Journal of Southeast University (English Edition) Vol.19 No.3 Sept. 2003 ISSN 1003—7985

A non-monotone Phase-1 method in linear programming

Pan Pingqi Li Wei

(Department of Mathematics, Southeast University, Nanjing 210096, China)

Abstract: To gain superior computational efficiency, it might be necessary to change the underlying philosophy of the simplex method. In this paper, we propose a Phase-1 method along this line. We relax not only the conventional condition that some function value increases monotonically, but also the condition that all feasible variables remain feasible after basis change in Phase-1. That is, taking a purely combinatorial approach to achieving feasibility. This enables us to get rid of ratio test in pivoting, reducing computational cost per iteration to a large extent. Numerical results on a group of problems are encouraging.

Key words: linear programming; Phase-1; ratio-test-free; pivoting rule

The simplex method moves on the polyhedron, from vertex to adjacent vertex along edges while increasing (without the loss of generality) the objective function value. Based on such a philosophy, standard simplex-like methods (either of Phase-1 or Phase-2) work under the following conditions:

① Some function value increases monotonically;

② All feasible variables remain feasible after basis change.

P. Wolfe[1] first proposed the idea of dropping ②. In his composite Phase-1 approach, the work of achieving feasibility is combined, to some degree, with the work of achieving optimality whereas, instead of condition ②, the amount of total infeasibility is reduced. Istvan Maros[2] and K.Belling-Seib[3] suggested their successively improved schemes to reduce computational cost at each iteration. However, these methods still require extra computational effort per iteration although they usually require fewer iterations.

On the other hand, as far as the authors know, all existing simplex variants work under condition ①. Assuming non-degeneracy, the function will increase strictly monotonically, guaranteeing the termination of the solution process; yet, this is of only conceptual or pedagogical interest because the non-degeneracy assumption does not coincide with the fact that cycling occurs very frequently in practice.

In order to gain more superior computational efficiency, some new algorithms have been developed in recent years[4-6]. In this paper, we change the philosophy of the simplex methods further, not only relaxing condition ① but also condition ② in Phase-1, that is, taking a purely combinatorial approach to achieving feasibility. This also enables us to get rid of the ratio test in pivoting, reducing computational cost per iteration to a large extent.

In section 1, we propose some ratio-test-free pivoting rules, the use of which results in different Phase-1 procedures. In section 2, as an example, we incorporate one of the rules with the classical Phase-2 of the simplex method to form a general purpose algorithm. Finally, in section 3, we present numerical results of our preliminary tests.

1 Phase-1 Pivoting Rules

Consider a linear programming problem in the following form:

$$\left.\begin{aligned} &\max\ x_0\\ &\text{s.t. } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\\ &x_j \geqslant 0 \qquad j = 1,\cdots,n \end{aligned}\right\} \tag{1}$$

where $\boldsymbol{A} \in \mathbf{R}^{m\times n}$ with $\operatorname{rank}(\boldsymbol{A}) = k(k \leqslant m \leqslant n)$.

Using notation

$$x_0 = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x},\ \boldsymbol{A} := \begin{bmatrix} -1 & \boldsymbol{c}^{\mathrm{T}} \\ 0 & \boldsymbol{A} \end{bmatrix}$$

we construct the following linear programming (LP) problem from (1)

$$\left.\begin{aligned} &\max\ x_0\\ &\text{s.t. } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\\ &x_j \geqslant 0 \qquad j = 1,\cdots,n \end{aligned}\right\} \tag{2}$$

where $\boldsymbol{A} \in \mathbf{R}^{(m+1)\times(n+1)}$ with $\operatorname{rank}(\boldsymbol{A}) = k+1(k \leqslant m+1 \leqslant n+1)$, and $\boldsymbol{x} := (x_0\ \boldsymbol{x}^{\mathrm{T}})^{\mathrm{T}}$, $\boldsymbol{b} := (0\ \boldsymbol{b}^{\mathrm{T}})^{\mathrm{T}}$. We will not handle (1) directly, but handle (2) instead.

In the sequel, the i-th row of a matrix $\bullet$ will be

Received 2003-01-03.

