ΑΦΙΕΡΩΜΕΝΟ ΣΤΟΥΣ ΓΟΝΕΙΣ ΜΟΥ

ΣΠΥΡΟ ΔΕΣΠΟΙΝΑ

DEDICATED TO MY PARENTS

SPIROS DESPINA

CONSTANTINOS SPIROS' ILIOPOULOS
B.Sc. (Athens), M.Sc. (Warwick)

COMPUTATIONAL PROBLEMS

IN THE THEORY OF ABELIAN GROUPS

Ph.D. THESIS

Warwick University
Department Of Computer Science
Coventry CV4 7AL
England

August 1983

ΑΦΙΕΡΩΜΕΝΟ ΣΤΟΥΣ ΓΟΝΕΙΣ ΜΟΥ

ΣΠΥΡΟ ΔΕΣΠΟΙΝΑ

DEDICATED TO MY PARENTS

SPIROS DESPINA

CONTENTS

CHAPTER II
COMPUTATIONAL PROBLEMS ON FINITE ABELIAN GROUPS REPRESENTED BY AN EXPLICIT SET GENERATORS

ACKNOWLEDGEMENTS

I am indebted to my supervisor Dr. Meurig Beynon for the enormous amount of time spent with me, the effort and his continuous help during the two years of research for this degree. It is a pleasure working with him and I certainly enjoyed our mathematical debates.

I should like to thank Professor Mike Paterson for many rewarding discussions.

I should also like to thank Professor John Dixon (Carleron University, Ottawa) for his helpful comments on my work on Chapters I and III. Algorithm I.7.1 and Proposition II.9.4 are due to him (any errors are due to me).

I am grateful to Professor Christos Papadimitriou (National Tech. Univ. of Athens and M.I.T.) for helpful comments and for pointing out to me the reference [19].

It is a pleasure to acknowledge the financial support of the Greek section of research and scholarships "Alexander S. Onassis" Public Benefit Foundation for the academic year 1982-83.

I would like to thank my parents for their encouragement, love and support. I heartily dedicate this work to them.

CHAPTER 0

INTRODUCTION

1. SUMMARY

In this thesis, the worst-case time complexity bounds on the algorithms for the problems mentioned below have been improved.

A. Algorithms on abelian groups represented by a set of defining relations for computing:
   (i) a canonical basis for finite abelian groups
   (ii) a canonical basis for infinite abelian group

B. Algorithms for computing:
   (i) Hermite normal form of an integer matrix
   (ii) The Smith normal form of an integer matrix
   (iii) The set of all solutions of a system of Diophantine Equations

C. Algorithms on abelian groups represented by an explicit set of generators for computing:
   (i) the order of an element (space complexity is only improved)
   (ii) a complete basis for a finite abelian group
   (iii) membership-inclusion testing
   (iv) the union and intersection of two finite abelian groups

D. A classification of the relative complexity of computational problems on abelian groups (as above), factorization and primility testing.

E. Algorithms on abelian subgroups of the symmetric group for computing:
   (i) the complete structure of a group
   (ii) membership-inclusion testing
   (iii) the union of two abelian groups
   (iv) the intersection of two abelian groups.

2. INTRODUCTION

In this thesis we investigate a large class of computational problems in the theory of abelian groups. Abelian groups are studied in three forms of representation:
(i) represented by a set of defining relations, (ii) represented by an explicit set of generators and (iii) represented as subgroups of a permutation group.

The algorithms presented here are considered from the point of view of worst-case complexity; mostly we consider the worst-case time complexity and in few places the space complexity. All algorithms mentioned below have been shown to have better worst-case complexity upper bounds than the bounds already cited in the literature. A brief outline of the algorithms included is given here; for a fuller summary of the restuls see the introduction of each chapter.

In the case which the group is presented by a set of defining relations, algorithms for computing the canonical structure of a finite or infinite abelian group is presented; algorithms for the closely related problems of computing the Hermite and Smith normal form of an integer matrix are also given. Among the applications of the above algorithms are methods for solving systems of Linear Diophantine Equations, for computing the characteristic polynomial of a matrix and for the problem of integer linear programming.

In the case of an abelian group represented by an explicit generating set an algorithm for computing the complete structure is presented; algorithms for membership-inclusion testing and for computing the intersection of groups are also given. These algorithms have applications to factorization and public key cryptosystems. Further a classification of the relative complexity of the factorization problem, primality testing and several group-theoretic problems is given.

In the case of abelian permutation groups, an algorithm for computing the complete structure together with algorithms for membership-inclusion testing are presented. Further an algorithm for computing the intersection of two abelian groups is given; this problem is related to the graph isomorphism problem.

Most of the algorithms are presented in an informal computer-like language named PIDGIN-ALGOL introduced by Aho, Hopcroft and Ullman in [2].

By Proposition II.3.5 is denoted the 5-th proposition of the 3rd section of Chapter II. If the chapter number (in Roman) is omitted, then we refer to the current chapter.

Coventry, August 1983

C.S.I.

3. PRELIMINARIES

A. Worst-Case Complexity Bounds For Numerical Algorithms

The computational complexity is measured in terms of *elementary operations*. An elementary operation is a Boolean operation on a single binary bit or pair of bits or an input or shift of a binary bit.

For ease in bounding operation counts the following convention is established throughout:

$$\log n = \begin{cases} \log_2 n & \text{if } n > 4 \\ & \text{if } n < 4 \end{cases}$$

Moreover the cardinality number of a set S is denoted by $|S|$.

<u>THEOREM 3.1</u>. (Schönhage and Strassen [45])

There exists an algorithm for multiplying two integers of length n bits in $M(n)$ elementary operations, where

$$M(n) = cn \log n \log\log n$$

for some positive constant c. □

<u>THEOREM 3.2</u> (Cook [9])

There exists an algorithm for division of n bit integers in $M(n)$ elementary operations. □

The following theorem yields an upper bound on Knuth's algorithm (see [35]) for computing the greater common divisor of two integers. The bound is due to Schönhage [46].

THEOREM 3.3 (Extended Euclidean Algorithm-abbreviated EEA)

There exists an algorithm for computing the gcd r of two integers a_1, a_2 and two integers x_1 and x_2 such that

$$x_1 a_1 + x_2 a_2 = r$$

with

$$|x_1| < |a_2|/2 \qquad |x_2| < |a_1|/2$$

in $O(M(n) \log n)$ elementary operations, where $n = \log(\max\{|a_1|, |a_2|\})$. □

The following theorem due to Goppersmith-Winograd (see [10]) yields the current upper bound on matrix multiplication.

THEOREM 3.4

There exists an algorithm for multiplying two $n \times n$ matrices, which requires $O(n^\alpha)$ multiplications, where

$$\alpha = 2.495364\ldots\ . \quad □$$

In 1969 Strassen [51] gave the celebrated algorithm for matrix multiplication which requires $O(n^{\log 7}) = O(n^{2.81})$ multiplications. Since then a series of improvements (Pan, Bini, Winograd) have led to the bound of Theorem 3.4. It is worth mentioning that these improvements are of theoretical rather than practical interest, since the hidden constant of the symbol O is very large.

THEOREM 3.5 (Strassen [51])

The inverse of a $n \times n$ matrix may be computed with $O(n^\alpha)$ multiplications. □

A sorting algorithm and searching algorithm is presented below.

<u>THEOREM 3.6</u> (Heapsort, see Williams [53], Floyd [15])

There exists an algorithm sorting a list of n elements which requires $O(n \log n)$ comparisons. □

<u>THEOREM 3.7</u> (Binary search)

There exists an algorithm for searching for a particular element in a sorted list of n elements which requires $O(\log n)$ comparisons. □

B. Abelian Groups

A set $\{g_1, g_2, \ldots, g_n\}$ is called a *generating set* for an abelian group G, if for every $x \in G$ there exists integers a_i for $1 < i < n$ such that

$$x = g_1^{a_1} \ldots g_n^{a_n} .$$

The g_i's are called *generators*.

A set of relations $S = \{x_1^{a_{1i}} \ldots x_n^{a_{ni}} = 1$, for $1 < i < m$, $x_k x_j = x_j x_k, \forall\, k,j\}$ is said to be a set of *defining relations* for an abelian group G if every relation holding in G can be derived from S. The matrix

$$A = \begin{bmatrix} a_{11} & \cdots\cdots & a_{1m} \\ \vdots & & \\ a_{n1} & \cdots\cdots & a_{nm} \end{bmatrix}$$

associated with S *represents* the group G.

Suppose that G is a group and x is an element of G. If h is the smallest positive integer such that $x^h = 1$ is called the *order* of x denoted $|x|$. The least common multiple of the orders of all the elements of

G is called *exponent* of G. The total number of elements of G is called *order* of the group G denoted $|G|$.

The *direct product* of two abelian groups H and K both subgroups of a group G with $H \cap K = 1$ is said to be

$$H \times K = \{hk : H \in H,\ k \in K\}.$$

A cyclic group of order n is denoted by C(n).

Suppose that the abelian group G is decomposed in terms of cyclic groups

$$G = C(d_1) \times \ldots \times C(d_k) \times \underbrace{C(\infty) \times \ldots \times C(\infty)}_{m \text{ times}} \qquad (3.1)$$

If d_i divides d_{i+1} for $1 < i < k$, then the representation (1.1) defines the *canonical structure* of G.

<u>THEOREM 3.8</u> (H.J. Smith, [49])

The canonical structure of G is unique. □

Suppose that the abelian group G is decomposed in terms of cyclic groups

$$G = C(p_1^{\alpha_{11}}) \times \ldots \times C(p_1^{\alpha_{1k}}) \times G(p_2^{\alpha_{21}}) \times \ldots \times C(p^{\alpha_{\lambda\mu}}) \times \underbrace{C(\infty) \times \ldots \times C(\infty)}_{m \text{ times}} \qquad (3.2)$$

where the p_i's are distinct primes. The representation (3.2) defines the *complete structure* of G.

A set of elements $B = \{b_1, \ldots, b_n\}$ in an abelian group G is *independent* if a finite product $\prod_i b_i^{e_i} = 1$ only when $b_i^{e_i} = 1$ for every i. If an independent set B of an abelian group G also generates G, then it is said that B form a *basis* for G; it is denoted by $G = \langle\langle b_1 \ldots b_n \rangle\rangle$.

If the orders of the elements of a basis for an abelian group yield the canonical structure of the group, then the basis is called *canonical basis*. If the orders of the elements of a basis for an abelian group yield the complete structure of the group then the basis is called *complete basis*.

If G is a group, then the set

$$Z(G) = \{x : xy = yx \quad \forall\, y \in G\}$$

under the group operation of G forms a group called the *centre* of G.

The multiplicative group of integers modulo n is denoted by $\mathbf{Z}_n^*$. (integers modulo n relative prime to n under multiplication)

C. Riemann Hypotheses

The function

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}\text{, s complex number}$$

is called *Riemann zeta* function.

The hypothesis that the zeros of $\zeta(s)$ in the critical strip $0 < \mathrm{Re}(s) < 1$ ($\mathrm{Re}(s)$ denotes the real part of s) all lie on the line $\mathrm{Re}(s) = \frac{1}{2}$, is called *Riemann's Hypothesis*.

Dirichlet's L functions are defined by:

$$L(s,\chi) = \sum_{n=1}^{\infty} \chi(n)/n^s\text{, with s complex number}$$

where χ is a character (we call χ a character, if χ is a function over all group elements of a group $G=\{a_1,\ldots,a_h\}$ with properties (i) $\chi(a_i) \neq 0, \forall_i$ and (ii) $\chi(a_i)\,\chi(a_j) = \chi(a_i a_j), \forall_{i,j}$). Two well known characters are

(i) The Legendre symbol $\chi_p = (\frac{a}{p})$ defined by

$$(\frac{a}{p}) := \begin{cases} 1 & a^2 \equiv 1 \bmod p \\ -1 & a^2 \not\equiv 1 \bmod p \text{ and } \gcd(a,p) = 1, \text{ with } p \text{ prime} \\ 0 & \gcd(a,p) \neq 1 \end{cases}$$

(ii) The Jacobi symbol $\chi_{pq} = (\frac{a}{pq})$ defined by

$$(\frac{a}{pq}) := (\frac{a}{p})\ (\frac{a}{q}), \text{ with } p,q \text{ primes}$$

where $(\frac{a}{p})$ and $(\frac{a}{q})$ are Legendre symbols.

The hypothesis that the zeros of $L(s,\chi)$ in the critical strip $a < Re(s) < 1$ all lie on the line $Re(s) = \frac{1}{2}$, where χ is a Legendre or a Jacobi symbol with fixed denominators, is called *Extended Riemann's Hypothesis* (ERH).

Some of the proofs of the propositions of Section 9 of the second chapter depend on the truth of ERH.

CHAPTER I

COMPUTATIONAL PROBLEMS ON ABELIAN GROUPS REPRESENTED BY A SET OF DEFINING RELATIONS

1. INTRODUCTION

In this section computational problems on abelian groups represented with a set of defining relations and closely related problems are investigated. In particular, the problem of computing the order and the canonical structure of a finite or an infinite abelian group is examined. Consequently the closely related problems of computing the Hermite normal form and the Smith normal form of a non-singular integer matrix are considered. Moreover an effective way of solving systems of linear Diophantine equations is studied.

An algorithm for computing the order and the canonical structure of a finite abelian group is presented; it requires $O(s^5 M(s^2))$ elementary operations, where s is the size of the matrix associated with the set of defining relations. The most competitive algorithm for this problem is due to Chou-Collins (see [8]) which requires $O(s^{11})$ elementary operations.

An algorithm for computing the Hermite normal form of an integer matrix is given; it requires $O(s^3 M(s^2))$ elementary operations, where s is the size of the matrix. Also an algorithm for computing the Smith normal form of an integer matrix of size is in $O(s^5 M(s^2))$ elementary operation is presented. The upper bounds of the above two algorithms improve the upper bounds given in [8]; the algorithm for Hermite normal form is shown to be optimal as direct method. Moreover algorithms for computing the multiplier-matrices M, B and C such that MA and BAC are the Hermite and Smith normal forms of the given matrix A are presented;the algorithm for computing M requires $O(s^3 M(s^2))$ elementary operations and the algorithm for computing B and C requires $O(s^{5.49} M(s^2))$ elementary operations.

An algorithm for computing the canonical structure of an infinite abelian group in $O(s^5 M(s^2))$ elementary operations, is presented here.

An algorithm for computing the set of allsolutions (or a particular solution) or establishing that there is none, of systems of linear Diophantine equations is given; it requires $O(s^{3.49} M(s^2) + s^2 M(s^*))$ elementary operations where s is the size of the matrix A and s* the size of vector b (The system is $Ax = b$). This upper bound improves the Chou-Collins upper bound in [8] by at least a factor of $O(s^2)$. Moreover, it is better than the Frumkin's upper bounds on the computation of a particular solution and on solving a homogeneous system of linear Diophantine equations.

A simplified algorithm for computing the canonical structure of an abelian group having asymptotically the same complexity with algorithm mentioned above is given; its main difference is that it is not a direct method.

The chapter closes with a discussion about applications of the algorithms refered to above.

A.Preliminaries

DEFINITION 1.1

Suppose that A is an $m \times n$ matrix. If the entries of A are integers, then the *norm* $||A||$ of the matrix is said to be the integer

$$||A|| = \max_{i,j} \{|a_{ij}|\}.$$

If the matrix A is over the field of rationals (denoted by $\mathbb{Q}$), then the *norm* $||A||$ is defined to be the integer

$$||A|| = \max_{i,j} \{|r_{ij}|, |s_{ij}| : a_{ij} = \frac{r_{ij}}{s_{ij}} \text{ with } \gcd(r_{ij}, s_{ij}) = 1\}. \ \square$$

DEFINITION 1.2

The *size* s of an $m \times n$ matrix A with entries from $\mathbb{Z}$ or $\mathbb{Q}$ is defined to be the number

$$s = m + n + \log ||A|| \ . \ \square$$

DEFINITION 1.3

A square matrix A is called *singular* if its determinant (denoted det (A)) is zero, otherwise is called *non-singular*.

A square matrix A over the integers, whose determinant is of absolute value 1, is called *unimodular*. □

NOTATION 1.4

If A is an $m \times n$ matrix, then the $n \times m$ matrix A^t defined by

$$[A^t]_{ij} = a_{ji} ,$$

is said to be the *transpose* matrix of A. □

<u>NOTATION 1.4a.</u>

An $n \times n$ identity matrix is denoted by I_n and an $n \times n$ zero matrix is denoted by 0_n. □

<u>NOTATION 1.5</u>

Let A be an $m \times n$ matrix. For $1 < i < n$, the i-th column of A will be denoted by $COL_A(i)$ and for $1 < j < m$, the j-th row of A will be denoted by $ROW_A(j)$. When there is no ambiguity the subscript A is omitted. □

<u>DEFINITION 1.6</u>

An *elementary row-column operation* (abbreviated ERC operation) on a matrix with entries from a ring $(R, +, \cdot)$ is:

(i) The multiplication of all entries of a row (column) by -1 .

or (ii) The interchange of two rows (columns).

or (iii) The addition of a multiple (over R) of a row (column) to a different row (column).

An IRC operation over the ring of integers is called an *integer row-column operation* (abbreviated IRC operation). □

<u>DEFINITION 1.7</u>

The square matrices

$$E_j(a) = \begin{bmatrix} 1 & & & & & & 0 \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & a & & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ 0 & & & & & & 1 \end{bmatrix}, \qquad E_{ik}(a) = \begin{bmatrix} 1 & & & & & & 0 \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & \vdots & \ddots & & & \\ & & a & \cdots & 1 & & \\ & & & & & \ddots & \\ 0 & & & & & & 1 \end{bmatrix}$$

$\uparrow$ j-th column $\qquad$ $\uparrow$ k-th column i-th row are called *elementary matrices.* □

<u>THEOREM 1.8</u> (see Gantmacher [19]).

Every matrix can be written as a product of elementary matrices. □

<u>PROPOSITION 1.9</u>

An ERC operation on a matrix corresponds to a multiplication of the matrix by an elementary matrix. □

<u>DEFINITION 1.10</u>

An algorithm for transforming a matrix A to another matrix A* is called a *direct method* if the transformation is performed by means of a sequence of linear combinations of rows and columns of the matrix A. □

The above definition was introduced by Klyuyen and Kokovkin-Shchebak in [34].

2. IRC OPERATIONS-FINITENESS OF AN ABELIAN GROUP -THE ORDER OF AN ABELIAN GROUP

Suppose that an $m \times n$ matrix A represents the abelian group G. The multiplication of a row by -1 substitutes the relation $\prod_i x_i^{a_{is}} = 1$ for the relation $\prod_i x_i^{-a_{is}} = 1$. The addition of an integer multiple of a row to a different row substitutes the relations $\{\prod_i x_i^{a_{is}} = 1, \prod_i x_i^{a_{ik}} = 1\}$ for the relations $\{\prod_i x_i^{a_{is}} = 1, \prod_i x_i^{a_{ik}-pa_{is}} = 1\}$. The interchange of rows corresponds to the renaming of generators. The multiplication of a column by -1 substitutes the generator x^{-1} for the generator x. The addition of an integer multiple of a column to a different column changes the generators by substituting $\{xy^p,y\}$ for $\{x,y\}$. Hence the following lemma is shown:

<u>LEMMA 2.1</u>

The IRC operations on a matrix A respect the structure of the group G represented by A. □

Let A and G be as above and suppose that rank $(A) = r < n$. Then there exists a sequence of IRC operations to transform A to the matrix $A' := [A^*,0]$, where A^* is an $m \times r$ matrix.† Therefore the generators $x_{r+1},\dots,x_n$ corresponding to the $r+1,\dots,n$-th column of A' respectively are free and thus G is infinite. Hence the following lemma is proved:

<u>LEMMA 2.2</u>

Suppose that an $m \times n$ matrix A represents the abelian group G. Then G is finite if and only if rank $(A) = n$. □

†See [48].

One can compute the order or a multiple of the order of an abelian group G (If G is infinite group, then one can compute the order or a multiple of the order of the finite component[†] of G) in the following way:

(i) Case rank (A) = m = n. In this case G is finite and it is not difficult to show that the determinant of A is equal to the order $|G|$ of G.

(ii) Case rank (A) = n < m. In this case G is finite. Moreover it is not difficult to show that

$$|G| = \gcd \{\det (M): M \text{ is an } n \times n \text{ non-singular submatrix of } A\}$$

(Using Gaussian elimination, one can transform A to the matrix $\begin{bmatrix} M^* \\ 0 \end{bmatrix}$, where M^* is an n x n matrix. Then $|G| = \det (M^*)$ and in [19] one can see that $\det (M^*) = \gcd \{\det(M), M \text{ as above}\}$)

Since the number of n x n non-singular submatrices of A is $\binom{m}{n}$ (see [19]), the above formula does not yield an efficient way of computing $|G|$. By computing the determinant of n x n non-singular submatrix M one can compute a multiple of the order $|G|$.

(iii) Case of rank (A) < n. In this case G is infinite. Let $G = H \times K$ where H is a subgroup containing all the finite order elements of G. Then

$$|H| = \gcd \{\det(M): M \text{ is an } r \times r \text{ non-singular submatrix of } A\}.$$

Since the number of r x r submatrices of A is $\binom{m}{r}\binom{n}{r}$ (see [19]), the above formula does not yield an efficient way of computing $|H|$ but by computing the determinant of such a matrix M one can compute a multiple of the order of H.

[†]The subgroup of an abelian group containing all the finite order elements of G is called, the finite component of G.

Papadimitriou and Steiglitz [43] refer to the existence of a polynomial time upper bound on Gaussian elimination over the rationals. Strassen in [51] yields a polynomial time upper bound on the number of multiplications needed for Gaussian elimination over a field but this does not imply necessarily a polynomial time upper bound in terms of elementary operations. The author knows of no reference to an explicit upper bound on the number of elementary operations required for Gaussian elimination which is shown below:

ALGORITHM 2.3 (Gauss)

INPUT: An $m \times n$ matrix A with entries from rationals (Wl.o.g. assume $m > n$)

OUTPUT: The rank r of A and the determinant d of an $r \times r$ non-singular submatrix of A

```
begin
  $i \leftarrow 0$
1. repeat
        $i \leftarrow i+1$:
        if $a_{ii} = 0$ then
         interchange ROW($i$) and ROW($k$), COL($i$) and COL($\lambda$), where $a_{k\lambda} \neq 0$,
                                                                   $k, \lambda \geq i$;
2.      $\mu_j \leftarrow a_{ji}/a_{ii}$ for $i < j \leq m$;
3.      ROW($j$) $\leftarrow$ ROW($j$) $- \mu_j$ ROW($i$) for $i < j \leq m$;
4.  until ROW($j$) $= (0,\dots,0)$ for all $j > i$;
  $r \leftarrow i$;
5.$d \leftarrow \prod_{i=1}^{r} a_{ii}$;
end. □
```

PROPOSITION 2.4

Algorithm 2.3 correctly computes r and d in $O(nmr\, M(\log|d^*|))$ elementary operations, where $|d^*| = \max\{|\det(M)| : M$ is an $r \times r$ non-singular submatrix of $A\}$.

Proof

Let $A^{(i)}$ denote A at the beginning of the i-th iteration of loop 1-4. Step 2 requires at most m divisions comprising $O(mM(\log\|A^{(i)}\|))$ elementary operations. Step 3 requires at most mn multiplications comprising $O(mnM(\log\|A^{(i)}\|))$ elementary operations. Hence algorithm 2.3 requires

$$O(\sum_{i=1}^{r} mn\, M(\log\|A^{(i)}\|))$$

elementary operations.

It is known (see [19], p.26 formula (12)) that $\|A^{(i)}\|$ is at most the largest in absolute value determinant of an ixi submatrix of A. Therefore the algorithm requires $O(mnr\, M(\log|d^*|))$ elementary operations. □

THEOREM 2.5 (Hadamard)

Suppose that A is an $n \times n$ square matrix and d its determinant. Then

$$|d| < \prod_{i=1}^{n} (\sum_{s=1}^{n} a_{is}^2)^{1/2}. \quad □$$

COROLLARY 2.6

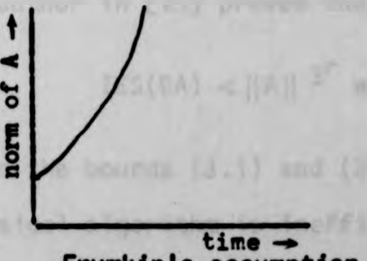
Algorithm 2.3 terminates in $O(s^3 M(s^2))$ elementary operations, where s is the size of A. □

3. A DIRECT METHOD FOR COMPUTING THE STRUCTURE OF FINITE ABELIAN GROUPS

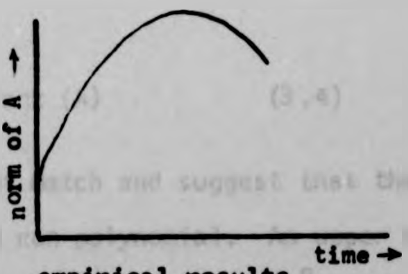
The classical algorithm (see Smith [48], Sims [50]) for transforming a matrix via IRC operations to a diagonal one may produce very large entries in the matrix at the intermediate steps. This effect is called *intermediate expression swell*[†] (abbreviated IES), see McClellan [39]. Frumkin in [17] observed that the IES of Bradley's algorithm (see [5], it is a slightly improved version of the classical algorithm) denoted IES(BA), can be higher than 2^{2^v}, where $v = \max\{m,n\}$, where the initial matrix is an $m \times n$ matrix. In [16] Frumkin using heuristic arguments indicated that:

(i) $\quad \mathrm{IES(BA)} > \|A\|^{(1+\varepsilon)^n}$ for some $\varepsilon > 0$ (3.1)

under the assumption that the entries of the matrix increase after a step. The two diagrams below illustrate the assumed and the empirical growth of the entries



Frumkin's assumption



empirical results

[†]The largest in absolute value entry of the martix throughout the computation is the IES of the algorithm

(ii) $\quad \mathrm{IES}(BA) < (n\,\|A\|)^n$ $\quad$ (3.2)

under the assumption that there exist integers $x_1,\ldots,x_n$ such that

$$\sum_{i=1}^{n} x_i a_i = \gcd(a_1,\ldots,a_n) \text{ and } \|x_i\| = O(\|a_i\|^{1/n})$$

$$\text{with } a_i \in \mathbb{Z},\ 1 < i < n$$

But empirical tests (see Bradley [5]) show that

$$\|x_i\| = O(\|a_i\|). \qquad (3.3)$$

Also one using that F_{n-1} and F_{n-2} are the smallest in absolute value coefficients of F_n and F_{n+1} respectively in the equation

$$F_n F_{n-1} - F_{n-2}\,F_{n+1} = \gcd(F_n, F_{n+1}) = 1,$$

where F_i is the i-th Fibonacci number, one can show that there exists an infinite sequence of numbers satisfying (3.3).

Therefore (3.3) suggests that Frumkin's upper bound is invalid. The author in [29] proved that

$$\mathrm{IES}(BA) < \|A\|^{3^r} \text{ with } r = \operatorname{rank}(A) \qquad (3.4)$$

Hence the bounds (3.1) and (3.4) almost match and suggest that the classical algorithm is inefficient and non-polynomial. An upper bound on the complexity time of the classical algorithm of $O(9^r s^8)$ elementary operations, where s is the size of the matrix, is given by the author in [29].

The first polynomial time algorithm for the problem mentioned above was given by Kannan and Bachem in [33]. Chou and Collins in [8] improved the complexity bounds of the Kannan-Bachem algorithm given in [33], and, by slightly modifying their algorithm improved the upper bound on the magnitude on the entries over the transformation. From [8] one can derive an upper bound of $O(s^{11})$ elementary operations on an algorithm for transforming a matrix A of size s to a diagonal one. Moreover the growth of the entries at all steps of the Chou-Collins algorithm is bounded by $O(s^2)$.

An algorithm running asymptotically faster than the algorithms mentioned above is given below. Knowing that the computation of a multiple of the order is not hard, a modified form of the classical algorithm is applied to the matrix given by the proposition below.

PROPOSITION 3.1

Suppose that A is an $m \times n$ matrix over $\mathbb{Z}$ representing a finite abelian group G. If B is an $n \times n$ submatrix of A with rank (B) = n, then the $(m+n) \times n$ matrix

$$K = \begin{bmatrix} A \\ dI_n \end{bmatrix} \quad \text{with } d = \det(B)$$

represents a group isomorphic to G.

Proof

The matrix K merely represents a group with defining relations

$$R \cup \{x_i^d = 1,\ 1 \leq i \leq n\}$$

isomorphic to G, where R is the set of defining relations of G and d is a multiple of the order $|G|$. □

Kamman-Bachem in [33] refer that Wolsey, Hu, Frumkin et al., suggested the use of arithmetic modulo d in order to avoid IES; they gave an example of a group represented by the matrix

$$A = \begin{bmatrix} 5 & 26 \\ 2 & 11 \end{bmatrix}$$

whose determinant is 3. Taking the entries of A modulo 3, one can see that the matrix

$$\begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$$

represents an infinite abelian group non-isomorphic to the group represented by A. Despite the Kannan-Bachem worries that working modulo d over the matrix A is not always valid, this arithmetic modulo d has been employed and formulated in a correct way, in the procedures of the new algorithm.

