by another suitable condition:

Theorem 3.5. *Under the same assumption of Lemma 3.3, the tableau restored from (2.13) gives an optimal solution to the original program (2.1) if only those components of the relative price row of (2.4) are changed which correspond to non-basic variables of (2.13), i.e. $J \cap J_{\tilde{B}}$ is empty.*

Proof. Note that $\tilde{z}'$ and $\tilde{c}'$ in (2.13) can be written alternatively in term of the basis:

$$\tilde{z}' = \hat{c}_{\tilde{B}} \tilde{B}^+ b, \tag{3.3}$$

$$\tilde{c}' = \hat{c}_{\tilde{B}} \tilde{B}^+ A - \hat{c}. \tag{3.4}$$

According to Lemma 3.3, B_* is an optimal basis for program (2.9) and, besides, we have

$$\hat{c}_{B_*} B_*^+ b = c_{B_*} B_*^+ b \equiv z_*,$$

implying that the optimal values of program (2.9) and of the original program (2.1) are equal. Thus, since (2.13) is an optimal tableau for (2.9), it holds that

$$\tilde{z}' = z_*. \tag{3.5}$$

Then, combining (3.5), (3.3), (2.15) and emptiness of $J \cap J_{\tilde{B}}$ leads to

$$z_* = c_{\tilde{B}} \tilde{B}^+ b = \tilde{z}, \tag{3.6}$$

implying that the tableau restored from (2.13) reaches an optimal solution to (2.1).

4. Two-Phase Algorithms

It is clear that the termination of the procedure described in Section 2 occurs if either $\bar{b}_r \geq 0$, implying the achievement of primal feasibility, or the set J' defined by (2.11b) is empty, implying infeasibility of the program—this is valid for *both* the perturbed and the original programs since the recovery affects the relative price row only.

The dual procedure described in Section 2 is combined with the primal simplex method to form Algorithm 4.1 below. Note that there the dual Phase-1 consists of Steps 2 through 8, the primal Phase-2 consists of Phase-2 Steps 8 through 14, and symbols ia and ic are controlling parameters, defined as follows:

ia =1 or 2: proceeding with Phase-1 or Phase-2;

ic=0 or 1: proceeding with relative price row unchanged or changed.

Algorithm 4.1. Let $B^+ \in R^{k\times m}$ be the Moore-Penrose inverse of an initial basis. Given constants $\delta_j > 0, j = 1, \ldots, n$, and a tolerance ϵ such that $0 < \epsilon \leq \min\{\delta_j \mid j = 1, \ldots, n\}$, this algorithm solves linear program (2.1).

1. Set $ia = 1$ and $ic = 0$.
2. Compute $\bar{z}, \bar{c}$ and $\bar{b}$ by (2.5).
3. If the index set, defined by (2.7), is nonempty, set $\bar{c}_j = \delta_j,\ j \in J$, and $ic = 1$.
4. Determine the row index r according to (2.10).
5. If $\bar{b}_r \geq -\epsilon$, then:

(i) stop if $ic = 0$;

(ii) restore $\bar{z}$ and $\bar{c}$ by (2.5a) and (2.5b), respectively;

(iii) set $ia = 2$, and then go to Step 10.

6. Stop if index set J', defined by (2.11b), is empty.
7. Determine the column index s by (2.11).
8. Update B^+, $\bar{c}$ and $\bar{b}$ by (2.12), (2.14b) and (2.14c), respectively.
9. Go to Step 4 if $ia = 1$.
10. Determine s such that

$$\bar{c}_s = \min\{\bar{c}_j \mid j \in \bar{J}_B\} \tag{4.1}$$

where $\bar{J}_B$ is defined by (2.3).

11. Stop if $\bar{c}_s \geq -\epsilon$
12. Stop if the row index set

$$I' = \{i \mid (B^+a_s)_i > 0,\ i = 1, \ldots, k\} \tag{4.2}$$

is empty.