Foundation item: The National Natural Science Foundation of China (19971014).

Biography: Pan Pingqi(1942—), male, professor; panpq@seu.edu.cn.

designated by $(\cdot)_i$. Let $\boldsymbol{B} \in \mathbf{R}^{(m+1)\times(k+1)}$ be the current basis with some index set

$$J_B = \{j_0 \equiv 0, j_1, \cdots, j_k\} \tag{3}$$

and let $\boldsymbol{B}^+ \in \mathbf{R}^{(k+1)\times(m+1)}$ be the Moore-Penrose inverse of $\boldsymbol{B}$. Then x_{j_i}, $i = 0,1,\cdots, k$ are the related basic set of variables, and $\boldsymbol{B}_0^+ \boldsymbol{b}$ is the objective function value at the current basic solution. Introduce notation

$$J_N = \{1, \cdots, n\} - J_B \tag{4}$$

Suppose that the basis $\boldsymbol{B}$ is infeasible, i.e., the row index set

$$I = \{i \mid (\boldsymbol{B}^+)_i \boldsymbol{b} < 0, i = 1, \cdots, k\} \tag{5}$$

is nonempty. For making a basis change, the following ratio-test-free pivoting rules may be employed.

Rule 1 Select the pivot-row index r such that

$$r = \operatorname{argmin}\{(\boldsymbol{B}^+)_i \boldsymbol{b} \mid i \in I\} \tag{6}$$

If the column index set

$$J = \operatorname{argmin}\{j \mid (\boldsymbol{B}^+)_r \boldsymbol{a}_j < 0, j \in J_N\} \tag{7}$$

is nonempty, select the pivot-column index s such that

$$s = \operatorname{argmin}\{(\boldsymbol{B}^+)_r \boldsymbol{a}_j \mid j \in J\} \tag{7}$$

Suppose that r and s have been determined. Then the according new basis can be written as

$$\bar{\boldsymbol{B}} = \boldsymbol{B} + (\boldsymbol{a}_s - \boldsymbol{a}_{j_r}) \boldsymbol{e}_r^{\mathrm{T}} \tag{8}$$

where $\boldsymbol{e}_r$ is the identity $(k+1)$-vector with the r-th component 1; and from Sherman-Morrison formula we know that its Moore-Penrose inverse is

$$\bar{\boldsymbol{B}}^+ = \boldsymbol{B}^+ + \frac{(\boldsymbol{e}_r - \boldsymbol{B}^+ \boldsymbol{a}_s)(\boldsymbol{B}^+)_r}{(\boldsymbol{B}_r^+)\boldsymbol{a}_s} \tag{10}$$

Thus one step is completed. The step is repeated until either set I is empty, indicating achievement of feasibility, or I is nonempty but J is empty, indicating infeasibility of the linear program.

Let us examine effects of such a basis change. The value of the new basic set of variables can be represented in terms of their predecessors, i.e.,

$$(\bar{\boldsymbol{B}}^+)_i \boldsymbol{b} = \begin{cases} (\boldsymbol{B}^+)_i \boldsymbol{b} - ((\boldsymbol{B}^+)_r \boldsymbol{b}/(\boldsymbol{B}^+)_r \boldsymbol{a}_s)(\boldsymbol{B}^+)_i \boldsymbol{a}_s \\ \qquad i = 0, \cdots k; i \neq r \\ (\boldsymbol{B}^+)_r \boldsymbol{b}/(\boldsymbol{B}^+)_r \boldsymbol{a}_s \qquad i = r \end{cases} \tag{11}$$

Since the variable x_{j_r} becomes nonbasic, its value increases from $(\boldsymbol{B}^+)_r \boldsymbol{b} < 0$ up to 0; on the other hand, the variable x_s becomes basic, and its value increases from 0 up to $(\boldsymbol{B}^+)_r \boldsymbol{b} > 0$, where inequality results from (11), (6) with (5) and (8) with (7). So, at the price of not maintaining current primal and dual feasibility, rule 1 eliminates one infeasibility at a time, involving computation of two vectors only, and making the basis change numerically stable due to favoring pivot with the largest possible absolute value. However, it might be more plausible to increase the value, if possible, or otherwise to decrease it as little as possible. This idea leads to the following variant.

