Πρακτικά 17ου Συνεδρίου Ε.Ε.Ε.Ε. «Διαχείριση Κινδύνων»
Πανεπιστήμιο Πατρών, 16-18 Ιουνίου 2005, Σελ. 677-687

On the basis inverse of the exterior point simplex algorithm

El-Said Badr[1], K. Paparrizos, N. Samaras and A. Sifaleras[2]

University of Macedonia, Department of Applied Informatics, 156 Egnatia Str., 54006 Thessaloniki, E-mail: {paparriz, samaras, sifalera}@uom.gr

Abstract

The main feature of simplex type algorithms is that they can be interpreted as a method following simplex paths that lead to the optimal vertex. Exterior Point Simplex Algorithms (EPSA) differs from classical simplex algorithm in the sense that its basic solution is not feasible. EPSA is sufficiently fast for large-scale sparse linear problems. Recall that the total computational effort of an iteration of simplex type algorithms is dominated by the determination of the basis inverse B^{-1}. This inverse does not have to be computed from scratch at any iteration. In this paper we present an analysis of two well-known updating schemes for basis inverse: (i) The Product Form of the Inverse (PFI) and (ii) A Modification of the Product Form of the Inverse (MPFI) and incorporate it with EPSA. Computational results with a subset of benchmark problems from NETLIB are also presented.

Keywords: Linear Programming, Simplex Type Algorithms, Revised Form, Basis Inverse

[1] Research supported by the Greek Scholarship Foundation, (I.K.Y).
[2] Research supported by Operational Program for Educational and Vocational Training II (EPEAEK II), and particularly the program HRAKLEITOS.

Περί της Αντιστροφής της Βάσης του Αλγορίθμου Εξωτερικών Σημείων

El-Said Badr, K. Paparrizos, N. Samaras and A. Sifaleras

University of Macedonia, Department of Applied Informatics, 156 Egnatia Str., 54006 Thessaloniki, E-mail: {paparriz, samaras, sifalera}@uom.gr

Περίληψη

Το κύριο χαρακτηριστικό των αλγορίθμων τύπου simplex είναι ότι μπορεί να ερμηνευτούν ως μια μέθοδος η οποία ακολουθεί simplex μονοπάτια, τα οποία οδηγούν στη βέλτιστη κορυφή. Οι Αλγόριθμοι Εξωτερικών Σημείων (ΑΕΣ) διαφέρουν από τον κλασικό αλγόριθμο simplex στο ότι η βασική τους λύση μπορεί να μην είναι εφικτή. Οι ΑΕΣ είναι ικανοποιητικά γρήγοροι όταν εφαρμόζονται σε μεγάλης κλίμακας αραιά γραμμικά προβλήματα. Η πιο χρονοβόρα διαδικασία σε μια επανάληψη τύπου simplex είναι ο υπολογισμός της αντίστροφης της βάσης, B^{-1}. Η αντίστροφη της βάσης δεν χρειάζεται να υπολογίζεται από την αρχή σε κάθε επανάληψη. Σε αυτήν την εργασία παρουσιάζεται μια ανάλυση δυο γνωστών τρόπων ανανέωσης της αντίστροφης: (i) Η Product Form of the Inverse (PFI) και (ii) Μια τροποποίηση της Product Form of the Inverse (MPFI) και εφαρμόζονται στον αλγόριθμο EPSA. Επίσης, παρουσιάζονται υπολογιστικά αποτελέσματα σε ένα υποσύνολο μετροπρογραμμάτων από τη συλλογή NETLIB.

Λέξεις Κλειδιά: Γραμμικός Προγραμματισμός, Αλγόριθμοι Τύπου Simplex, Αναθεωρημένη Μορφή, Αντιστροφή Βάσης

1. Introduction

Let us consider the following Linear Programming (LP) problem:

$$\begin{array}{rrcl} \min & c^T x & & \\ \text{s.t.} & Ax & = & b \\ & x & \geq & 0 \end{array} \qquad \text{(LP.1)}$$

where $A \in \Re^{m \times n}$, $c, x \in \Re^n$, $b \in \Re^m$ and T denotes transposition. Let us assume first that A is linearly independent and has full rank, rank(A) = m, $1 \leq m < n$. Given a linear problem (LP.1) a basis A_B is a square non-singular submatrix of A.