<u>ALGORITHM 3.2</u>

<u>INPUT</u> : The matrix K of Proposition 3.1 and an integer ρ, $1 < \rho < n$

<u>OUTPUT</u>: A $(m+n) \times n$ matrix whose ρ-column is of the form $(k_{1\rho},\dots,k_{\rho\rho},0,\dots,0)^t$; this matrix is a transformation of K via IRC operations.

```
Procedure ELIMINATECOL (K,d,ρ)
  begin
    if $k_{\rho\rho} \neq 0$ then
      begin
1.      $r \leftarrow \gcd(k_{\rho\rho}, k_{\rho+1,\rho})$;
2.      compute integers $x_1, x_2$: $x_1 k_{\rho\rho} + x_2 k_{\rho+1,\rho} = r$ with
          $|x_1| < |k_{\rho+1,\rho}|/2$, $|x_2| < |k_{\rho\rho}|/2$;
3.      $y_1 \leftarrow k_{\rho+1,\rho}/r$;
4.      $y_2 \leftarrow -k_{\rho\rho}/r$;
5.      $K \leftarrow \left[\begin{array}{c|cc|c} I_{\rho-1} & \multicolumn{2}{c|}{0} & 0 \\ \hline & y_1 & y_2 & \\ 0 & x_1 & x_2 & 0 \\ \hline 0 & \multicolumn{2}{c|}{0} & I_{m+n-\rho-1} \end{array}\right] . K$;
        comment The result of step 5 is a matrix K having $k_{\rho\rho} = 0$
                and $k_{\rho+1,\rho} = r$.
                Note the $(m+n) \times (m+n)$ matrix is unimodular
      end
6.  $s \leftarrow \gcd(k_{\rho+1,\rho}, k_{\rho+2,\rho}, \ldots, k_{m\rho}, d)$;
7.  compute integers $t_j$ for $\rho+1 \leq j \leq m+1$: $\sum_{j=\rho+1}^{m} t_j k_{j\rho} + t_{m+1} d = s$;
      $t_j \leftarrow t_j \pmod d$ for $\rho+1 \leq j \leq m$:
      $t_{m+1} \leftarrow s - \sum_{j=\rho+1}^{m} t_j k_{j\rho}$;
8.  $ROW(\rho) \leftarrow ROW(\rho) + \sum_{j=\rho+1}^{m} t_j ROW(j) + t_{m+1} ROW(m+\rho)$;
    comment The entry $k_{\rho\rho}$ of K is equal to s. Note that the computation
            of $t_{m+1}$ is not necessary,since one can merely assign $k_{\rho\rho} = s$
```

```
9.  ROW(i) ← ROW(i) - ($k_{i\rho}$/s) ROW(i), for  $\rho+1 \le i \le m$;
    comment Now COL($\rho$) = ($k_{1\rho}$,...,$k_{\rho\rho}$,0,...,0)$^t$
10. ROW(i) ← ROW(i) - ($\lfloor k_{ij}/d \rfloor$) ROW(m+j), $1 \le i \le m$, $1 \le j \le n$;
    comment At this step all the entries of K are reduced mod d.
    return K;
end. □
```

REMARK 3.3

One may speed up (in practice) the above procedure by making the following modifications :

(i) At the beginning of the procedure ELIMINATECOL one can check if there exists a $k_{i\rho}$ for some $\rho \le i \le m$ such that $k_{i\rho}$ divides $k_{j\rho}$ for every $\rho \le j \le m+\rho$ and if it exists, then interchange ROW(i) and ROW(ρ) and go to step 9.

(ii) At the beginning of the procedure one may check whether or not there exists a $k_{i\rho}$ for some $\rho \le i \le m$ such that $k_{i\rho} \neq 0$ and if it exists then interchange ROW(i) and ROW(ρ) and go to step 6.

An amount of computation may be saved with the above modifications but one can observe that the worst-case complexity will remain asymptotically the same. □

REMARK 3.5

Suppose that the matrix K of Proposition 3.1 is transformed to K^* after an application of the procedure ELIMINATECOL (K,d,ρ). Since every IRC operation can be expressed as matrix multiplication of K by a unimodular matrix (Proposition 1.9), it is not difficult to modify the procedure to compute a unimodular square matrix L such that

$$LK = K^*. \quad \square$$

3.6 A WORKED EXAMPLE

Let

$$K = \begin{bmatrix} 26 & 38 & 60 \\ 12 & 18 & 30 \\ 36 & 54 & 102 \\ 22 & 12 & 48 \\ 144 & 0 & 0 \\ 0 & 144 & 0 \\ 0 & 0 & 144 \end{bmatrix} \quad \text{with } |d| = |\det(B)| = \left|\det\left(\begin{bmatrix} 26 & 38 & 60 \\ 12 & 18 & 30 \\ 36 & 54 & 102 \end{bmatrix}\right)\right| = 144$$

Now we apply ELIMINATECOL (K, 144, 1).

First one computes $x_1 = 1$, $x_2 = -2$ such that $1.26 + (-2)12 = \gcd(26,12)$.

Moreover $y_1 = 6$ and $y_2 = -13$

At step 5 we have that

$$K \leftarrow \left[\begin{array}{cc:c} 6 & -13 & \\ 1 & -2 & 0 \\ \hdashline 0 & & I_5 \end{array}\right] . K = \begin{bmatrix} 0 & -6 & -30 \\ 2 & 2 & 0 \\ 36 & 54 & 102 \\ 22 & 12 & 48 \\ & 144I_3 & \end{bmatrix}$$

At step 6 we have that s=2 and at step 7 one can compute $t_1=1$, $t_2=0$, $t_3=0$ and $t_4 = 0$ such that

$$1.2 + 0.36 + 0.22 + 0.6 = \gcd(2,36,22) = 2.$$

At step 8 we have

$$K = \begin{bmatrix} 2 & -4 & -30 \\ 2 & 2 & 0 \\ 36 & 54 & 102 \\ 22 & 12 & 48 \\ & 144I_3 & \end{bmatrix}$$

At step 9 we have

$$K = \begin{bmatrix} 2 & -4 & -30 \\ 0 & 6 & 30 \\ 0 & 126 & 642 \\ 0 & 56 & 378 \\ & 144I_3 & \end{bmatrix}$$

At step 10 we have

$$K = \begin{bmatrix} 2 & -4 & -30 \\ 0 & 6 & 30 \\ 0 & 126 & 66 \\ 0 & 56 & 90 \\ & 144I_3 & \end{bmatrix}$$

which is the output matrix. □

<u>PROPOSITION 3.7</u>

The procedure ELIMINATECOL terminates in

$$O(mnM(\log(m\,||K||)) + m\,M(\log ||K||)\log\log ||K||)$$

elementary operations in the worst-case and the size of the output matrix is $O(m+n+\log ||K||)$.

<u>Proof</u>

Steps 1-2 require $O(M(\log ||A||)\log\log ||A||)$ elementary operations for an application of the E.E.A. and by Theorem 0.3.3.

$$|x_1| < |k_{\rho+1,\rho}|/2, \quad |x_2| < |k_{\rho\rho}|/2 \qquad (1.1)$$

Steps 3-4 require $O(M(\log \|A\|)$ elementary operations for divisions.

In view of (1.1) step 5 requires at most 2n multiplications comprising $O(nM(\log \|A\|)$ elementary operations. If A^* denotes the matrix after step 5, then

$$\|A^*\| < \|A\|^2$$

Steps 6-7 require $O(mM(\log \|K\|)\ \mathrm{loglog}\ \|K\|)$ elementary operations for $O(m)$ applications of E.E.A. and

$$\|t_i\| := \max_i \{|t_i|\} < \|K\| \text{ .}$$

Step 8 requires at most mn multiplications comprising $O(mnM(\log \|K\|)$ elementary operations for divisions/multiplications.

Step 10 requires $O(mn\ M(\log(m\ \|K\|)))$ elementary operations.

The bound of the output matrix follows from step 10. □

REMARK 3.8

The dominant complexity of the procedure ELIMINATECOL is $O(mn\ M(\log(m\ \|K\|)))$, except in the case of enormous $\|K\|$;

$$\|K\| > 2^{2^{n\log m + \varepsilon}} \text{ . } \square$$

REMARK 3.9

One can show that a modification of the procedure ELIMINATECOL for computing the matrix L defined in Remark 3.5 can be done in such a way that the worst-case complexity is increased only by a constant factor and

$\log \|L\| = O(\log \|K\|)$. □

<u>ALGORITHM 3.10</u>

<u>INPUT</u> : The integer matrix K of Proposition 3.1 and an integer $\rho : 1 < \rho < m$

<u>OUTPUT</u>: An $(m+n) \times n$ matrix whose ρ-row is of the form $(k_{\rho 1}, \ldots, k_{\rho\rho}, 0, \ldots, 0)$; this matrix is a transformation of the matrix K via IRC operations.

(Note that procedure ELIMINATEROW is almost symmetric with the procedure ELIMINATECOL; steps 9 and 11 are the only non-symmetries)

```
Procedure ELIMINATEROW ( K,d,ρ)
  begin
    if  $k_{\rho\rho} \neq 0$ then
     begin
1.     $r \leftarrow \gcd(k_{\rho\rho}, k_{\rho,\rho+1})$;
2.     compute $x_1, x_2 : x_1 k_{\rho\rho} + x_2 k_{\rho,\rho+1} = r$ with $|x_1| < |k_{\rho,\rho+1}|/2, |x_2| < |k_{\rho\rho}|/2$;
3.     $y_1 \leftarrow k_{\rho,\rho+1}/r$;
4.     $y_2 \leftarrow -k_{\rho\rho}/r$;
       $K \leftarrow K \left[\begin{array}{c|cc|c} I_{\rho-1} & 0 & & 0 \\ \hline 0 & y_1 & x_1 & 0 \\ & y_2 & x_2 & \\ \hline 0 & 0 & & I_{n-\rho-1} \end{array}\right]$
5'.    $k_{m+\rho,\rho} \leftarrow d$; $k_{m+\rho+1,\rho+1} \leftarrow d$; $k_{m+\rho+1,\rho} \leftarrow 0$; $k_{m+\rho,\rho+1} \leftarrow 0$;
       comment The above operation can be expressed as a sequence
       of row operations.  See comment below.
     end
```

```
6.  $s \leftarrow \gcd(k_{\rho,\rho+1},\dots,k_{\rho n})$;

7.  Compute $t_j$ for $\rho+1 < j < n$: $\sum_{j=\rho+1}^{n} t_j k_{\rho j} = s$;

    $t_j \leftarrow t_j \bmod d$ for $\rho+1 < j < n$;

8.  $COL(\rho) \leftarrow COL(\rho) + \sum_{j=\rho+1}^{n} t_j\, COL(j)$;

9.  $ROW(m+j) \leftarrow ROW(m+j) - t_j ROW(m+1)$, for $\rho+1 < j < n$;
    comment This step is not necessarily executed, since one can
          merely assign $k_{m+\rho+1,\rho} = \dots = k_{m+n,\rho} = 0$.

10. $COL(j) \leftarrow COL(j) - (k_{\rho j}/s)\, COL(\rho)$, $\rho+1 < j < n$;
11. $ROW(m+\rho) \leftarrow ROW(m+\rho) - \sum_{j=\rho+1}^{m} (k_{\rho j}/s)\, ROW(m+j)$;

    comment This step is not necessarily executed, since one can
          merely assign $k_{m+\rho,\rho+1} = \dots = k_{m+\rho,n} = 0$
12. $ROW(i) \leftarrow ROW(i) - (\lfloor k_{ij}/d \rfloor) ROW(m+j)$, $\rho < i < n$, $\rho < j < m$;
    return $K$;
end. □
```

<u>Comment</u>

If M is the inverse of $\begin{bmatrix} y_1 & x_1 \\ y_2 & x_2 \end{bmatrix}$, then one can see that

$$\left[\begin{array}{c:c:c} I_{m+\rho-1} & 0 & 0 \\ \hdashline 0 & M & 0 \\ \hdashline 0 & 0 & I_{m+n-\rho-1} \end{array}\right] . K \text{ is equivalent to step 5' and thus it can be}$$

expressed as a sequence of row operations. □

REMARK 3.11

One may speed up the above procedure by modifying it in a similar way as procedure ELIMINATECOL was suggested to be modified at Remark 3.3. □

REMARK 3.12

Suppose that K^* is the output matrix of ELIMINATEROW (K,d,ρ). Then it is not difficult to modify ELIMINATEROW in order to compute two unimodular matrices L and R such that

$$LKR = K^*. \quad □$$

PROPOSITION 3.13

The procedure ELIMINATEROW terminates in

$$O(mn\, M(\log(m\,\|K\|)) + n\, M(\log \|K\|)\log\log \|K\|)$$

elementary operations and the size of the output matrix is $O(m+n + \log \|K\|)$.

Proof

It is similar to Proposition 3.7. □

REMARK 3.14

One can show that the computation of the matrices L and R of Remark 3.12 can be done without any asymptotic increase in the worst-case complexity of the procedure ELIMINATEROW and

$\log(\max\{\|L\|, \|R\|\}) = O(\log \|K\|)$. □

3.15 A WORKED EXAMPLE

Let

$$K = \begin{bmatrix} 26 & 38 & 60 \\ 12 & 18 & 30 \\ 36 & 54 & 102 \\ 22 & 12 & 48 \\ & 144I_3 & \end{bmatrix} \text{ with } |d| = \det\left(\begin{bmatrix} 26 & 38 & 60 \\ 12 & 18 & 30 \\ 36 & 54 & 102 \end{bmatrix}\right) = 144.$$

In order to eliminate the 1-st row of K one can apply ELIMINATEROW (K,d,1). Then at step 2 one can find $x_1 = -16$ and $x_2 = 11$ such $(-16).26 + 11.38 = \gcd(26,38)$. Moreover $y_1 = 19$ $y_2 = -13$.

At step 5 we have

$$K \leftarrow K \left[\begin{array}{cc:c} 19 & -16 & \\ -13 & 11 & 0 \\ \hdashline & 0 & I_5 \end{array}\right] = \left[\begin{array}{ccc} 0 & 2 & 60 \\ -6 & 6 & 30 \\ -18 & 18 & 102 \\ 262 & -220 & 48 \\ 144.19 & -144.16 & 0 \\ -144.13 & 144.11 & 0 \\ 0 & 0 & 144 \end{array}\right]$$

which is equivalent to the matrix

$$\begin{bmatrix} 0 & 2 & 60 \\ -6 & 6 & 30 \\ -18 & 18 & 102 \\ 262 & -220 & 48 \\ & 144I_3 & \end{bmatrix}$$

Now at step 6 one can compute $t_1 = 1$, $t_2 = 0$ such that

$$1.2 + 0.60 = \gcd(2,60)$$

At step 7 we have

$$\begin{bmatrix} 2 & 2 & 60 \\ 0 & 6 & 30 \\ 0 & 18 & 102 \\ 42 & -220 & 48 \\ 144 & 0 & 0 \\ 144 & 144 & 0 \\ 144 & 0 & 144 \end{bmatrix} \xrightarrow{\text{step 8}} \begin{bmatrix} 2 & 2 & 60 \\ 0 & 6 & 30 \\ 0 & 18 & 102 \\ 42 & -220 & 48 \\ \multicolumn{3}{c}{144I_3} \end{bmatrix}$$

At step 10 we have

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 6 & 30 \\ 0 & 18 & 102 \\ 42 & -262 & -1212 \\ 144 & -144.2 & -144.30 \\ 0 & 144 & 0 \\ 0 & 0 & 144 \end{bmatrix} \xrightarrow{\text{step 11}} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 6 & 30 \\ 0 & 18 & 102 \\ 42 & -262 & -1212 \\ \multicolumn{3}{c}{144I_3} \end{bmatrix}$$

and at step 12 we have

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 6 & 30 \\ 0 & 18 & 102 \\ 42 & -118 & -60 \\ \multicolumn{3}{c}{144I_3} \end{bmatrix}$$

which the output matrix. □

ALGORITHM 3.16

INPUT : An $m \times n$ matrix representing a finite abelian group G

OUTPUT: The canonical structure of the group G

```
begin
1.  d ← the determinant of an $n \times n$ non-singular submatrix of A;
2.  K ← $\begin{bmatrix} A \\ dI_n \end{bmatrix}$;
3.  for $\rho$ = 1 to n do
      begin
        repeat
5.          ELIMINATEROW (K,d,$\rho$);
6.          ELIMINATECOL (K,d,$\rho$);
7.      until either $k_{\rho\rho} \mid k_{\rho i}$ for all $\rho + 1 < i < n$ or $k_{\rho\rho} = 0$
        $k_{\rho i} \leftarrow 0$ for $\rho + 1 < i < n$;
8.    end
    If $k_{ii} = 0$ for some i then interchange ROW(i) and ROW(m+i);
    A ← the n top rows of K;
    comment The only non-zero elements of A are the diagonal entries.
9.  for p = 1 to n do
      begin
10.     for q = p+1 to n do
          begin
            h ← $a_{pp}$;
            $a_{pp} \leftarrow \gcd(a_{qq}, h)$;
            $a_{qq} \leftarrow a_{qq} . h/a_{pp}$;
11.     end
12. end
end. □
```

PROPOSITION 3.17

Algorithm 3.16 correctly computes a canonical basis for the abelian group G in

$$O(mn(n + \log\log \|K\|)\log \|K\| M(\log \|K\|) + mn^2 M(\log|d^*|))$$

elementary operations, where $d^* = \max \{|d^*| : d^*$ determinant of an $n \times n$ submatrix of A}.

Proof

The computation of a multiple of the order of the group (steps 1-2) requires $O(mn^2 M(\log |d^*|)$ elementary operations using Gaussian elimination.

Steps 5 and 6 require $O(m(n+\log\log \|K\|)M(\log \|k\|))$ elementary operations from Propositions 3.7 and 3.13 and using the facts that

$$m > n \text{ and } \|K^{(i)}\| < d < \|K\| \quad \forall i$$

where $K^{(i)}$ denotes the matrix K at the i-th iteration of the loop 4-7. The number of iterations required by loop 4-7 is at most $\log |d|$, since

$$|k_{pp}^{(p)}| < |k_{pp}^{(i-1)}|/2$$

Hence loop 3-8 requires $n\log |d| = n \log \|K\|$ iterations. (3.1)
Loop 9-12 required $O(n^2)$ applications of EEA and $O(n^2)$ multiplications/divisions comprising $O(n^2M(\log|d|)\log\log|d|) = O(n^2M(\log \|K\|)\log\log \|K\|)$ elementary operations.

From the above analysis the proposition follows. □

COROLLARY 3.18

Algorithm 3.16 terminates in

$$O(mn^2[n + \log(n \log(n \|A\|))] \log(n \|A\|) M(n \log(n \|A\|)))$$

elementary operations.

Proof

The result follows from Proposition 3.17 and using the fact that

$$\max \{d^*, \|K\|\} < \max\{\|A\|, |d|, |d^*|\} < (n \|A\|)^n. \quad \square$$

COROLLARY 3.19

If a finite abelian group is represented by a matrix of size s, then one can compute its canonical structure in $O(s^5 M(s^2))$ elementary operations. □

Chou and Collins in [8] propose an algorithm for computing the canonical structure of a finite (or infinite) abelian group by means of their LDSMKB algorithm and the algorithm SNF given by Kannan-Bachem [33]. The Chou-Collins LDSMKB algorithm for triangularization of an integer matrix requires $O(mn^3[n+n \log n \|A\|)]^2$ elementary operations. The Kannan-Bachem algorithm (named SNF) requires $n^2 \log(n \|A\|)$ applications of a triangularization algorithm. Hence the computations of the structure can be done for the proposed Chou-Collins algorithm in

$$O(mn^5 \log(n \|A\|)[n + n \log (n \|A\|)]^2 = O(s^{11}) \qquad (3.2)$$

elementary operations. Therefore the upper bound (3.2) has been improved by a factor of $O(s^3)$.

This section closes by computing the canonical structure of the group represented by the matrix

$$\begin{bmatrix} 5 & 26 \\ 2 & 11 \end{bmatrix}$$

(Kannan-Bachem example - see below Proposition 3.1).

$$K = \begin{bmatrix} 5 & 26 \\ 2 & 11 \\ 3 & 0 \\ 0 & 3 \end{bmatrix} \xrightarrow{\text{ELIMINATEROW}} \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 3 & 0 \\ 0 & 3 \end{bmatrix} \xrightarrow{\text{ELIMINATECOL}} \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 3 & 0 \\ 0 & 3 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$$

Therefore the group is cyclic of order 3.

4.THE COMPUTATION OF THE HERMITE AND SMITH NORMAL FORMS OF AN INTEGER MATRIX

A. Hermite Normal Form

THEOREM 4.1 (Hermite, see [23])

Given a non-singular $n \times n$ integer matrix A, there exists an $n \times n$ unimodular matrix M such that MA = T is upper triangular with positive diagonal elements. Furthermore, each off-diagonal element of T is non-positive and strictly less in absolute value than the diagonal element in its row. □

DEFINITION 4.2

The matrix T of Theorem 4.1 is called the *Hermite normal form* (abbreviated HNF) of the matrix A. □

It is not difficult to modify ELIMINATECOL (K,d,p) to obtain a procedure ELIMINATECOL* (K,d,p) of the same asymptotic complexity, for eliminating all entries below the diagonal element of the column of K.

ALGORITHM 4.3

INPUT : An $n \times n$ non-singular matrix

OUTPUT: The HNF matrix T of A and a unimodular matrix M such that

MA = T

```
begin
1.  $d \leftarrow \det(A)$;
2.  $K \leftarrow \begin{bmatrix} A \\ dI_n \end{bmatrix}$;
3.  for i = 1 to n do
      begin
        ELIMINATECOL* (K,d,i);
4.    end
    $T^* \leftarrow [k_{ij}]$ for $1 < i, j < n$;
    comment The matrix T* is upper triangular.  Note that det (T*) = d.
    $K \leftarrow \begin{vmatrix} T^* \\ dI_n \end{vmatrix}$
5.  ROW(i) ← -ROW(i) for each $k_{ii} < 0$;
    comment  The diagonal entries are positive.  In the next loop the
    entries to the right of the diagonal will be reduced .
6.  for i = 1 to n do
      begin
7.      for j = 1 to i-1 do
          begin
            ROW(j) ← ROW(j) - $\lfloor k_{ij}/k_{ii} \rfloor$ ROW(i);
          end
8.    end
    $T \leftarrow [k_{ij}]$ for $1 < i, j < n$
10. solve the system $X \cdot T = A$;
    comment It is not hard to solve the above system, since T is
    triangular .
11. $M \leftarrow X^{-1}$;
    comment  Use the algorithm given by Proposition 0.3.5.
end. □
```

PROPOSITION 4.4

Algorithm 4.3 correctly computes the HNF the matrix A and the unimodular matrix M.

Proof

There exists an $n \times n$ integer matrix W such that

$$WA = dI_n$$

Let $L_1 = \begin{bmatrix} I_n & 0 \\ W & I_n \end{bmatrix}$, then

$$L_1 \begin{bmatrix} A \\ 0_n \end{bmatrix} = K \qquad (4.1)$$

Using Remark 3.5 there exists a $2n \times 2n$ matrix L_2 such that

$$L_2K = \begin{bmatrix} T^* \\ 0 \end{bmatrix} \qquad (4.2)$$

From Proposition 1.9 there exists a $2n \times 2n$ unimodular matrix L_3 such that

$$L_3 \begin{bmatrix} T^* \\ 0 \end{bmatrix} = \begin{bmatrix} T \\ dI_n \end{bmatrix} \qquad (4.3)$$

Let $L = L_3L_2L_1$. Let $\begin{bmatrix} \Lambda_1 & \Lambda_2 \\ \Lambda_3 & \Lambda_4 \end{bmatrix}$ be a partition of L, where Λ_i for $1 \leq i \leq 4$ is an $n \times n$ matrix.

Then from (4.1), (4.2) and (4.3) follows that

$$L \begin{bmatrix} A \\ 0_n \end{bmatrix} = \begin{bmatrix} T \\ dI_n \end{bmatrix}$$

Hence $\Lambda_1 A = T$. The determinant of T is d, since T has the same diagonal elements as T^* and $\det(T^*) = d$. Hence $\det(\Lambda_1) = 1$.

PROPOSITION 4.4

Algorithm 4.3 correctly computes the HNF the matrix A and the unimodular matrix M.

Proof

There exists an $n \times n$ integer matrix W such that

$$WA = dI_n$$

Let $L_1 = \begin{bmatrix} I_n & 0 \\ W & I_n \end{bmatrix}$, then

$$L_1 \begin{bmatrix} A \\ 0_n \end{bmatrix} = K \qquad (4.1)$$

Using Remark 3.5 there exists a $2n \times 2n$ matrix L_2 such that

$$L_2 K = \begin{bmatrix} T^* \\ 0 \end{bmatrix} \qquad (4.2)$$

From Proposition 1.9 there exists a $2n \times 2n$ unimodular matrix L_3 such that

$$L_3 \begin{bmatrix} T^* \\ 0 \end{bmatrix} = \begin{bmatrix} T \\ dI_n \end{bmatrix} \qquad (4.3)$$

Let $L = L_3L_2L_1$. Let $\begin{bmatrix} \Lambda_1 & \Lambda_2 \\ \Lambda_3 & \Lambda_4 \end{bmatrix}$ be a partition of L, where Λ_i for $1 < i < 4$ is an $n \times n$ matrix.

Then from (4.1), (4.2) and (4.3) follows that

$$L \begin{bmatrix} A \\ 0_n \end{bmatrix} = \begin{bmatrix} T \\ dI_n \end{bmatrix}$$

Hence $\Lambda_1 A = T$. The determinant of T is d, since T has the same diagonal elements as T^* and $\det(T^*) = d$. Hence $\det(\Lambda_1) = 1$.

Moreover one can see that the matrix T satisfies all the conditions of Theorem 4.1 and thus T is the HNF of the matrix A. □

PROPOSITION 4.5

Algorithm 4.3 computes the NHF of A and the matrix M in

$$O(n^2[n + \log(n \log(n \|A\|))]M(n\log(n \|A\|)))$$

elementary operations. Moreover $\log \|M\| = O(n \log (n \|A\|))$.

Proof

The computation of the determinant requires $O(n^3M(\log d))$ elementary operations. Loop 3-4 require $O(n^2(n + \log\log \|K\|)M(\log \|K\|))$ elementary operations using Proposition 3.7. (See Remark above algorithm 4.3).

One can show easily that loop 6-9 requires $O(n^3)$ multiplications comprising $O(n^3M(\log d)$ elementary operations.

The computation of X is not difficult, since T is triangular. It requires $O(n^2)$ divisions comprising $O(n^2M(\log d))$ elementary operations.

The computation of the inverse of X requires $O(n^3M(\log d))$ elementary operations by Proposition 3.5.

Hence using that

$$\|K\| < \max \{\|A\| \; |d|\} < (n \|A\|)^n$$

the proposition follows. □

Chou-Collins in [8] gave an algorithm computing the HNF of a matrix A which requires $O(n^4[n + n \log(n \|A\|)]^2)$ elementary operations. Hence the upper bound is improved by a factor of $O(n^2)$.

COROLLARY 4.6

There exists an algorithm computing the HNF of a matrix A of sizes and the unimodular M of Theorem 4.1 in $O(s^3 M(s^2))$ elementary operations. □

The following corollary is given in order to show the optimality of algorithm 4.3 as direct method.

COROLLARY 4.7

Algorithm 4.3 is a direct method and it requires $O(n^3)$ multiplications in order to compute the HNF of an $n \times n$ matrix A. □

In [34] Klyuyev and Kokovkin-Shchebak proved that Gaussian elimination is an optimal direct method. Note that in general Gaussian elimination is not optimal (see [51]). The lower bound on the number of multiplications necessary to transform a matrix with entries from a field to a triangular matrix, is given below.

THEOREM 4.8 (Klyuyen-Kokovkin-Shchebak)

A direct method for triangularizing an $n \times n$ matrix with entries from a field requires exactly $\frac{1}{6} n(n+1)(2n+1)-n$ multiplications. □

Directly from Corollary 4.7 and Theorem 4.8 follows that

PROPOSITION 4.9

Algorithm 4.3 is optimal within a constant factor as direct method. □

B. Smith Normal Form

THEOREM 4.10 (H.J. Smith, see [49])

Given a non-singular $n \times n$ integer matrix A, there exists $n \times n$ unimodular matrices B and G such that $D = BAC$ is a diagonal matrix with positive diagonal elements such that $d_{11} \mid d_{22} \mid \cdots \mid d_{nn}$. □

DEFINITION 4.11

The matrix D of Theorem 4.10 is called the *Smith normal form* (abbreviated SNF) of the matrix A. □

PROPOSITION 4.12

The SNF of an $n \times n$ matrix A can be computed in

$$O(n^3 \log(n \,\|A\|)[n + \log(n \log(n \,\|A\|))]M(n \log(n \,\|A\|)))$$

elementary operations.

Proof

It readily follows from Corollary 3.18. □

REMARK

Using Proposition 3.17 one can have a expression of the upper bound of Proposition 4.12 in terms of $\|K\| = \max\{\|A\|, \det(A)\}$; the upper bound is the same as that of Proposition 3.17. □

An upper bound of $O(n^6 \log(n\|A\|)\,[n + n\log(n\|A\|)]^2)$ on the computational complexity of SNF was given by Chou-Collins in [8] (see (3.2)). Therefore this upper bound is improved by at least a factor of $O(n^3)$.

In order to give an algorithm computing the matrices B and C of Theorem 4.10, the following problem is considered:

FROBLEM 4.13

Suppose that A is an $n \times n$ non-singular integer matrix with determinant d. Compute an integral solution of the system

$$XA = dI_n \tag{4.4}$$

where X is an $n \times n$ matrix of unknown variables. □

PROPOSITION 4.14

There exists an algorithm computing a particular solution for (4.4) in

$$O(n^2[n + \log(n\log(n\|A\|))]M(n\log(n\|A\|)))$$

elementary operations. Moreover $\log\|X\| = O(n\log(n\|A\|))$.

Proof

Using algorithm 4.3, one can compute a matrix T, the HNF of A^t and a matrix M such that

$$MA^t = T$$

Hence the system (4.4) is equivalent to

$$TX^t = M\cdot(dI_n)$$

whose solution is easily computed, since T is triangular.

The complexity of the method follows from the analysis in Proposition 4.5. □

Proposition 4.15

There exists an algorithm computing the matrices B and C of Theorem 4.10 in

$$O(n^{\alpha+2} \log(n \|A\|)\log(n \log(n \|A\|))M(n \log(n \|A\|)))$$

elementary operations, where $O(n^{\alpha})$ is an upper bound on the number of multiplications required for multiplication of two $n \times n$ matrices.

Proof

One can use the following method of computing the matrices B and C.

(i) Compute the matrix D, the SNF of A, using algorithm 3.14. From Remarks 3.5, 3.11 one can see that algorithm 3.14 can be modified to yield unimodular matrices $L_1,\dots,L_\lambda$ of dimension $2n \times 2n$ and $R_1,\dots,R_\mu$ of dimension $n \times n$ such that

$$L_\lambda \cdots L_1 \begin{vmatrix} A \\ dI_n \end{vmatrix} R_1 \cdots R_\mu = \begin{vmatrix} 0 \\ dI_n \end{vmatrix} \qquad (4.5)$$

(ii) Compute an integral solution of the system

$$XA = dI_n$$

using the algorithm given by Proposition 3.14. Moreover let

$$L^* = \begin{bmatrix} I_n & 0 \\ X & I_n \end{bmatrix}$$

(iii) Compute the matrices $L = \prod_{i=1}^{\lambda} L_i \cdot L^*$ and $R = \prod_{i=1}^{\mu} R_i$.

