Some Performance Studies in Exact Linear Algebra

George Havas[1] and Clemens Wagner[2]

[1] Centre for Discrete Mathematics and Computing
Department of Computer Science and Electrical Engineering
The University of Queensland, Queensland 4072, Australia
havas@csee.uq.edu.au
http://www.it.uq.edu.au/~havas/

[2] Fachgruppe Praktische Informatik
Fachbereich Elektrotechnik und Informatik
Universität-GHS Siegen
D-57078 Siegen, Germany
wagner@informatik.uni-siegen.de
http://pi.informatik.uni-siegen.de/clemens/

Abstract. We consider parallel algorithms for computing the Hermite normal form of matrices over Euclidean rings. We use standard types of reduction methods which are the basis of many algorithms for determining canonical forms of matrices over various computational domains. Our implementations take advantage of well-performing sequential code and give very good performance.

1 Introduction

Algorithms for exact linear algebra have been much studied. Many different strategies for calculation of canonical forms of matrices have been proposed. A comprehensive bibliography and a number of earlier methods for integer matrices are examined in [8], including references to various polynomial time algorithms. Some parallel and some more recent methods are described in [16,9,6,22,20,11,12,21]. Reduction methods for general Euclidean rings are studied in detail in [23].

We concentrate on algorithms which use reduction as their underlying principle. In spite of the fact that the worst case performance of reduction methods can be exponentially bad (see [4] and [13]), such techniques provide the basis for many sequential implementations. We do not consider modular methods in this paper. Rather we study other recent implementations which focus on finding well-performing algorithms and good heuristics for reduction methods. We start by outlining the mathematical background. We then show how to extend sequential algorithms to a parallel environment. We finish by presenting some sample performance figures and outlining recommendations for choosing appropriate parallel algorithms.

2 Mathematical background

A commutative ring $\mathbf{R}$ with identity 1 is Euclidean if there is a value function $\varphi : \mathbf{R}^* \to \mathbb{N}_0$ (where $\mathbf{R}^* = \mathbf{R} \setminus \{0\}$ and $\mathbb{N}_0$ is the set of nonnegative integers) such that the following properties hold for $a \in \mathbf{R}$ and $b \in \mathbf{R}^*$.

1. For $a \neq 0$, $\varphi(ab) \geq \varphi(a)$.
2. There exist $q, r \in \mathbf{R}$ with $a = qb + r$, such that either $r = 0$ or $\varphi(b) > \varphi(r)$.

Paradigm examples of Euclidean rings are $\mathbb{Z}$ (the ring of integers, with absolute value as value function) and $\mathbb{F}[x]$ (the ring of univariate polynomials with coefficients in a field $\mathbb{F}$, with degree as value function).

An element $a \in \mathbf{R}$ is a unit if it has an inverse $a^{-1} \in \mathbf{R}$ such that $aa^{-1} = a^{-1}a = 1$. The set $U(\mathbf{R})$ of all units of $\mathbf{R}$ is a multiplicative group. Elements $a, b \in \mathbf{R}$ are associates if there exists a unit $c \in \mathbf{R}$ such that $a = bc$ and we write $a \sim b$. Association is an equivalence relation with equivalence classes $[a] := \{b \in \mathbf{R} \mid a \sim b\}$. A subset $R \subseteq \mathbf{R}$ is a representative set for $\mathbf{R}$ if: $\{[a] \mid a \in R\} = \mathbf{R}/\sim$; and $\forall a, b \in R$, $a \not\sim b$.

For a Euclidean ring $\mathcal{R}$ a function $\rho : \mathbf{R} \times \mathbf{R}^* \to \mathbf{R}$ is called a residue class system if for all $a, a' \in \mathbf{R}$ and $b \in \mathbf{R}^*$

$$\begin{aligned} \rho(a,b) &\in \{a - qb \mid q \in \mathbf{R}\}, \\ \varphi(\rho(a,b)) &< \varphi(b), \qquad \text{and} \\ \rho(a,b) = \rho(a',b) &\iff \exists t \in \mathbf{R} : a = a' + tb. \end{aligned}$$

Let $M \subset \mathbf{R}$ with $M \neq \{0\}$ be a finite, nonempty subset of $\mathbf{R}$. The greatest common divisor of M ($\gcd(M)$) is the equivalence class $[g]$ such that: $\forall a \in [g]$, $a \mid M$; and $\forall b \in \mathbf{R}$ with $b \mid M$, $b \mid [g]$. If we have a representative set R for $\mathbf{R}$ and $d \in \gcd(M) \cap R$ then d is uniquely determined. Further background material on Euclidean rings is given in [18,7,5].