EPSA is a simplex type algorithm which was originally developed by Paparrizos (1991) for the assignment problem. Later, Paparrizos (1993) generalized his exterior point method to the general linear problem by developing a dual in nature, algorithm. Moreover, (Karagiannis et al., 2005) developed an exterior type algorithm for the minimum cost network flow problem. A common feature of almost all simplex type algorithms is that they can be interpreted as a procedure following simplex type paths that lead to the optimal solution. It has been pointed out that the geometry of EPSA reveals that this algorithm is faster than the classical simplex algorithm. An extended computational study (Paparrizos et al., 2003) between EPSA and simplex algorithm shows that the former is up to 10 times faster than the latter on medium size linear problems.

Any iteration of simplex type algorithms is relatively expensive. The total work of an iteration of simplex type algorithms is dominated by the determination of the basis inverse, $(A_B)^{-1}$. This inverse however, does not have to be computed from scratch at each iteration. Simplex type algorithms maintain a factorization of basis A_B, and update this factorization on every iteration. There are several schemes for updating basis inverse. The most well-known schemes are (i) the Product Form of the Inverse (PFI), (ii) the Elimination Form of the Inverse (EFI), and (iii) a recently developed by (Benhamadou, 2002).

In this paper we apply the schemes (i) and (iii) to EPSA and perform a computational study using a subset of NETLIB test problems. EPSA using the updating scheme (iii) is about 1.6 times faster than EPSA using the (PFI) in terms of CPU time. In order to prove his method, Benhamadou, used the canonical linear programming form. This means that all the technological constraints are inequalities. We investigate how Benhamadou's (2002) method behaves in real world problems that have a considerable number of equality constraints.

The paper is organized as follows. In section 2 we give a brief description of EPSA. In section 3 we present an analysis of the updating schemes. An illustrative example is given in section 4. Our computational results are presented in section 5 and, finally in section 6 we conclude.

2. EPSA Description

It is more convenient that the algorithm is being described using the revised form of the simplex algorithm. Consider the problem (LP.1). With the matrix A partitioned as $A = (A_B, A_N)$, and with a corresponding partitioning and ordering of x and c, (LP.1) is written as

$$\begin{array}{lrcl} \min & c_B^T x_B + c_N^T x_N & & \\ \text{s.t.} & A_B x_B + A_N x_N & = & b \\ & x_B, x_N & \geq & 0 \end{array}$$

where B and N are subsets of indices. The columns of A which belong to B are called basic and the remaining ones are called nonbasic. The basic solution corresponding to A_B is $\{x_B = (A_B)^{-1}b,\ x_N = 0\}$. A solution $x = (x_B, x_N)$ is feasible if $x \geq 0$. Otherwise it is called infeasible. A given basis is dual feasible if $s = c - A^T w \geq 0$, where $w^T = (c_B)^T(A_B)^{-1}$ are the simplex multipliers and s are the dual slack variables.

EPSA generates two paths toward the optimal solution. One path is infeasible (exterior) and the other is feasible. So EPSA doesn't need to proceed by examining one such edge after another along the polyhedron of the feasible region. Therefore, avoiding the feasible region EPSA can follow shorter paths to the optimal solution. A formal description of EPSA in the revised form is given below. Proof of correctness of the above algorithm can be found in (Paparrizos et al., 2001).

EPSA Algorithm.

Step 0. ***(Initialization).*** Start with a feasible basic partition (B, N). Compute the vectors and matrices $(A_B)^{-1}$, x_B, w, s_N. Find the sets $P = \{j \in N: s_j < 0\}$ and $Q = \{j \in N: s_j \geq 0\}$. Choose an arbitrary vector $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_{|P|}) > 0$, compute s_0 using the relation

$$s_0 = \sum_{j \in P} \lambda_j s_j$$

and the vector

$$d_B = -\sum_{j \in P} \lambda_j h_j$$

with

$$h_j = (A_B)^{-1} A_{.j}$$

Step 1. ***(Test of termination).***
i. *(Test of optimality).* If $P = \emptyset$, STOP. Problem (LP.1) is optimal.
ii. *(Choice of leaving variable).* If $d_B \geq 0$, STOP. If $s_0 = 0$, problem (LP.1) is optimal. Otherwise, choose the leaving variable $x_{B[r]} = x_k$ using the relation

$$\alpha = \frac{x_{B[r]}}{-d_{B[r]}} = \min\left\{ \frac{x_{B[i]}}{-d_{B[i]}} : d_{B[i]} < 0 \right\}$$

If $\alpha = +\infty$ problem (LP.1) is unbounded.