13. Determine r such that

$$\bar{b}_r/(B^+a_s)_r = min\{\bar{b}_i/(B^+a_s)_i \mid i \in I'\} \tag{4.3}$$

14. Go to Step 8.

Based on properties of the dual and primal pivoting rules, we state:

Theorem 4.2. *Assuming dual non-degeneracy for Phase-1 and primal non-degeneracy for Phase-2, Algorithm 4.1 terminates at either*

(a) *Step 5(i) or 11, with an optimal solution of (2.1) reached; or*

(b) *Step 6, indicating infeasibility of the program;or*

(c) *Step 12, indicating upper unboundedness of it.*

It is seen that dual and/or primal degeneracy may still have adverse effects on the computational performance of Algorithm 4.1, as in the classical dual and/or primal methods. For solving highly degenerate problems efficiently, it might be advantageous to perturb degenerate entries encountered at each iteration throughout the solution process. Moreover, the algorithm can be modified so that it goes back to its Phase-1 portion, and restarting anew after Phase-2 steps are carried out; such circle is repeated again and again until the program is solved. When δ_j's are small, however, it might be likely for the algorithm to terminate after only one circle, because the perturbed program could be close to the original one in this case.

The preceding ideas are implemented in Algorithm 4.3 below, where the dual Phase-1 consists of Steps 2 through 8, while the primal Phase-2 consists of Steps 8 through 15. It includes three controlling parameters: ia and ic are as the same as those in Algorithm 4.1, while an additional parameter is defined by

ib=0 or 1: proceeding with the right-hand side unchanged or changed.

Algorithm 4.3. Assuming the same input as that of Algorithm 4.1, this algorithm solves linear program (2.1).

1. Set $ia = 1, ib = 0$ and $ic = 0$.

2. Compute $\bar{z}, \bar{c}$ and $\bar{b}$ by (2.5).

3. If the index set

$$J = \{j \mid \bar{c}_j < \epsilon,\ j \in \bar{J}_B\}, \tag{4.4}$$

is nonempty, set $\bar{c}_j = \delta_j,\ \forall j \in J$ and $ic = 1$.

4. Determine index r by (2.10).

5. If $\bar{b}_r \geq -\epsilon$, then:

(i) stop if $ic = 0$;

(ii) restore $\bar{z}$ and $\bar{c}$ by (2.5a) and (2.5b), respectively;

(iii) set $ia = 2, ic = 0$, and then go to Step 10.

6. Stop if J', defined by (2.11b), is empty.

7. Determine s by (2.11).

8. Update B^{+}, $\bar{c}$ and $\bar{b}$ by (2.12), (2.14b) and (2.14c), respectively.

9. Go to Step 3 if $ia = 1$.

10. Determine index s by (4.1).

11. If $\bar{c}_s \geq -\epsilon$, then

(i) stop if $ib = 0$;

(ii) restore $\bar{z}$ and $\bar{b}$ by (2.5a) and (2.5c), respectively;

(iii) set $ia = 1, ib = 0$, and then go to Step 4.

12. Stop if I', defined by (4.2), is empty.

13. If the row index set $I = \{i \mid \bar{b}_i < \epsilon,\ i \in I'\}$ is nonempty, set $\bar{b}_i = \delta_i,\ \forall i \in I$ and $ib = 1$.

14. Determine r according to (4.3)
15. Go to Step 8.

Theorem 4.4. *Outlets of Algorithm 4.3 have the same meanings and end products as those of Algorithm 4.1.*

Remark 4.5. Algorithm 4.3 terminates if dual and primal degeneracy occur only finitely many times.

5. Computational Results

To corroborate our theory and to gain an idea of the numerical behavior of the proposed method, we have performed some computational experiments.

Coded in FORTRAN 77 program, Algorithms 4.1 and 4.3 were implemented, and compared with the two-phase revised simplex algorithm using Dantzig's original rule. We used a crash procedure, similar to Algorithm 6.5 of [15], to provide the Moore-Penrose inverse of an initial basis. Values of the parameters taken were $\epsilon = 10^{-6}$ and $\delta_j = 10^{-6}$, $\forall j = 1, \ldots, n$. The machine precision used was about 16 decimal digits.

Test problems fall into four groups. The first group involves 113 arbitrarily collected text problems with up to 22 decision variables and strict inequality constraints. It would be interesting to see how the new Algorithms behave with Klee-Minty problems (see, for example, [18]); so, we put in the second group three such problems, designated by KM1, KM2 and KM3, respectively. The third group involves 7 randomly produced problems. And the fourth has 4 larger problems available.

In terms of iteration numbers required, we summarize in Table 1 numerical results obtained with Groups 1 and 2, and in Table 2 with Groups 3 and 4. The tables also give iteration number ratios by running new algorithms and the classical method. We point out that computational cost per iteration for the three algorithms are about equal.

The results show that both new algorithms outperformed the conventional simplex algorithm with the tested problems significantly. In fact, this is the case even for each of the problems. So, it seems that the new algorithms are superior to the simplex algorithm with problems of such sizes.