Matrices A and B with entries in a Euclidean ring $\mathcal{R}$ are column equivalent if there exists a unimodular matrix V such that $A = BV$. Matrix V corresponds to a sequence of elementary column operations: multiplying a column by a unit of $\mathcal{R}$; adding any multiple by a ring element of one column to another; or interchanging two columns.

For any matrix B over a Euclidean ring $\mathcal{R}$ with representative system R there exists a unique lower triangular matrix H which is column equivalent to B and which satisfies the following conditions. Let r be the rank of B.

1. The first r columns of H are nonzero and the remaining columns are zero.
2. For $1 \leq j \leq r$ let $H_{i_j,j}$ be the first nonzero entry in column j. Then $i_1 < i_2 < \ldots < i_r$.
3. $H_{i_j,j} \in R \setminus \{0\}$, for $1 \leq j \leq r$.
4. For $1 \leq k < j \leq r$, $H_{i_j,k} = \rho(H_{i_j,k}, H_{i_j,j})$.

This matrix is called the column Hermite normal form (HNF) of the given matrix B and has many important applications. As already mentioned,

there are many algorithms based on reduction methods for computing the HNF. Descriptions of such methods for canonical form computation in Euclidean rings (sometimes specialized to the integers) in the literature include [18,7,5,19].

3 Sequential algorithms

Deterministic polynomial-time HNF algorithms (non-modular) include those of Kannan and Bachem [14], Chou and Collins [3], and Havas, Majewski and Matthews [12] for the integers; of Kannan [15] for $\mathbb{Q}[x]$; and of Wagner [23] for $\mathbb{F}_q[x]$. Heuristic algorithms (often faster and/or "better") include those of Havas and Majewski [9] for the integers and of Wagner [23] for $\mathbb{F}_q[x]$. All in all, even the sequential algorithm situation is a quite complicated story, which is addressed in much more detail in [23]. We do not go into this further here, but rather build parallel algorithms based upon effective sequential ones.

4 Parallel implementations

The problem we consider is: Given $A \in \mathrm{Mat}_{m\times n}(\mathbf{R})$, compute in parallel H the HNF of A together with a unimodular matrix V such that $H = AV$. To unify the computation we actually compute the HNF of a working matrix $W = \begin{pmatrix} A \\ I_n \end{pmatrix}$. Let K be the Hermite normal form of W. The first m rows of K are the Hermite normal form of A and the last n rows of K give a unimodular transformation matrix V.

A *parallel computer* $\mathcal{P} := \{\pi_0, \ldots, \pi_{N-1}\}$ consists of N processors with distinct memory and a communication network. Let

$$\ell(l,\tau) := \left\lfloor \frac{l}{N} \right\rfloor + \begin{cases} 1 & \text{if } \tau < (l \bmod N) \\ 0 & \text{otherwise} \end{cases}$$

for each $0 \le \tau \le N-1$.

We use the matrix distribution model from [24]. Each processor π_τ on the parallel computer stores part of the working matrix W corresponding to a $(\ell(m,\tau)+1) \times n$ matrix $A^{(\tau)}$ and a $\ell(n,\tau) \times n$ matrix $V^{(\tau)}$ which are submatrices of the working versions of A and the multiplier $V \in \mathrm{GL}_n(\mathbb{F}[x])$, respectively. An extra row, row $\ell(m,\tau)+1$ of $A^{(\tau)}$ is used to control the computations. We call this the *computation row.*

The matrix is distributed rowwise to processors, but in stripes **not** in blocks. For each operational row we also do a broadcast. Thus, we distribute the input matrix A by storing the ith row of A in row i' of matrix $W^{(\tau)}$ on processor π_τ where $\tau := (i-1) \bmod N$ and $i' := \lfloor (i-1)/N \rfloor + 1$ for $1 \le i \le m$. The parallel, hybrid HNF algorithm PARALLEL-HNF is given by pseudocode in Figure 1. It uses the standard DIV operator for the appropriate Euclidean ring and calls two subprocedures: COMPUTE-GCD and PARALLEL-ROD.