The computation can be done by multiplying in the list $\Lambda = \{L_1, \ldots, L_\lambda, L^*\}$ pairwise from left to right so to obtain a new list $\{L_1L_2, L_3L_4, \ldots\}$; then repeating this process until a list with a single element viz. the product $L_1L_2 \ldots L_\lambda L^*$ is obtained

(iv) Let $L = \begin{bmatrix} \Lambda_1 & \Lambda_2 \\ \Lambda_3 & \Lambda_4 \end{bmatrix}$,

where Λ_i for $1 < i < 4$ are matrices of $n \times n$ dimension. Then let $B = \Lambda_1 + \Lambda_2 X$ and $C = R$.

The correctness follows from the facts that

$$BAC = D$$

and since C is unimodular and $|\det(A)| = |\det(D)|$, the matrix B is unimodular.

The complexity required for the computation of L_i's, R_i's and X is given by Propositions 3.15 and 4.14.

The computation of L and R requires

$$\sum_{i=1}^{\sigma} v/2^i \quad \text{with } v = \max\{\lambda,\mu\},\ \sigma = \lceil \log v \rceil$$

matrix multiplications comprising

$$O(n^{\alpha} \sum_{i=1}^{\sigma} [(v/2^i)\, M(2^i \log \|Q\|)])$$

elementary operations, where $\|Q\| = \max_{i,j} \{ \|L_i\|, \|R_j\|, L^*\}$.

Now from Remarks 3.9, 3.14 and (3.1) follows that

$$v < n \log d < n^2 \log(n\|A\|)$$

and $\|Q\| < \max\{\|A\|, d\} < (n\|A\|)^n$.

Hence the running time of (iii) requires
$O(n^{\alpha+2} \log(n\|A\|) \log(n \log(n\|A\|) M(n \log(n\|A\|))$ elementary operations. □

<u>COROLLARY 4.16</u>

There exists an algorithm for computing the matrices B and C of Theorem 4.10 in $O(s^{\alpha+3} \log s\, M(s^2))$ elementary operations, where s is the size of A. □

<u>REMARK</u>

Using the remark below Proposition 4.12 and Proposition 4.15 one can derive an expression of the upper bound of Proposition 4.15 in terms of $\|K\| = \max\{\|A\| \, |\det(A)|\}$, which is

$O(n^{\alpha+1} \log \|K\| \log \|K\| M(\log \|K\|))$ elementary operations. □

Now the following problem whose solution will be used for the computation of the canonical structure of an infinite abelian group and for the solution of a system of linear Diophantine equations, is considered.

PROBLEM 4.17

Suppose that A is an $m \times n$ integer matrix with rank n. Compute a unimodular $(m+n) \times (m+n)$ matrix L such that

$$L \begin{bmatrix} A \\ 0_n \end{bmatrix} = \begin{bmatrix} T \\ 0_n \end{bmatrix} \quad (4.6)$$

where T is an upper triangular $n \times n$ matrix. □

ALGORITHM 4.18

INPUT : The matrix A of Problem 4.17

OUTPUT: A matrix L satisfying (4.6)

```
begin
  $B \leftarrow$ an $n \times n$ non-singular submatrix of A;
  $d \leftarrow \det(B)$;
  find an integral solution of the system: $XA = dI_n$;
  $L^* \leftarrow \begin{bmatrix} I_n & 0 & 0 \\ 0 & I_{m-n} & 0 \\ X & 0 & I_n \end{bmatrix}$;
  $K \leftarrow \begin{bmatrix} A \\ dI_n \end{bmatrix}$;
  for i = 1 to n do
```

```
    begin
        ELIMINATECOL* (K,d,i);
        Let $L_i$ be the matrix L defined in Remark 3.5;
    end
    $L \leftarrow \prod_{i=1}^{n} L_i \cdot L^*$;
end. □
```

PROPOSITION 4.19

Algorithm 4.18 correctly computes L in

$$O(n[n^{\alpha}\log n + mn + n^2\log(n \log(n \|A\|))]M(n \log(n \|A\|)))$$

elementary operations, where $O(n^{\alpha})$ denotes an upper bound on the number of multiplications required for multiplication of two $n \times n$ matrices. Moreover

$$\log \|L\| = O\ (n^2 \log (n \|A\|).$$

Proof

The correctness of the algorithm is obvious.

The time required for the computation of d and B is given by Proposition I.2.4. . Using Proposition 3.7, one can find the time required for an application of the procedure ELIMINATECOL*.

The computation of the matrix L requires

$$\sum_{i=1}^{[\log n]} (n/2^i)$$

matrix multiplications comprising

$$O(n^{\alpha}\sum_{i=1}^{[\log n]}(n/2^i)M(2^i\log\|K\|)) = O(n^{\alpha+1}\log n\, M(n\log(n\|A\|)))$$

elementary operations. Moreover from Remark 3.9 and Proposition 4.14

$$\max\{\|L_i\|\} = O(\|K\|) \text{ and } \|L^*\| = O(\|K\|)$$

hence $\log\|L\| = O(n\log\|K\|) = O(n^2\log(n\|A\|))$. □

<u>REMARK 4.20</u>

Suppose that L and A are as in Problem 4.18. Let

$$L = \begin{bmatrix}\Lambda_1 & \Lambda_2\\ \Lambda_3 & \Lambda_4\end{bmatrix}$$

where Λ_1 is an $m \times m$ matrix, Λ_2 is an $m \times n$ matrix, Λ_3 is an $n \times m$ matrix and Λ_4 is an $n \times n$ matrix.

Then

$$\Lambda_1 A = \begin{bmatrix} T \\ 0_{m-n}\end{bmatrix}$$

but Λ_1 is not necessarily unimodular. The author does not know of an efficient way of computing a unimodular matrix Λ such that

$$\Lambda A = \begin{bmatrix} T \\ 0_{m-n}\end{bmatrix}$$

and this will lead to some inelegancies in the presentation of algorithms for computing the structure of infinite abelian groups and for solving systems of linear Diophantine equations. □

<u>REMARK 4.21</u>

The algorithm of Proposition 4.17 and the algorithm of Proposition 4.19 make use of a "fast matrix multiplication" algorithm; Coppersmith's and Winograd's algorithm (see Proposition 0.3.4 and Remark below) for matrix multiplication is faster than the obvious way for matrix multiplication, only for matrices with very large numbers of rows(columns); therefore for practical purposes for the above two algorithms is suggested the use of the usual way of multiplying matrices.

5. A DIRECT METHOD FOR COMPUTING THE STRUCTURE OF INFINITE ABELIAN GROUPS

Suppose that A is an $n \times n$ integer matrix representing an infinite abelian group G. Then the matrix A is singular. In order to transform A to diagonal form, one may use the following algorithm.

ALGORITHM 5.1

INPUT : An $m \times n$ singular matrix A representing the infinite abelian group G.

OUTPUT: The canonical structure of G

```
begin
1.  $r \leftarrow \mathrm{rank}(A)$
2.  $B \leftarrow$ an $r \times r$ nonsingular submatrix of A;
3.  $A \leftarrow \begin{bmatrix} B & C \\ D & E \end{bmatrix}$;
    comment The matrix D is an $(n-r) \times r$ matrix.  C is an $r \times (n-r)$ matrix
    and E is an $(m-r) \times (n-r)$ matrix.  The transformation is done with
    column and row interchanges.

    $A \leftarrow \begin{bmatrix} A \\ \hline 0_r \quad 0 \end{bmatrix}$;
    comment Note that the addition of trivial relations to the set of
    defining relations of G does not change its structure.
4.  Compute a unimodular $(m+r) \times (m+r)$ matrix L such that:
    $L \begin{bmatrix} B \\ D \\ 0_r \end{bmatrix} = \begin{bmatrix} T \\ 0 \end{bmatrix}$, where T is an $r \times r$ upper triangular matrix;
    comment One can use algorithm 4.18, since rank $\left(\begin{bmatrix} B \\ D \\ 0_r \end{bmatrix}\right) = r$.
5.  $A \leftarrow LA$;
6.  Let $A = \begin{bmatrix} T & M \\ 0 & 0 \end{bmatrix}$;
7.  for $i = r$ down to 1 do
      begin
```

```
8.  $COL(j) \leftarrow COL(j) - (\lfloor m_{ij}/t_{ii} \rfloor)\ COL(i)$ for $r+1 < j < n$;
9.  end
10. $A^* \leftarrow [T,M]^t$
    comment  $A^*$ is an $n \times r$ matrix with rank $r$ and thus represents an
    finite abelian group, say $G^*$.
11. compute the canonical structure of $G^*$ using algorithm 3.15;
end. □
```

PROPOSITION 5.2

Algorithm 5.1 correctly computes the canonical structure of G.

Proof

(i) All steps are expressed in terms of IRC operations which respect the structure of the group G.

(ii) In step 6 the bottom right corner $(m-r) \times (m-r)$ submatrix of A is a matrix where all entries are zeros, because if its j-th column had a non-zero entry for some j, then A would have r+1 linearly independent columns (the first r columns of A and the j-th of the submatrix) which contradicts the fact that $r = \text{rank}(A)$.

(iii) In step 10, A^* represents the finite group G^*. Let G' be the maximal finite subgroup of G. It will be shown that G^* is isomorphic to G'. Let $A_1 = [T,M]$. Since A_1 represents G, there exists unimodular matrices L and R such that:

$$L\,A_1 R = D \tag{5.1}$$

where D represents the canonical structure of G. Moreover

$$G' = G(d_{11}) \times \dots \times G(d_{rr}) \tag{5.2}$$

From (5.1) follows that

$$R^t A_1^t L^t = D^t \text{ or } R^t A^* L^t = D^t$$

and D^T represents the canonical structure of G^*. Therefore

$$G^* = G(d_{11}) \times \ldots \times G(d_{rr})$$

and using (5.2) follows that $G^* \approx G'$. □

<u>PROPOSITION 5.3</u>

Algorithm 5.1 computes the canonical structure of G in

$$O(r[r^{\alpha}+mr+nr\log(r\|A\|)(r+\log(r\log(r\|A\|)))]M(r\log(r\|A\|)$$

$$+ mn\,M(r^2\log(r\|A\|)) + r^2 n\,M(r^3\log(r\|A\|)))$$

elementary operations.

<u>Proof</u>

Step 1-2 requires $O(mnr\,M(\log|d^*|)$ elementary operations by Proposition 2.4, where $|d^*| = \max\{|d|: \text{determinant of an } r \times r \text{ submatrix of } A\}$.

The running time of step 4 is given by Proposition 4.19.

Step 5 requires $O(mn)$ multiplications comprising $O(mn\,M(r^2\log(r\|A\|)))$ elementary operations, using that $\log\|L\| = O(r^2\log(r\|A\|))$ by Proposition 4.19. Loop 7-9 requires $O(r^2 n)$ multiplications comprising $O(r^2 n\,M(r^2\log(r\|A\|)))$ elementary operations.

One can observe that

$$\|A^*\| \leq |d| < (r\,\|A\|\,)^r \qquad (5.3)$$

From Proposition 3. 16 and (5.3) one can derive the running time of step 11 comprising $O(nr^2 \log(r\,\|A\|\,)[r+\log(r \log(r\,\|A\|\,))]M(r\log(r\,\|A\|\,)))$ elementary operations.

The proposition follows from the fact that $|d^*| < (r\,\|A\|\,)^r$. □

<u>COROLLARY 5.4</u>

There exists an algorithm computing the canonical structure of an infinite abelian group G represented by a matrix A of size s, in $O(s^5M(s^2))$ elementary operations.

<u>Proof</u>

From Proposition 5.3 and using the fact that $\alpha < 3$ ($\alpha = 3$ in the classical algorithm for matrix multiplication). □

The Chou-Collins algorithm for computing the structure of finite groups is discussed below. Corollary 3.19 can be used for infinite abelian group as well. Hence the (3.2) upper bound of $O(s^{11})$ in the case of infinite groups has been also improved by at least a factor $O(s^3)$.

6. A DIRECT METHOD FOR THE SOLUTION OF SYSTEMS OF LINEAR DIOPHANTINE EQUATIONS

Let A be an $m \times n$ matrix with integer entries and b an $n \times 1$ vector with entries from the integers. Then the system of equations

$$Ax = b, \; x \in Z^{n \times 1} \qquad (6.1)$$

is called a *system of linear Diophantine equations.*

The computation of a solution or all (if any) of the system (6.1) is closely related to the triangularization of the matrix A. If the matrix A of (6.1) is of rank n, then (6.1) has exactly one solution or none and this can be found by means of Gaussian elimination. In the case in which the rank r of A is less than n, the system (6.1) has an infinite number of solutions (n-r linearly independent solutions) or the system is inconsistent. In this case the classical algorithm for solution of a system of linear Diophantine equations makes use of the classical triangularization algorithm (Smith [48], Bradley [5]) and therefore has the problem of "intermediate expression swell".

The first polynomial algorithms for solution of (6.1) were given by Frumkin in [17] (see also [16]) in some special cases. Frumkin's algorithm for computing a *particular* solution of (6.1) or establishing that there is not one requires in worst-case $O(n^2m^2\log(n\,||A||)M(n\log(n\,||A||)) + n^2M(n\log(n\,||A||) + ||b||) = O(s^5M(s^2) + s^2M(s^*))$ where s is the size of the matrix A and s* the size of the vector b. Moreover Frumkin in [17] gave an algorithm for computing the set of all solutions (if any) of an *homogeneous system* of linear Diophantine equations (that is (6.1) with b = (0,...,0)) which requires $O(n^3m\log(n\,||A||)M(m\log m\log(m\,||A||))) = O(s^5M(s^2\log s))$ elementary operations.

The best known polynomial algorithm for solving a general system of linear Diophantine equations is given by Chou-Collins in [8]. Their algorithm requires $O(n^3(m+n)[n + r\log(r\,\|A\|\,)]^2 + r(m+n)\log\|b\|\,[n + r\log(n\,\|A\|\,)]) = O(s^8 + s^4 s^*)$ elementary operations for the computation of the set of all the solutions of (6.1) if any, where s is the size of A and s* is the size of the vector b.

An algorithm for solving (6.1) whose upper bound improves the Frumkin's and Chou-Collins' upper bounds is presented below:

<u>ALGORITHM 6.1</u>

```
  INPUT : The system of equations (6.1)
  OUTPUT: A set of all integral solutions of the system (6.1), if any.
 begin
1.r ← rank(A);
2.B ← an r × r non-singular submatrix of A;
3.compute unimodular matrices $L_1$, $R_1$ such that: $L_1AR_1 = \begin{bmatrix} B & C \\ D & E \end{bmatrix}$;
  comment The matrix D is an (m-r) × r matrix, C is an r × (n-r) matrix
  and is an (m-r) × (n-r) matrix. The transformation is done with column
  and row interchanges.
4.A ← $L_1AR_1$;
5.b ← $L_1 \cdot b$;
  A ← $\left| A \begin{matrix} 0_r \\ 0 \end{matrix} \right|$;
6.compute a unimodular (n+r) × (n+r) matrix L such that: $L[B,C,0_r]^t = \begin{bmatrix} T \\ 0_n \end{bmatrix}$;
  comment The matrix T is upper triangular. The (n+r) × r matrix $[B,C,0_r]^t$
  has rank r and therefore one may use algorithm 4.18 for the computation of L.
  $R_2 \leftarrow L^t$;
```

```
7.  $A \leftarrow AR_2$;
8.  $R \leftarrow R_1R_2$;
    $T^* \leftarrow T^t$
    Let $A = \begin{bmatrix} T^* & 0 \\ M & 0 \end{bmatrix}$;
9.  If the system $Az = b$ has not an integral solution then
        comment One can compute a solution (if any) easily, since $T^*$ is a
        lower triangular matrix;
        return "the system is inconsistent"
10. else
        let z be a solution;
11. $w \leftarrow R \cdot z$: let $w = (w_1,\dots,w_{n+r})^t$;
    $\bar{w} \leftarrow (w_1,\dots,w_n)^t$;
    comment The vector $\bar{w}$ is a particular solution of the system (6.1);
12. $R^* \leftarrow [r_{ij}]$, $r+1 < j < n+r$, $1 < i < n$;
13. return $\{x = \bar{w} + R^*(t_1,\dots,t_n)\}$;
end. □
```

PROPOSITION 6.2

Algorithm 6.1 correctly computes the set of all solutions of the system.

Proof

Let $\vec{x} = (x_1,\dots,x_{n+r})^t$ and $\vec{x} := R(y_1,\dots,y_r,t_1,\dots,t_n)^t + w$.

It will be shown that $A'\vec{x} = b$ where $A' := \left[A \middle| \begin{smallmatrix} 0_r \\ 0 \end{smallmatrix} \right]$ and that $y_1 = y_2 = \dots = y_r = 0$.

Using that $LA'w = LA'Rz = Lb$ one can show that

$$LA'\vec{x} = LA'R(\vec{y},\vec{t})^t + LA'w = \begin{vmatrix} T^* & 0 \\ M & 0 \end{vmatrix} (\vec{y},\vec{t})^t + Lb = Lb$$

and moreover $LA'R(\vec{y},\vec{t})^t = 0$

which implies $\vec{y} = 0$.

Hence $\vec{x}$ yields the set of all solutions of the system $A\vec{x} = b$ which is equivalent to (6.1) and thus it is not difficult to see that $\vec{x}$ is the set of all solutions of (6.1). □

PROPOSITION 6.3

Algorithm (6.1) computes the set of all integral solutions of (6.1) in $O(r[r^{\alpha}\log r + mr + r^2\log(r\log(r\,||A||\,))]M(r\log(r\,||A||\,)) + mn\,M(r^2\log(r\,||A||\,) + \log||b||\,))$ elementary operations.

Proof

Steps 1-4, 6-8 require

$$O(r[r^{\alpha}\log r + mr + r^2\log(r\log(r\,||A||\,))]M(r\log(r\,||A||\,)) + mn\,M(r^2\log(r\,||A||\,)))$$

elementary operations. Their analysis is the same as in Proposition 5.3 (steps1-5).

Step 5 requires $O(n\log||b||\,)$ elementary operations using the fact that the rows of L_1 are of the form $(0,\dots,0,1,0,\dots,0)$.

Steps 9-10 require $O(mn)$ divisions/multiplications comprising $O(mnM(r^2(\log(r\,||A||\,) + \log||b||\,)$ elementary operations, since if A^* denote the matrix A at step 8 then

$$||A^*|| = ||L_1AR_1R_2|| < ||L_1||\;||A||\;||R_1||\;||R_2|| < ||A||\,(r\,||A||\,)^{r^2}$$

using that $||L_1|| = ||R_2|| = 1$ and $||R_2|| < (r\,||A||\,)^r$ by Proposition 4.19

Step 11 requires $O(mn\ M(r^2 \log(r\ \|A\|\ ) + \log \|b\|\ )$ elementary operations for multiplications.

From the above analysis the proposition follows. □

<u>COROLLARY 6.4</u>

There exists an algorithm for computing the set of all the solutions, if any, of a linear Diophantine system (6.1) in

$$O(s^{\alpha+1} \log s\ M(s^2) + s^2 M(s^*))$$

elementary operations, where s is the size of A and s* the size of b.

<u>Proof</u>

From Proposition 6.3 using Winograd's (see [54]) result that

$$\alpha > 2. \quad □$$

Suppose that

$$x = \bar{w} + R^*(t_1,\dots,t_n)^t \qquad (6.2)$$

is a general solution of (6.1). If the elements of the set $S = \{\bar{w},\ COL_{R^*}(i), \forall i\}$ are linearly independent, then S is called a *basis* of the general solution of (6.1).

<u>PROPOSITION 6.5</u>.

There exists an algorithm for computing a basis for the solutions of (6.1) in

$$O(s^{\alpha+1} \log sM(s^2) + s^2 M(s^*))$$

elementary operations, where s is the size of A and s* the size of b.

Proof

One can compute (6.2) using algorithm 6.1. Then using algorithm one can triangularize R* in similar way with the triangularization of A by algorithm 6.1. If the triangular form of R* is

$$\left[T\,\middle|\,\begin{matrix}0_{m-r}\\0_{m-r}\end{matrix}\right]$$

then $x = \bar{w} + T(t_1,\dots,t_n)^t$

is a basis for all solutions. The analysis follows from Propositions 6.3 and 4.19. □

Using the Coppersmith's-Winograd's result that $\alpha = 2.49\ldots$ one can see that algorithm (6.1) requires $O(s^{5.49}\log^2 s\log\log s + s^2 s^*\log s^*\log\log s^*)$ elementary operations. Therefore the Frumkin's upper bounds on the computation of a particular solution and the computation of a general solution of homogeneous systems are improved by a factor at least $O(s)$. Moreover the Chou-Collins upper bound on the computation of a basis of all solutions of a general system is improved by at least a factor of $O(s^2)$.

This section closes with a result on systems over the ring $\mathbb{Z}_k$, which will be used in later sections.

PROPOSITION 6.6 (Hu, see [25])

Suppose that A and b of (6.1) have entries from $\mathbb{Z}_k$†. There exists an algorithm for solving (6.1) over $\mathbb{Z}_k$ in $O(t^3 M(\log k)$ elementary operations, where $t = \max\{n,m\}$. □

†Note that $\mathbb{Z}_k = \mathbb{Z}/k\mathbb{Z}$ with $k \in \mathbb{N}$.

7. AN ALTERNATIVE ALGORITHM FOR COMPUTING THE STRUCTURE OF ABELIAN GROUPS

In the previous sections, direct methods for computing the structure of abelian groups represented by a set of defining relations, are presented. In the case both finite and infinite abelian groups, the methods require $O(s^5M(s^2))$ elementary operations, where s is in the size representing the group. An algorithm† for the same problem is given below, which requires asymptotically the same time as the methods above. This algorithm is presented because of its simplicity and the fact that its worst-case complexity bound is better by a constant factor than the bounds of the direct method. The disadvantage of the algorithm is that it is a non-direct method and thus it is difficult to compute the multiplier-matrices corresponding to the transformation.

The proposed method for diagonalization of an integer matrix A is the following:

(i) Compute an $r \times r$ non-singular submatrix B of the matrix A, where r is the rank of A

(ii) Compute the determinant d of the matrix B

(iii) Use the classical algorithm for diagonalization of the matrix A applying arithmetic modulo d.

(iv) If the computed diagonal matrix is

$$\begin{bmatrix} d_1 & & & & & 0 \\ & \ddots & & & & \\ & & d_r & & & 0 \\ & & & 0 & & \\ & 0 & & & \ddots & 0 \end{bmatrix}$$

† Due to W.M. Beynon, J.D. Dixon and C.S. Iliopoulos.

then $d_i' = \gcd(d_i, d)$ for $1 \le i \le r$ are the components of the canonical structure of the group represented by A.

It is known that the order of a group G (finite part of G if G is infinite) is given by

$$|G| = \gcd\ \{\det(B) : B \text{ is an } r \times r \text{ submatrix of } A\} \qquad (7.1)$$

Since

$$|G| = \gcd_B\ \{\det(B) \bmod d,\ d\}$$

where B as in (7.1), the correctness of the method follows.

A formal way of describing the above method is the following:

<u>ALGORITHM 7.1</u>

```
  INPUT : An $m \times n$ integer matrix A
  OUTPUT: The canonical structure of the abelian group represented by A.
begin
  $r \leftarrow \text{rank}(A)$
  $B \leftarrow$ an $r \times r$ non-singular submatrix of A;
  $d \leftarrow \det(B)$;
  $K \leftarrow \begin{bmatrix} A \\ dI_n \end{bmatrix}$
  comment observe than rank(k) = n.
  diagonalize the matrix k using algorithm 3.16;
  $k_{ii} \leftarrow 0$ for $r < i \le n$;
end.
```

<u>COMMENT</u>

One can observe that the above method coincides with algorithm 3.16 when A represents a finite abelian group.

<u>PROPOSITION 7.2</u>

Algorithm 7.1 correctly computes the canonical structure of G in $O(s^5M(s^2))$ elementary operations, where s is the size of A. □

8. APPLICATIONS

The diagonalization of an integer matrix is shown to have an application:

A. computing the canonical structure of a finite or infinite abelian group represented by a set of defining relations.

B. computing the set of all solutions (or a particular solution or a basis for the general solution) if any, of systems of linear Diophantine equations.

One can find further applications in:

C. Geometry Of Numbers

Suppose that $a_1,\dots,a_n$ are linear independent vectors in $Z^{n \times 1}$ and

$$\Lambda(a_1,\dots,a_n) = \{x : x = \sum_{i=1}^{n} u_i a_i,\ u_i \in Z\}$$

Then Λ is a *lattice* with basis $\{a_1,\dots,a_n\}$.

Using HNF and SNF algorithms one can compute a triangular or a canonical basis for the lattice. Moreover one can compute the structure of a sublattice given the structure of the lattice. For further details see Cassels [7].

D. Matrix Theory

Suppose that A is a square matrix with entries from R[x] where R is a ring and R[x] is the ring of all polynomial with coefficients from R. The using the algorithms described in the previous sections one can compute the HNF and SNF of the matrix; this can be done using an algorithm computing the gcd of two polynomials (see Knuth [35]).

For further details see Gantmacher [19].

E. Linear Algebra

The method mentioned in the previous sections can be used for the computation of the invariant polynomials of the characteristic equations, the elementary divisors and the eigenvalues of a matrix with entries from a ring R.

The *characteristic equation* of matrix is said to be the equation

$$AX = \lambda X, \ \lambda \in R.$$

The *invariant polynomials* $L_1(\lambda),\ldots,L_r(\lambda)$ are the diagonal elements of the SNF of the matrix $A-\lambda I_n$ and det $|A-\lambda I_n| = \prod_i L_i(\lambda)$ is the *characteristic polynomial* of A.

One can factor $L_i(\lambda)$ in such a way that

$$L_i(\lambda) = p_1^{\alpha_{1i}} \ldots p_s^{\alpha_{1s}} \text{ for } 1 \le i \le r$$

where $\alpha_{ij} \in Z$ and p_i is a linear function[†] of λ for $1 \le i \le s$. Then the p_i's are called the *elementary divisors* of $A-\lambda I_n$ and the roots of the p_i's for $1 \le i \le n$ are the *eigenvalues* of A.

Therefore one can readily see the application of the SNF algorithm in the computation of the above values. For further details one can see Lancaster [37] and Gantmacher [19]. For factorization of polynomials see Aho et al [2].

F. System Theory

The use of the HNF and SNF algorithm is essential for the solution of *linear modular systems* in system theory and circuit theory. For details see Zadeh and Polak [55].

[†]That is $a\lambda + b$ for a, b in R

G. Integer Programming

An *integer linear programming problem* (abbreviated ILP) has the form:

$$\text{minimize } z = \sum_{i=1}^{n} a_{i1}x_i \geq 0$$

$$\text{subject to } \sum_{i=1}^{n} a_{ij}x_i = b_j \text{ for } 2 \leq j \leq m \qquad (8.1)$$

$$\text{and } \quad x_i \geq 0 \text{ integer for } 1 \leq i \leq n$$

The ILP is a well-known NP-complete problem see Garey-Johnson [20].

Dropping the constraint $x_i \geq 0$ for $1 \leq i \leq n$, then the problem has a polynomial time solution. One can solve the system

$$\left[\begin{array}{c|c} A & \begin{matrix} 1 \\ 0 \\ \vdots \\ 0 \end{matrix} \end{array}\right] (\vec{x},z)^t = \begin{bmatrix} 0 \\ \cdot b \end{bmatrix}, \ \vec{x},z \text{ integer}$$

using algorithm 6.1.

Further in order to obtain the minimal solution for z from the general solution

$$z = p_1t_1 + \ldots + p_rt_r$$

one can compute t_i's, for $1 \leq i \leq r$ such that

$$\sum_i t_ip_i = |\gcd\{p_1,\ldots,p_r\}|$$

using the EEA algorithm.

The SNF algorithm also has a standard application in the method of ILP solution described in Hu [25] p.325.

H. Algebraic Group Theory

For computations over ideals of rings, see Newman [42] .

CHAPTER II

COMPUTATIONAL PROBLEMS ON FINITE ABELIAN GROUPS REPRESENTED BY AN EXPLICIT SET OF GENERATORS

1. INTRODUCTION

In this chapter computational problems in abelian groups represented by a set of generators are investigated. The upper bounds on the time complexity of the algorithms presented here are polynomial in terms of the order of the group and exponential in terms of the size of the input.

In the construction and analysis of the algorithms presented here, certain assumptions on the representation of the group elements are done. It is assumed that every element of a group G has a binary representation of length at most $O(\log |G|)$. One can see that this convention is reasonable, since the $|G|$ elements of a group G can be assigned an integer of the set $\{1,2,3,\dots,|G|^c\}$ via an injective function, where c is a positive constant independent of G. It is also necessary to consider ξ the number of elementary operations required for a group operation. In Section 9 it is assumed that $\xi = O(\log^c |G|)$ for some positive constant c which does not depend on G, although in all other sections the time complexity bounds are functions of ξ. For example in Z_n^* an element can be represented with at most logn bits and a group operation requires only M(n) elementary operations. Similarly the above assumed bounds apply to permutation groups and to the form class group (see [47], [30] and [31]).

In Section 2 Shanks' algorithm for computing the order of a group element together with an algorithm for computing a power of a group element is presented. Moreover an algorithm for computing the order of a group element with better space complexity than Shanks' algorithm is presented (it is assumed that the order of the group is given).

In the third section an algorithm for computing a set of defining relations for a p-group $H^* = \langle H,x\rangle$, where H has a known basis, is presented; it requires $O(|H^*|^{1/2+\epsilon}\xi)$ elementary operations.

In Section 4, two algorithms for computing a basis for H^* given a set of defining relations are presented; both require polynomial time in terms of the size of the input.

In section 5 an algorithm for computing a basis for a finite abelian group G in $O(|G|^{1/2+\epsilon}\xi)$ elementary operations is given. This upper bound improves Savage's bound of $O(|G|^2\xi)$ (see [32])

In Section 6 a membership testing algorithm is given.

In Section 7 the problem of computing a basis for a subgroup represented by a set of generators, of a group with known basis is investigated. The existence of a polynomial time algorithm for this problem is proved.

In Section 8 algorithms for computing a basis for the union and intersection of two finite abelian groups F and G is investigated. An upper bound on their time complexity of $O((|F||G|)^{1/2}\xi)$ is proved.