As far as the new algorithms concerned, the performance of Algorithm 4.3 appears to be better than that of Algorithm 4.1 although margins are small. It may be expected that difference between them is larger with highly degenerate problems.

Certainly, drawing general and final conclusions requires a more extensive computational study with large-scale problems, as is only possible with some sparsity-exploiting version of the new method. At least, however, we feel safe to conclude that the proposed perturbation algorithms are attractive, and deserves a further investigation.

TABLE 1

	$A.4.3$	$A.4.1$	$Clas.$	$C. : A.4.3$	$C. : A.4.1$
Total for Group 1	454	460	1252	2.76	2.72
KM1: n=8	11	11	255	23.18	23.18
KM2: n=10	13	15	1023	78.69	68.20
KM3: n=12	17	19	4095	240.88	215.53
Total for Group 2	41	45	5373	131.05	119.40

TABLE 2

$Problem$	M	N	$A.4.3$	$A.4.1$	$Clas.$	$C. : A.4.3$	$C. : A.4.1$
1	23	49	30	30	60	2.00	2.00
2	25	50	23	36	49	2.13	1.36
3	22	45	21	21	41	1.95	1.95
4	25	45	23	26	48	2.09	1.85
5	25	45	19	34	39	2.05	1.15
6	25	48	14	15	34	2.43	2.27
7	24	48	21	19	42	2.00	2.21
Total for Group 3			151	181	313	2.07	1.73
1	27	51	9	9	27	3.00	3.00
2	28	56	4	4	38	9.50	9.50
3	55	137	93	95	170	1.83	1.79
4	56	138	72	79	172	2.39	2.18
Total for Group 4			178	187	407	2.29	2.18

References

[1] E.M.L. Beale, An alternative method for linear programming, *Proceedings of Cambridge Philosophical Society,* **50** (1954), 513–523.

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

[3] A. Charnes, Optimality and degeneracy in linear programming, *Econometrica,* **20** (1952), 160–170.

[4] M. Benichou, J.M. Gauthier, G. Hentges, G. Ribiere, The effcient solution of large-scale linear programming problems, *Mathematical Programming,* **13** (1977), 280–322.

[5] G.B. Dantzig, Application of the simplex method to a transportation problem, in T.C. Koopman (ed.), Activity Analysis of Production and Allocation, John Wiely & Sons, Inc., New York, 1951, 359–373.

[6] G.B. Dantzig, The Dual Simplex Algorithm, RAND Report RM-1270, The RAND Corporation, Santa Monica, CA, 1954.

[7] G.B. Dantzig, Linear Programming and Extensions, Princeton University Press, Princeton, New Jersey, 1963.

[8] S.I. Gass, Linear Programming (Fifth Edition), Mcgraw-HillBoo Company, New York, 1985.

[9] P.E. Gill, Walter Murray, M.A. Saunders, M.H. Wrighr, A practical anticycling procedure for linearly constrained optimization, *Mathematical Programming,* **45** (1989), 437–474.

[10] D. Goldfarb, J.K. Reid, A practicable steepest-edge simplex algorithm, *Mathematical Programming,* **12,** 361–371.

[11] R.G. Jeroslow, The simplex algorithm with the pivoting rule of maximizing criterion improvement, *Disc. Math.,* **4** (1973), 367–377.

[12] P.M.J. Harris, Pivot selection methods of the Devex LP code, *Mathematical Programming Study,* **4** (1975), 30–57.

[13] C.E. Lemke , The dual method of solving the linear programming problems, *Naval Research Logistics Quarterly,* **1** (1954), 36–47.

[14] P.Q. Pan, Practical finite pivoting rules for the simplex method, *OR Spektrum,* **12** (1990), 219–225.

[15] P.Q. Pan, A simplex-like method with bisection for linear programming, *Optimization,* **22**:5 (1991), 717–743.

[16] P.Q. Pan, A variant of the dual pivot rule in linear programming, *Journal of Information & Optimization Sciences,* **15**:3 (1994), 405–413.

[17] P.Q. Pan, New Perturbation Simplex Algorithms for Linear Programming, Accepted for publication by *Journal of Computational Mathematics.*

[18] A. Schrijver, The Theory of Linear and Integer Programming, John Wiley & Sons, Chichester, 1986.

[19] P. Wolfe, A technique for resolving degeneracy in linear programming, *Journal of the Society for SIAM,* **11** (1963), 205–211.