Step 2. ***(Choice of entering variable).*** Compute the vectors $H_{rP} = ((A_B)^{-1})_{r.}A_P$ and
$H_{rQ} = ((A_B)^{-1})_{r.}A_Q$. Also find the ratios θ_1 and θ_2 using the relations

$$\theta_1 = \frac{-s_p}{h_{rp}} = \min\left\{\frac{-s_j}{h_{rj}} : h_{rj} > 0 \text{ and } j \in P\right\}$$

and

$$\theta_2 = \frac{-s_q}{h_{rq}} = \min\left\{\frac{-s_j}{h_{rj}} : h_{rj} < 0 \text{ and } j \in Q\right\}$$

and determine indexes t_1 and t_2 such that $P(t_1) = p$ and $Q(t_2) = q$. If $\theta_1 \leq \theta_2$, set $l = p$. Otherwise, set $l = q$. The nonbasic variable x_l enters in the basis.

Step 3. ***(Pivoting).*** Set $B[r] = l$. If $\theta_1 \leq \theta_2$ set $P = P\setminus\{l\}$ and $Q = Q \cup \{k\}$. Otherwise, set $Q[t_2] = k$. Using the new partition (B, N), where N = (P, Q), update the vectors and matrices $(A_B)^{-1}$, x_B, w, s_N. Also, update the vector $\overline{d}_B$. If $l \in P$ set $\overline{d}_{B[r]} \leftarrow d_{B[r]} + \lambda_l$. Go to Step 1.

3. Analysis of the updating schemes

Matrix multiplication is a basic operation in linear algebra which has numerous applications to the theory and practice of computation. Several applications are due to the fact that matrix multiplication is a substantial part of several successful algorithms for other computational problems of operations research, such as the solution of linear equations, matrix inversion, etc. The computational time required for matrix multiplication is the dominant part of the total computational time required for the solution of such problems.

Proposition 1. The multiplication of two dense square m x m matrices involves m^3 multiplications and $m^3 - m^2$ additions. Hence, the overall complexity is $\Theta(m^3)$.

Proposition 2. Two bases $(A_B)_{[1]}$, $(A_B)_{[2]} \in \Re^m$ are called neighbouring if they differ from each other only in one column.

In each iteration, simplex type algorithms move from a basis to a neighbouring one until termination. The operations of simplex type iteration are defined in terms of the inverse of the current basis. Determining $(A_B)^{-1}$ is a substantial computational effort. Should it be computed at each iteration, then it would be computationally prohibitive.

To justify the next analysis we need the following notations:

$x \otimes y$	:	outer product of the vectors $x, y \in \Re^n$
l	:	index of the entering variable
k	:	index of the leaving variable
$A_{.l}$	:	the lth column of A
$(A_B)^{-1}_{r.}$	:	the rth row of basis inverse
$(\overline{A}_B)^{-1}_{r.}$	:	the matrix $(A_B)^{-1}$ with the rth row put to zero

3.1 Product Form of the Inverse

The key idea is the following. The current basis inverse $(\tilde{A}_B)^{-1}$ can be computed from the previous inverse $(A_B)^{-1}$ with a simple pivoting operation. Namely, we have

$$(\tilde{A}_B)^{-1} = E^{-1}(A_B)^{-1} \qquad (1)$$

where E^{-1} is the inverse of the eta-matrix. We use the following steps to compute $(\tilde{A}_B)^{-1}$.