In Section 9 the relative complexity of the problems mentioned above, the problem of factorization and primality testing is examined. A classification of the complexity of the problems is established. Moreover the role of the Extended Riemann Hypothesis in speeding algorithms and improving bounds is investigated.

In Section 10 some applications of the algorithms of this chapter are discussed.

2. THE COMPUTATION OF THE ORDER OF AN ELEMENT OF A GROUP

The procedure ORDER(x) given below is due to Daniel Shanks (see [47]); given an element x of a finite abelian group of unknown order, it computes the order of x by means of the "baby-giant step" strategy.

ALGORITHM 2.1

```
Procedure ORDER(x)
 begin
   k ← 0; h ← 0;
1. repeat
     k ← k+1
     $r \leftarrow \lceil 2^{k/2} \rceil$
2.   compute the set $L_1 = \{x^i : 0 \le i \le r\}$
3.   compute the set $L_2 = \{x^{jr} : 0 \le j \le r\}$
4.   Sort the set $L_1$;
5.   for each $\ell_2 \in L_2$ do
       begin
         if $\ell_2 = \ell_1$ for some $\ell_1 \in L_1$ then
           comment Testing whether or not $\ell_2 \in L_1$ one may use binary search.
           $L \leftarrow \{\ell_1 = \ell_2$ equivalent to $x^{i_\lambda} = x^{j_\lambda r}$ for some $i_\lambda, j_\lambda\}$ ;
6.     end
7.   $h \leftarrow \min \{|i_\lambda - j_\lambda r|\}$;
8. until $h \ne 0$;
   return h;
 end. □
```

PROPOSITION 2.2

The procedure ORDER correctly computes the order of the element x in $O(|x|^{1/2}(\log^2|x| + \xi))$ elementary operations.

Proof

The correctness of the procedure ORDER follows from the fact that it merely computes the order of x by means of direct search for matches of the form

$$x^i = x^{jr} \text{ with } i - jr \neq 0$$

and the order $|x|$ is the minimal $|i-jr| > 0$ deduced from them.

Let r_k denote r at the k-iteration of loop 1-8. The computation of the set L_1 and L_2 requires $O(r_k)$ group operations. Step 4 requires $O(r_k \log r_k)$ comparisons using "heap sort" (see Proposition I.1.1) comprising $O(r_k \log r_k \log |x|)$ elementary operations. Loop 5-6 requires at most r_k applications of "binary search" and therefore requires $O(r_k \log r_k)$ comparisons comprising $O(r_k \log r_k \log |x|)$ elementary operations, using Proposition I.1.2.

Now let the n-th iteration be the last one. Then

$$|x| \geq r_{n-1}^2$$

because $x^{jr-i} \neq 1$ for $1 \leq i, j \leq r_{n-1}$. Therefore

$$r_{n-1} = \lceil 2^{(n-1)/2} \rceil \leq |x|^{1/2} \text{ implies that } n = \log |x| - 1$$

Hence, from the above analysis the procedure requires

$$O(\sum_{i=1}^{n} r_k\xi + \log |x| \sum_{i=1}^{n} r_k \log r_k) = O(\sqrt{|x|}\,(\log^2|x| + \xi))$$

elementary operations. □

Sattler and Schnorr in [44] gave a probabilistic algorithm computing the order of an element of an abelian group G. The expected computational time is exactly the same as the time required by Shanks' algorithm. The space complexity of their algorithm requires a constant number of registers (strictly speaking it requires $O(\log |x|)$ bits of memory). Shanks' method requires $O(|x|^{1/2} \log^2 |x|)$ bits of memory as one can observe easily. Hence their probabilistic method is very practical, since problems of this size can overload the memory of a computer. Note that the order of the element since it is computed by the probabilistic algorithm given in [44] it has a certificate of correctness.

Furthermore an algorithm computing the order of an element x of an abelian group G, given x and $|G|$ is presented below; its computational time complexity is of $O(|G|^{1/2+\epsilon})$ elementary operations and its space complexity is of $O(\log |G|)$ bits of memory. One of the applications of the algorithm may be on computations in $\mathbb{Z}_p^*$, since it is known that $|\mathbb{Z}_p^*| = p-1$.

<u>ALGORITHM 2.3</u>

<u>INPUT</u> : An element x of a group G and the order $|G|$ of G

<u>OUTPUT</u> : The order h of x

```
begin
  factor $|G|$;
  comment Use the naive method of trial and error.
  Let $|G| = p_1^{\alpha_1} \dots p_k^{\alpha_k}$;
  $x_i \leftarrow x^{q_i}$ with $q_i = |G|/p_i^{\alpha_i}$ for $1 \le i \le k$;
  compute the minimal $\beta_i$ for $1 \le i \le k$ : $x_i^{p_i^{\beta_i}} = 1$;
  $n \leftarrow \prod_{i=1}^{n} p_i^{\beta_i}$;
end. □
```

PROPOSITION 2.4

Algorithm 2.3 correctly computes the order of x in $O(|G|^{1/2} M(\log |G|) + \log^2|x|\xi)$ elementary operations. Moreover its space complexity is $O(\log |G|)$ binary bits of memory. □

One may improve the time complexity of the algorithm by using a more sophisticated factorization algorithm requiring small memory space.

This section closes with an algorithm for computing a power of a group element. This "power algorithm" is referred in Hindu manuscript 200 B.C. (See Knuth [35])

ALGORITHM 2.5

```
Procedure POWER (x,k)
  begin
```

```
1.  compute $a_i \in \{0,1\}$: $k = \sum_{i=0}^{n} a_i 2^i$;
    $y \leftarrow x$; $z = 1$;
3.  for $i = 0$ to $n$ do
      begin
        if $a_i = 0$ then
          $y \leftarrow y^2$
4.  else
5.        $z \leftarrow zy$;
6.    end
    return $x^k := z$;
end. □
```

PROPOSITION 2.6

Algorithm 2.5 computes x^k in $O(\log k\ M(\log k) + \xi \log k)$ elementary operations.

Proof

Step 1 requires at most $\log k$ divisions comprising $O(\log k\ M(\log k))$ elementary operations. Moreover $n = O(\log k)$.

Using the fact that steps 3-4 require 2 group operations one can see that loop 2-6 requires $O(\log k)$ group operations. □

3. THE COMPUTATION OF A SET OF DEFINING RELATIONS FOR THE p-GROUP $< H,x >$

In this section the problem of computing a set R of defining relations for the p-group $H^* = <H,x>$ is investigated; given a basis for H, the orders of the basis elements, an element x and its order, compute the set R. The procedure DEFIREL presented below computes a set R as it is deduced by the following proposition:

PROPOSITION 3.1

Suppose that $H = <<b_1,\ldots,b_n>>$, $H^* = <H,x>$ are finite abelian p-groups. If $|b_i| = p^{\alpha_i}$ and $|x| = p^h$, then there exists an integer $0 < k \le h$ such that

$$R = \{x^{p^k} = \prod_{i=1}^{n} b_i^{\delta_i}\} \cup \{b_i^{p^{\alpha_i}} = 1 \text{ for } 1 \le i \le n\} \tag{3.1}$$

for some $0 \le \delta_i < p^{\alpha_i}$, $1 \le i \le n$, is a set of defining relations for H^* and p^k is the smallest possible exponent of x in this form of relations.

Proof

Since the b_i's are independent, the only existing relations of x and b_i's are of the form

$$x^{\gamma} = \prod_{i=1}^{n} b_i^{\theta_i} \quad \gamma,\ \theta_i \in Z \tag{3.2}$$

Let $\mu > 0$ be the exponent of x with the smallest value in one of the relations of the form (3.2)

$$x^{\mu} = \prod_{i=1}^{n} b_i^{\delta_i} \qquad \mu , \delta_i \in Z \qquad (3.3)$$

It will be shown that μ divides the exponent of x in every relation of the form (3.2).From (3.2), (3.3) can be deduced that

$$x^{\gamma-\lambda\mu} = \prod_{i=1}^{n} b_i^{\theta_i - \lambda\delta_i} \qquad (3.4)$$

for some $\lambda \in Z$ such that $0 \le \gamma - \lambda\mu < \mu$. If $\gamma-\lambda\mu > 0$, then the exponent $\gamma-\lambda\mu$ of x in (3.4) is positive and smaller than μ contradicting with the definiton of μ. Therefore $\gamma-\lambda\mu = 0$ and thus μ divides γ.

Now consider the relation

$$x^{p^h} = \prod_{i=1}^{n} b_i^{o}$$

Since μ divides p^h, it follows that $\mu = p^k$ for some $k > 0$. □

Any set of relations R* for H* in terms of b_i's is polynomial time reducible to the set R of Proposition 3.1. A method for constructing R from R* in polynomial (almost linear) time in terms of the size of R* is given below.

PROPOSITION 3.2

Given R* a set of defining relations for the abelian p-group $H^* = \langle\langle\langle b_1,\dots,b_n\rangle\rangle,x\rangle$, the order p^{α_i} of b_i for $1 \le i \le n$ and the order p^h of x, then there exists an algorithm for computing the set R of Proposition 3.1 in $O((m \log\log \|R^*\| + n)M(\log \|R^*\|))$ elementary operations, where m is the number of relations in R* and $\|R^*\|$ denotes the maximum absolute value of the exponent of b_i's and x in the relations of R*.

Proof

Suppose that $R^* = \{x^{h_i} = \prod_{j=1}^{n} b_j^{c_{ij}},\ 1 \le i \le m\}$.

Compute $s = \gcd(h_1,\dots,h_m)$ and $t = h_\lambda/s$ for some λ such that $h_\lambda \neq 0$. Then compute $\delta_j = c_{\lambda j}/t \bmod p^{\alpha_j}$ for $1 \le j \le n$. It is not difficult to see† that

$$R' = \{x^s = \prod_{i=1}^{n} b_i^{\delta_i}\} \cup \{b_i^{p^{\alpha_i}} = 1,\ 1 \le i \le n\}$$

is the required set†. The analysis follows from Proposition 0.3.3 and 0.3.2. □

The following procedure computes the set R of Proposition 3.1 by means of the "baby-giant step" strategy introduced in [47].

ALGORITHM 3.3

```
procedure DEFIREL ($b_1,\dots,b_n$, x, $p^{\alpha_1},\dots,p^{\alpha_n}$, $p^h$);
 begin
   $r_i \leftarrow \lceil p^{\alpha_i/2} \rceil$ for $1 \le i \le n$; $L \leftarrow \emptyset$;
1. compute the sets $L_{1i} = \{b_i^k : 0 \le k \le r_i\}$ for $1 \le i \le n$;
2. compute the sets $L_{2i} = \{b_i^{kr_i} : 0 \le k \le r_i\}$ for $1 \le i \le n$;
3. compute the set $L_3 = \{x^{p^k} : 0 \le k \le h\}$;
4. compute the set $L_1 = \{\psi_0\psi_1\cdots\psi_n : \psi_0 \in L_3,\ \psi_i \in L_{1i}$ for $1 \le i \le n\}$;
5. compute the set $L_2 = \{w_1w_2\dots w_n : w_i \in L_{2i}$ for $1 \le i \le n\}$;
6. sort the elements of the set $L_2$;
7. for each $\ell_1 \in L_1$ do
     begin
```

† One can see that the construction of R' from R* (as above) is equivalent to the transformation of the matrix associated with R* to the matrix associated with R' using ELIMINATECOL.

```
        if $\ell_1 \in L_2$ then
            $L \leftarrow \{\ell_1 = \ell_2$ for some $\ell_2 \in L_2\}$;
            comment To test whether or not $\ell_1 \in L_2$ one may use "binary search"
            The set L contains relations of the form

                $x^{p^k} \prod_{i=1}^{n} b_i^{\theta_i} = \prod_{i=1}^{n} b_i^{\phi_i r_i}$

            with $1 \le \theta_i, \phi_i \le r_i$ for $1 \le i \le n$ and $0 \le k \le h$
8.  end
    Let $\Lambda$ be the relation in L which x has the smallest exponent;
    $R \leftarrow \{b_i^{p^{\alpha_i}} = 1,$ for $1 \le i \le n\} \cup \{\Lambda\}$ ;
end. □
```

<u>PROPOSITION 3.4</u>

The procedure DEFIREL correctly computes a set of defining relations for $H^* = \langle H,x\rangle$ in $O(|H^*|^{1/2}[\log|H^*| \log^2|H^*| + \xi])$ elementary operations.

<u>Proof</u>

The correctness of DEFIREL follows from the fact that it merely computes the set R as it is given in Proposition 3.1 by means of direct search for a match

$$x^{p^\lambda} = \prod_{i=1}^{n} b_i^{\gamma_i} \text{ for some } 0 \le \lambda \le h$$

using that there exist integers ϕ_i, θ_i for $1 \le i \le n$ such that

$$\gamma_i = \phi_i r_i + \theta_i \text{ for } 1 \le i \le n$$

with $1 \le \phi_i, \theta_i \le r_i$.

The computation of the sets L_{1i} and L_{2i} requires $O(r_i)$ group operations for $1 \le i \le n$. Hence steps 1-2 require $O(\sum_{i=1}^{n} r_i) = O(|H^*|^{1/2})$ group operations. Step 3 requires just $h = O(\log |H^*|)$ group operations.

The computation of L_1 in step 4 requires

$$|L_1| = |L_3| \prod_{i=1}^{n} |L_{1i}| = h \prod_{i=1}^{n} r_i = O(|H^*|^{1/2} \log |H^*|)$$

group operations. Similarly step 5 requires $|L_2| = O(|H^*|^{1/2})$ group operations. Step 6 requires $O(|L_2|\log|L_2|) = O(|H^*|^{1/2} \log |H^*|)$ comparisons by Proposition 0.3.6 comprising $O(|H^*|^{1/2} \log^2|H^*|)$ elementary operations. Loop 7-8 requires $|L_2|$applications of the "binary search" algorithm comprising

$O(|L_2|\log|L_1|) = O(|H^*|^{1/2} \log^2|H^*|)$ comparisons, or $O(|H^*|^{1/2}\log^3|H^*|)$ elementary operations.

From the above analysis the proposition follows. □

3.5 A WORKED EXAMPLE

Let $H = \langle\langle 3,31\rangle\rangle \subseteq Z^*_{32}$ and $x = 15$. The order of 3, 31 and 15 is 8, 2 and 2 respectively. A set of defining relations for $H^* = \langle 3, 31, 15\rangle$ will be computed using DEFIREL $(H, 15, 2^3, 2,2)$.

First we have $r_1 = \lceil 2^{3/2} \rceil = 3$ and $r_2 = \lceil 2^{1/2} \rceil = 2$.

At step 1 one can compute

$L_{12} = \{3^0 = 1,\ 3^1 = 3,\ 3^2 = 9,\ 3^3 = 27\}$ and $L_{21} = \{31^0 = 1, 31^1 = 31, 32^2 = 1\}$

(W.l.o.g. let $L_{21} = \{31^0 = 1,\ 31^1 = 31\}$).

At step 2 one can compute

$L_{21} = \{3^{0.3} = 1,\ 3^3 = 27,\ 3^6 = 25,\ 3^9 = 3\}$ and $L_{22} = \{31^0 = 1,\ 31^2 = 1,\ 31^4 = 1\}$

(W.l.o.g. let $L_{22} = \{31^0 = 1\}$).

At step 3 one can compute $L_3 = \{15^{2^0} = 15,\ 15^{2^1} = 1\}$.

At step 4 one can compute

$L_1 = \{15^1 3^0 31^0 = 15,\ 15^1.3^1.31^0 = 13,\ 15^1.3^2.31^0 = 7,\ 15^1.3^3.31^0 = 21,$
$15^1.3^0.31^1 = 17,\ 15^1.3^1.31^1 = 19,\ 15^1.3^2.31^1 = 25,\ 15^1.3^3.31^1 = 12,$
$15^2.3^0.31^0 = 1,\ 15^2.3^1.31^0 = 3,\ 15^2.3^2.31^0 = 9,\ 15^2.3^3.31^0 = 27,$
$15^2.3^0.31^1 = 31,\ 15^2.3^1.31^1 = 29,\ 15^2.31^2 31^1 = 23,\ 15^2.3^3 31^1 = 5\}$

At step 5 one can compute

$L_2 = \{3^0 31^0 = 1,\ 3^3 31^0 = 27,\ 3^6.31^0 = 25,\ 3^4.31^0 = 3\}$

At step 6 the list L_2 is sorted and after loop 7-8 one can obtain

$L = \{15^1.3^2.31^1 = 3^6.31^0,\ 15^2.3^0.31^0 = 3^0.31^0, 15^2.3^1.31^0 = 3^9.31^0,$
$15^2.3^3.31^0 = 3^3.31^0\}$

Therefore $R = \{15^{2^0} = 3^4.31,\ 3^8 = 1,\ 31^2 = 1\}$ is a set of defining relations for $\langle 3,31,15 \rangle \subseteq Z_{32}^*$. □

4. THE COMPUTATION OF A BASIS FOR THE p-GROUP $< H,x >$

In this section the following problem is considered:

<u>PROBLEM 4.1</u>

Given the finite abelian p-groups $H = \ll b_1,\dots,b_n \gg$, $H^* = \langle H,x \rangle$, with $|b_i| = p^{\alpha_i}$ for $1 \le i \le n$ and $|x| = p^h$ for some prime p and a set R of defining relations for H^*, compute a basis for H^*. $\square$

In view of Proposition 3.2 every set of defining relations for H^* in terms of b_i's and x can be reduced to the set given by Proposition 3.1 in polynomial time (almost linear) in terms of the size of the input set. Hence w.l.o.g one may assume that the set R of defining relations given in Problem 4.1 is that of Proposition 3.1.

Two algorithms for computing a basis for H^* of Problem 4.1 are presented below. The first method is based on the matrix associated with the set of defining relations and makes use of the direct method presented in Chapter I, Section 3. The second method is based on case analysis and makes use of group properties.

<u>PROPOSITION 4.2</u>

There exists an algorithm for computing a basis for H^* of Problem 4.1 $O(\log |H^*|)^{4.41} \log\log |H^*| \, M(\log |H^*|) + \log^2 |H^*|\xi)$ elementary operations.

<u>Proof</u>

The set R of the defining relations for the group G is associated with the $(n+1) \times (n+1)$ matrix

$$M = \begin{bmatrix} p^{\alpha_1} & \ddots & 0 & 0 \\ 0 & & p^{\alpha_n} & 0 \\ -\delta_1 & \cdots & -\delta_n & p^h \end{bmatrix}$$

The order $|H^*|$ of H^* is given by

$$|H^*| = \det(M) = p^h \prod_{i=1}^{n} p^{\alpha_i} \qquad (4.1)$$

Using algorithm I.3.16 one can compute a diagonal matrix D which represents the canonical structure of H^*. Moreover using the algorithm of Proposition I.4.15 one can compute unimodular matrices B and C such that

$$BAC = D$$

Then it is not difficult to show that

$$b_i^* := b_1^{c_{i1}} \cdots\cdots b_n^{c_{in}} x^{c_{i,n+1}} \qquad 1 \le i \le n+1 \qquad (4.2)$$

is a basis for H^*. The computation of the b_i^*'s can be done by means of computing the $b_k^{c_{jk}}$'s using the "power algorithm" 2.5.

The time complexity of the method described above follows from Propositions I.3. 16, II.2.6 and remark below corollary I.4.16 using the facts (4.1) and that

$$n = O(\log |H^*|). \qquad \square$$

Another algorithm for Problem 4.1 is given below; its worst-case complexity time upper bound is shown to be better than the upper bound proved for the method of Proposition 4.2.

ALGORITHM 4.3

INPUT : A set of defining relations for H*

$$R = \{x^{p^k} = \prod_{i=1}^{n} b_i^{\delta_i},\ b_i^{p^{\alpha_i}} = 1 \text{ for } 1 \le i \le n,\ x^{p^h} = h\} \quad (4.3)$$

where x, b_i's are as in Problem 4.1

OUTPUT : A basis for H* = <H,x>

```
Procedure BASIS (H,x,R)
 begin
   case k of
     begin
1.   0:  return H* = <<b_1,...,b_n>>;
2.   h:  return H* = <<b_1,...,b_n,x>>;
     end
3.   $r \leftarrow \log_p (\gcd \{p^k, \delta_1, \ldots, \delta_n\})$;
4.   $\gamma_i \leftarrow -\delta_i/p^r$ for $1 \le i \le n$;
     $w \leftarrow p^{k-r}$;
5.   $u \leftarrow x^w \prod_{i=1}^{n} b_i^{\gamma_i}$;                    (4.4)
   if r = k then
6.    return  H* = <<b_1,...,b_n,u>>;
   else
```

```
        $t \leftarrow$ index $\{\gamma_t : \gcd(\gamma_t, p) = 1\}$;
        if $r = 0$ then
            $H' \leftarrow \langle\langle b_1,\dots,b_{t-1}, b_{t+1},\dots,b_n\rangle\rangle$;
7.          $R' \leftarrow \{x^{p^{k+\alpha_t}} = \prod_{i=1}^{n} b_i^{\delta_i p^{\alpha_t}}, b_i^{p^{\alpha_i}} = 1$ for $1 \le i \le n, i \ne t, x^{p^h} = 1\}$;
8.          BASIS $(H', x, R')$;
        else
            $H'' \leftarrow \langle\langle b_1,\dots,b_{t-1}, b_{t+1},\dots,b_n,u\rangle\rangle$ ;
9.          $\lambda \leftarrow \alpha_t + k - r$;
10.         compute an integral solution of the system:
                $\delta_i(p^{\lambda-k} - p^{-r}z) = z_i + \gamma_i z, z_t = 0, 1 \le i \le n$
11.         $R'' \leftarrow \{x^{p^\lambda} = u^z \prod_{i=1, i\ne t}^{n} b_i^{z_i}, b_i^{p^{\alpha_i}} = 1$ for $1 \le i \le n, i \ne t, x^{p^h} = 1\}$;   (4.5)
12.         BASIS $(H'', x, R'')$;
        end
    end. □
```

<u>PROPOSITION 4.4</u>

Algorithm 4.3 correctly computes a basis for H^*.

<u>Proof</u>

<u>Case $k = 0$</u> In this case

$$x = \prod_{i=1}^{n} b_i^{\delta_i}$$

therefore $x \in H = H^* = \langle\langle b_1, \ldots, b_n \rangle\rangle$.

<u>Case $k = h$</u>. In this case the set $\{b_1, \ldots, b_n, x\}$ has independent elements and thus correctly $H^* = \langle\langle b_1, \ldots, b_n, x \rangle\rangle$.

<u>Case $k = r$</u>. It is not difficult to show that $\{b_1, \ldots, b_n, u\}$ is a set of independent elements using the facts: (i) p^k is the smallest possible exponent of x in relations of the form (3.1) and (ii) $\{b_1, \ldots, b_n\}$ is a set of independent elements. Moreover $x \in \langle b_1, \ldots, b_n, u \rangle$, because

$$x = \prod_{i=1}^{n} b_i^{-\gamma_i} u$$

Hence $H^* = \langle\langle b_1, \ldots, b_n, u \rangle\rangle$.

<u>Case $r < k$</u>. In this case there exists an integer t such that

$$\gcd(\gamma_t, p) = 1$$

and thus there exist integers μ and v such that

$$\mu\gamma_s + vp^{\alpha_t} = 1.$$

Therefore $b_t = b_t^{\mu\gamma_s + vp^{\alpha_t}} = (u\, x^{-p^w} \prod_{\substack{i=1 \\ i \neq t}}^{n} b_i^{-\gamma_i}) \in \langle b_1, \ldots, b_n, u, x \rangle$ (4.6)

Moreover one can observe that the order of u is p^r.

<u>Subcase r = 0</u>. Then $u = 1$ and $H^* = \langle H', x\rangle$. The correctness of the computation of the set R' of the defining relations for H' follows from (4.3) and the fact that $|b_t| = p^{\alpha_t}$. Therefore it is sufficient to call recursively BASIS for a simpler problem, since $|H'| \le |H'|/p$.

<u>Subcase r ≠ 0</u>. As in the case $k = r$ it is not difficult to show that the set $\{b_1,\ldots,b_{t-1}, b_{t+1},\ldots,b_n,u\}$ has independent elements. From (4.6) follows that $H^* = \langle H'', x\rangle$. The set R" of the defining relations for H" is correctly computed since

$$\text{(i)}\ \lambda = \log_p(|H^*|/|H''|) = \log_p \frac{\prod_{i=1}^{n} |b_i|\, p^k}{\prod_{\substack{i=1\\ i\ne t}}^{n} |b_i| \cdot |u|} = \log_p(|b_t|p^k/|u|)$$

and (ii) Using (4.4) and (4.5) follows that

$$x^{p^\lambda - wz} = \prod_{i\ne t} b_i^{z_i+\gamma_i z}\, b_t^{\gamma_t z}$$

and by definition p^k divides $p^\lambda - wz$, hence using (4.3)

$$x^{p^k(p^{\lambda-k}-p^{-r}z)} = \prod_i b_i^{\delta_i(p^{\lambda-k}-p^{-r}z)} = \prod_{i\ne t} b_i^{z_i+\gamma_i z}\ b_t^{\gamma_t}$$

and thus follows the system of step 10.

Hence in this case suffices to call the procedure BASIS recursively for a simpler problem, since $|H''| \le |H|/p$. □

<u>PROPOSITION 4.5</u>

Algorithm 4.3 computes a basis for H^* in

$$O(\log^2 |H^*| \, (M(\log |H^*|) + \xi))$$

elementary operations.

<u>Proof</u>

Step 3 requires $O(nM(\log |H^*|) \log\log |H^*|)$ elementary operations for an application of the Extended Euclidean Algorithm using Proposition 0.3.3.

Step 4 requires $O(nM(\log |H^*|))$ elementary operations for divisions.

Step 5 requires $O(\log (w \prod_{i=1}^{n} \gamma_i)) = O(\log |H^*|)$ group operations for n+1 applications of the "power algorithm" by Proposition 2.5.

The solution of the system of step 10 requires $O(nM(\log |H^*|))$ elementary operations for divisions.

The recursive application of the procedure BASIS is done at most $\log |H^*|$ times, since

$$|H'| \leq |H|/p \text{ and } |H'| \leq |H|/p$$

From the above analysis and the fact that $n \leq \log |H^*|$ the proposition follows. □

5. THE COMPUTATION OF THE STRUCTURE OF A FINITE ABELIAN GROUP

An algorithm for computing the order and the complete structure of a finite abelian group G represented by a set of generators is presented below; it makes use of the procedures ORDER, DEFIREL and BASIS presented in the previous sections.

<u>ALGORITHM 5.1</u>

<u>INPUT</u> : A set of generators $\{g_1,\dots,g_n\}$ for the finite abelian group G

<u>OUTPUT</u>: The order, the complete structure of G and a set of basis elements for G.

```
begin
1. $\sigma_i \leftarrow$ ORDER $(g_i)$ for $1 \le i \le n$;
2. compute the set $P = \{p: p$ prime of $\sigma_i$ for some $1 \le i \le n\}$
   comment This can be done by factoring all $\sigma_i$'s using Shanks' method
   (see [47 ])
3. for each $p \in P$ do
     begin
       $\lambda_i \leftarrow$ the largest integer such that $p^{\lambda_i}$ divides $\sigma_i$, for $1 \le i \le n$;
4.     $x_i \leftarrow g_i^{w_i}$ with $w_i = \sigma_i/p^{\lambda_i}$ for $1 < i < n$,
       $H_0 \leftarrow \langle 1 \rangle$;
5.     for s = 1 to n do
         begin
6.         $R_s \leftarrow$ DEFIREL $(H_{s-1}, x_s)$
7.         $H_s \leftarrow$ BASIS $(H_{s-1}, x_s, R_s)$
8.         end
```

```
    return the order and the basis's elements of the p-component $G_p$ of G;
    comment This information is included in $H_n$; see Propositions 4.4 and 4.5.
9. end
 return $|G| = \prod_{p \in P} |G_p|$;
end. □
```

PROPOSITION 5.2

Algorithm 5.1 correctly computes the order and the structure of the group G in $O(n|G|^{1/2}(\log |G|)^3 (\log |G| + \xi))$ elementary operations.

Proof

Step 1 requires $O(n|G|^{1/2}(\log^2 |G| + \xi))$ elementary operations, using Proposition 2.2.

Step 2 requires $O(n|G|^{1/5+\epsilon})$ elementary operations for factorizations.

Step 4 requires only $O(n \log |G|)$ group operations using the "power algorithm" 2.5.

Step 6 requires $O(|G|^{1/2} \log^2|G| + \xi))$ elementary operations from Proposition 3.4.

Step 7 requires $O(\log^2|G| (M(\log |G| + \xi))$ elementary operations from Proposition 4.5.

Hence loop 5-8 requires $O(n|G|^{1/2}\log^2|G|(\log|G| + \xi))$ elementary operations and loop 3-9 requires $O(n|G|^{1/2} \log^3|G| (\log|G| + \xi))$ elementary operations, since

$$|P| = O(\log |G| / \log\log |G|).$$

From the above analysis the proposition follows. □

An algorithm for computing a basis for a group G in $O(|G|^2\xi)$ elementary operations due to Savage is mentioned in [32]. Hence one can see that the upper bound has been improved by at least a factor of $O(|G|)$.

6. MEMBERSHIP-INCLUSION TESTING

Given a generating set $<g_1,\ldots,g_n>$ for a finite abelian group G and an element $h \in H \supseteq G$, an algorithm for testing whether or not h belongs to G is presented below.

ALGORITHM 6.1

```
begin
  compute the order $|G|$ and a basis for every p-component† $G_p$ of G;
  comment The computation is done as in algorithm 4.1.
  Let $G_p = << b_{1p},\ldots,b_{n_p,p} >>$ for every p-component;
  $\beta \leftarrow$ ORDER(h);
  if $\beta \nmid |G|$ then
     return "h does not belong to G";
  Let $\beta = p_1^{\lambda_1} \ldots p_k^{\lambda_k}$, where $p_i$'s are distinct primes;
  comment For factorization of $\beta$ use the prime factor of $|G|$ which are known.
  $\alpha_i \leftarrow \beta/p_i^{\lambda_i}$ for $1 \le i \le k$;
  $x_i \leftarrow h^{\alpha_i}$ for $1 \le i \le k$;
  for i = 1 to k do
   begin
      search for integers $0 \le \delta_j < |b_{jp_i}|$ for $1 \le j \le n_{p_i}$ : $x_i = \prod_{j=1}^{n} b_{jp_i}^{\delta_j}$;
      comment This can be done using the "baby-giant step" strategy in a
      similar manner as in the procedure DEFIREL.
      If the required $\delta_j$'s do not exist then
         return "h does not belong to G";
   end
  return "h belongs to G";
end. □
```

† The p-Sylow subgroup

<u>PROPOSITION 6.2</u>

Algorithm 6.1 correctly decides whether or not h belongs to G in

$$O(|h|^{1/2} (\log |h| + G) + n|G|^{1/2} \log^3|G|(\log |G| + \xi))$$

elementary operations.