PARALLEL-HNF$(W^{(\tau)}, \alpha, \beta)$
input τ: processor index
$W^{(\tau)}$: π_τ-part of a full column-rank, distributed matrix W
α, β: non-negative integers
$k \leftarrow 1$
$(i_1, \ldots, i_n) \leftarrow \mathbf{0}$
while $k \leq n$ **do**
  $r \leftarrow 0$
  $i \leftarrow 1$
  $s \leftarrow \min\{k + \alpha - 1, n\}$
  $f \leftarrow k$
  while $r < s$ **do**
    $f \leftarrow \max\{r + 2, f\}$
    $\mu \leftarrow (i - 1) \bmod N$
    if $\mu = \tau$ **then**
      $j \leftarrow \lfloor (i-1)/N \rfloor + 1$
      broadcast $W^{(\tau)}_{j,r+1}, W^{(\tau)}_{j,f}, \ldots, W^{(\tau)}_{j,s}$ **to all other**
    else
      $j \leftarrow \ell(m, \tau) + 1$ ▷ *Index of computation row*
      receive $W^{(\tau)}_{j,r+1}, W^{(\tau)}_{j,f}, \ldots, W^{(\tau)}_{j,s}$ **from** μ
    if $i = i_{r+1}$ **then**
      for $l \leftarrow f$ **to** s **do**
        $q \leftarrow \text{DIV}(W^{(\tau)}_{j,l}, W^{(\tau)}_{j,r+1})$
        $W^{(\tau)}_{*,l} \leftarrow W^{(\tau)}_{*,l} - qW^{(\tau)}_{*,r+1}$
    $W^{(\tau)} \leftarrow$ COMPUTE-GCD$(W^{(\tau)}, j, r+1, f, s)$
    if $W^{(\tau)}_{j,r+1} \neq 0$ **then**
      $(i_1, \ldots, i_n) \leftarrow (i_1, \ldots i_r, i, i_{r+1}, \ldots, i_{n-1})$
      $r \leftarrow r + 1$
  $W^{(\tau)} \leftarrow$ PARALLEL-ROD$(W^{(\tau)}, r, \beta, i_1, \ldots, i_r)$
  $k \leftarrow k + \alpha$
return $W^{(\tau)}$

Fig. 1. Parallel hybrid Hermite normal form algorithm

A call COMPUTE-GCD(B, i, j_0, j_1, j_2) takes a matrix B with n columns, an integer $i \geq 1$ and $1 \leq j_0 < j_1 \leq j_2 \leq n$ as input, where i is a valid row index of B. It produces a right equivalent matrix B' as output where

$$B'_{i,j_0} = \gcd(B_{i,j_0}, B_{i,j_1}, \ldots, B_{i,j_2}) \text{ and } B'_{i,j} = 0$$

for $j_1 \leq j \leq j_2$. This algorithm computes B' from B by *unimodular column operations* (*swapping* of two columns, *multiplying* a column with an unit, and *adding a multiple of one column to another column*). There are various different methods to obtain B' from B (e. g. [1,2,10,23]). By using gcd algorithms whose execution depends only on the entries in the operational row we need no additional communication for this purpose.

The function PARALLEL-ROD reduces the off-diagonal entries. It is a hybrid algorithm which is controlled by parameter $\beta \in \mathbb{N}_0$. For $\beta + 1$ greater than or equal to the rank of the input matrix it is a parallel variant of the standard reduction algorithm. For $\beta = 1$ this algorithm is a parallel version of Chou-Collins' reduction method. If we choose β equal to zero the algorithm does not change the input matrix. A more detailed description of Parallel-ROD algorithm can be found in [24].

Theorem 1. *Let $A \in \mathrm{Mat}_{m\times n}(\mathbf{R})$ with rank r be in echelon form. Let $1 \leq \beta \leq r$ and $q := \lfloor \frac{r-2}{\beta} \rfloor$. Then the* PARALLEL-ROD *algorithm uses $\frac{q(q+1)}{2}\beta + r$ broadcasts to transfer at most $\frac{r^2+r+q^2\beta+q\beta}{2}$ ring elements.*

A proof is given in [24].

The PARALLEL-HNF algorithm divides the input matrix into vertical blocks of width α. For $k := \alpha, 2\alpha, \ldots, (p-1)\alpha, n$ (where $p := \lceil \frac{n}{\alpha} \rceil$) the HNF of the leading k-column submatrix is computed. This is shown in Figure 2.