Compute the pivot column $h_l=(A_B)^{-1}A_{.l}$
Compute the column vector

$$v = \begin{bmatrix} -\frac{h_{1l}}{h_{rl}} & \cdots & \frac{1}{h_{rl}} & \cdots & -\frac{h_{ml}}{h_{rl}} \end{bmatrix}^T$$

Replace the *r*th column of an identity matrix with the column vector v. This matrix is the inverse of the Eta-matrix.

Compute the new basis inverse using relation (1).

The inverse of the Eta-matrix has the following form

$$E^{-1} = I - \frac{1}{h_{rl}}(h_l - e_l)e_l^T = \begin{bmatrix} 1 & & -h_{1l}/h_{rl} & & \\ & \ddots & \vdots & & \\ & & 1/h_{rl} & & \\ & & \vdots & \ddots & \\ & & -h_{ml}/h_{rl} & & 1 \end{bmatrix}$$

If the current basis inverse is computed using regular multiplication in relation (1), then the complexity of the (PFI) is $\Theta(m^3)$. It can be seen that the matrix E^{-1} has a special structure. Specifically, E^{-1} is sparse and on every iteration it has $2m - 1$ nonzero elements. In order to count the total number of multiplications and additions needed for this updating scheme, we make the following assumptions:

(i) We count only the multiplication between two nonzero numbers, $\alpha\beta$, where ($\alpha\in\Re^*$-{1} and $\beta\in\Re^*$) or ($\alpha\in\Re^*$ and $\beta\in\Re^*$-{1}) and

(ii) We count the addition between two nonzero numbers.

The above updating scheme of the inverse is simplified in Fig. 2.

$$\begin{array}{cc} E^{-1} & (A_B)^{-1} \\ \begin{bmatrix} 1 & & -h_{1l}/h_{rl} & & \\ & \ddots & \vdots & & \\ & & 1/h_{rl} & & \\ & & \vdots & \ddots & \\ & & -h_{ml}/h_{rl} & & 1 \end{bmatrix} & \begin{bmatrix} x & x & x \\ \vdots\ddots & \vdots\ddots & \vdots \\ x & x & x \\ \vdots\ddots & \vdots\ddots & \vdots \\ x & x & x \end{bmatrix} \end{array}$$

Figure 1: *Updating Scheme of the PFI*

In the *m*th iteration the basis inverse has m^2 nonzero elements. The *r*th row of E^{-1} has only one nonzero element. All the other rows (m-1) have 2 nonzero elements. In order to compute the product in Figure 1, according to the above assumptions, we need

(m – 1) rows x m multiplications + 1 x m multiplications = m^2 multiplications

and

(m – 1) rows x m additions = m^2 – m additions

The total cost of the above method is $2m^2$-m operations (multiplications and additions).

3.2 A Modification of Product Form of the Inverse

This updating scheme presented by (Benhamadou, 2002). The key idea is the following. The current basis inverse $(\tilde{A}_B)^{-1}$ can be computed from the previous inverse $(A_B)^{-1}$ with a simple outer product of two vectors and one matrix addition. Namely we have

$$(\tilde{A}_B)^{-1} = (\bar{A}_B)^{-1}_{r.} + v \otimes (A_B)^{-1}_{r.} \tag{2}$$

A formal description of this method follows.

Compute the pivot column $h_l=(A_B)^{-1}A_{.l}$

Compute the vector

$$v = \begin{bmatrix} -\frac{h_{1l}}{h_{rl}} & \cdots & \frac{1}{h_{rl}} & \cdots & -\frac{h_{ml}}{h_{rl}} \end{bmatrix}^T$$

Compute the outer product $v \otimes (A_B)^{-1}_{r.}$

Set the *r*th row of $(A_B)^{-1}$ equal to zero. Save the result in $(\bar{A}_B)^{-1}_{r.}$

Compute the new basis inverse using relation (2).