<u>Proof</u>

The correctness of the algorithm is implied from the following fact: There exist integers γ_i for $1 \leq i \leq k$ such that

$$a_1\gamma_1 + a_2\gamma_2 + \dots + a_k\gamma_k = 1 \tag{6.1}$$

Then it follows that

$$h = h^{\sum_{i=1}^{k} a_i\gamma_i} = \prod_{i=1}^{k} (h^{a_i})^{\gamma_i} = \prod_{i=1}^{k} x_i^{\gamma_i} \tag{6.2}$$

Hence h belongs to G if and only if x_i belongs to G_{p_i} for $1 \leq i \leq k$, using the fact that $x_i = h^{a_i}$ for $1 \leq i \leq k$.

The upper bound on the time complexity of algorithm 6.1 follows from Proposition 2.2, 3.4 and 5.2. □

Suppose that F and G are finite abelian groups, both subgroups of an abelian group and F and G, are represented by a set of generators. For testing whether or not either $F = G$ or $F \subset G$ or $G \subset F$, it is sufficient to test each generator for membership of the one group in the other. Hence using algorithm 6.1 and proposition 6.2 one can show the following proposition:

PROPOSITION 6.3

Suppose that $F = \langle f_1,\dots,f_k\rangle \subseteq H \supseteq G = \langle g_1,\dots,g_n\rangle$ are finite abelian groups. Then the equality-inclusion test can be done in

$$O((k+n)\sqrt{|F| + |G|}\,\log^3(|F| + |G|)\,[\log(|F| + |G| + \xi])$$

elementary operations. □

COROLLARY 6.4

Given a finite abelian group $G = \langle g_1,\dots,g_n\rangle$ and an element h in G, there exists an algorithm for computing an expression for h in terms of the generators g_i's in $O(|G|^{1/2}(\log^2|G| + \xi))$ elementary operations.

Proof

Here an additive notation for the group G is used. The computation of the required expression of h can be done by means of direct search for a match of the form

$$h = m_1g_1 + \dots + m_ng_n \qquad (6.3)$$

using the "baby-giant step" strategy. This requires $O(\mu^{1/2}\log\mu(\log\mu + \xi))$ elementary operations, with $\mu = \prod_{i=1}^{n} |m_i|$ (proof similar to that of Proposition 2.2).

Now it will be shown that there exist m_i's for $1 \le i \le n$ such that $\mu = O(|G|)$. Let the $n \times n$ triangular matrix

$$A = \begin{bmatrix} \alpha_{11} & \cdots\cdots & a_{1n} \\ & a_{22} & \vdots \\ 0 & \ddots & \vdots \\ & & a_{nn} \end{bmatrix}$$

be the associated matrix[†] with a set of defining relations for G in terms of g_i's.

If (6.3) holds, then it is implied that

$$A'\vec{g} := \begin{bmatrix} m_1 & \cdots & m_n \\ a_{11} & \cdots & a_{1n} \\ & \ddots & \vdots \\ 0 & & a_{nn} \end{bmatrix} \begin{bmatrix} g_1 \\ \vdots \\ \vdots \\ g_n \end{bmatrix} = \begin{bmatrix} h \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

One can reduce the m_i's of L.H.S. $(n+1) \times n$ matrix A' modulo a_{ii} for $1 \le i \le n$, via integer-row operations (see algorithm I.4.2). This corresponds to a multiplication of A' by a unimodular $(n+1) \times (n+1)$ matrix

$$L := \begin{bmatrix} 1 & & & \\ & 1 & 0 & \\ * & & \ddots & \\ * & * & & \ddots \\ \vdots & \vdots & & \\ * & * & \cdots * & 1 \end{bmatrix}$$

where * denotes an entry.

Then it follows that

$$LA'\vec{g} = L \begin{bmatrix} h \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} h \\ * \\ \vdots \\ * \end{bmatrix}$$

Hence if $(m_1', \ldots, m_n')$ denotes the first row of the matrix L A', then

[†]Since every matrix can be triangularized, w.l.o.g. A is assumed to be triangular. See proposition I.4.1.

$$h = m_1'g_1 + \dots + m_n'g_n$$

Moreover

$$|m_i'| \le |a_{ii}| \text{ for } 1 \le i \le n$$

which implies $\mu = \prod_{i=1}^{n} |m_i'| \le \prod_{i=1}^{n} |a_{ii}| = |G|$.

Therefore the proposition follows. $\square$

<u>REMARK 6.5</u>

One can see that using algorithm 6.1 one can compute an expression of h in terms of the elements of a basis (see (6.1), (6.2)). $\square$

7. THE STRUCTURE OF A SUBGROUP OF A GROUP WITH KNOWN STRUCTURE

In this section the following problem is considered:

<u>PROBLEM 7.1</u>

Given that F is a finite abelian group with a known complete basis and G is a subgroup of F generated by a set of generators whose elements are expressed in terms of the basis's elements of F, compute the complete structure of G. □

An algorithm for problem 7.1 which runs in polynomial time is presented below; it makes use of the procedure DEFIREL* given below analogous to the procedure DEFIREL of Section 3.

<u>ALGORITHM 7.2</u>

<u>INPUT</u> : The abelian group $H = \langle\langle h_1,\dots,h_n\rangle\rangle$ and $H^* = \langle H,h_{n+1}\rangle$ both subgroups of the group $F = \langle\langle f_1,\dots,f_m\rangle\rangle$, the orders $|f_i|$ for $1 \le i \le m$ and the relations $h_i = \prod_{j=1}^{m} f_j^{\gamma_{ij}}$ for $1 \le i \le n+1$.

<u>OUTPUT</u>: A set of defining relations for the group H^*

$$R = \{h_{n+1}^{\delta_{n+1}} = \prod_{i=1}^{n} h_i^{\delta_i}\} \cup \{h_i^{\alpha_i} = 1 \text{ for } 1 \le i \le n+1\}$$

where $\alpha_i = |h_i|$ for $1 \le i \le n+1$.

```
Procedure DEFIREL*($H, h_{n+1}, h_i = \prod_{j=1}^{n} f_j^{\gamma_{ij}}$ for $1 \le i \le n+1$, $|f_j|$ for $1 \le i \le m$)
  begin
    compute an integral solution of the system:
        $\sum_{i=1}^{n} x_i\gamma_{ij} = x_{n+1}\,\gamma_{n+1,j} + y_j|f_j|,\ 1 \le j \le m\quad x_i, y_i \in Z_{|F|}$;
    comment This computation is done using the algorithm of Proposition
    I. 6.6.
    Let $(\vec{x},\vec{y})^t = K.\vec{w}$ be the set of all the solutions with
    $K \in Z^{(n+m+1)\times(n+m+1-r)}$;
    comment The integer r is the rank of the system and $\vec{w}$ is a vector of
            variables
    compute $\theta_i$ for $1 < i < n+m+1-r$: $\sum_{i=1}^{n+m+1-r} k_{n+1,i}\theta_i = |\gcd(k_{n+1,1},\dots,$
                                                              $k_{n+1,n+m+1-r})|$;
    comment By valuing $\vec{w} := \vec{\theta}$ the $x_{n+1}$ is became minimal.
    $(\delta_1,\delta_2,\dots) \leftarrow K.\vec{\theta}$;
    $\delta_i \leftarrow \delta_i(\mathrm{mod}(h_i))$ for $1 \le i \le n$;
    $R = \{h_{n+1}^{\delta_{n+1}} = \prod_{i=1}^{n} h_i^{\delta_i}\} \cup \{h_i^{\alpha_i} = 1$ for $1 \le i \le n\}$
    comment Note that $|h_i| = \mathrm{lcm}_j\,\{|f_j|/\gcd(|f_j|,\gamma_{ij})\}$ for $1 \le i \le n$.
    return R;
  end. □
```

PROPOSITION 7.3

Procedure DEFIREL* correctly computes a set of defining relations for H* in $O(n+m)^3$ $M(\log |F|)$ elementary operations.

Proof

Computing x_i's for $1 \le i \le n+1$ such that

$$h_{n+1}^{x_{n+1}} = h_1^{x_1} \ldots h_n^{x_n}$$

is equivalent to compute solutions of the system

$$\prod_{j=1}^{m} f_j^{\sum_{i=1}^{n} \gamma_{ij} x_i} = \prod_{j=1}^{m} f_j^{\gamma_{jn+1} x_{n+1}}$$

which is equivalent to solving

$$\sum_{i=1}^{n} \gamma_{ij}\, x_i = \gamma_{jn+1}\, x_{n+1} + y_j |f_j| \quad 1 \le j \le m,\ x_i,\ y_j \in Z_{|F|}$$

Step 2 guarantees the minimality of x_{n+1} and thus the correctness proof is completed.

Step 1 requires $O([n+m]^3$ $M(\log|F|))$ elementary operations for the computation of the set of all the solutions of the system using Proposition I.6.6.

Step 2 merely requires $O([n+m]$ $M(\log|F|)$ $\log\log|F|)$ elementary operations for applying the E.E.A. and step 3 requires $O((n+m)^2$ $M(\log|F|))$ elementary operations.

Using the fact that $\log\log |F| < m$, the proposition follows. □

ALGORITHM 7.4

INPUT : A finite abelian group $F = \langle\langle f_1,\dots,f_n\rangle\rangle$ $|f_i|$ for $1 \le i \le n$, a subgroup $G = \langle g_1,\dots,g_k\rangle$ and a set of relations

$$g_j = \prod_{i=1}^{n} f_i^{\gamma_{is}} \text{ for } 1 \le j \le k.$$

OUTPUT: The complete structure of G and a complete basis for G.

```
begin
  for each p-component of F do
   begin
     $H \leftarrow \langle g_1^{(p)}\rangle$
     comment The sets $\{g_1^{(p)},\dots,g_k^{(p)}\},\{f_1^{(p)},\dots,f_n^{(p)}\}$ are the generating
     sets for the p-components of G and F respectively
     for $2 \le i \le k$ do
       begin
         $R_i \leftarrow$ DEFIREL*$(H_p, g_1^{(p)}, g_j^{(p)} = \prod_{i=1}^{n} (f^{(p)})^{\gamma_{is}}$ for $1 \le j \le i, |f_i^{(p)}|\ \forall i)$;

         $H_p \leftarrow$ BASIS$(H_p, g_i, R_i)$;
       end
     return $H_p$;
   end
end. □
```

PROPOSITION 7.5

Algorithm 7.4 correctly computes a complete basis for G in

$$O(k(\log|F|)^3\, M(\log|F|) + \log^2|G|\xi)$$

elementary operations.

<u>Proof</u>

The correctness of the algorithm follows from the correctness proofs for the procedures DEFIREL* and BASIS.

The upper bound on the time complexity of the algorithm, follows from Propositions 3.4 and 7.3, since

$$m \le \log |G| \text{ and } n \le \log |F|$$

where m denotes the number of the basis elements for G. □

8. THE COMPUTATION OF THE STRUCTURE OF THE UNION AND INTERSECTION OF ABELIAN GROUPS

A. UNION OF FINITE ABELIAN GROUPS

Suppose that the finite abelian groups $F = \langle f_1,\dots,f_m\rangle$ and $G = \langle g_1,\dots,g_n\rangle$ subgroups of the same group are given. If $\langle F \cup G\rangle$ is an abelian group, then for the computation of its order and its structure, it is sufficient to apply algorithm 4.1 for $H = \langle f_1,\dots,f_m, g_1,\dots,g_n\rangle$. Since $|H| = O(|F|\ |G|)$, the computation of a basis for $\langle F \cup G\rangle$ requires $O(m+n)\ \sqrt{|F||G|}$.

B. INTERSECTION OF FINITE ABELIAN GROUPS

In case which $\langle F \cup G\rangle$ is abelian, the computation of the order of the group $\langle F \cap G\rangle$ can be done by computing the orders of the groups F,G and $\langle F \cup G\rangle$ and then

$$|\langle F \cap G\rangle| = |F| \cdot |G| / |\langle F \cup G\rangle|$$

The computation of the structure of the group $F \cap G$ (and its order in the case which $\langle F \cup G\rangle$ is non-abelian) is apparently difficult since it seems that finding a generating S for $F \cap G$ is hard. The computation of S and consequently its complete structure can be done using the following algorithm.

ALGORITHM 8.1

```
INPUT  :  Two finite abelian groups $F = <f_1,\ldots,f_m>$ and
          $G = <g_1,\ldots,g_n>$ with $<F \cup G>$ abelian group.
OUTPUT:   The complete structure and a complete basis for $<F \cap G>$
begin
1. compute a basis $<<b_1,\ldots,b_n>>$ for F using algorithm 5.1;
2. compute a basis $<<\gamma_1,\ldots,\gamma_\lambda>>$ for G using algorithm 5.1;
3. compute $x_i$'s, $z_i$'s: $b_1^{x_1} \ldots b_k^{x_k} = \gamma_1^{z_1} \ldots \gamma_\lambda^{z_\lambda}$ with $|x_i| \le |b_i|$ and
                                                        $|z_j| < |\gamma_j|$;    (8.1)
   comment This is done using the "baby-giant step" strategy.
4. $S \leftarrow \{s: s = b_1^{x_1} \ldots b_n^{x_n}$ for every match in (8.1)$\}$;
5. compute a basis for $<F \cap G> = <S>$ using algorithm 7.4.;
end. □
```

PROPOSITION 8.2

Algorithm 8.1 correctly computes the complete structure and the order of $F \cap G$ in $O((|F|\ |G|)^{1/2+\epsilon})$ elementary operations.

Proof

Step 3 requires $O(\sqrt{|F|\ |G|}\ (\log|F|\ |G|) + \xi))$ elementary operations for the computation of the matches (8.1)[†] and it is not difficult to see that

$$|S| = O(\sqrt{|F|\ |G|})$$

Then the upper bound on the complexity of the algorithm follows from propositions 5.2 and 7.3. □

[†]Using that $|x_i| \le |b_i|, \forall_i$ and $|z_j| \le |\gamma_j|, \forall_j$, follows that the number of matches of the form (8.1) is equal to

$$\prod_i |b_i| \prod_j |\gamma_j| = O\ (|F|\ |G|)$$

It has been shown that the steps required by the "baby-giant step" strategy is equal the square root of the number of matches.

PAGE
MISSING

ALGORITHM 8.1

```
INPUT  :  Two finite abelian groups $F = <f_1,\dots,f_m>$ and
          $G = <g_1,\dots,g_n>$ with $<F \cup G>$ abelian group.
OUTPUT:   The complete structure and a complete basis for $<F \cap G>$
begin
1. compute a basis $<<b_1,\dots,b_n>>$ for F using algorithm 5.1;
2. compute a basis $<<\gamma_1,\dots,\gamma_\lambda>>$ for G using algorithm 5.1;
3. compute $x_i$'s, $z_i$'s: $b_1^{x_1} \dots b_k^{x_k} = \gamma_1^{z_1} \dots \gamma_\lambda^{z_\lambda}$ with $|x_i| \le |b_i|$ and
                                                   $|z_j| < |\gamma_j|$;    (8.1)
   comment This is done using the "baby-giant step" strategy.
4. $S \leftarrow \{s: s = b_1^{x_1} \dots b_n^{x_n}$ for every match in (8.1)$\}$;
5. compute a basis for $<F \cap G> = <S>$ using algorithm 7.4.;
end. □
```

PROPOSITION 8.2

Algorithm 8.1 correctly computes the complete structure and the order of $F \cap G$ in $O((|F|\ |G|)^{1/2+\epsilon})$ elementary operations.

Proof

Step 3 requires $O(\sqrt{|F|\ |G|}\ (\log|F|\ |G|) + \epsilon))$ elementary operations for the computation of the matches (8.1)† and it is not difficult to see that

$$|S| = O(\sqrt{|F|\ |G|})$$

Then the upper bound on the complexity of the algorithm follows from propositions 5.2 and 7.3. □

†Using that $|x_i| \le |b_i|, \forall_i$ and $|z_j| \le |\gamma_j|, \forall_j$, follows that the number of matches of the form (8.1) is equal to

$$\prod_i |b_i| \prod_j |\gamma_j| = O(|F|\ |G|)$$

It has been shown that the steps required by the "baby-giant step" strategy is equal the square root of the number of matches.

9. RELATIVE COMPLEXITY OF THE PROBLEMS IN CHAPTER II

A. Computing The Order Of An Element And Factoring An Integer

Comparison of the complexities of computing two distinct functions can be done formally as follows.

DEFINITION 9.1

Suppose that f and g are functions; f is called *polynomial time reducible* to g, if there exists a Turing machine which on inputs n and g(n) computes f(n) in $O(\log^c n)$ steps for some constant c depending on f and g. The relation is denoted by

$$f <_p g$$

If $f <_p g$ and $g <_p f$, then f is called *polynomial time equivalent* to g denoted $f \approx_p g$. □

The functions which are investigated in this subsection are the following:

DEFINITION 4.2

Suppose that h is the exponent of $\mathbb{Z}_n^*$. Then define the function

$$\theta(n) := h^{\lceil \log n \rceil}$$

and let $\Theta(n)$ denote the number of elementary operations required by an optimal algorithm for computing $\theta(n)$.

Moreover define the function

$$f^*(n) := (p_1,\dots,p_k)$$

where the p_i's are all the distinct prime divisors of n and let F(n) denote the number of elementary operations required by an optimal algorithm for factoring the integer n. $\square$

Two useful facts about the multiplicative group of integers mod n are given in the following two theorems; both assume the truth of the Extended Riemann Hypothesis.

THEOREM 9.3 (E.R.H) Ankeny-Montgomery (see [41])

There exists an absolute constant $c > 0$ such that if

$$\psi: \mathbb{Z}_n^* \to G \text{ , where } G \text{ is a finite abelian group}$$

is a non-trivial group homomorphism, then there exists a prime p such that

$$\psi(p) \neq 1 \text{ and } p \in [2, c(\log n)^2]. \quad \square$$

THEOREM 9.4 (E.R.H), Dixon [13]

There exists a polynomial time algorithm for computing a generating set for $\mathbb{Z}_n^*$.

Proof

Let B be the set of all primes in the interval $[2, c\log^2 n]$. It will be shown that $\mathbb{Z}_n^* = \langle B \rangle$.

Let $G = \mathbb{Z}_n^*/\langle B \rangle$ and ψ the canonical homomorphism

$$\psi: \mathbb{Z}_n^* \to G.$$

The kernel of ψ is $\langle B\rangle$ and thus

$$\psi(q \bmod n) = 1 \qquad \forall\ q \text{ in } B.$$

By Theorem 9.3 ψ is trivial and thus $G = 1$ and $Z_n^* = \langle B\rangle$. □

Two well-known functions are the following:

DEFINITION 9.5

Let $n = p_1^{\alpha_1} \ldots p_m^{\alpha_m}$ distinct primes p_i, $1 < i < m$. Then

$$\phi(n) := p_1^{\alpha_i - 1} \ldots p_m^{\alpha_m - 1} (p_1 - 1) \ldots (p_m - 1)$$

is the *Euler function.* Note that $|\mathbb{Z}_n^*| = \phi(n)$.
Moreover let

$$\lambda'(n) := \mathrm{lcm}\,\{p_1 - 1\ \ldots, p_m - 1\}. \quad \square$$

The relation between the computational complexity of f^* and an integer function g is studied in the following lemma due to Miller (see [40]).

LEMMA 9.6 (E.R.H)

If $g(n)$ is a function satisfying:

(i) $\lambda'(n)$ divides $g(n)$

(ii) $\log(g(n)) = O(\log^c n)$ for some constant c depending on the function g)

then $\qquad f^* <_p g.$ □

PROPOSITION 9.7 (E.R.H.)

$$f^* \approx_p \theta$$

Proof

(i) First one can compute a generating set S for $\mathbb{Z}_n^*$ in polynomial time using the algorithm of Theorem 9.4. Knowing the prime factorization of $n = p_1^{\alpha_1} \dots p_k^{\alpha_k}$ one can compute the order $|g|$ of each g in S in polynomial time; this can be done by raising g^m with $m = n/p_i^{\alpha_i}$ in the smaller power of p_i such that $(g^m)^{p_i^{\delta_i}} = 1$, for $1 < i < k$ and then $|g| = p_1^{s_1} \dots p_k^{s_k}$. Hence $\theta(n)$ can be computed in polynomial time as

$$\theta(n) = (\mathrm{lcm}\ \{|g|: g \in S\})^{\lceil \log n \rceil}$$

Therefore

$$\theta <_p f^*$$

(ii) Since

$$\lambda'(n) \text{ divides } \theta(n)$$

and $\qquad \log(\theta(n)) = \log(h^{\lceil \log n \rceil}) = O(\log^2 n)$

it follows from Lemma 9.6 that

$$f^* <_p \theta. \quad \square$$

Now the relation of the complexity of factoring and computing the order of a element of a group is investigated.

DEFINITION 9.8

Problem T Given a generating set S for an abelian group G of order at most n and a element x in G, compute the order of x. □

Let $T_{S,x}(n)$ denote the time in terms of elementary operations required by an optimal algorithm for problem T and let

$$T(n) := \max_{S,x} \{T_{S,x}(n) : x \in G = \langle S \rangle \text{ and } |G| < n\}. \quad \square$$

PROPOSITION 9.11 (E.R.H.)

For some polynomial p the following holds

$$T(n) \succ \Theta(n)/p(\log n)$$

Proof

One can compute $\theta(n)$ in the following way. First compute a generating set S for $\mathbb{Z}_n^*$ in $p_1(\log n)$ elementary operations for some polynomial p_1 using the algorithm of Theorem 9.4. Second compute the orders $|g|$ of each $g \in S$ using optimal algorithms; this requires

$$\sum_{g \in S} T_{S,g}(n)$$

elementary operations. The value of $\theta(n)$ is given by $(\operatorname{lcm}\{|g| : g \in S\})^{\log n}$ which can be computed in $p_2(\log n)$ elementary operations. Hence the computation of $\theta(n)$ in the above way requires

$$\Theta^*(n) := p_1(\log(n)) + \sum_{g \in S} T_{S,g}(n) + p_2(\log n)$$

elementary operations.

Now using that $|S| = p_4(\log n)$ for some polynomial p_4 follows that

$$\Theta^*(n) < p_1(\log n) + p_2(\log n) + p_4(\log n)\, T(n)$$

or for some polynomial p follows that

$$\Theta^*(n) < p(\log n)\, T(n) \tag{9.1}$$

Therefore from the obvious relation

$$\Theta(n) < \Theta^*(n)$$

and (9.1) the proposition follows. □

<u>COROLLARY 9.12</u> (E.R.H.)

For some polynomial p, the following holds

$$T(n) > F(n)/p(\log n)$$

<u>Proof</u>

From Proposition 9.7 it follows that

$$\Theta(n) = p_1(\log n) + F(n)$$

for some polynomial p_1. Therefore the corollary follows from Proposition 9.11. □

B. Relative Complexity Of Problems In Abelian Groups

In the previous subsection the complexity relationship between functions is considered; here the relation of the complexity of problems is considered.

<u>DEFINITION 9.13</u>

A problem A is *polynomial time reducible* to a problem B denoted

$A <<_p B$

if the existence of a polynomial time algorithm for solution of problem B implies the existence of a polynomial time algorithm for problem A. If $A <<_p B$ and $B <<_p A$, then problem A is *polynomial time equivalent* to problem B denoted $A \approx_p B$. □

<u>PROPOSITION 9.14</u> (transitivity of $A <<_p B$)

Suppose that X, Y and Z are problems. If $X <<_p Y$ and $Y <<_p Z$, then

$X <<_p Z$. □

<u>PROPOSITION 9.15</u> (E.R.H)

If problem F is the problem of factoring an integer n, then

$F <<_p T$.

<u>Proof</u>

Assume the existence of a polynomial time algorithm for T. From Proposition 9.12 it follows that F(n) is a polynomial p(log n). □

<u>DEFINITION 9.16</u>

<u>Problem C</u>. Suppose that G is an abelian group with $|G| < n$. Given a set of generators S for G, compute a canonical basis for G. □

Let $C_{G,S}(n)$ denote the time in terms of elementary operations by an optimal algorithm for problem C and let

$$C(n) = \max_{G,S} \{C_{G,S}(n) : G = \langle S\rangle, |G| < n\}. \quad \square$$

<u>DEFINITION 9.17</u>

<u>Problem D</u>. Suppose that G is an abelian group with $G < n$. Given a set of generators S for G, compute a set of defining relations for G. □

Let $D_{G,S}(n)$ denote the time in terms of elementary operations required by an optimal algorithm for problem D and let

$$D(n) := \max_{G,S} \{D_{G,S}(n) : G = \langle S\rangle, |G| < n\}. \quad \square$$

<u>PROPOSITION 9.18</u>

The following holds:

$$C <<_p D.$$

<u>Proof</u>

Assume that there exists an algorithm for computing a set of defining relations for $G = \langle S\rangle$ in polynomial time, say $p_1(\log n)$. Then using algorithm I.3.1 one can compute a canonical basis for G in $p_2(\log n)$ elementary operations for some polynomial p_2 . Therefore there exists an algorithm for problem C which computes the canonical basis for G in $p(\log n)$ elementary operations, where $p = p_1 + p_2$. □

<u>DEFINITION 9.19</u>

<u>Problem B</u> Suppose that G is an abelian group with $|G| < n$. Given a set of generators S for G, compute a complete basis for G.

Let $B_{G,S}(n)$ denote the time in terms of elementary operations required by an optimal algorithm for problem B and let

$$B(n) := \max_{G,S} \{B_{G,S}(n) : G = \langle S\rangle, |G| < n\}. \quad \square$$

PROPOSITION 9.20

The following holds:

$$C <<_p B$$

Proof

Suppose that the complete structure of G is

$$G = \prod_{i=1}^{j_1} C(p_1^{\alpha_{i1}}) \times \dots \times \prod_{i=1}^{j_k} C(p_k^{\alpha_{ik}}) \text{ for } p_i,\ 1 < i < k$$

distinct primes and $j_1 > j_2 > \dots > j_n$, $\alpha_{1\ell} < \alpha_{2\ell} < \dots < \alpha_{j_\ell \ell}$ for $1 < \ell < k$.

Then

$$G = \prod_{i=1}^{j_\ell} C(d_i) \text{ with } d_i = \prod_{\substack{\ell=1 \\ j_\ell > j_i}}^{k} p_\ell^{\alpha_{\ell i}}$$

yields the canonical structure of G.

Therefore the existence of a polynomial time algorithm for problem B implies the existence of a polynomial time algorithm for problem C. $\square$

DEFINITION 9.21

Problem 0 Suppose that G is an abelian group with $|G| < n$. Given a set of generators S for G compute the order of G.

Let $\Theta_{G,S}(n)$ denote the time in terms of elementary operations required by an optimal algorithm for Θ and let

$$\Theta(n) = \max_{G,S} \{\Theta_{G,S}(n) : G = \langle S \rangle, |G| < n\}. \quad \square$$

<u>PROPOSITION 9.22</u>

The following holds

$$\Theta <<_p C$$

<u>Proof</u>

It is obvious. $\square$

<u>PROPOSITION 9.23</u>

The following holds:

$$T <<_p \Theta$$

<u>Proof</u>

Assume the existence of a polynomial time algorithm for problem Θ. Then using this algorithm one can compute the order of the group $\langle X \rangle$. o.e.d. $\square$

<u>DEFINITION 9.24</u>

<u>Problem C_γ</u> Suppose that G is an abelian group and $G < n$. Given a generating set S for G, test whether or not there exists an $h \in G$ such that $G = \langle h \rangle$ and compute a_g for all $g \in S$ such that $h^{\alpha_g} = g$. Let $C_{\gamma\ G,S}(n)$ denote the time in terms of elementary operations required by an optimal algorithm for problem $C\gamma$ and let

$$C_\gamma(n) = \max_{G,S} \{C_{\gamma\ G,S}(n) : G = \langle S \rangle,\ |G| < n\}. \quad \square$$

<u>PROPOSITION 9.25</u>

The following holds

$$C_\gamma <<_p C$$

<u>Proof</u>

It is obvious. $\square$

<u>DEFINITION 9.26</u>

<u>Problem E</u>. Suppose that G is an abelian group with $|G| < n$. Given a set S of generators for G and an element x in G, compute an expression, if any, for x in terms of the generators g in $S' \subseteq S$, such that

$$x = \prod_{g \in S'} g^{\alpha_g} \text{ with } \alpha_g < |g| \quad \forall g \in S'$$

Let $E_{G,S,x}(n)$ denote the time in terms of elementary operations required by an optimal algorithm for problem E and let

$$E(n) := \max_{G,S,x} \{E_{G,S,x}(n) : G = \langle S \rangle,\ x \in S,\ |G| < n\}. \quad \square$$

<u>ALGORITHM 9.27</u>

<u>INPUT</u> : An algorithm for problem E and an element x of G

<u>OUTPUT</u>: The order of G

```
Procedure ORD(X)
  begin
1.    compute an expression of x in terms of $x^2$ using the algorithm for E;
      if such an expression exists then
        let $x = x^{2k}$, where k as in step 1;
2.      return $|x| = 2k-1$;
      else
3.      $|x| \leftarrow 2.ORD(x^2)$;
       return $|x|$;
  end. □
```

PROPOSITION 9.28

The following holds:

$$T <<_p E$$

Proof

Assume the existence of a polynomial algorithm for problem E. Then algorithm 9.27 correctly computes the order of x in polynomial time.

If at step 1 an expression is computed, then $|x|$ is an odd number and thus $|x| = |x^2|$. Hence it is not difficult to see that $|x| = 2k-1$.

If there is no such expression, then x is even and the problem is reduced to the computation of the order of x^2. The recursive calls of the procedure ORD can be at most $\log |x| < \log n$.

Therefore algorithm 9.27 requires polynomial time in log n. □

PROPOSITION 9.29

The following holds:

$$E \ll_p D$$

Proof

Assume the existence of a polynomial time algorithm for problem D. Then let $S^* = S' \cup \{x\}$. In polynomial time one can obtain a set of defining relations for G^* generated by S^* using the algorithm for problem D.