Rule 2 Select r by (6). If set J defined by (7) is nonempty, select s such that

$$s = \operatorname{argmin}\{(\boldsymbol{B}^+)_0 \boldsymbol{a}_j \mid j \in J\} \tag{12}$$

Because of possible choice of a small pivot, the preceding rule can be numerically unstable. This shortcoming can be avoided by the following rule:

Rule 3 Select r by (6). Let $\delta > 0$ be a small number and J defined by (7) be nonempty. If the following set

$$J' = \{j \mid (\boldsymbol{B}^+)_r \boldsymbol{a}_j < -\delta, j \in J\} \tag{13}$$

is also nonempty, select s such that

$$s = \operatorname{argmin}\{(\boldsymbol{B}^+)_0 \boldsymbol{a}_j \mid j \in J'\} \tag{14}$$

else such that

$$s = \operatorname{argmin}\{(\boldsymbol{B}^+)_r \boldsymbol{a}_j \mid j \in J\} \tag{15}$$

In addition to the most infeasible variable x_{j_r}, it seems to be attractive in row pivoting to take into account the other infeasible variables as well as x_0. The extra computational effort involved in doing so is limited. Consider the sum

$$x_0 + \sum_{i \in I - \{r\}} x_{j_i} = \boldsymbol{v}\boldsymbol{b} - \sum_{j \in J_N} (\boldsymbol{v}\boldsymbol{a}_j) x_{j_i} \tag{16}$$

where $\boldsymbol{v}$ is the row vector,

$$\boldsymbol{v} = (\boldsymbol{B}^+)_0 + \sum_{i \in I - \{r\}} (\boldsymbol{B}^+)_i \tag{17}$$

giving the measurement $\boldsymbol{v}\boldsymbol{b}$ of the other infeasibility, combined with the value of x_0. The basic idea of the following rule is to increase the value of the preceding sum, or at least to decrease it as little as possible.

Rule 4 Select r by (6). If set J defined by (7) is nonempty, select s such that

$$s = \operatorname{argmin}\{\boldsymbol{v}\boldsymbol{a}_j \mid j \in J\} \tag{18}$$

Also, to avoid too small a pivot being chosen, the following variant may be employed:

Rule 5 Select r by (6). Let $\delta > 0$ be a small number and J defined by (7) be nonempty. If set J' defined by (13) is also nonempty, select s such that

$$s = \operatorname{argmin}\{\boldsymbol{v}\boldsymbol{a}_j \mid j \in J'\} \tag{19}$$

else such that

$$s = \operatorname{argmin}\{(\boldsymbol{B}^+)_r \boldsymbol{a}_j \mid j \in J\} \tag{20}$$

2 General Algorithm

Now we can describe the general algorithm for solving the linear programming (2). After achieving feasibility by using any of the rules, given in the previous section, Phase-2 of the simplex method is

immediately applicable to this purpose. As an example, the following model algorithm is formed by combining rule 1 with the classical Phase-2, where Phase-1 consists of steps 1 to 6, whereas Phase-2 consists of steps 7 to 12.

Algorithm 1 Given the Moore-Penrose inverse $\boldsymbol{B}^+ \in \mathbf{R}^{(k+1)\times(m+1)}$ of an initial basis, this algorithm solves linear programming (2).

Step 1 Go to step 7 if set I defined by (5) is empty;

Step 2 Determine pivot-row index r by (6);

Step 3 Stop if set J defined by (7) is empty;

Step 4 Determine pivot-column index s by (8);

Step 5 Update $\boldsymbol{B}^+$ by (10);

Step 6 Go to step 1;

Step 7 Determine s such that $s = \operatorname{argmin}\{(\boldsymbol{B}^+)_0 \boldsymbol{a}_j \mid j \in J_N\}$;

Step 8 Stop if $(\boldsymbol{B}^+)_0 \boldsymbol{a}_s \geqslant 0$;

Step 9 Stop if set $I' = \{i \mid (\boldsymbol{B}^+)_i \boldsymbol{a}_s > 0,\ i = 1, \cdots, k\}$ is empty;

Step 10 Determine r such that $r = \operatorname{argmin}\{(\boldsymbol{B}^+)_i \boldsymbol{b} / (\boldsymbol{B}^+)_i \boldsymbol{a}_s \mid i \in I'\}$;

Step 11 Update $\boldsymbol{B}^+$ by (10);

Step 12 Go to step 7.