The above updating scheme of the inverse is simplified in Fig. 2.

$$(A_B)^{-1}_{r.} : \begin{bmatrix} b_{r1} & \cdots & b_{rr} & \cdots & b_{rm} \end{bmatrix}$$

$$(\tilde{A}_B)^{-1} = \begin{bmatrix} b_{11} & \cdots & b_{1m} \\ \cdots & \cdots & \cdots \\ 0 & 0 & 0 \\ \cdots & \cdots & \cdots \\ b_{m1} & \cdots & b_{mm} \end{bmatrix} + \begin{bmatrix} -\frac{h_{1l}}{h_{rl}} \\ \vdots \\ \frac{1}{h_{rl}} \\ \vdots \\ -\frac{h_{ml}}{h_{rl}} \end{bmatrix} \qquad [mxm]$$

Figure 2: *A modification of the (PFI)*

The outer product requires m^2 multiplications and the addition of two matrices requires m^2 additions. The total cost of the above method is $2m^2$ operations (multiplications and additions). Hence, the complexity is $\Theta(m^2)$.

4. An illustrative example

An illustrative example of the previously mentioned updating schemes in section 3, will be given. Assume there is the following linear problem,

$$\begin{array}{rrrrrrrrl} \min & x_1 & + & x_2 & - & 4x_3 & & & \\ \text{s.t.} & x_1 & + & x_2 & + & x_3 & \leq & 9 & \\ & x_1 & + & x_2 & - & x_3 & \leq & 2 & \\ & -x_1 & + & x_2 & + & x_3 & \leq & 4 & \\ & x_1 & & x_2 & & x_3 & \geq & 0 & \end{array} \qquad \text{(LP.2)}$$

First we introduce the slack variables x_4, x_5 and x_6. In matrix notation the problem (LP.2) is written,

$$\begin{array}{ll} \min c^T x & \\ \text{s.t} & Ax = b \\ & x \geq 0 \end{array}$$

where

$$c^T = \begin{bmatrix} 1 & 1 & -4 & 0 & 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 1 & 2 & 1 & 0 & 0 \\ 1 & 1 & -1 & 0 & 1 & 0 \\ -1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 9 \\ 2 \\ 4 \end{bmatrix}$$

The first feasible basic partition is B=[4 5 6] and N=[1 2 3]. Hence,

$$(A_B)^{-1} = A_B = [A_{.4} \; A_{.5} \; A_{.6}]$$

According to Step 1 and Step 2 of the EPSA x_6 and x_l is the leaving and entering variable respectively. Leaving variable found in the r = 3 row. The pivot column is $h_l = [1 \;\; 1 \;\; -1]^T$.

4.1 Update Basis Inverse Using Eta-matrix

Compute the vector

$$v = \begin{bmatrix} -\frac{h_{1l}}{h_{rl}} & L & \frac{1}{h_{rl}} & L & -\frac{h_{ml}}{h_{rl}} \end{bmatrix}^T = \begin{bmatrix} 1 & 1 & -1 \end{bmatrix}^T$$

Put the vector v in the 3rd column of an identity matrix $E^{-1} = I_{3x3}$. The inverse of the Eta-matrix has the form

$$E^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & -1 \end{bmatrix}$$

Using relation (1) the new basis inverse is

$$(\tilde{A}_B)^{-1} = E^{-1}(A_B)^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & -1 \end{bmatrix}$$

4.2 Update Basis Inverse Using Outer Product & Matrix Addition

Compute the vector v=[1 1 -1]^T. The outer product $v\otimes(A_B)^{-1}_{r.}$ gives

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & -1 \end{bmatrix}$$

Set the 3rd row of $(A_B)^{-1}$ equal to zero. Save the result in $(\overline{A}_B)^{-1}_{r}$. Compute the new basis inverse using relation (2). The new basis inverse is

$$(\tilde{A}_B)^{-1} = (\overline{A}_B)^{-1}_{r.} + v\otimes(A_B)^{-1}_{r.} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & -1 \end{bmatrix}$$

5. Computational Study

The updating schemes described in section 3 have been experimentally implemented. In this section we describe our numerical experiments and present computational results, which demonstrate updating schemes efficiency on a set of test problems obtained by NETLIB (www.netlib.org/lp). Compiled using the MATLAB v7.01 SP2 with default options, all runs were carried out under MS Windows XP Pro system, having P4 3.06 GHz and 512 Mb RAM.