Let the defining relations for G^* in additive notation be:

$$A[x,g_1,\ldots,g_k]^t = [0,\ldots,0]^t \text{ with } S' = \{g_1,\ldots,g_k\} \qquad (9.2)$$

where A is an $m \times n$ integer matrix.

Now one can triangularize A via IRC operations to the matrix

$$\begin{vmatrix} T \\ 0 \end{vmatrix} \text{ with } T \text{ an } (n+1) \times (n+1) \text{ upper triangular matrix} \qquad (9.3)$$

This can be done in polynomial time by Proposition I.4.2.
From (9.2) and (9.3) follows that

$$t_{11}x + t_{12}g_1 + \ldots + t_{1,n+1}g_n = 0 \qquad (9.4)$$

By definition of the set of defining relations, if $t_{11} \neq 1$, then does not exist a relation required by problem E and if $t_{11} = 1$, then (9.4) is the required relation. ▫

PROPOSITION 9.30

The following holds:

$$T \ll_p C_Y$$

Proof

Assume the existence of a polynomial time algorithm for problem C_Y. Then using this algorithm for the group generated by $\{x,1\}$ one can see that $\langle x,1\rangle$ is cyclic and the algorithm also yields the relations $x = x$ and $1 = x^h$. Therefore the order h of x is computed in polynomial time. □

C. Relative Compexity Of Decision Problems

The first decision problem considered is the following:

DEFINITION 9.31

Problem C_Y^* Given a generating set S for a finite abelian group G with $|G| < n$ decide whether or not G is cyclic.

Let $C_Y^*{}_{G,S}(n)$ denote the time in elementary operations required by an optimal algorithm for problem C_Y^* and let

$$C_Y^*(n) := \max \{C_Y^*{}_{G,S}(n) : \langle S\rangle = G \text{ and } |G| < n\}. \quad \square$$

PROPOSITION 9.32

The following holds

$$C_Y^* <<_p C_Y$$

Proof

It is obvious. □

The second decision problem is a version of problem T.

DEFINITION 9.33

Problem T^* Given a generating set S for a finite abelian group G with $|G| \leq n$, an element x in G and an integer k, decide whether or not the order of x is less than k.

Let $T^*_{G,S,x,k}(n)$ denote the time in terms of elementary operations required by an optimal algorithm for problem T^* and let

$$T^*(n) := \max \{T^*_{G,S,x,k}(n) : G = \langle S \rangle,\ x \in G,\ |G| \leq n\}. \quad \square$$

PROPOSITION 9.34

The following hold:

(i) $T \approx_p T^*$

(ii) $T^*(n) \leq T(n) \leq |\log n|\ T^*(n)$

Proof

(i) It is obvious that $T^* \ll_p T$.

Now assume the existence of a polynomial time algorithm for T^*. Then using "binary search" together with the algorithm T^* one can compute the order of x. A sketch of the method is the following:
Check whether or not $|x| < \frac{n}{2}$. If the answer is "yes", then $0 < |x| < \frac{n}{2}$ else $\frac{n}{2} \leq |x| \leq n$. Then check whether or not $|x| < m$ where m is the middle of the interval in which the order was estimated to fall. By repeating this process one can find the order in at most $|\log n|$ repetitions since the interval is halved every time. Hence $T \ll_p T^*$.

(ii) It follows from above. $\square$

In order to measure the "hardness" of the problem C^*_Y, the well-known problem of "primality testing" is considered along with some results on Z^*_n.

<u>DEFINITION 9.35</u>

<u>Problem P</u> Given an integer n decide whether or not n is a prime. Let P(n) denote the time in elementary operations required by an optimal algorithm for problem P. □

<u>DEFINITION 9.36</u>

An integer n is said to be *Carmichael number* if

(i) $n = p_1 \dots p_k$ where p_i's are distinct primes with $k > 2$

(ii) $p_i - 1$ divides $n-1$ for $1 < i < k$. □

<u>LEMMA 9.37</u>

If an abelian group G is cyclic, then it contains exactly $\phi(p)$ elements of order p in every p-Sylow subgroup of G.

<u>Proof</u>

Let $G = \langle x \rangle$ and $Z^*_p = \{k_1, \dots, k_{\phi(p)}\}$. If $|G| = p_1^{\alpha_1} \dots p_\lambda^{\alpha_\lambda}$, then let $d_i = |G|/p_i$ for $1 < i < \lambda$. Moreover let $x_j = x^{k_j d_i}$ for every k_j in Z^*_p. Then it is not difficult to see that $s = \{x_1, \dots, x_{\phi(p)}\}$ are all the elements of G of order p. Because if y has order p_i, then using that $y = x^q$ for some q follows that $x^{qp_i} = 1$ and that $q \equiv 0 \bmod d_i$ which implies

$$q = \mu \, d_i$$

with μ relative prime to p_i and thus $y \in S$. □

LEMMA 9.38

Suppose that n is Carmichael number. Then Z_n^* is not cyclic.

Proof

Let $n = p_1 \ldots p_k$ where p_i for $1 < i < k$ are prime. Now consider the equation

$$x^{p_\lambda} = 1 \bmod(n) \text{ for some } 1 < \lambda < k \qquad (9.5)$$

If the number of solutions of the equation

$$x^{p_\lambda} \equiv 1 \bmod p_i \text{ for some } 1 < i < k \qquad (9.6)$$

is k_i, then the number ℓ of solutions modulo n of (9.5) is

$$\ell = \prod_{i=1}^{k} k_i \text{ (see Apostol [4]).}$$

First observe that $k_i > 1$, since 1,-1 are solutions of (9.6).

Moreover every $h \in Z_{p_\lambda}^*$ is a solution of the equation

$$x^{p_\lambda} \equiv 1 \bmod p$$

and thus $k_\lambda > \phi(p_\lambda)$.

Therefore $\ell > \phi(p_\lambda)$. This implies that the number of elements of order p_λ is greater than $\phi(p_\lambda)$ and by Lemma 9.41 Z_n^* is not cyclic. □

PROPOSITION 9.39

An integer n is prime if and only if

(i) The integer n is not prime power

(ii) The group Z_n^* is cyclic

(iii) Every $h \in Z_n^*$ satisfies the equation

$h^{n-1} \equiv 1 \bmod n$. (Fermat's criterion)

Proof

If n is prime, then it is well known that $\mathbb{Z}_n^* = \phi(n) = n-1$ and (i) - (iii) hold.

Assume now that n is not prime, $n = p_1^{\alpha_1} \ldots p_k^{\alpha_k}$ for p_i $1 < i < k$ distinct primes and (i) - (iii) hold.

Since (iii) holds, it follows that

$\phi(n)$ divides $n-1$

which implies that $p_i^{\alpha_i - 1}$ divides $n-1$ for $1 < i < k$. Therefore it follows that $a_i = 1$ for $1 < i < k$.

If $k > 3$, then n is a Carmichael number and therefore $\mathbb{Z}_n^*$ is not cyclic by Proposition 9.38 contradicting with the assumption of (ii).

If $k = 2$, then

$\phi(n) = (P_1-1)(P_2-1)$ divides $P_1P_2-1 = (P_1-1)(P_2-1) + (P_1-1) + (P_2-1)$

which is not difficult to be shown invalid. □

As a bridge between the problem $C_Y^\#$ and P the following problem is used.

DEFINITION 9.40

Problem Z The same as problem $C_Y^\#$ but with $G = \mathbb{Z}_n^*$. The definitions of $Z_{\mathbb{Z}_n^*,S}(n)$ and $Z(n)$ are similar to $C_Y^\#{}_{G,S}(n)$ and $C_Y^\#(n)$ respectively. □

PROPOSITION 9.41

The following holds:

$$Z <<_p C^*_Y. \quad \square$$

The problem Z is used as bridge between the problem C^*_Y and P in order to avoid the assumption of E.R.H for the construction of the generating set of $\mathbb{Z}^*_n$ in which case the reduction is dependant on the truth of E.R.H.

PROPOSITION 9.42

The following holds:

(i) $P << Z$.

(ii) $P(n) \leq Z(n) + p(\log n)$

Proof

(i) Assume the existence of a polynomial time algorithm for Z. Then one can check the conditions (i) and (ii) of Proposition 9.43 in polynomial time. One can check whether or not $g^{n-1} = 1$ for each $g \in S$ by means of the "power algorithm" and thus checking whether or not (iii) holds requires polynomial time.

(ii) It follows from above. $\square$

D. Relative Complexity Of Intersection Problems

DEFINITION 9.43

Problem OI Given abelian groups $G = \langle S \rangle$ and $F = \langle S' \rangle$ with $|G| \leq n$ and $|F| \leq n$ and the fact that $\langle F \cup G \rangle$ abelian, compute the order of $\langle F \cap G \rangle$.

Let $OI_{G,S,F,S'}(n)$ denote the time in elementary operations required by an optimal algorithm for problem OI and let

$$OI(m,n) := \max_{G,S,F,S'} \{OI_{G,S,F,S'}(n,m) : G=\langle S\rangle, F=\langle S'\rangle, |G|<n, |F|<m, \langle F\ G\rangle_{\lambda} \text{ abelian}\} \quad \square$$

<u>PROPOSITION 9.44</u>

The following holds:

$$|\Theta(n,m) - OI(n,m)| < \Theta(n) + \Theta(m) + p(\log nm)$$

where $\Theta(n,m) = \max \{\Theta_{H,S}(nm) : H = F \cup G = \langle S\rangle,\ |F| < n \quad |G| < m\}$

<u>Proof</u>

It is well known that

$$|\langle F \cup G\rangle| = |F| \cdot |G| / |\langle F \cap G\rangle| \qquad (9.7)$$

Now one using an optimal objection for computing F, G and $F \cup G$ can compute $\langle F \cap G\rangle$ by using (9.7). Therefore

$$OI(n,m) < \Theta(n,m) + \Theta(n) + \Theta(m) + p(\log mn)$$

for some polynomial p.

Similarly using an optimal algorithm for computing $|F|$, $|G|$ and $|\langle F \cap G\rangle|$ one can compute $|\langle F \cup G\rangle|$ by using (9.7). Therefore

$$\Theta(m,n) < OI(n,m) + \Theta(n) + \Theta(m) + p(\log mn) \qquad (9.8)$$

Hence the proposition follows. $\square$

<u>PROPOSITION 9.45</u>

The following holds

$$\Theta(m,n) < c\ OI(n,m) + p(\log mn)$$

for some constant $c > 1$ and some polynomial p.

<u>Proof</u>

From (9.8) using that

$$O(m) + O(n) < \left(1 - \frac{1}{c}\right) \Theta(m,n). \quad \square$$

<u>PROBLEM 9.46</u>

<u>Problem OI*</u> As problem OI without the restriction that $\langle F \cup G \rangle$ is abelian. In a similar way to OI(n) the complexity of OI* denoted OI*(n,m) is defined. $\square$

<u>PROPOSITION 9.47</u>

The following holds:

$$OI^*(n,m) > c\ \Theta(m,n) + p(\log mn)$$

for some constant $c < 1$ and some polynomial p.

<u>Proof</u>

From Proposition 9.45 and the fact that

$$OI^*(n,m) > OI(n,m). \quad \square$$

<u>PROPOSITION 9.45</u>

The following holds

$$\Theta(m,n) \leq c\ OI(n,m) + p(\log mn)$$

for some constant $c > 1$ and some polynomial p.

<u>Proof</u>

From (9.8) using that

$$O(m) + O(n) \leq (1 - \frac{1}{c})\ \Theta(m,n). \quad \square$$

<u>PROBLEM 9.46</u>

<u>Problem OI*</u> As problem OI without the restriction that $\langle F \cup G \rangle$ is abelian. In a similar way to OI(n) the complexity of OI* denoted OI*(n,m) is defined. $\square$

<u>PROPOSITION 9.47</u>

The following holds:

$$OI^*(n,m) \geq c\ \Theta(m,n) + p(\log mn)$$

for some constant $c < 1$ and some polynomial p.

<u>Proof</u>

From Proposition 9.45 and the fact that

$$OI^*(n,m) \geq OI(n,m). \quad \square$$

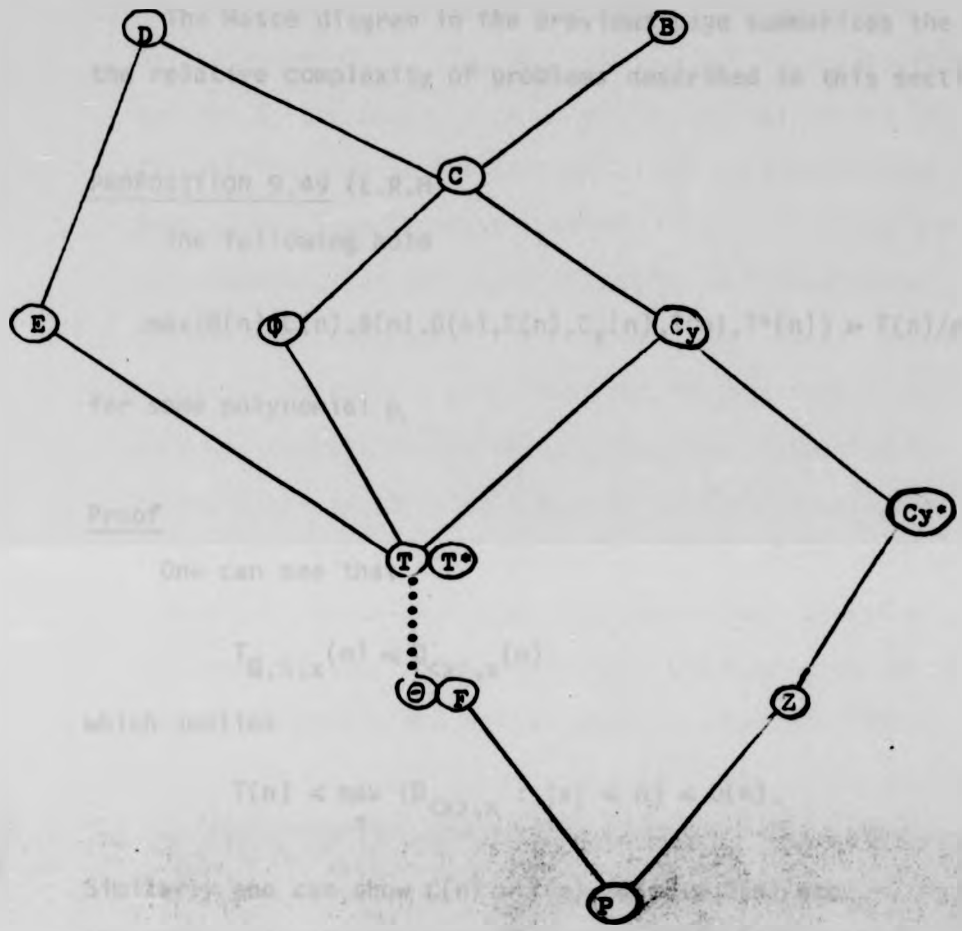
One can define problems of computing the canonical structure (problem (I), complete structure (problem BI), a set of defining relations (problem DI) for the intersection of two abelian groups. It is not difficult to show that the following hold:

(i) $CI <<_p DI$, (ii) $CI <<_p BI$, (iii) $OI <<_p CI$

Their proof are similar to the proofs of Propositions 9.18, 9.20 and 9.22. Moreover one can define the problem of deciding whether or not the intersection of two groups is cyclic (problem $C_Y I$). It is not difficult to prove that

$$CY <<_p C_Y I$$

HASSE DIAGRAM



The dotted line denotes the assumption of E.R.H.

E. Conclusions

The Hasse diagram in the previous page summarizes the results on the relative complexity of problems described in this section.

<u>PROPOSITION 9.49</u> (E.R.H)

The following hold

$$\max\{D(n), C(n), \Theta(n), B(n), E(n), C_\gamma(n), T(n), T^*(n)\} \succ F(n)/p(\log n) \qquad (9.9)$$

for some polynomial p.

<u>Proof</u>

One can see that

$$T_{G,S,x}(n) \prec D_{\langle x\rangle,x}(n)$$

which implies

$$T(n) \prec \max_{x}\{D_{\langle x\rangle,x} : |x| \prec n\} \prec D(n).$$

Similarly one can show $C(n) \succ T(n)$, $\Theta(n) \succ T(n)$ etc.
Therefore using Proposition 9.11 the proposition follows. □

In the sections 1-8 upper bounds are given for the above problems are given; it is shown that the complexity of all the problems in L.H.S. of (9.9) is $O(n^{\frac{1}{2}+\epsilon})$ elementary operations. The best upper bound for F is $O(\log n)^{c\log\log\log n})$ elementary operations due to Adleman [1]; the proof of the bound depends on the truth of various number-theoretic assumptions. The best-known upper bound for F (depending on the truth of ERH) is $O(n^{1/5+\epsilon})$ due to D. Shanks [47] (see [30]).

Let us consider the role of the Riemann hypothesis which is unsolved for more than a century. One can see that assuming E.R.H. one can compute a generating set for $\mathbb{Z}_n^*$ by picking up all the primes in the interval [1, $(\log^2 n]$. Without E.R.H to construct a generating set for $\mathbb{Z}_n^*$ one has to pick up all the primes of the interval $[1, ch^{\frac{1}{4}+\epsilon}]$; this is the best known unconditional upper bound due to Burgess [6] (see Lagarias and Odlyzko [36]). Therefore one can see that the assumption of the truth of E.R.H. is "responsible" for the gap between the upper bound for problem T and Shanks' bound for factorization: Shanks' constructs a generating set for the form class group which can be found quickly $(O(\log^2 n))$ assuming the truth of E.R.H. but unconditionally the computation is of $O(n^{\frac{1}{4}+\epsilon})$. An additional reason is that Odlyzko's proof for the $O(n^{1/5+\epsilon})$ upper bound on Shanks' algorithm assumes the truth of E.R.H. One paradoxical aspect of Shanks' algorithm is that primality depends on E.R.H. but compositeness is proved unconditionally since one can check whether the "factors" are factors.

In Adleman's factorization algorithm the situation is similar. The unconditional bound for his algorithm by Rumely (see [1]) is

$$O(e^{c(\log n)^{0.4999}})$$

Assuming that the density of Eulidean primes† is "small" (see [1], p.403) it can be shown that Adleman's algorithm terminates in

$$O(\log n^{c \text{ logloglog } n})$$

elementary operations.

† A prime number p is called "Eulidean prime", if $p = 2p_1 \ldots p_k + 1$, where P_i is a prime, for $1 \leq i \leq k$, for some integer k.

Hence one can claim that the assumption for the density of Euclidean primes is "responsible" for the gap between the upper bounds for T and F. Note that the correctness of Adleman's algorithm is unconditional only the analysis of its running time depends on assumptions.

It is not difficult to verify that $C^*_Y(n) > P(n)$. Using the algorithm for problem C one can obtain an $O(n^{\frac{1}{2}+\epsilon})$ upper bound for problem C^*_Y. The best known "primality testing" algorithms are due to Miller [40]. He gave an unconditional algorithm for problem P which requires $O(n^{1/7+\epsilon})$ elementary operations and one depending on the truth of E.R.H. which requires $O(\log^4 n \text{ loglog } n)$ elementary operations. Both the correctness and the analysis of the conditional algorithm depend on the assumption of E.R.H. Once again one can see the dramatic influence of E.R.H. on proving the bounds and speeding up the algorithms. Also from the above one may suggest that the bound given for problem C^*_Y can be improved.

Also using the facts that $OI^*(n,m) \approx \mathcal{M}(n,m) \approx \mathcal{M}(nm)$ and $OI(n,m) \approx \mathcal{M}(n,m) \approx \mathcal{M}(nm)$ and the argument that the bound for problem $\mathcal{M}$ is reasonable, one can suggest that the bounds for OI and OI* proved in section 8 are not weak.

10. APPLICATIONS

A. Factorization And Form Class Group

Shanks' algorithm for factorization makes use of the form class group (see [47] and [30]); that is the group formed by the binary quadratic forms (a,b,c) with determinant $D = b^2 - ac$, where $D < 0$ is the number given for factoring. In order to compute a complete factorization of D, it is sufficient to compute generators for every cyclic subgroup of the elementary 2-group of the form class group. In particular the factors are given from the ambiguous forms, that is the binary forms of order 2. An ambiguous binary quadratic form is of the form either

$$(a, a, c) \text{ or } (a, b, a) \text{ or } (a, o, c)$$

Therefore if one can compute an ambiguous form, then a factorization of D either $D = a(a-c)$ or $D = (b-a)(b+a)$ or $D = ac$ is obtained. Hence one can use the algorithms described in the previous sections for factorization and further for investigation of the form class group

B. Galois Theory

The algorithm for computing the basis of a group represented by a set of generators can be used in the computation of algebraic functions which are determined by the Galois group of the extension of the field by the functions. For details see JÁ Ja' [32].

C.CRYPTOGRAPHY

Algorithm 23may be used for computing discrete logarithms (see [44] and [35]). The same algorithm may have applications in the public key cryptosystem introduced by Diffie and Helmann in [14].

CHAPTER III

COMPUTATIONAL PROBLEMS ON ABELIAN SUBGROUPS OF THE SYMMETRIC GROUP

1. INTRODUCTION

In this section abelian subgroups of the symmetric group represented by a set of generators are investigated. In general computing the order, the canonical structure, a set of defining relations a basis et al for abstract abelian groups require exponential time,but it will be shown that such computations in abelian permutation groups require polynomial time of elementary operations in terms of the number of symbols permuted by the group. Furst et al in [18] showed that the computing of the order of a permutation group and membership testing in a permutation group can be done in polynomial time.

In Section 2 an "elementary orbit" algorithm is presented; it requires $O(kn\log^2 n)$ elementary operations for computing the orbits of a group $G \subseteq S_n$,which is represented with k generators. Moreover algorithms computing the canonical and the complete structure (consequently the order) of an abelian group $G \subseteq S_n$ are given. Both require $O(kn^4 \log n \log\log n)$ elementary operations, where k is the number of generators. Furst et al in [18] gave an algorithm computing the order and a set of "strong generators" for a permutation group which requires $O(kn^2 + n^6)$ elementary operations.

In Section 3 algorithms testing membership and inclusion are presented; their complexity is $O(kn^4 \log n \log\log n)$ elementary operations. Furst's et al membership testing in a permutation group given in [18] requires $O(k\, n^2 + n^6)$ elementary operations.

In Section 4 the problem of computing the intersection of two abelian groups is considered; it is an important problem due to its links with graph isomorphism problems. Three algorithms for computing the intersection of the abelian groups F and G both subgroups of S_n are given; one algorithm for computing $\langle F \cap G \rangle$ when $\langle F \cup G \rangle$ is abelian and the other two for computing $\langle F \cap G \rangle$ when $\langle F \cup G \rangle$ is non-abelian. The time complexity of all the intersection algorithms is of $O((k+m+n)n^4 \log n \log\log n)$ elementary operations.

Finally in Section 5 several applications of the algorithms described in previous sections are discussed.

A.Preliminaries

Let G be a permutation group, subgroup of the symmetric group S_n which operates on the set $I_n = \{1,\dots,n\}$. A set $\Delta \subseteq I_n$ is called a *fixed block* of G if

$\Delta = \{g(\delta) : g \in G,\ \delta \in \Delta\}$.

Let $g = c_1 c_2 \dots c_k$ in G, where the c_i's are disjoint cycles and let $\Gamma \subseteq I_n$. If Γ is a set of all points permuted by $c_i, \forall\, i \in I$, for some set $I \subseteq I_n$ then $g^\Gamma = \prod_{i \in I} c_i$ is called the *restriction* of g to Γ. Note that if $\alpha \in \Delta \subseteq I_n$ and $g(\alpha) \notin \Delta$ then g^Δ is not defined.

Let Δ be a fixed block of the permutation group G. Then

$G^\Delta = \{g^\Delta : g \in G\}$

is called the *constituent* of G on Δ. Note that G^Δ is a group.

Let $G \subseteq S_n$ operating on I and $\Gamma \subseteq I_n$. Then

$G^\Gamma = \{g^\Gamma : g^\Gamma$ is well-defined and $g \in G\}$

is called a *pseudo-constituent* of G on Γ. Note that G^Γ is a group.

A group $G \subseteq S_n$ is called *transitive* if the only fixed blocks of G are the trivial fixed blocks ϕ and I_n.

A minimal fixed block $\Delta \neq \phi$ of a group $G \subseteq S_n$ is called an *orbit* of G.

Let G be a permutation group operating on $I \subseteq I_n$. If for each $\alpha \in I$ and for each non-trivial $g \in G$, $g(\alpha) \neq \alpha$ then G is called *semiregular*.

A semiregular and transitive group is called *regular*.

PROPOSITIUN 1.1 (Wielandt [52])

A transitive abelian group $G \subseteq S_n$ is semiregular.

Proof

Let G operating on I. Assume the existence of a non-trivial g in G such that

$$g(\alpha) = \alpha \text{ for some } \alpha \in I$$

Then for every f in G follows that

$$g(f(\alpha)) = f(g(\alpha)) = f(\alpha) \tag{1.1}$$

Let $\Delta_\alpha = \{f(\alpha):f \in G\}$. Then Δ_α is a fixed block and since G is transitive, follows

$$\Delta_\alpha = I \tag{1.2}$$

Hence from (1.1) and (1.2) follows that g = 1 contradicting the non-triviality of g. □

PROPOSITION 1.2 (Wielandt [52])

Suppose that G is a regular abelian group on n symbols. Then

$$|G| = n$$

Proof

Let $\Delta_\alpha = \{g(\alpha):g \in G\}$ for some $\alpha \in I_n$. Then Δ_α is a fixed block and since G is transitive follows that

$$\Delta = I_n$$

Moreover

$$|\Delta_\alpha| = |G|$$

because if $f(\alpha) = g(\alpha)$ for some f and g in G, then it follows that $(f^{-1}g)(\alpha) = \alpha$ contradicting the semiregularity of G. □

PROPOSITION 1.3 (Dixon [12], Exercise 2.41)

Suppose that G is an abelian subgroup of S_n. Then

$$|G| \le 3^{n/3}$$

Proof

If G is transitive, then $|G| = n \le 3^{n/3}$.

Let $\Gamma_1,\ldots,\Gamma_t$ be the orbits of G and let $|\Gamma_i| = k_i$ for $1 \le i \le t$. Then G acts transitively on orbits and hence

$$|G| \le \prod_{i=1}^{t} |G^{\Gamma_i}| \le \prod_{i=1}^{t} k_i \le 3^{k_i/3} \le 3^{n/3}. \quad \Box$$

2. THE COMPUTATION OF THE ORDER AND THE STRUCTURE OF AN ABELIAN SUBGROUP OF THE SYMMETRIC GROUP

In this section the problem of computing the order and the structure of an abelian permutation group, given a set of generators, is investigated. One can observe that the order of an abelian subgroup of S_n can be as big as $3^{n/3}$ and thus an application of algorithm II.5.1. (which requires $O(3^{n/6})$ elementary operations) is impractical. One may use, Furst's et al algorithm for computing the order in polynomial time; but their method does not yield information about the structure of a group, when the group is abelian. Therefore an efficient algorithm for computing the canonical structure and the complete structure of an abelian group is needed.

First the "elementary orbit algorithm" for computing the orbits of one abelian group $G \leq S_n$ is considered. The main concept of the algorithm is demonstrated by the following proposition.

<u>LEMMA 2.1.1</u> (See Wielandt [52], p.4)

Suppose that G is a permutation group. Two points α and β lie in the same orbit if and only if $\beta = g(\alpha)$ for some $g \in G$.

<u>PROPOSITION 2.1.2</u>

Suppose that the abelian group $G = < g_j = c_{1j} \dots c_{\nu_j j}$, for $1 \leq j \leq k >$ is a subgroup of S_n (the c_{ij}'s are disjoint cycles). Let $S_{\ell j}$ denote the set of points permuted by the cycle $c_{\ell j}$ and

$\Gamma^{(1)} = S_{\lambda\mu}$ for some integers λ and μ

and

$$\Gamma^{(i+1)} = \Big(\bigcup_{S_{\ell j} \cap \Gamma^{(i)} \neq \phi} S_{\ell j} \Big) \cup \Gamma^{(i)} \qquad (2.1)$$

Then (i) $\Gamma^{(r+1)} = \Gamma^{(r)}$ for some integer r

(ii) $\Gamma^{(r)} = \Gamma$ is an orbit of G

<u>Proof</u>

(i) It is obvious.

(ii) First it will be shown that $\Gamma \subseteq \Delta$, where Δ is the orbit of G containing $S_{\lambda\mu}$. We shall use induction.

For i=1, we have $\Gamma^{(1)} \subseteq \Delta$. Assume that $\Gamma^{(i)} \subseteq \Delta$. Then let $\alpha \in \Gamma^{(i+1)} - \Gamma^{(i)}$. Let $\alpha \in S_{\ell j}$. Since $S_{\ell j}$ intersects $\Gamma^{(i)}$, we have that $g_j^m(\alpha) \in \Gamma^{(i)}$ for some integer m and thus $\alpha \in \Delta$ o.e.d.

Now it will be shown that $\Delta \subseteq \Gamma$. Let $\beta \in \Delta$ and $\alpha \in \Gamma$. Then by lemma 2.1.1 we have that

$g(\alpha) = \beta$ for some $g \in G$ ($\alpha \in \Gamma \subseteq \Delta$ and Δ is an orbit)

Since $g = g_1^{m_1} \dots g_k^{m_k}$ for some integer m_i, $1 \leq i \leq k$, we have that

$\beta = g_1^{m_1}(g_2^{m_2}(\dots\dots(g_k^{m_k}(\alpha))\dots)$

One can see that $\gamma_k := g_k^{m_k}(\alpha) \in \Gamma$, $\gamma_{k-1} := g_{k-1}^{m_{k-1}}(\gamma_k) \in \Gamma$, ..., $\beta = g_1^{m_1}(\gamma_2) \in \Gamma$ (by definition of Γ).