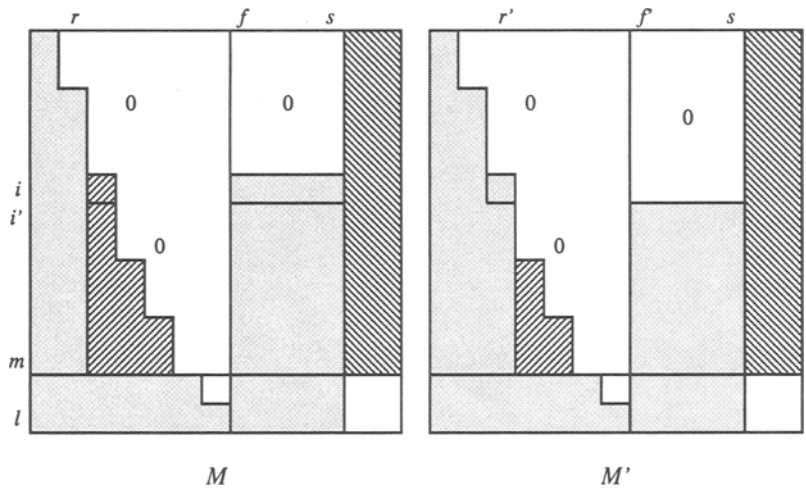

Fig. 2. Computing the HNF of the leading k columns, for $k = \alpha, 2\alpha, \ldots$

Theorem 2. *Let $A \in \mathrm{Mat}_{m\times n}(\mathbf{R})$ with rank r and $1 \leq \alpha, \beta \leq n$. Then the* PARALLEL-HNF *algorithm uses $O(\frac{n^3}{\alpha\beta} + \frac{nm}{\alpha})$ broadcasts with distributed $W := \begin{pmatrix} A \\ I_n \end{pmatrix}$, α and β as input. It transfers $O(\frac{n^3}{\alpha})$ ring elements.*

For $\mathbf{R} = \mathbb{F}[x]$ the procedure uses $O((m+n-r)n^5\beta^2 4^\alpha \|A\|^2)$ field operations. At most $O(\frac{n^4\beta 2^\alpha \|A\|}{\alpha})$ field elements are transferred via broadcasts.

Proof. Transforming a $(m+n) \times s$ principal submatrix of W with $s \in \{\alpha, 2\alpha, \ldots, (p-1)\alpha, n\}$ into echelon form requires at most $m+n-r$ broadcasts. Transforming the echelon form into HNF requires (by Theorem 1) $\frac{q(q+1)}{2}\beta$ with $q := \left\lfloor \frac{s-2}{\beta} \right\rfloor$ broadcasts. In total this leads to the broadcast bound

$$
\begin{aligned}
& p(m+n-r) + \sum_{s=\alpha,\ldots,(p-1)\alpha,n} \left(\frac{1}{2} \cdot \left\lfloor \frac{s-2}{\beta} \right\rfloor \left(\left\lfloor \frac{s-2}{\beta} \right\rfloor + 1 \right) \beta + s \right) \\
& \leq p(m+n-r) + \sum_{i=1}^{p} \left(i\alpha + \left\lfloor \frac{i\alpha - 2}{2} \left(\frac{i\alpha - 2}{\beta} + 1 \right) \right\rfloor \right) \\
& \leq p(m+n-r) + \alpha \frac{p(p+1)}{2} + \sum_{i=1}^{p} \left\lfloor \frac{(i\alpha)^2}{2\beta} + \frac{i\alpha}{2} \right\rfloor \\
& = \underbrace{p(m+n-r)}_{\in O(n(n+m)/\alpha)} + \underbrace{\alpha \frac{p(p+1)}{2}}_{\in O(n^2/\alpha)} + \Bigg\lfloor \underbrace{\alpha^2 \frac{2p^3+3p^2+p}{12\beta}}_{\in O(n^3/(\alpha\beta))} + \underbrace{\alpha \frac{p(p+1)}{4}}_{\in O(n^2/\alpha)} \Bigg\rfloor \\
& \in O\left(\frac{n^3}{\alpha\beta} + \frac{nm}{\alpha} \right)
\end{aligned}
$$