Table 1 presents the statistics for 16 benchmarks by alphabetical order. The first column contains the name of the problem (1), while the other two columns (2)-(3) contain the size of problems. The number of rows doesn't include the objective function. In column (4) the number of nonzero elements of A and c is displayed. Finally, the density percentage contained in column (5).

Table 1: *Test set statistics*

Name	**m**	**n**	**nnz(A)+nnz(c)**	**Density**
(1)	**(2)**	**(3)**	**(4)**	**(5)**
adlittle	56	97	465	8,56%
bandm	305	472	2659	1,85%
degen2	444	534	4449	1,88%
e226	223	282	2767	4,40%
israel	174	142	2358	9,54%
lotfi	153	308	1086	2,30%
sc105	105	103	281	2,60%

sc205	205	203	552	1,33%
scagr7	129	140	553	3,06%
sctap1	300	480	2052	1,42%
sctap3	1480	2480	10734	0,29%
share1b	118	225	1182	4,45%
share2b	96	79	730	9,62%
ship04l	402	2118	8450	0,99%
ship08s	778	2387	9501	0,51%
stocfor1	117	111	474	3,65%

Table 2 collects information on the performance of the mentioned above updating schemes. Columns of Table 2 contains name of the problem, problem size, the number of iterations, niter and the CPU time, cpu in seconds for the updating schemes Product Form of the Inverse (PFI) and Modified Product Form of the Inverse (MPFI).

Table 2: *Computational Results*

			PFI		**MPFI**		**speed-up**
name	**m**	**n**	**niter**	**cpu**	**niter**	**cpu**	
adlittle	56	97	140	0,33	140	0,33	1,00
bandm	305	472	485	5,86	485	3,00	1,95
degen2	444	534	1869	54,42	1436	17,87	3,05
e226	223	282	693	3,04	693	1,92	1,58
israel	174	142	306	0,89	306	0,65	1,37
lotfi	153	308	565	1,73	565	1,01	1,71
sc105	105	103	111	0,14	111	0,08	1,75
sc205	205	203	260	0,97	260	0,71	1,37
scagr7	129	140	141	0,45	141	0,25	1,80
sctap1	300	450	676	2,56	704	6,31	0,41
sctap3	1480	2480	2421	42,02	2421	37,56	1,12
share1b	118	225	458	1,40	458	0,56	2,50
share2b	96	79	356	1,08	356	0,48	2,25
ship04l	402	2118	1282	6,05	1282	13,78	0,44
ship08s	778	2387	764	3,46	764	7,71	0,45
stocfor1	117	111	138	0,31	138	0,11	2,82
Mean value							**1,60**

From the above results we make the following observations: (i) The modified Product Form of the Inverse (MPFI) is in most problems faster than PFI (in 12 out of 16) and (ii) Using MPFI the total problem set execution time was 1.60 times faster than PFI.

6. Conclusions

In this paper we have presented an analysis of two well-known updating schemes for the basis inverse. The computational study of section 5 indicates that the modification of the Product Form of the Inverse is 1.60 times faster than the Product Form of the Inverse in most problems. Theoretically, both methods require the same number of multiplications and additions.

References

Benhamadou, M. (2002). On the simplex algorithm ‘revised form’, *Advances in Engineering Software*, 33, 769-777.

Karagiannis P., Paparrizos K., Samaras N. and Sifaleras A. (2005) A new simplex type algorithm for the minimum cost network flow problem, *presented at the 7th Balkan Conference on Operational Research (BACOR ΄05), 25-28 May, Constanta, Romania.*

Paparrizos, K. (1991). An infeasible exterior point simplex algorithm for assignment problems, *Mathematical Programming*, 51, 45-54.

Paparrizos, K. (1993). An exterior point simplex algorithm for general linear problems, *Annals of Operation Research*, 47, 497-508.

Paparrizos, K., Samaras, N. and Tsiplidis, K. (2001). Pivoting algorithm for (LP) generating two paths, in: M. P. Pardalos, A. C. Fluodas (Eds.), *Encyclopedia of Optimization*, 4, 302-306, Kluwer Academic Publishers.

Paparrizos, K. Samaras, N. and Stephanides, G. (2003). An efficient simplex type algorithm for sparse and dense linear programs, *European Journal of Operational Research*, 148, 323-334.