Hence $\Gamma = \Delta$.□

<u>ALGORITHM 2.1</u>

<u>INPUT</u>: A set of generators $\{g_1, \dots, g_k\}$ for the abelian subgroup G of the symmetric group S_n

```
    OUTPUT: All the orbits $\Gamma_t$ of G
begin
    t ← 1;
    Let $g_j = c_{1j}\, c_{2j} \ldots c_{v_j j}$ for $1 \le j \le k$;
    comment The $c_{ij}$'s are disjoint cycles of the generator $g_j$, for $1 \le j \le k$
            $S_{ij} \leftarrow \{\alpha : \alpha$ is affected by $c_{ij}\}$ for $1 \le i \le v_j$, for $1 \le j \le k$;
    comment Note that $S_{ij}$ is the set of all points affected by the cycle
            $c_{ij}$ of the generator $g_j$ for $1 \le i \le v_j$ and for $1 \le j \le k$
1.  Sort $S_{ij}$ for $1 \le i \le v_j$, for $1 \le j \le k$;
2.  $L(j) \leftarrow \{1,\ldots,v_j\}$ for $1 \le j \le k$;
    comment Note that if $\ell \in L(j)$ for some $1 \le j \le k$, then $c_{\ell j}$ is the $\ell$-th
            cycle of the generator $g_j$.  If $c_{\ell j}$ is  a cycle of $g_j$ and
            $\ell \notin L(j)$, then $c_{\ell j}$ is called "used" cycle.  A cycle c becomes
            "used", when the set of all points affected by c has been found to
            be a subset of the currently computed orbit (named $\Gamma_t$, see below)
3.  INDEX $[\alpha,j] \leftarrow (m,j)$ such that $\alpha \in S_{mj}$, for $1 \le j \le k$, for every $\alpha \in I_n$;
    comment The INDEX$[\alpha,j]$ is an array which yields the indices of a
            cycle (of the generator $g_j$) which moves the point $\alpha$.
4.  Repeat
        $S \leftarrow S_{\lambda\mu}$ for some $L(\mu) \ne \phi$ ;
        $L(\mu) \leftarrow L(\mu) - \{\lambda\}$ ;
        $\Gamma_t \leftarrow S$;
    comment The set S contains all points affected by the cycle $c_{\lambda\mu}$
            and thus there exists an orbit $\Gamma_t$ which contains S.  Note
            that $c_{\lambda\mu}$ becomes "used" and thus $\{\lambda\}$ is subtracted from $L(\mu)$.
```

```
5.  Repeat
    $\alpha \leftarrow$ an element of S;
6.      $S^* \leftarrow \{(v,j): (v,j) = INDEX[\alpha,j]$ and $v \in L(j)$, for $1 \leq i \leq k\}$
    comment The set $S^*$ is the set of indices $(v,j)$ of the cycles $c_{vj}$ that
            are not "used" and move $\alpha$.
7.  for each $(v,j)$ in $S^*$ do
        begin
8.      $S \leftarrow S \cup (S_{vj} - \Gamma_t)$
9.      $\Gamma_t \leftarrow \Gamma_t \cup S_{vj}$;
    comment The cycle $c_{vj}$ moves at least one point (the point $\alpha$)
            of the orbit $\Gamma_t$ and thus every point affected by $c_{vj}$ belongs
            to $\Gamma_t$. Moreover the points added to $\Gamma_t$ at step 9 are added
            to S in order to pick up those not "used" cycles which move
            the points of $S_{vj} - \Gamma_t$.
            Also steps 8-9 are executed simultaneously; use "binary search"
            for each $\alpha \in c_{vj}$ for testing whether or not $\alpha$ belongs to $\Gamma_t$.
            If $\alpha \notin \Gamma_t$, then S becomes $S \cup \{\alpha\}$ and insert $\alpha$ in $\Gamma_t$ in such a
            way that $\Gamma_t$ remains sorted, "binary search" yields the
            "correct position" of $\alpha$ in the sequence of $\Gamma_t$ (see [2], p. 113).
10.     $L(j) \leftarrow L(j) - \{v\}$;
    comment Note that $c_{vj}$ became "used" at step 9
11.     end
    $S \leftarrow S - \{\alpha\}$ ;
    comment The set $\{\alpha\}$ is subtracted from S, since all cycles moving
            $\alpha$ are "used".
```

```
12. Until  S = $\phi$
    Comment Now the orbit $\Gamma_t$ has been computed, since all cycles
            moving elements of S are "used".
    return $\Gamma_t$;
    t ← t+1;
    comment The computation of a new orbit starts.
13. Until  L(j) = $\phi$  for every $1 \leq j \leq k$
    comment All cycles are "used" and thus all orbits are computed.
    end.
```

2.2.A WORKED EXAMPLE

Suppose that G = < (123) (789) ($\beta\gamma\delta\varepsilon\zeta\theta$) (456),

($\beta\varepsilon$) (15) (43) ($\delta\theta$) (789) ($\gamma\zeta$) (26), (456) ($\beta\delta\zeta$) (789) ($\gamma\varepsilon\theta$) (132)

In order to select the S_{ij}'s of (2.1) , we make use of two arrays, L(j) and INDEX[α,j].

The array L(j) contains the first index of each not "used" cycle of the generator g_j, for $1 \leq j \leq k$. Here we have L(1) = {1,2,3,4}, L(2) = {1,2,3,4,5,6,7} and L(3) = {1,2,3,4,5}.

The matrix INDEX [α,j] yields the indices of that cycle of the generator g_j, which moves the point α. Here we have that

INDEX [α,j]

α	1	2	3	4	5	6	7	8	9	β	γ	δ	ε	ζ	θ
j=1	(1,1)	(1,1)	(1,1)	(4,1)	(4,1)	(4,1)	(2,1)	(2,1)	(2,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)	(3,1)
j=2	(2,2)	(7,2)	(3,2)	(3,2)	(2,2)	(7,2)	(5,2)	(5,2)	(5,2)	(1,2)	(6,2)	(4,2)	(1,2)	(6,2)	(4,2)
j=3	(5,3)	(5,3)	(5,3)	(1,3)	(1,3)	(1,3)	(3,3)	(3,3)	(3,3)	(2,3)	(4,3)	(2,3)	(4,3)	(2,3)	(4,3)

As we did in 2.1.2 we choose $S=S_{\lambda\mu}=\{1,2,3\}$. Hence c_{11} becomes "used" and $L(1)=\{2,3,4\}$. Also the orbit Γ_1 contains at least $\{1,2,3\}$.
Let us choose $\alpha=1$ (after step 5). Then the set S^* yields the indices of all not "used" cycles moving 1 (The computation of S^* is done for efficiency reasons via the arrays INDEX and L)
Here we have

$$S^* = \{(2,2),\ (5,3)\}$$

since c_{22} and c_{33} move 1, which belongs to Γ_1.
Now since $c_{22}=(15)$ and $c_{53}=(132)$, we have that Γ_1 contains at least $\{1,2,3,5\}$. Also c_{11} and c_{53} become "used" and thus $L(2)=\{1,3,4,5,6,7\}$ and $L(3)=\{1,2,3,4\}$. Moreover S becomes $\{2,3,5\}$, since we considered all cycles moving 1 and we want to consider all cycles moving 5.
Note that for the computation of the orbit Γ_1 is not necessary to consider all the cycles which move points belonging to S, but the reason for considering these cycles is that since we find them, we can remove them (from the array L) and thus the computation of the other orbits of the groups will be more efficient.
Now choose $\alpha=2$ (after step 5). Then we have that

$$S^* = \{(7,2)\}$$

because c_{72} moves 2. Note that c_{53} is "used" ($5 \notin L(3)$). Since 2 is in Γ_1 and $c_{72}=(26)$, Γ_1 contains at least $\{1,2,3,5,6\}$. Then c_{72} becomes "used" and $L(2)=\{1,3,4,5,6\}$. Also we have $S=\{3,5,6\}$.
Again choose $\alpha=3$ (after step 5). Then we have that

$$S^* = \{(3,2)\}$$

since c_{32} moves 3. Now $c_{32}=(43)$, hence Γ_1 contains $\{1,2,3,4,5,6\}$.
Also we have $L(2)=\{1,4,5,6\}$ and $S=\{4,5,6\}$.

Similarily for $\alpha=4$, the cycles c_{41} and c_{13} are removed (that is $L(1) = \{2,3\}$ and $L(3) = \{2,3,4\}$) and Γ_1 remaines unchanged. For $\alpha=5,6$ no change occurs. From proposition 2.1.2 follows that Γ_1 is an orbit of G. In a similar way all other orbits of G are computed.

<u>PROPOSITION 2.3</u>

Algorithm 2.1 correctly computes the orbits Γ_t for the abelian group $G \subseteq S_n$ in $O(kn \log^2 n)$ elementary operations.

<u>Proof</u>

The correctness of the algorithm follows from proposition 2.1.2 and the comments of algorithm 2.1.

Step 1 requires $O(\sum_{j=1}^{k} \sum_{i} |c_{ij}|\log|c_{ij}|)$ comparisons for sorting comprising $O(kn \log^2 n)$ elementary operations, since

$$\sum_{i}|c_{ij}| \le n$$

and a comparison requires $O(\log n)$ elementary operations.

Steps 2-3 merely require $O(n)$ operations for the construction of the lists L and INDEX.

Step 6 requires $O(k)$ elementary operations for computing INDEX $[\alpha, j]$ for $1 \le j \le k$ and k applications of "binary search" for testing whether or not $v \in L(j)$. Hence step 6 requires $O(k \log^2 n)$ elementary operations, since $|L(j)| \le n$.

Steps 9-10 requires $|c_{v_j}|$ applications of "binary search" comprising $O(|c_{v_j}|\log|\Gamma_t|)$ comparisons.

It is not difficult to see that S^* passes through all cycles which permute an element of the orbit. Hence loop 7-11 requires $O(\sum_{c_{v_j} \subseteq \Gamma_t} |c_{v_j}|\log|\Gamma_t|\log n) = O(k(|\Gamma_t|\log|\Gamma_t|\log n)$ elementary operations.

Moreover one can see that all the symbols of Γ_t pass through Λ and thus loop 5-12 has at most $|\Gamma_t|$ iterations.

Therefore loop 5-12 requires $O(k|\Gamma_t|\log|\Gamma_t| \log n + k \log|\Gamma_t|\log^2 n)$ elementary operations.

Finally the algorithm requires

$$O(k \log|\Gamma_t|\log n \, (\sum_{t} |\Gamma_t| + \log n)) = O(kn \log^2 n)$$

elementary operations. □

Algorithm 2.1 makes use of the data structure used in the "disjoint-set algorithm" (UNION-FIND algorithm) given in Aho et al in [2] p. 124; algorithm 2.1 is not a straightforward application of the "disjoint-set algorithm", since the first one makes use of UNION of sets not necessarily disjoint (step 10) and the latter deals only with UNION of disjoint sets. One can find data structures used in a faster version of the UNION-FIND algorithm given in [2], p. 129, the use of which will improve the upper bound of the "elementary orbit algorithm" by a factor of $O(\log^{\varepsilon} n)$, with $0 < \varepsilon < 1$.

<u>PROPOSITION 2.4</u>.

There exists an algorithm for computing the constituent of G on each orbit of G in $O(k\ n \log^2 n)$ elementary operations, given a set of generators for G.

<u>Proof</u>

One may use algorithm 2.1 for computing the orbits Γ_t, $1 \le t \le \lambda$. Then it is not difficult to see that in $O(kn \log^2 n)$ elementary operations one can compute g^{Γ_t} for each generator g of G for $1 \le t \le \lambda$ by means of "binary search"; sort the Γ_i's and then test whether or not a symbol α permuted by the cycle c_{ij} of g_j belongs to Γ_i. Then

$$g_j^{\Gamma_i} = \prod_i c_{ij},$$

with c_{ij} such that α is permuted by c_{ij} and $\alpha \in \Gamma_i$. Then

$$G^{\Gamma_i} = \langle g_1^{\Gamma_i}, \ldots, g_k^{\Gamma_i} \rangle. \quad \square$$

<u>ALGORITHM 2.5</u>

```
INPUT  An abelian group G subgroup of the symmetric group represented
       by a set of generators $<g_1,\dots,g_k>$
OUTPUT The canonical structure and the canonical basis for G.
begin
1.  compute a generating set for $G^{\Gamma_t}$ for each orbit $\Gamma_t$ using the
    algorithm of III.2.1;
2.  compute a complete basis for $G^{\Gamma_t}$ for every t using algorithm II.5.1;
    $\Gamma \leftarrow \prod_t G^{\Gamma_t}$;
3.  compute the complete structure of G using algorithm II.7.4;
    comment The group G is a subgroup of the group $\Gamma$ whose the
    structure is known (If $g \in G$, then $g = \prod_t g^{\Gamma_t} \in \prod_t G^{\Gamma_t} = \Gamma$ )
end. □
```

<u>PROPOSITION 2.6</u>

Algorithm 2.5 correctly computes the complete structure of G in $O(k(\log|\Gamma|)^3 M(\log|\Gamma|) + kn \log^2 n + n^{3/2}\log^3 n)$ elementary operations.

<u>Proof</u>

The correctness of the algorithm depends on the correctness of algorithms III.2.1, II.5.1 and II.7.4 already proved.

Step 1 requires $O(kn \log^2 n)$ elementary operation by Proposition III.2.3.

Step 2 requires $O(\sum_t |\Gamma_t|^{1/2} \log^3 |\Gamma_t| (\log|\Gamma_t| + G)$ elementary operations by Proposition 5.2 and the fact that $|G^{\Gamma_t}| = |\Gamma_t|$

(Proposition 1.2). Since a group operation requires O(n) elementary operations and $\sum_t |\Gamma_t| = n$.

Step 3 requires $O(k\,(\log|\Gamma|)^3\,M(\log|\Gamma|))$ elementary operations by Proposition II.7.5.
The result follows from the above analysis. □

<u>COROLLARY 2.7</u>

Algorithm 2.5 correctly computes the complete structure of the abelian group $G \subseteq S_n$ in $O(kn^4 \log\log\log n)$ elementary operations.

<u>Proof</u>

By Proposition III.1.3 follows that

$$|\Gamma| \leq 3^{n/3}$$

Therefore the result follows from Proposition 2.6. □

Comparing algorithm 2.5 and the algorithm for computing the order of a permutation group given in [18] one can observe the following:
(i) Furst's et al algorithm does not yield any information about the complete structure of the group, in the case of abelian groups.
(ii) Furst's et al algorithm requires $O(kn^2 + n^6)$ elementary operations for computing the order. Under the reasonable assumption that the number of generators $k = O(\log|G|) = O(n)$ (see [18], Hoffman [24], Theorem 5), the upper bound on the time complexity of algorithm 2.5 is better at least by a factor O(n) than the upper bound on the time complexity of Furst's et al algorithm.

(iii) Furst's et al algorithm depends on the number of symbols permuted, since it computes all the cosets G_i for $1 \leq i \leq n$, where G_i is the subgroup G which contains permutations fixing the symbols $\{1,\dots,i\}$. Algorithm 2.5 mainly depends on the number of orbits of the group as it is shown by Proposition 2.6. Therefore the computation of the order of groups whose orbits are large, requires time of $O(k(\log \prod_t |\Gamma_t|^{4+\epsilon})$ elementary operations which can be much better than the worst-case upper bound of $O(k\ n^{4+\epsilon})$ elementary operations. For example with $|\Gamma_1| = \frac{n}{3}$, $|\Gamma_2| = \frac{n}{2}$ and $\Gamma_3 = \frac{n}{6}$, algorithm 2.5 requires $O(k(\log \frac{n^3}{36})^{4+\epsilon} + kn\log n + n^{3/2} \log^4 n) = O(kn \log n + n^{3/2} \log^4 n)$ elementary operations but the time required by Furst's et al algorithm in this case is a polynomial of degree 6.

3. MEMBERSHIP-INCLUSION TESTING AND COMPUTING THE UNION OF ABELIAN GROUPS

A. MEMBERSHIP TESTING

Testing whether or not a permutation h belongs to an abelian group $G = \langle h_1,\dots,h_k\rangle$ subgroup of the symmetric group S_n can be done in polynomial time in terms of n in the following way:

ALGORITHM 3.1

```
begin
1. compute a generating set for the constituents $G^{\Gamma_1},\dots,G^{\Gamma_t}$ for all
   orbits $\Gamma_1,\dots,\Gamma_t$;
2. compute a basis for $G^{\Gamma_i}$ for $1 < i < t$ using algorithm II.5.1;
   Let $\Gamma_i = \langle\langle \gamma_{1i},\dots,\gamma_{s_i i}\rangle\rangle$ for $1 < i < t$;
3. compute the order $|G|$ of G and a basis for $G = \langle\langle g_1,\dots,g_v\rangle\rangle$;
   if h can not be expressed in terms of $\gamma_{ij}$'s then
      return "h does not belong to G"
   else
4. compute an expression of h in terms of $\gamma_{ij}$'s;
5. comment The testing whether or not h can be expressed in terms of
   $\gamma_{ij}$'s and the computation of the actual expression if any can be done
   by making use of the "baby-giant step" strategy for searching for a
   match of the form
```

$$h \mid \prod_{\substack{i=1\\ i\neq\lambda}}^{t} \Gamma_i = \gamma_{1\lambda}^{Y_1} \cdots \gamma_{s_\lambda\lambda}^{Y_{s\lambda}} \quad \text{for } 1 < \lambda < t.$$

```
6.  let $h = \prod_{i,j} \gamma_{ij}^{\alpha_{ij}}$;

7.  let $g_\mu = \prod_{i,j} \gamma_{ij}^{\beta_{ij}^{(\mu)}}$   for $1 < \mu < v$;

    comment The expression of $g_\mu$'s is done in a similar way with the
    computation of h in terms of $\gamma_{ij}$'s.

8.  Let S be the linear Diophantine system: $\sum_{\lambda=1}^{v} b_{ij}^{(\mu)} x_\lambda = \alpha_{ij} + z_{ij}|\gamma_{ij}| \;\forall i,j$
                                                                    over $\mathbf{Z}_{|G|}$;

9.  If S has a solution then
        comment Testing for solvability of s can be done by means of
        algorithm I.6.6.
        return "h belongs to G"
    else
        return "h does not belong to G"
end. □
```

PROPOSITION 3.2

Algorithm 3.1 correctly decides whether or not h belongs to G in $O(k\ n^4 \log n \log\log n)$ elementary operations

Proof

The system S of step 8 follows from the equations

$$h = \prod_{i,j} \gamma_{is}^{\alpha_{is}}$$

$$h = g_1^{x_1} \cdots g_v^{x_v}$$

$$g_\mu = \prod_{i,j} \gamma_{ij}^{\beta_{ij}^{(\mu)}} \quad \text{for } 1 < \lambda < v$$

which implies that

$$\prod_{i,j} (\gamma_{ij})^{\sum_{\lambda=1}^{v} b_{ij}^{(\mu)} x_\lambda} = \prod_{i,j} \gamma_{ij}^{\alpha_{is}}$$

Therefore it is not difficult to show the correctness of the algorithm.

Step 1 requires $O(kn \log^2 n)$ elementary operations by Proposition 2.3.

Step 2 requires t applications of algorithm II.5.1 and thus using Propositions II.5.2 and III.1.2 follows that it requires at most $O(k \sum_{i=1}^{t} |\Gamma_i|^{\frac{1}{2}} \log^4 n) = O(kn^{\frac{1}{2}} \log^4 n)$ group operations comprising $O(k\, n^{3/2} \log^4 n)$ elementary operations.

Step 3 requires $O(k\, n^4 \log n \log\log n)$ elementary operations by Proposition 2.6. The computation of an expression of $h/\prod_{i \neq \lambda} \Gamma_i$ in terms of $\gamma_{i\lambda}$'s requires $O(\prod_{i=1}^{s_\lambda} |\gamma_{i\lambda}|^{\frac{1}{2}}) = O(|\Gamma_\lambda|^{\frac{1}{2}})$ group operation for using the "baby-giant step" strategy. Hence the expression of h in terms of γ_{ij}'s requires $(\sum_{\lambda=1}^{t} |\Gamma_\lambda|^{\frac{1}{2}} n) = O(n^{3/2})$ elementary operations, using that a group operation requires $O(n)$ elementary operations.

The analysis of step 7 is similar to step 4 and thus step 7 requires $O(k\, n^{3/2})$ elementary operations.

By Proposition I.6.6 one can decide about the solvability of the system S in $O((v+n)n^2 M(\log |G|)) = O((n+\log|G|)n^2 M(\log|G|))$ elementary operations, using the fact that $v < \log |G|$, since the g_i's form a basis. Therefore by Proposition 1.3, $|G| < 3^{n/2}$, therefore follows that step 8 requires $O(n^4 \log n \log\log n)$ elementary operations. □

<u>COROLLARY 3.3</u>

Suppose that h is an element of the abelian group $G = \langle g_1,\ldots,g_k\rangle \subseteq S_n$. There exists an algorithm for computing an expression of h in terms of the generators of G in $O(kn^4 \log n \log\log n)$ elementary operations.

<u>Proof</u>

One can solve the system of step 8 of algorithm 3.1 using Hu's algorithm of Proposition I.6.6 and thus to compute the expression

$$h = g_1^{x_1} \ldots g_k^{x_k} . \quad \square$$

B. <u>INCLUSION TESTING</u>

Testing whether or not $G = \langle g_1,\ldots,g_k\rangle$ is a subgroup of $F = \langle f_1,\ldots,f_m\rangle$, where F and G are abelian subgroups of the symmetric group, can be done by testing whether or not g_i for $1 \leqslant i \leqslant k$ belong to F using algorithm 3.1; this computation requires $O((k+m)n^4 \log n \log\log n)$ elementary operations by Proposition 3.2.

In the case of multiple membership testing one need not repeat algorithm 3.1 for all elements for testing. A "multiple membership testing" algorithm can be formulated as follows:

(i) Test the first element for membership using algorithm 3.1. From an application of algorithm 3.1 one can obtain the following information:

(1) A complete basis for $\Gamma = \prod_{j=1}^{t} \langle\langle \gamma_{1j},\ldots,\gamma_{s_j j} \rangle\rangle$

(2) A complete basis for $G = \langle\langle g_1,\ldots,g_v \rangle\rangle$

(3) A triangular matrix T such that

$$T(\vec{x},\vec{y}) = [a_{ij}] \quad \text{over } \mathbb{Z}_{|G|}$$

which yields a particular solution of the system S at step 9 of algorithm 3.1.

(ii) For each of the remaining elements for testing compute an expression in terms of γ_{ij}'s. This can be done as the computation of h in step 6 and requires $O(n^{3/2})$ elementary operations. If one of these elements fails to be expressed in terms of γ_{ij}'s, then it is not a member of the group G.

(iii) Let $h = \prod_{i,j} \gamma_{ij}^{\alpha_{ij}}$ be the expression of one of the elements for testing. Then it suffices to prove that

$$T(\vec{x},\vec{y}) = [a_{ij}] \text{ over } \mathbb{Z}_{|G|}$$

is solvable in order to establish membership of h in G. This can be done in $O(n^2\, M(\log |G|)) = O(n^3 \log n \log\log n)$ elementary operations.

Using the above "multiple membership testing" algorithm, it is not difficult to show that the inclusion testing mentioned in the beginning of this subsection can be done in $O((kn^4 + mn^3) \log n \log\log n)$ elementary operations.

Furst et al in [18] gave an $O(kn^2 + n^6)$ elementary operations algorithm for membership testing in a permutation group; any additional membership testing requires merely $O(n^2)$ elementary operations. It is difficult to compare, Furst's et al and the above "multiple membership testing" algorithms, since their input is different. But the upper bound proved in [18] seems superior to our bound in terms of worst-case complexity. It may be practical to use the method described above for abelian groups, since its complexity depends on the orbits and this may give better running times that Furst's et al algorithm (see discussion below Proposition 2.7 paragraph (ii))

C. THE COMPUTATION OF THE STRUCTURE OF UNION OF GROUPS

In the case which the union of two abelian groups F and G subgroups of S_n is an abelian group, the computation of the structure of $\langle F \cup G \rangle$ can be done merely by means of algorithm 2.5 applied on the union of the generating sets F and G. This computation requires $O(k+m)n^4 \log n \log\log n)$ elementary operations by Proposition 2.7, where k and m are the cardinality numbers of the generating sets of F and G respectively.

4. THE COMPUTATION OF THE SRUCTURE OF THE INTERSECTION OF ABELIAN GROUPS

The problem of computing the intersection of two abelian subgroups of the symmetric group is considered in this section. The problem is investigated in two separate cases (i) when the union of the groups is an abelian group and (ii) when the union of the group does not respect commutativity.

In order to be able to separate the above two cases one needs an algorithm for testing the group $<F \cup G>$ for commutativity. If $F = <f_1,\dots,f_m>$ and $G = <g_1,\dots,g_k>$ are abelian subgroups of S_n, then by testing whether or not $f_i\, g_j = g_j f_i$ for $1 < i < m$ and $1 < j < k$ one can decide about the commutativity of $<F \cup G>$ in $O(km)$ group operations comprising $O(kmn)$ elementary operations.

A. <u>CASE OF <F ∪ G> ABELIAN</u>

An algorithm for computing the structure of $<F \cap G>$, given generating sets for the abelian groups F and G and the fact that $<F \cup G>$ is abelian, is presented below. An outline of the algorithm is the following; it computes the structure of $<F \cup G>$ and a generating set for $<F\cap G>$, then since $<F\cap G> \subseteq <F \cup G>$ and $<F \cup G>$ has a known structure one can apply algorithm II.7.4 in order to compute the structure of $<F \cap G>$.

ALGORITHM 4.1

INPUT : The generating sets $\{f_1,\dots,f_m\}$ and $\{g_1,\dots,g_k\}$ for the abelian group F and G respectively, where $F \subseteq S_n$, $G \subseteq S_n$ and the fact that $\langle F \cup G \rangle$

OUTPUT: The order and a complete basis for the abelian group $\langle F \cap G \rangle$.

```
begin
1.  compute a basis $\langle\langle h_1,\dots,h_\mu \rangle\rangle$ for $F \cup G$ using algorithm 2.5;
2.  $d \leftarrow |\langle F \cup G \rangle|$;
3.  compute $\alpha_{ij}$'s and $\beta_{ij}$'s: $f_i = h_1^{\alpha_{1i}} \dots h_\mu^{\alpha_{\mu i}}$ for $1 < i < m$,
    $g_j = h_1^{\beta_{1i}} \dots h_\mu^{\beta_{\mu i}}$ for $1 < i < k$
    comment This can be done by means of the algorithm of Corollary 3.3.
4.  Let S be the linear Diophantine system: $\sum_{j=1}^{m} \alpha_{ij} x_i = \sum_{j=1}^{k} \beta_{ij} z_j + y_i |h_i|$
    for $1 < i < \mu$ over $\mathbb{Z}_d$;
5.  solve S using the algorithm of Proposition I.6.6;
6.  let $(\vec{x},\vec{z},\vec{y}) = K.(t_1,\dots,t_v)^t$ be the set of all solutions of $S$;
    comment The matrix K is of dimension $(m+k+1) \times v$.
7.  $\delta_j \leftarrow \prod_{i=1}^{\mu} f_i^{k_{ji}}$ for $1 < j < v$;
8.  compute the complete structure of $\langle \delta_1,\dots,\delta_v \rangle$ using algorithm II.7.4
    comment Now $\langle F \cap G \rangle = \langle \delta_1,\dots,\delta_n \rangle$ is subgroup of $\langle F \cup G \rangle$ whose
    structure is known.
end. □
```

PROPOSITION 4.2

Algorithm 4.1 correctly computes the order and complete structure of the group <F ∩ G>.

Proof

The system S of step 4 is deduced from the equations

$$f_1^{x_1} \ldots f_m^{x_m} = g_1^{z_1} \ldots g_k^{z_k}$$

$$f_i = h_1^{\alpha_{1i}} \ldots h_\mu^{\alpha_{\mu i}} \quad \text{for } 1 < i < \mu$$

$$g_j = h_1^{\beta_{1j}} \ldots h_\mu^{\beta_{\mu j}} \quad \text{for } 1 < j < \mu$$

which implies that

$$\prod_{i=1}^{\mu} (h_i)^{\sum_{j=1}^{m} \alpha_{ij} x_j} = \prod_{i=1}^{\mu} (h_i)^{\sum_{j=1}^{k} \beta_{ij} z_j}$$

Moreover if the set of all solutions of the system S is that of step 6, then it follows that

$$\prod_{j=1}^{v} \left(\prod_{i=1}^{m} f_i^{k_{ji}} \right)^{t_j} = \prod_{j=1}^{v} \left(\prod_{i=1}^{k} g_i^{k_{j,i+m}} \right)^{t_j}$$

equivalent to

$$\prod_{j=1}^{v} \delta_j^{t_j} = \prod_{j=1}^{v} \left(\prod_{i=1}^{v} g_i^{k_{j,i+m}} \right)^{t_j}$$

which implies that $\{\delta_1, \ldots, \delta_v\}$ is a generating set for <F ∩ G>. Hence the correctness of the algorithm follows. □

PROPOSITION 4.3

Algorithm 4.1 requires $O((m+k)n^4 \log n \log\log n)$ elementary operations for the computation of the complete structure of $\langle F \cap G \rangle$

Proof

Step 1-2 requires $O((m+k)n^4 \log n \log\log n)$ elementary operations by Proposition 2.7.

Step 3 requires $O((m+k)n^4 \log n \log\log n)$ elementary operations for expressing the f_i's and g_j's in terms of the basis elements of G by Proposition 3.3.

Step 5 requires $O(m+k)n^3 \log n \log\log n)$ elementary operations by Proposition I.6.5 and the fact that $\mu < m+k$.

Step 7 requires $O(\log |F|)$ group operations for applying the "power algorithm" comprising $O(n^2)$ elementary operations, since $|F| < 3^{n/2}$ by Proposition 1.3.

Moreover one can observe that

$$v < m+k$$

and thus by Proposition II.7.5, step 6 requires $O(m+k)n^4 \log n \log\log n)$ elementary operations. □

B. THE CASE OF $\langle F \cup G \rangle$ NON-ABELIAN

Given generating sets for F and G abelian subgroups of the symmetric group, where the union forms a non-abelian group, two algorithms for computing the structure of their intersection are presented here.