Based on discussions made in section 1, and well-known properties of the simplex methods, we state:

Theorem 1 Assuming finiteness of Phase-1, and non-degeneracy throughout Phase-2, the algorithm terminate at either ① Step 8, with an optimal solution reached; or ② Step 3, indicating infeasibility of (2); or ③ Step 9, indicating upper unboundedness of (2).

The question remaining now is whether or not Phase-1 is finite. Because of finiteness of the number of bases, it is finite if and only if cycling occurs, i.e., some basis is repeated infinitely many times. We point out that all pivoting rules, given in section 1, do not monotonically increase (or decrease) some function value, and it has not been possible to rule out the possibility of cycling. However, like the well-known cycling risk for the classical pivoting rule, this might not be important in practice; and it might be very likely for cycling to occur in practice only rarely, as is supported by our computational experiment.

3 Computational Experiment

In this section, we report numerical results of our computational tests, giving some insight into the behavior of the new method.

We designed a crash procedure, based on the plausible characterization of optimal basis[7], to create the Moore-Penrose inverse of an initial basis as input. The procedure is embodied in the model algorithm below.

Algorithm 2 Given pivoting indices α_j, $j = 1, \cdots, n$, and a small number $\varepsilon > 0$, this algorithm produces a basis B and its Moore-Penrose inverse $\boldsymbol{B}^+$ with "ε-rank" k,

Step 1 Set $J_N = \{1, \cdots, n\}$ and determine $s_0 = 0$;

Step 2 Set $\boldsymbol{A}_1 = -\boldsymbol{e}_0$ and $\boldsymbol{A}_1{}^+ = -\boldsymbol{e}_0^{\mathrm{T}}$;

Step 3 Set $k = 0$ and $r = 0$;

Step 4 Set $k = k + 1$;

Step 5 Set $r = r + 1$;

Step 6 Determine $s_r = \operatorname{argmax}\{\alpha_j \mid j \in J_N\}$;

Step 7 Set $J_N = J_N - \{s_r\}$;

Step 8 Compute $\boldsymbol{d}_k = \boldsymbol{A}_{k-1}{}^+ \boldsymbol{a}_{s_r}$ and $\boldsymbol{c}_k = \boldsymbol{a}_{s_r} - \boldsymbol{A}_{k-1}\boldsymbol{d}_k$;

Step 9 If $\|\boldsymbol{c}_k\|_2^2 < \varepsilon$, and $r < n$, go to step 5;

Step 10 If $\|\boldsymbol{c}_k\|_2^2 \geqslant \varepsilon$, set $\boldsymbol{A}_k = (\boldsymbol{A}_{k-1} \mid \boldsymbol{a}_{s_r})$ and compute $\boldsymbol{A}_k{}^+ = \left(\dfrac{\boldsymbol{A}_{k-1}^+ - \boldsymbol{d}_k \boldsymbol{c}_k^+}{\boldsymbol{c}_k^+}\right)$;

Step 11 If $k < m$ and $r < n$, go to step 4;

Step 12 Set $\boldsymbol{B} = \boldsymbol{A}_k$ and $\boldsymbol{B}^+ = \boldsymbol{A}_k{}^+$, and stop.

Algorithm 1 can serve as a frame to construct variants by employing different pivoting rules. In our tests, the following 7 codes in Fortran 77 were tested, and compared with the revised two-phase simplex algorithm using Dantzig's original rule:

① Algorithm 1 with (8) replaced by (18);

② Algorithm 1 with (8) replaced by (19), (20) with (13), where $\delta = 0.01$;

③ Algorithm 1 with (8) replaced by (14), (15) with (13), where $\delta = 0.1$;

④ The same as ②, but where $\delta = 0.1$;

⑤ The same as ③, but where $\delta = 0.01$;

⑥ Algorithm 1 with (8) replaced by (12);

⑦ Algorithm 1.