To compute the echelon form we do not need to broadcast the linearly dependent rows. Thus, the number of broadcast ring elements can be majorized by

$$\sum_{i=1}^{s} (1 + (\alpha - 1)) = s\alpha$$

for a $(m+n) \times s$ principal submatrix of W. Transforming the echelon form to Hermite normal form requires broadcasting at most $O(r^2)$ ring elements. Computing the Hermite normal form of all $(m+n) \times s$ principal submatrices of W with $s \in \{\alpha, 2\alpha, \ldots, (p-1)\alpha, n\}$ we need

$$
\begin{aligned}
O(n\alpha + \sum_{i=1}^{p-1} i\alpha^2) + pO(r^2) & \subseteq O(n\alpha + \frac{p(p-1)}{2}\alpha) + O(pr^2) \\
& \subseteq O(n\alpha) + O(\frac{n^2}{\alpha}) + O(\frac{n^3}{\alpha}) \subseteq O(\frac{n^3}{\alpha})
\end{aligned}
$$

ring elements to be broadcast.

The proofs of the other two estimates are quite lengthy. They can be found in [23].

5 Performance examples

We have implemented this and related algorithms in C/C++ on the IBM SP2 at the GMD in Sankt Augustin. We have used the xlC compiler and

the message passing library MPL (both IBM products). We have used the Sorting-GCD algorithm, due to Majewski-Havas [17], for the implementation of the COMPUTE-GCD function, where we used a heap for determining the polynomial with the largest degree or the integer with the largest absolute value, respectively in a subvector.

We have done many practical studies with these algorithms. In this paper we give some details of the behavior of PARALLEL-HNF for some random matrices over $\mathbb{F}_2[x]$ and $\mathbb{Z}$. Thus we used an input matrix over $\mathbb{F}_2[x]$ which is a random 80×80 matrix, where the degree of each entry is less than or equal to 80. The rank of this matrix is $r = 80$. The input matrix over $\mathbb{Z}$ is a random 100×100 matrix. The absolute value of each entry is less than or equal to 64.

Table 1 and Figures 3 and 4 show the results of experiments in which we varied α and β. We used 16 nodes of the SP2. The first row of each measurement is the total running time (*minutes:seconds.hundredth*). The second row gives the maximum degree ($\mathbb{F}_2[x]$), or the number of bits in the largest absolute value ($\mathbb{Z}$) which arose during the computation. In Figure 3 the x-axis shows α while the y-axes show run times and maximum degrees. In Figure 4 the x-axis shows α while the y-axes show run times and maximum number of bits.

$\mathbb{F}_2[x]$				$\mathbb{Z}$			
α	$\beta = 1$	$\beta = \alpha$	$\beta = n+1-\alpha$	α	$\beta = 1$	$\beta = \alpha$	$\beta = n+1-\alpha$
1	06:53.08 12708	06:57.10 12708	06:52.68 12717	1	00:20.35 1401	00:20.28 1401	00:18.92 1412
3	05:34.19 12708	07:14.12 15073	10:28.21 18646	3	00:23.25 1698	00:22.91 1698	00:21.99 2138
5	05:45.01 13701	06:41.58 15016	10:51.69 23377	5	00:19.68 1598	00:19.59 1744	00:22.39 2639
8	04:43.10 12232	05:06.79 19614	10:02.18 26823	8	00:20.57 1641	00:20.35 2374	00:21.78 3146
16	04:11.13 12384	04:40.08 27387	09:05.69 33377	20	00:21.01 1263	00:22.65 5851	00:27.84 6506
20	04:28.23 11270	05:13.12 29951	08:40.16 35593	40	00:31.24 1464	00:40.86 13977	00:46.43 14266
40	03:49.84 9982	06:12.17 37468	06:02.98 37883	60	00:43.38 2043	01:14.43 20222	01:11.70 20222
60	03:32.68 11270	08:11.38 35593	05:00.42 30719	80	01:37.72 4544	03:09.35 64587	01:49.23 51291
80	01:33.32 9955	07:01.55 43486	01:32.47 9955	100	11:36.60 20987	11:36.20 20987	11:38.67 20987

