Bull. Math. Soc. Sci. Math. Roumanie
Tome 53(101) No. 3, 2010, 225–230

Version of block Lanczos-type algorithm for solving sparse linear systems

by
M.A. Cherepniov

Dedicated to the memory of Laurenţiu Panaitopol (1940-2008) on the occasion of his 70th anniversary

Abstract

A new version of block algorithm for solving sparse systems of linear equations over $GF(2)$ is described in the talk. The algorithm involves block Pade approximations. The running time of the algorithm can be made less then running time of the known Wiedemann-Coppersmith algorithm by selection larger value of block factor with fixed RAM volume.

Key Words: Linear sparse systems over GF(2), Padé approximations, Lanczos-type algorithm, Integer factorization.
2010 Mathematics Subject Classification: Primary 11Y05; Secondary 11-02.

1 Introduction

Solving large sparse linear systems is usually utilized in modern algorithms of integer factorization. Frequently Wiedemann-Coppersmith algorithm [5] is used for this purpose because of the possibility utilization of different calculation sites without commutation links between them. Algorithm divides into three parts:

- Calculation of coefficients of series.
- Calculation of the approximation for the series.
- Calculation of the solution with the help of coefficients of this approximation.

Time, needed for the steps 1 and 3, much more than for the step 2. To the other hand step 2 demands utilization of the cluster with big RAM volume. In

this paper we propose some knew version of the algorithm, described in [1], that requires less RAM memory volume for the step 2 than Wiedemann-Coppersmith algorithm with Thomé [7] proposal. If we select block factor bigger to stay RAM volume on the original level we obtain better upper bounds for running time of the steps 1 and 3.

2 Definitions and utilized results

Let $\mathbb{F} = GF(2)$, and d denote upper bound of number of nonzero elements in every row of linear system matrix, s - block factor. So ns - number of vectors in X and in right hand side of linear system:

$$DX = B, D \in \mathbb{F}^{M \times N}, M \leq N; B, X \in \mathbb{F}^{N \times ns}. \tag{1}$$

As in [2] we denote c ratio between runtime of one machine word's passing between nods and runtime of one arithmetic operation with machine words. As was described there running time for one double matrix multiplication bounded with value $4Nc$ with number of required such multiplications $2\frac{N}{ns}$ for the step 1 , and $\frac{N}{ns}$ for the step 3 (Wiedemann-Coppersmith algorithm have no more than $2\frac{N}{ns}$ single matrix multiplications in the step 3). Since in algorithms [1] and [4] we use double multiplication with matrixes D and D^T (see [2])- optimal number of calculation nods, that makes time for arithmetic equal to time for sending, will be $s\frac{d}{c}$.

Time for only arithmetic operations in the step 2 of Wiedemann-Coppersmith-Thomé algorithm bounded in [7] with value

$$O(Nns(ns + log_2 N)log_2 Nlog_2 log_2 N)$$

that for parallel implementation on the cluster with required RAM volume about 1 TB, when $N \approx 1,9 \cdot 10^8, s = 8, n = 64$, was approximately 17,3 hours.

3 The algorithm

We construct our algorithm according to [1]. On the step 1 we calculate approximately $2\frac{N}{ns}$ coefficients of the series

$$\alpha = \sum_{i=0}^{\infty} \alpha_i \lambda^{-i}, \alpha_i = B^T A^i B, A = D^T D. \tag{2}$$

On the step 2 we calculate approximations of the series (2). At the beginning for some t_0, which chouse we describe below, for every $t \in \{1, \ldots, t_0\}$ we calculate $Q^{(t)}(\lambda)$ such that

$$\alpha(\lambda)Q^{(t)}(\lambda) - P^{(t)}(\lambda) = \sum_{i=t+1}^{\infty} \rho_i^{(t)} \lambda^{-i}, degQ^{(t)}(\lambda) \leq t, degP^{(t)}(\lambda) \leq t,$$

with the help of the matrix

$$\begin{pmatrix} \alpha_{t+1} & \dots & \alpha_2 & \alpha_1 \\ \alpha_{t+2} & \dots & \alpha_3 & \alpha_2 \\ \dots & \dots & \dots & \dots \\ \alpha_{2t} & \dots & \alpha_{t+1} & \alpha_t \\ \lambda^t Q^{(0)} & \dots & \lambda Q^{(0)} & Q^{(0)} \\ I_n & \dots & O_n & O_n \\ \dots & \dots & \dots & \dots \\ O_n & \dots & I_n & O_n \\ O_n & \dots & O_n & I_n \end{pmatrix} \begin{pmatrix} \upsilon_t \\ \cdot \\ \cdot \\ \cdot \\ \upsilon_1 \\ \upsilon_0 \end{pmatrix} = \begin{pmatrix} O_n \\ \cdot \\ \cdot \\ O_n \\ Q^{(t)} \\ Q^{(t)}_{t+1} \\ \cdot \\ \cdot \\ Q^{(t)}_0 \end{pmatrix}. \tag{3}$$