The machine precision was about 16 decimal places. For each code, the tolerance used was 10^{-6}. In algorithm 2, $\varepsilon = 10^{-3}$ was taken. The test problems fall into three groups. The first group includes 91 problems with strict inequality constraints and up to 22 decision variables and constraints; the second involves 4 larger sparse problems in the standard form, for each its price coefficients were simply taken to be pivoting indices used as inputs of algorithm 2; and the third are three Klee-Minty problems.

In terms of optional the number of iterations required, numerical results obtained are summarized in Tab.1, where the 4 problems of group 2 are designated by P_1, P_2, P_3 and P_4, and Klee-Minty problems by KM_1, KM_2 and KM_3, respectively.

In Tab.1, the 8 codes are ranked according to totals of iterations required for each on 91 problems of group 1. Roughly speaking, such ranking also coincides with the case on group 2 or on group 3. Obviously, the performance of the classical method, which is on the rightest of the table, is inferior to all the new codes on problems of each group overall. It might be worth mentioning that this is the case even for each of the problems only except one: P_4 of group 2 when solved by using ⑥. We point out that computational cost with algorithm 2 alone is limited, and that cost per iteration, associated with the new codes, is significantly less than that associated with the classical method.

As far as only new codes are concerned, those with a rule taking all infeasible variables as well as x_0 into account performed better than the others.

Tab.1 Computational results in iterations

Problems	①	②	③	④	⑤	⑥	⑦	Dantzig's
Total for group 1	329	330	334	338	347	356	363	1 026
$P_1: m = 27\ n = 51$	7	5	11	5	12	13	8	27
$P_2: m = 28\ n = 56$	4	4	5	4	5	6	4	38
$P_3: m = 55\ n = 137$	51	58	52	57	81	85	59	170
$P_4: m = 56\ n = 138$	116	118	111	118	166	293	148	172
Total for group 2	178	185	179	184	264	397	219	407
$KM_1: m = n = 8$	11	11	11	11	5	5	11	255
$KM_2: m = n = 10$	15	15	15	15	11	5	15	1 023
$KM_3: m = n = 12$	19	19	18	19	12	5	19	4 095
Total for group 3	45	45	44	45	28	15	45	5 373

Finally, we report that the test problems are all arbitrarily collected ones, and no cycling has been observed so far. We conclude that these results do show promise of the new method although further computational tests are expected to be done.

References

[1] Wolfe P. The composite simplex algorithm [J]. *SIAM Review*, 1965, **7**: 42 - 54.

[2] Maros Istvan. A general Phase-I method in linear programming [J]. *European Journal of Operations Research*, 1986, **34**: 64 - 77.

[3] Belling-Seib K. An improved general Phase-I method in linear programming [J]. *European Journal of Operations Research*, 1988, **36**: 101 - 106.

[4] Pan P Q. A projective simplex method for linear programming [J]. *Linear Algebra and its Applications*, 1999, **29**(2): 99 - 125.

[5] Pan P Q. A new perturbation simplex algorithm for linear programming [J]. *Journal of Computational Mathematics*, 1999, **17**(3): 233 - 242.

[6] Pan P Q. A projective simplex algorithm using LU decomposition [J]. *Computers and Mathematics with Applications*, 2000, **39**(1): 187 - 208.

[7] Pan P Q. Practical finite pivoting rules for the simplex method [J]. *OR Spektrum*, 1990, **12**: 219 - 225.

线性规划非单调一阶段算法

潘平奇　　李　炜

(东南大学数学系, 南京 210096)

摘　要　为了获取计算的高效率,有必要修正单纯形算法的原则.本文提出了一个新的单纯形一阶段算法.与传统单纯形算法不同的是,新算法不仅不要求目标函数值单调变化,且在一阶段的迭代过程中也不必保持变量的可行性,而是采用纯组合的方法去达到可行.这样摆脱了迭代时的比值检验,减少了每次迭代的计算工组量.理论分析及数值计算结果表明新算法的前景令人鼓舞.

关键词　线性规划; 一阶段; 无比检验; 主元规则

中图分类号　O221.1