Table 1. Effect of varying α and β

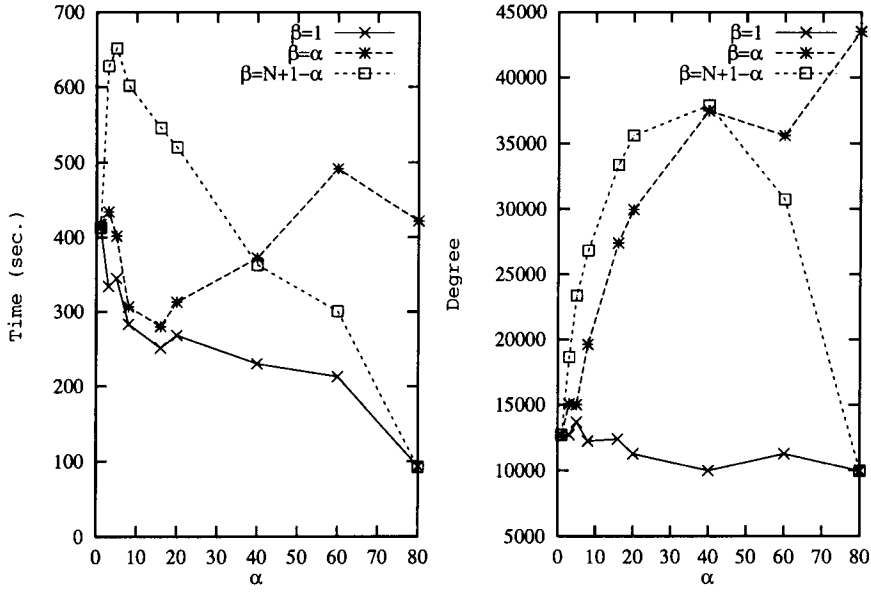

Fig. 3. Effect of varying α and β for a matrix over $\mathbb{F}_2[x]$: (a) running times, (b) maximum degree

6 Concluding Remarks

The reuse of sequential code for parallel implementations is well supported by the operation row concept. We have used earlier sequential GCD algorithms for our parallel implementations. This leads to parallel implementations for Hermite normal form computation with good speed-up. In the integer case the hybrid algorithm gives best results for small α and $\beta = n$. For $\mathbf{R} = \mathbb{F}_2[x]$ we get the best performance for $\alpha = n$ and small β.

For *integer* matrices with *large* rank, the new procedures are fastest. For distributed computation it is good to use the parallel, hybrid procedure with *small* $\alpha(< 10)$ and $\beta := n$. These procedures produce very good transformation matrices (i.e., entries with small absolute value) if the rank of the matrix is less than the number of columns. The transformation matrices are almost as good as ones obtained using LLL lattice basis reduction methods, which are orders of magnitude slower. For matrices over $\mathbb{F}_2[x]$ and for *integer* matrices with *small* rank, "Gaussian elimination" type procedures ($\alpha := n$) are fastest. For distributed computation use *small* $\beta(\in \{1, 2\})$.

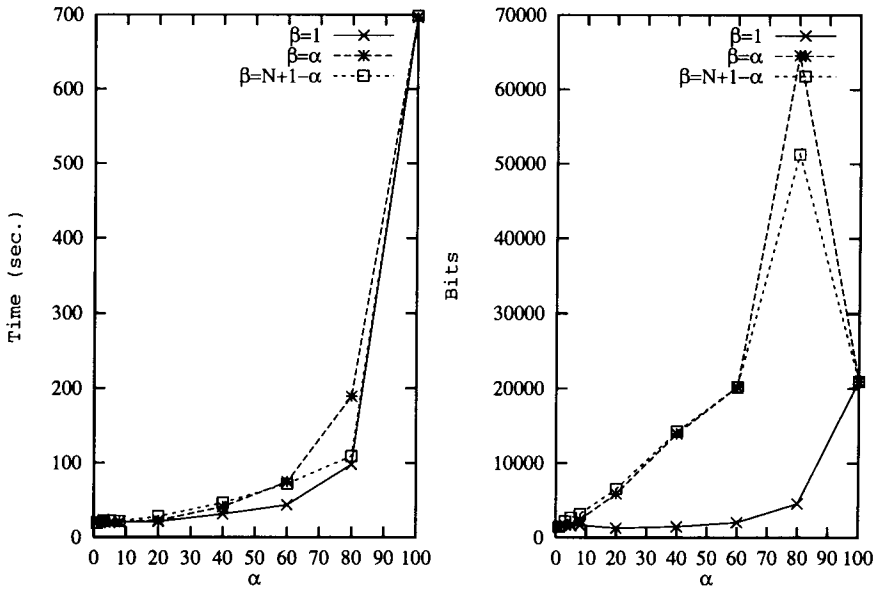

Fig. 4. Effect of varying α and β for an integer matrix: (a) running times, (b) maximum number of bits needed

Acknowledgements

The first author was partially supported by the Australian Research Council.