The upper part of the left matrix has a size of t by $t+1$ blocks from $\mathbb{F}^{ns\times ns}$. Therefore, corresponding homogeneous system has a full-rank solution $\bar{v} \in \mathbb{F}^{(t+1)ns\times ns}$, and the resulting matrix approximation $Q^{(t)}(\lambda)$ has a full rank over the field $\mathbb{F}$.

For every $Q^{(i)}(\lambda)$ we obtain $\tilde{Q}^{(i)}(\lambda) = \sum_{j=0}^{i} \tilde{Q}^{(i)}_j \lambda^j, \tilde{Q}^{(i)}_j \in \mathbb{F}^{ns\times ns}$, where $\tilde{Q}^{(i)}_i$- nonsingular and

$$\alpha(\lambda)\tilde{Q}^{(t)}(\lambda) - \tilde{P}^{(t)}(\lambda) = \sum_{i=t+1-\zeta(t)}^{\infty} \tilde{\rho}^{(t)}_i \lambda^{-i}, deg\tilde{Q}^{(t)}(\lambda) \le t, deg\tilde{P}^{(t)}(\lambda) \le t,$$

for not very big $\zeta(t)$ as was described in [1]. Then according to the same work we obtain $Q^{(t)}(\lambda), \tilde{Q}^{(t)}(\lambda) t > t_0$ with the help of $Q^{(i)}(\lambda), \tilde{Q}^{(i)}(\lambda), i \in \{t-1, \dots, t-\theta(t)\}$. An upper bound for $\theta(t)$ that makes success finish of our algorithm with probability not less than $0,99$, obtained in [3]

$$\theta(t) \le 2,7 \cdot log_2 \frac{N}{ns} + 19. \tag{4}$$

So t_0 we must take near this bound.

If we denote

$$U_i = \tilde{Q}^{(i)}(A, B) = \sum_{j=0}^{i} A^j B \tilde{Q}^{(i)}_j \in \mathbb{F}^{N\times ns}. \tag{5}$$

we obtain according to [1]

$$U_i^T A U_t = \sum_j \tilde{Q}^{(i)T}_j \tilde{\rho}^{(t)}_{j+1}. \tag{6}$$

So, when $i+1 < t+1-\zeta(t)$, this equals to zero. As we can obtain from [3], the mean value of $\zeta(t)$ is no more than 2. So we can sequentially orthogonalize blocks U_i and form blocks $W_i, i = 1, \dots, m-1$, from their columns, with transfer some of them to the next steps. Then

$$X = \sum_{i=0}^{m-1} W_i (W_i^T A W_i)^{-1} W_i^T B, \tag{7}$$

that gives solution of the system like in the algorithm of P.Montgomery [4]. Scalar products $W_i^T A W_i$ we will obtain from (6). We have (see [1])

$$U_i = A U_{i-1} C_{i-1} + U_{i-1} c_{i-1} + \cdots + U_{i-\theta(i)+1} c_{i-\theta(i)+1} + V_{i-\theta(i)} c_{i-\theta(i)},$$

for some $C_i, c_i \in \mathbb{F}^{ns \times ns}$, where $V_k = Q^{(k)}(A, B)$, and $W_i = \sum_{k=0}^{\Delta(i)} U_{i-k}\delta_k, \delta_k \in \mathbb{F}^{ns \times ns}, \Delta(i) \approx \zeta(i), B = U_0$. So elements $W_i^T B$ may be obtain by the recurrence

$$U_i^T B = C_{i-1}^T U_{i-1}^T A B + \sum_{k=1}^{\theta(i)-1} c_{i-k}^T U_{i-k}^T B + c_{i-\theta(i)}^T V_{i-\theta(i)}^T B.$$

On the step 2 of our algorithm we will replace vectors U_i from (5) with the set of their coefficients $\tilde{Q}_j^{(i)}$ in the bases $A^j B$, where $j = 0, 1, \ldots, i$. We can done analogous replacement with vectors W_i and X. If we denote corresponding coefficients of X as X_i, on the step 3 of our algorithm we obtain

$$X = \sum_{j=0}^{j_0} A^j B X_j, j_0 \approx \frac{N}{ns},$$

from $A^j B$, that we must calculate for the second time. Note, that in corresponding place in Wiedemann-Coppersmith algorithm we have $j_0 \leq 2\frac{N}{ns}$.