References

1. W. A. Blankinship. A new version of the Euclidian algorithm. *Amer. Math. Monthly* **70** (1963) 742–745.
2. G. H. Bradley. Algorithm and bound for the greatest common divisor of n integers. *Comm. ACM* **13** (1970) 433–436.
3. T-W.J. Chou and G.E. Collins. Algorithms for the solution of systems of linear Diophantine equations. *SIAM J. Comput.* **11** (1982) 687–708.
4. X.G. Fang and G. Havas. On the worst-case complexity of integer gaussian elimination. *ISSAC'97* (Proc. 1997 Internat. Sympos. Symbolic Algebraic Comput.), ACM Press (1997) 28–31.
5. K.O. Geddes, S.R. Czapor and G. Labahn. *Algorithms for Computer Algebra.* Kluwer Academic Publishers, 1992.
6. M. Giesbrecht. Fast computation of the Smith normal form of an integer matrix. *ISSAC'95* (Proc. 1995 Internat. Sympos. Symbolic Algebraic Comput.), ACM Press (1995) 110–118.
7. B. Hartley and T.O. Hawkes. *Rings, Modules and Linear Algebra.* Chapman and Hall, 1976.

8. G. Havas, D.F. Holt and S. Rees. Recognizing badly presented **Z**-modules. *Linear Algebra Appl.* **192** (1993) 137–163.
9. G. Havas and B.S. Majewski. Hermite normal form computation for integer matrices. *Congressus Numerantium* **105** (1994) 87–96.
10. G. Havas and B.S. Majewski. Extended gcd calculation. *Congressus Numerantium* **111** (1995) 104–114.
11. G. Havas and B.S. Majewski. Integer matrix diagonalization. *J. Symbolic Computation* **24** (1997) 399–408.
12. G. Havas, B.S. Majewski and K.R. Matthews. Extended gcd and Hermite normal form algorithms via lattice basis reduction. *Experimental Mathematics* **7** (1998) 125–135.
13. G. Havas and C. Wagner. Matrix reduction algorithms for Euclidean rings. *Proc. 1998 Asian Symposium on Computer Mathematics*, Lanzhou University Press (1998) 65–70.
14. R. Kannan and A. Bachem. Polynomial algorithms for computing Smith and Hermite normal forms of an integer matrix. *SIAM J. Comput.* **8** (1979) 499–507.
15. R. Kannan. Solving systems of linear equations over polynomials. *Theoretical Computer Science* **39** (1985) 69–88.
16. E. Kaltofen, M. S. Krishnamoorthy, and B. D. Saunders. Parallel algorithms for matrix normal forms. *Linear Algebra Appl.* **136** (1990) 189–208.
17. B. S. Majewski and G. Havas. A solution to the extended gcd problem. *ISSAC'95* (Proc. 1995 Internat. Sympos. Symbolic Algebraic Comput.), ACM Press (1995) 248–253.
18. M. Newman. *Integral Matrices.* Academic Press, 1972.
19. C.C. Sims. *Computation with finitely presented groups.* Cambridge University Press (1994).
20. A. Storjohann. Near optimal algorithms for computing Smith normal forms of integer matrices. *ISSAC'96* (Proc. 1996 Internat. Sympos. Symbolic Algebraic Comput.), ACM Press (1996) 267–274.
21. A. Storjohann. Computing Hermite and Smith Normal Forms of Triangular Integer Matrices. *Linear Algebra Appl.* **282** (1998) 25–45.
22. A. Storjohann and G. Labahn. Asymptotically fast computation of Hermite normal forms of integer matrices. *ISSAC'96* (Proc. 1996 Internat. Sympos. Symbolic Algebraic Comput.), ACM Press (1996) 259–266.
23. C. Wagner. *Normalformberechnung von Matrizen über euklidischen Ringen.* PhD thesis, Institut für Experimentelle Mathematik, Universität-GH Essen, 1997. Published by Shaker-Verlag, 52013 Aachen/Germany, 1998.
24. C. Wagner. Fast parallel Hermite normal form computation of matrices over F[x]. *Euro-Par'98 Parallel Processing*, Lecture Notes Comput. Sci. **1470** (1998) 821–830.