Estimation time for the 1 and the step 3 of our algorithm is approximately the same to Wiedemann-Coppersmith algorithm. These steps can be made on s non linked clusters, that have RAM volume suffice to store matrix A. The step 2 demand much more RAM volume. In method, described in [6], it was approximately 1TB for $N \approx 1,9 \cdot 10^8, s = 8$. In our algorithm from (4) we have upper bound

$$\frac{N}{ns}(ns)^2(2,7 \cdot log_2 \frac{N}{ns} + 21) = Nns(2,7 \cdot log_2 \frac{N}{ns} + 21),$$

and for such N, s this upper bound approximately equals to 500 GB. Running time of the step 2 of our algorithm may be approximately bounded with the value of number of arithmetic operations by formula

$$O(\sum_{i=1}^{\frac{N}{ns}} i \cdot ns \cdot ns \cdot s) = O(N^2 s). \tag{8}$$

This calculations have no recursive components and no matrix polynomial multiplications. They are linear combinations of blocks from $\mathbb{F}^{ins \times ns}$ with the

coefficients in $\mathbb{F}^{ns\times ns}$, multiplying of blocks from $\mathbb{F}^{ins\times ns}$ and some other relatively fast operations like multiplications and additions of matrixes from $\mathbb{F}^{ns\times ns}$. In [3] was calculated that value of θ is approximately $3,67$. So constant in formula (8) is not big (approximately 5) and with the help of not very hard parallelization with not heavy commutation, we can effectively divide bound (8) to the number of calculation nods, that have common RAM memory volume as was mentioned above. As was described in [2], optimal value of calculation nodes for 1 and step 3 is $O(s^2)$, when $s = O(N^{\frac{1}{3}})$. Utilization of such quantity of calculation nodes in the step 2 makes its running time asymptotically equal to the running time of steps 1 and 3.

4 Conclusions

Proposed version is similar to original Wiedemann-Coppersmith algorithm, described in [5]. Same techniques gives in [1] an algorithm that is similar to original Montgomery algorithm [4]. Because of the simplicity of step 2 of described version it may be simply reconstructed [8] to obtain smaller runtime and to avoid demands of big RAM volume and big number of cores in step 2 when using additional nodes in the steps 1 and 3 . So we can avoid utilization of strong cluster for step 2 .

References

[1] Cherepniov M.A. Block Lanczos-type algorithm for solving sparse linear systems. Diskr. Math. (in Russian), V.20, n.1, 2008, p.145-150.

[2] Cherepniov M.A. Some estimations of performance of parallel algorithms for solving large systems over GF(2). IEEE Tran. Proc. of the conference PARCa 2010 (Tambov) (to appear).

[3] Astakhov V. Estimates of the rutime and memory requirements of the new algorithm of solving large sparse linear systems over field with two elements. IEEE Tran. Proc. of the conference PARCa 2010 (Tambov) (to appear).

[4] Montgomery P.L. A Block Lanczos Algorithm for Finding Dependencies over GF(2). EUROCRYPT'95, LNCS Vol. 921 (1995), p.106-120

[5] Coppersmith D. Solving Homogeneous Linear Equations Over GF(2) via Block Wiedemann Algorithm. Mathematics of Computation, Vol. 62, No. 205, (Jan. 1994), p.333-350.

[6] Kleinjung T., Aoki K., Franke J., Lenstra A.K., Thomé E., Bos J.W., Gaudry P., Kruppa A., Montgomery P.L., Osvik D.A., Riele H., Timofeev A., Zimmermann P. Factorization of a 768-bit RSA modulus. version 1.0, January 7, 2010. http://eprint.iacr.org/2010/006.pdf

[7] THOMÉ E, Subquadratic computation of vector generating polynomials and improvement of the block Wiedemann algorithm. Journal of Symbolic Computation. (to appear)

[8] Optimized algorithm for solving sparse linear systems over $GF(2)$ with the help of Pade approximations. Nauchno-tekhnicheskiy sbornik ACRF N.572 (2009.)(in Russian), p.290-315.

Received: 20 May 2010.

Moscow State University, Russia
E-mail: cherepniov@gmail.com