An outline of the first algorithm is the following: one can compute bases for Γ and Δ , the groups formed by the direct product of the F^{Γ_i}'s and G^{Δ_i}'s for each orbit Γ_i and Δ_i of the group F and G respectively. Then using the reduction described in the Proposition 4.5 below, on Γ and Δ, one can compute groups $\Gamma^* \subseteq \Gamma$ and $\Delta^* \subseteq \Delta$ such that $\langle \Gamma^* \cup \Delta^* \rangle$ is abelian group and $\langle \Gamma^* \cap \Delta^* \rangle = \langle \Gamma \cap \Delta \rangle$. Moreover the groups $F^* = \langle F \cap (\Gamma \cap \Delta)\rangle$, $F' = \langle F \cup (\Gamma \cap \Delta)\rangle$, $G^* = \langle G \cap (F \cap (\Gamma \cap \Delta)\rangle$ and $G' = G \cup (F \cap (\Gamma \cap \Delta))\rangle$ are abelian. Hence one can use algorithm 4.1 for computing the structure of $\langle \Gamma \cap \Delta\rangle$, since $\langle \Gamma^* \cup \Delta^*\rangle$ is abelian, the structure of F^*, since F' is abelian and the structure of G^*, since G' is abelian. It is obvious that $G^*=\langle F \cap G\rangle$.

<u>PROPOSITION 4.5</u>

Suppose that $\Gamma_1,\dots,\Gamma_\lambda$ and $\Delta_1,\dots,\Delta_\mu$ are the orbits of the groups $F \subseteq S_n$ and $G \subseteq S_n$, and let $\Gamma_i^* \subseteq F^{\Gamma_i}$ for $1 < i < \lambda$, $\Delta_i^* \subseteq G^{\Delta_i}$ for $1 < i < \lambda$

$$\Gamma = \Gamma_1^* \times \dots \times \Gamma_\lambda^* \text{ and } \Delta = \Delta_1^* \times \dots \times \Delta_\mu^*$$

If $\quad \Gamma_1^{**} := \langle \Gamma_1^* \cap Z(\Gamma_1^* \cup (\Delta_1^*)^{\Gamma_1})\rangle$, $\Delta_1^{**} := \langle \Delta_1^* \cap Z(\Delta_1^* \cup (\Gamma_1^*)^{\Delta_1})$,

$$\Gamma^* := \Gamma_1^{**} \times \prod_{i=2}^{\lambda} \Gamma_i^* \text{ and } \Delta^* := \Delta_1^{**} \times \prod_{i=2}^{\mu} \Delta_i^* \text{ , then}$$

(i) The group $\langle \Gamma_1^{**} \cup \Delta_1^{**}\rangle$ is abelian

(ii) $\langle \Gamma^* \cap \Delta^*\rangle = \langle \Gamma \cap \Delta\rangle$

<u>**Proof**</u>

(i) Let $\gamma_1 \in \Gamma_1^{**}$ and $\delta_1 \in \Delta_1^{**}$. Assume that the restriction $\bar{\gamma}_1$ of γ_1 on Δ_1 does not belong to $(\Gamma_1^*)^{\Delta_1}$.

Then there is a cycle c of γ_1 which permutes some points $\{\delta_1',\ldots,\delta_k'\}$ of Δ_1 and some points $\{\gamma_1',\ldots,\gamma_m'\}$ of $\Gamma_1-\Delta_1$. Then it is not difficult to see that c does not commute with the elements of $(\Delta_1^*)^{\Gamma_1}$ and thus γ_1 does not commute with the elements of $(\Delta_1^*)^{\Gamma_1}$. Hence $\gamma_1 \notin Z(\Gamma_1^* \cup (\Delta_1^*)^{\Gamma_1})$ a contradiction.

Therefore $\bar{\gamma}_1 \in (\Gamma_1^*)^{\Delta_1}$, which implies γ_1 commutes with δ_1, since $\delta_1 \in z(\Delta_1^* \cup (\Gamma_1^*)^{\Delta_1})$.

(ii) Let $\alpha \in \langle\Gamma^* \cap \Delta^*\rangle$. Then it is obvious that $\alpha \in \langle\Gamma \cap \Delta\rangle$. Hence

$$\langle\Gamma^* \cap \Delta^*\rangle \subseteq \langle\Gamma \cap \Delta\rangle \qquad (4.1)$$

Now let $\alpha \in \langle\Gamma \cap \Delta\rangle$. Then

$$\alpha = \gamma_1 \ldots \gamma_\lambda \quad \text{for some } \gamma_i \in \Gamma_i,\ 1 < i < \lambda$$

Let $\tilde{\delta} \in (\Delta_1^*)^{\Gamma_1}$. Then there exists a $\delta \in \Delta_1^*$ such that

$$\delta = \tilde{\delta}\delta'$$

It is obvious that

$$\alpha\delta = \delta\alpha$$

which implies that

$$\gamma_1 \pi\gamma_1 \tilde{\delta}\delta' = \tilde{\delta}\delta'\gamma_1\pi\gamma_1$$

from which follows that

$$\gamma_1\tilde{\delta} = \tilde{\delta}\gamma_1.$$

Therefore $\gamma_1 \in Z(\Gamma_1^* \cup (\Delta_1^*)^{\Gamma_1})$ and thus $\alpha \in \Gamma^*$.

Moreover $\alpha = \delta_1 \dots \delta_m$ for some $\delta_i \in \Delta_i$, $1 \le i \le m$. Similarly it can be shown that $\delta_1 \in Z(\Delta_1^* \cup (\Gamma_1^*)^{\Delta_2})$, which implies $\alpha \in \Delta^*$. Therefore $\alpha \in \langle\Gamma^* \cap \Delta^*\rangle$ and thus

$$\langle\Gamma \cap \Delta\rangle \subseteq \langle\Gamma^* \cap \Delta^*\rangle \qquad (4.2)$$

The result follows from (4.1) and (4.2). $\square$

<u>ALGORITHM 4.6</u>

<u>INPUT</u> : Generating sets for the abelian groups F and G subgroups of the symmetric group with $\langle F \cup G\rangle$ non-abelian

<u>OUTPUT</u>: The complete structure and a complete basis for the intersection group $\langle F \cap G\rangle$

```
begin
1. compute a generating set for $F^{\Gamma_i}$ for each orbit $\Gamma_i$, $1 \le i \le \lambda$ of F;
2. compute a generating set for $G^{\Delta_i}$ for each orbit $\Delta_i$, $1 \le i \le \mu$ of G;
3. for i = 1 to $\lambda$ do
     begin
4.     for each $\Delta_j$ such that $\Delta_j \cap \Gamma_i = \emptyset$ do
         begin
5.         $F^{\Gamma_i} \leftarrow F^{\Gamma_i} \cap Z(F^{\Gamma_i} \cup (G^{\Delta_j})^{\Gamma_i})$;
6.         $G^{\Delta_j} \leftarrow G^{\Delta_j} \cap Z(G^{\Delta_j} \cup (F^{\Gamma_i})^{\Delta_j})$;
           comment The computation of the centre and the intersection
                   at the above two steps is done by testing each element
                   of $F^{\Gamma_i}(G^{\Delta_j})$ whether or not it satisfies the required
                   property.
7.       end.
```

```
 8.  end.
     $\Gamma^* \leftarrow F^{\Gamma_1} \times \dots \times F^{\Gamma_\lambda}$;
     $\Delta^* \leftarrow G^{\Delta_1} \times \dots \times G^{\Delta_\mu}$;
 9.  compute a complete basis for $<\Gamma^* \cap \Delta^*>$ using algorithm 4.1;
     comment The group $<\Gamma^* \cup \Delta^*>$ is abelian.
10.  compute a complete basis for $<F \cap \Gamma^* \cap \Delta^*>$ using algorithm 4.1;
     comment The group $<F \cup (\Gamma^* \cap \Delta^*)>$ is abelian.
11.  compute the complete structure of $<G \cap F \cap \Gamma^* \cap \Delta^*>$ using
     algorithm 4.1;
     comment The group $<G \cup (F \cap \Gamma^* \cap \Delta^*)>$ is abelian. Moreover the
             group $<G \cap F \cap \Gamma^* \cap \Delta^*> = <F \cap G>$;
end. □
```

<u>PROPOSITION 4.7</u>

Algorithm 4.6 correctly computes the canonical structure of $<F \cap G>$ in $O((m+k+n)n^4 \log n \log\log n)$ elementary operations.

<u>Proof</u>

The fact that $\Gamma^* \cup \Delta^*$ is abelian follows from the construction of Γ^* and Δ^* and Proposition 4.5. Therefore it is not difficult to see the correctness of the algorithm.

Steps 1-2 require $O((k+m)n \log^2 n)$ elementary operations by Proposition 2.1.

The computation of all elements of F^{Γ_i} and $(G^{\Delta_j})^{\Gamma_i}$ requires $O(|\Gamma_i| \log n)$ elementary operations and $O(|\Gamma_i|\ |\Delta_j| \log n)$ elementary operations respectively. Moreover the computation of the centre and the intersection at step 5 requires $O(|\Gamma_i|\ |\Delta_j|\ \log n)$ elementary operations for testing whether or not each element of F^{Γ_i} permutes with the elements of $(G^{\Delta_j})^{\Gamma_i}$. Similarly step 6 requires the same time.

Therefore loop 3-7 requires at most

$$O(\sum_{i=1}^{\lambda} |\Gamma_i|\ (\sum_{j=1}^{\mu} |\Delta_j|)\log n) = O(n^2 \log n)$$

elementary operations.

Using the fact that the generating sets of Γ^* and G^* is of cardinality at most $O(n)$ and Proposition 4.2, step 9 requires $O(n^5 \log n \log\log n)$ elementary operations.

Step 10 requires $O(k+n)n^4 \log n \log\log n)$ elementary operations by Proposition 4.2.

Step 11 requires $O(m+n)n^2 \log n \log\log n)$ elementary operations by Proposition 4.2.

The proposition follows from the above analysis. □

The skeleton of the second algorithm is similar to the first one; its main scope is the computation of two groups whose intersection contains the required intersection and their union is abelian. The computation of these two groups is done by means of fixed blocks of the required intersection group computed with the following algorithm.

<u>ALGORITHM 4.8</u>

<u>INPUT</u> : A set of generators $\{f_1,\ldots,f_k\}$ and $\{g_1,\ldots,g_m\}$ for the abelian groups F and G subgroups of S_n.

<u>OUTPUT</u> : A set of disjoint fixed blocks $\phi_1,\dots,\phi_v$ for the group $\langle F \cap G\rangle$ such that $I_n = \underset{i}{\cup}\, \phi_i$, $\phi_i = \Gamma_{j_i} \cap \Delta_{\ell_i}$ for some integers j_i and ℓ_i, for all $1 \le i \le v$, where Γ_j's are the orbits of F and Δ_j's are the orbits of G.

```
begin
1. compute the orbits $\Gamma_1,\dots,\Gamma_\lambda$ of the group F using algorithm 2.1;
2. compute the orbits $\Delta_1,\dots,\Delta_\mu$ of the group G using algorithm 2.1;
3. $L(j) \leftarrow \ell$: $j \in \Delta_\ell$, for $1 \le j \le \mu$ ; $M \leftarrow \emptyset$; $S_i \leftarrow \emptyset$ for $1 \le i \le \mu$;
4. for  $i = 1$ to $\lambda$ do
     begin
5.      $S_{L(j)} \leftarrow S_{L(j)} \cup \{j\}$ for every j in $\Gamma_i$;
6.      $M \leftarrow M \cup \{\{S_{L(j)}\}: S_{L(j)} \ne \emptyset\}$;
        $S_k \leftarrow \emptyset$ for $1 \le k \le \mu$;
7.   end
     Let $\phi_1,\dots,\phi_v$ be the elements of M;
end. □
```

<u>PROPOSITION 4.9</u>

Algorithm 4.8 correctly computes the set of disjoint fixed blocks ϕ_i of $\langle F \cap G\rangle$ in $O((k+m)n \log^2 n)$ elementary operations.

<u>Proof</u>

It is obvious that $\phi_i = \Gamma_{j_i} \cap \Delta_{\ell_i}$ for some integer j_i and ℓ_i, for all $1 \le i \le n$.

Now let $\alpha \in \phi_i$ and $g \in \langle F \cap G\rangle$. Since $\alpha \in \Gamma_{j_i}$, $\alpha \in \Delta_{\ell_i}$ it follows that $g(\alpha) \in \Gamma_{j_i}$ and $g(\alpha) \in \Delta_{\ell_i}$. Therefore $g(\alpha) \in \phi_i$ and thus ϕ_i is a fixed block of $\langle F \cap G\rangle$.

Steps 1-2 require $O((k+m)n \log^2 n)$ elementary operations by Proposition 4.2.

Step 3 requires $\sum_{i=1}^{\mu} |\Delta_i| = n$ elementary operations.

Step 5 requires $O(|\Gamma_i|)$ elementary operations for computing $L(j)$ for every $j \in \Gamma_i$ and for computing the S_i's. Step 6 merely requires $O(|\Gamma_i|)$ elementary operations. Therefore loop 4-7 requires $O(\sum_{i=1}^{\lambda} (|\Gamma_i|) = O(n)$ elementary operations. $\square$

<u>PROPOSITION 4.10</u>

Suppose that $\Gamma_1,\ldots,\Gamma_\lambda$ and $\Delta_1,\ldots,\Delta_\mu$ are the orbits of the abelian groups F and G subgroups of the symmetric group. Let $\Gamma = \prod_{i=1}^{\lambda} F^{\Gamma_i}$ and $\Delta = \prod_{i=1}^{\lambda} G^{\Gamma_i}$. If $\phi_1,\ldots,\phi_v$ are the fixed blocks of the group $\langle F \cap G\rangle$ defined in algorithm 4.8, then

$$\langle F \cap G\rangle \subseteq \langle \Gamma' \cap \Delta'\rangle$$

where

$$\Gamma' = \prod_{i=1}^{\lambda} F_i \text{ and } \Delta' = \prod_{i=1}^{\mu} G_i, \text{ with}$$

$$F_i = \langle g : g \in F^{\Gamma_i} \text{ and } g^{\phi_j} \text{ is well defined for } 1 \le j \le v\rangle$$

and $G_i = \langle g : g \in G^{\Delta_i}$ and g^{ϕ_j} is well defined for $1 \le j \le n\rangle$.

Proof

Let $h \in F \cap G$. Then there exists $\gamma_i \in F^{\Gamma_i}$ for $1 \le i \le \lambda$ and $\delta_i \in G^{\Delta_i}$ for $1 \le i \le \mu$, such that

$$h = \gamma_1 \cdots \gamma_\lambda = \delta_1 \cdots \delta_\mu$$

Moreover h^{ϕ_j} is well-defined for $1 \le j \le v$ and

$$h^{\phi_j} = \gamma_1^{\phi_j} \cdots \gamma_\lambda^{\phi_j} \qquad \text{for } 1 \le j \le v$$

where $\gamma_i^{\phi_j}$ for $1 \le i \le \lambda$ and $1 \le j \le v$ is well-defined, since

$$\phi_j \subseteq \Gamma_{k_j} \text{ for some } k_j \text{ and } \phi_j \cap \Gamma_i = \emptyset \text{ for every } i \neq k_j.$$

Therefore $\gamma_i \in F_i$ for $1 \le i \le \lambda$ and thus $h \in \Gamma'$.

Similarly it can be shown that $\delta_i \in G_i$ for $1 \le i \le \mu$ and thus $h \in \Delta'$.

Hence it follows that $h \in \Gamma' \cap \Delta'$. $\square$

PROPOSITION 4.11

Let Γ' and Δ' be as in Proposition 4.8,

$$\phi_j^* = \langle F^{\phi_j} \cap G^{\phi_j} \rangle \text{ and } \phi = \prod_{j=1}^{v} \phi_j^*$$

Then (i) the groups $\langle \Gamma' \cup \phi \rangle$ and $\langle \Delta' \cup \phi \rangle$ are abelian

(ii) The following holds

$$\langle F \cap G \rangle \subseteq \langle \Gamma' \cap \Delta' \cap \phi \rangle \quad .$$

<u>Proof</u>

(i) Let $\alpha \in \Gamma'$ with $\alpha = f_1 \ldots f_\lambda$ for $f_i \in F_i$, $1 \le i \le \lambda$ and $\beta \in \phi$ with $\beta = \theta_1 \ldots \theta_v$ for $\theta_i \in {}^*_i$, $1 \le i \le v$. From the definition of F_i it follows that

$$f_i = f_i^{\phi_1} f_i^{\phi_2} \ldots f_i^{\phi_v} \quad \text{for } 1 \le i \le \lambda.$$

It is obvious that $f_i^{\phi_j}$ commutes with θ_k for every i, j and $k \ne j$. Also $f_i^{\phi_j}$ commutes with θ_j, since $\theta_j \in F^{\phi_j}$ and $f_i^{\phi_j}$ is either 1 or $f_i^{\phi_j} \in (F^{\Gamma_i})^{\phi_j} = F^{\phi_j}$. Hence α and β commute and thus $\langle\Gamma' \cup \phi\rangle$ is abelian.

Similarly it can be shown that $\langle\Delta' \cup \phi\rangle$ is abelian.

(ii) Let $\alpha \in F \cap G$. Then $\alpha^{\phi_j} \in F^{\phi_j}$ and $\alpha^{\phi_j} \in G^{\phi_j}$. Therefore $\alpha^{\phi_j} \in \phi^*_j$. Since $\alpha = \alpha^{\phi_1} \alpha^{\phi_2} \ldots \alpha^{\phi_v}$, it follows that $\alpha \in \phi$. By Proposition 4.10, $\langle F \cup G\rangle \le \langle\Gamma' \cap \Delta'\rangle$, thus $\alpha \in \Gamma' \cap \Delta'$.

The proposition follows from above. □

<u>REMARK</u>

Since $\langle\Gamma' \cup \phi\rangle$ and $(\Delta' \cup \phi)$ are abelian, one can compute $\langle\Gamma' \cap \phi\rangle$ and $(\Delta' \cap \phi)$ using algorithm 4.1. Moreover one can compute $\langle(\Gamma' \cap \phi) \cap (\Delta' \cap \phi)\rangle = \langle\Gamma' \cap \Delta' \cap \phi\rangle$ using algorithm 4.1, since $\langle(\Gamma' \cap \phi) \cup (\Delta' \cap \phi)\rangle$ is abelian. □

<u>ALGORITHM 4.12</u>

<u>INPUT</u> : Generating sets $\{f_1,\dots,f_k\}$ and $\{g_1,\dots,g_k\}$ for the abelian groups F and G subgroups of S_n, with $\langle F \cup G\rangle$ non-abelian

<u>OUTPUT</u>: The canonical basis for $\langle F \cap G\rangle$

```
begin
1. compute $\phi_1,\dots,\phi_v$ using algorithm 4.8;
2. compute $F_1,\dots,F$ and $G_1,\dots,G_\mu$;
   comment This is done by direct computation.
   $\Gamma' \leftarrow F_1 \times \dots \times F_\lambda$;
   $\Delta' \leftarrow G_1 \times \dots \times G_\mu$;
3. compute $\phi_1^*,\dots,\phi_v^*$;
   comment This is done by direct computation of $(F^{\Gamma_{k_j}\phi_j}) \cap (G^{\Delta_{\ell_j}\phi_j}) = \phi_j^*$
   with $\phi_j \subset \Gamma_{k_j}$ and $\phi_j \leq \Delta_{\ell_j}$.
   $\phi \leftarrow \phi_1^* \times \dots \times \phi_v^*$;
4. compute a basis for $\Gamma' \cap \phi$ using algorithm 4.1;
   comment The group $\Gamma' \cup \phi$ is abelian.
5. compute a basis for $\Delta' \cap \phi$ using algorithm 4.1;
   comment The group $\Delta' \cup \phi$ is abelian.
6. compute a basis for $\Gamma' \cap \Delta' \cap \phi$ using algorithm 4.1;
   comment The group $(\Delta' \cap \phi) \cup (\Gamma' \cap \phi)$ is abelian
7. compute a basis for $F \cap \Gamma' \cap \Delta' \cap \phi$ using algorithm 4.1;
   comment The group $F \cup (\Gamma' \cap \Delta' \cap \phi)$ is abelian
8. compute a canonical basis for $G \cup F \cap \Gamma' \cap \Delta' \cap \phi)$ is abelian and
         $G \cap F \cap \Gamma' \cap \Delta' \cap \phi = F \cap G$.
end. □
```

PROPOSITION 4.13

Algorithm 4.12 correctly computes the canonical structure of $\langle F \cap G \rangle$ in $O(k+m+n)n^4 \log n \log\log n)$ elementary operations.

Proof

The correctness of the algorithm follows from Proposition 4.10, 4.11 and 4.2.

Step 1 requires $O(kn \log^2 n)$ elementary operations by Proposition 4.9. The computation of F_i requires $O(|\Gamma_i|^2)$ elementary operations for computing all elements of F^{Γ_i} and $O(|\Gamma_i| \log(\sum_{j=1}^{\nu} \phi_j)) = O(|\Gamma_i| \log n)$ elementary operations for testing each of them whether or not its restriction to ϕ_s for $1 < s < \mu$ is a permutation. Therefore step 2 requires $O(\sum_{i=1}^{\lambda} |\Gamma_i|^2) = O(n^2)$ elementary operations for computing the F_i and similarly $O(\sum_{i=1}^{\mu} |\Delta_i|^2) = O(n^2)$ elementary operations for computing the G_i's.

The computation of $(F^{\Gamma_{k_j}})^{\phi_s}$ and $(G^{\Delta_{\ell_j}})^{\phi_j}$ for $1 < j < \nu$ requires $O(|\Gamma_{k_j}|^2 + |\Delta_{\ell_j}|^2)$ elementary operations; one can show that $\phi_j^* = (F^{\Gamma_{k_j}})^{\phi_j} \cap (G^{\Delta_{\ell_j}})^{\phi_j}$ using the fact that $(F^{\Gamma_{k_j}})^{\phi_j} \neq 1$ and $(G^{\Delta_{\ell_j}})^{\phi_j}$ using the fact that $(F^{\Gamma_{k_j}})^{\phi_j} \neq 1$ and $(G^{\Delta_{\ell_j}})^{\phi_j} \neq 1$ for some k_j and ℓ_j only, since ϕ_j intersects just one orbit of F and one orbit of G. Therefore the computation of the ϕ_s^*'s can be done by sorting the elements of $(F^{\Gamma_{k_j}})^{\phi_j}$ (they are at most $|\phi_j|$) and testing for all g in $(G^{\Delta_{\ell_j}})^{\phi_j}$ (the are at most $|\phi_j|$) whether or not g belongs to $(F^{\Gamma_{k_j}})^{\phi_j}$. Therefore one can compute ϕ in $O(\sum_{i=1}^{\lambda} |\Gamma_i|^2 + \sum_{i=1}^{\mu} |\Delta_i|^2 + \sum_{j=1}^{\nu} |\phi_j| \log|\phi_j| \log n) = O(n^2)$ elementary operations.

Using Proposition 4.2 one can conclude that step 5-8 requires $O(k+m+n)n^4 \log n \log\log n)$ elementary operations. □

5. APPLICATIONS

A. Group Theory

The subsection deals with a negative result about applications of the algorithms for permutation groups on abstract groups. The gap between the upper bound of $O(|G|^{1/2+\epsilon})$ on the time required for computations (like order, canonical structure of a group) in abstract abelian groups represented by a set of generators and the upper bound of $O(\log^c |G|)$ for some constant on the time required for similar computations in abelian permutation groups is far too big. As a bridge between permutation groups and abstract groups is Cayley's theorem (see Hall [22]) saying that every group is isomorphic to a permutation group. It will be shown that given a set of generators for a group G, the time required by an optimal algorithm for computing the isomorphic image of G into a permutation group is $\Omega(|G|)$. Moreover, Cayley's construction is shown to be optimal within a polynomial factor.

Suppose that $G = \langle g \rangle$ cyclic elementary p-group for p prime. Then there exists a permutation π such that $\langle \pi \rangle \cong G$. Then the order of π is $|G| = p$ which implies that π permutes at least $|G|$ symbols. Therefore an optimal algorithm for computing an isomorphic permutation group to G requires $\Omega(|G|)$ elementary operations in the worst-case.

Moreover Cayley's construction of the isomorphic group requires $|G|$ elementary operations, given $G = \langle g_1,\dots,g_k \rangle$, The construction is the following: Define a permutation r_{g_i} for $1 < i < n$ acting on $\{g : g \in G\}$ such that $r_{g_i}(g) = gg_i \ \forall g \in G$. Then it is not difficult to show that

$$\langle r_{g_1},\dots,r_{g_k} \rangle \cong G.$$

It is not difficult to see that the computation of r_{g_1} requires $O(|G|)$ and then the isomorphic image of G requires $O(k|G|\xi)$ elementary operations. Assuming that $k \approx \xi \approx O(\log^c|G|)$ for some constant $c > 0$, Cayley's method is optimal within a factor of $O(\log^c|G|)$.

Moreover in the case which G is abelian one can construct a complete basis $<<b_1,\ldots,b_n>>$ for G in $O(|G|^{1/2+\epsilon}\xi)$ elementary operations. If c_i are disjoint cycles permuting elements in $\{1,\ldots,|G|\}$ and c_i is of length $|b_i|$ for $1 < i < n$, then

$$<c_1,\ldots,c_n> \approx G.$$

This computation requires $O(|G|)$ elementary operations and thus is optimal.

Therefore one can conclude that the representation of the group as permutation groups is powerful for computing the order of the group but the computation of an isomorphic permutation group to a given abstract group is intractable.

B. Graph Theory

The following two problems are polynomial time equivalent:

Group Intersection problem: Given two permutation groups, subgroups of the symmetric group S_n, compute a generating set for their intersection.

Graph Isomorphism Problem: Given two graphs determine whether or not they are isomorphic and if so, construct an isomorphic from the one to another.

There is no known polynomial time algorithm for both of the problems mentioned above. Hoffman in [24] suggests that their complexity lies between P and NP and that they do not seem to be candidates of the

It is not difficult to see that the computation of r_{g_i} requires $O(|G|)$ and then the isomorphic image of G requires $O(k|G|\xi)$ elementary operations. Assuming that $k \approx \xi \approx O(\log^c|G|)$ for some constant $c > 0$, Cayley's method is optimal within a factor of $O(\log^c|G|)$.

Moreover in the case which G is abelian one can construct a complete basis $\langle\langle b_1,\dots,b_n\rangle\rangle$ for G in $O(|G|^{1/2+\varepsilon}\xi)$ elementary operations. If c_i are disjoint cycles permuting elements in $\{1,\dots,|G|\}$ and c_i is of length $|b_i|$ for $1 < i < n$, then

$$\langle c_1,\dots,c_n\rangle \approx G.$$

This computation requires $O(|G|)$ elementary operations and thus is optimal.

Therefore one can conclude that the representation of the group as permutation groups is powerful for computing the order of the group but the computation of an isomorphic permutation group to a given abstract group is intractable.

B. Graph Theory

The following two problems are polynomial time equivalent:

Group Intersection problem: Given two permutation groups, subgroups of the symmetric group S_n, compute a generating set for their intersection.

Graph Isomorphism Problem: Given two graphs determine whether or not they are isomorphic and if so, construct an isomorphic from the one to another.

There is no known polynomial time algorithm for both of the problems mentioned above. Hoffman in [24] suggests that their complexity lies between P and NP and that they do not seem to be candidates of the

NP-complete class of problems (for definitions of P, NP and NP-complete, see Aho et al. [2]).

The graph isomorphism problem has been considered under constraints e.g. the graph is of bounded valence (see Luks [38]). The algorithms described in the previous sections of this chapter have no direct application on the graph isomorphism; they (algorithm 4.12.?) may help for the solution of the graph isomorphism problem under the constraint that the automorphism group of the graph is isomorphic.

C. Chemistry

It is well-known the connection of the representation of the molecules and the symmetric group (see [3]). Many of these representations form an abelian group (see [11], e.g. translation group). The computation of the structure of these groups aids to the computation of the orbitals of the atoms (see [3]).

REFERENCES

[1] Adleman, L.M. "On distinguishing prime numbers from composite numbers" 21st Annual IEEE Symposium on Foundations of Computer Science (1980).

[2] Aho,Hopcroft,Ullman, "The Design and Analysis of Computer Algorithms" Addison-Wesley (1974)

[3] Altmann, S.L., "Induced representations in crystals and molecules" Academic Press (1977)

[4] Apostol, T.M., "Introduction to Analytic Number Theory" Springer-Verlag (1976).

[5] Bradley, G.H., "Algorithms for Hermite and Smith Normal Matrices and linear Diophantine equations" Maths of Comp. v.25, 116 (1971).

[6] Burgess, D.A., "The distribution of quadratic residues and non-residues", Mathematica 4 (1957), pp. 106-112.

[7] Cassels, J.W.S., "An introduction to the geometry of numbers" Springer-Verlag (1959)

[8] Chou, T.J., Collins, G.E., "Algorithms for the solution of systems of linear diophantine equations" SIAM J. Computing 11 (1982)

[9] Cook, S.A., "The complexity of theorem-proving procedures" Proc. 3rd ACM Symp. on Th. of Comp. (1971) pp. 151-158.

[10] Coppersmith, D., Winograd G., "On the asymptotic complexity of matrix multiplications" in 22-nd Annual symposium on FOCS (1981).

[11] Cotton, F.A., "Chemical Applications of Group Theory" John Wiley (1963).

[12] Dixon, J.D., "Problems in Group Theory " Dover, New York (1973)

[13] Dixon, J.D., "Factorization and Primality Tests" (to appear)

[14] Diffie, W., Helman, M., "New directions in cryptography" IEEE Trans. Inform. Theory IT-221 6 (Nov 1976), 644-654.

[15] Floyd, R.W., "Algorithm 245 : treesort 3" Comm. ACM 7:12, p. 701.

[16] Frumkin, M.A., Polynomial time algorithms in the theory of linear diophantine equations", M. Karpinski ed., Fundamentals on Computation Theory, Springer Lecture Notes in CS 56 (1977), pp. 386-392.

[17] Frumkin, M.A., "An application of modular arithmetic to the construction of algorithms for solving systems of linear equations" Soviet Math. Doxl. Vol. 17 (1976) No. 4.

[18] Furst, M., Hopcroft, J., Luks, E., "Polynomial time algorithms for permutation groups" in 21st Annual Symposium on FOCS (1980) pp. 36-41.

[19] Gantmacher, F.R.,"Matrix Theory", Vol. I, Chelsea (1960)

[20] Garey, M.R., Johnson, "Computers and Intracrability - A Guide to the Theory of NP-completeness" W.H. Freeman, San Francisco (1979).

[21] Gauss, C.F.,"Disqusitiones Arithmeticae", English transl. Yale Univ. Press, (1966).

[22] Hall, M.Jr, "The theory of Groups" McMillan New York (1959)

[23] Hermite, C.,"Sur l'introduction des variables continus dans la theorie des no bres", J.R. Augeur. Math., 41 (1851), pp. 191-216.

[24] Hoffman, C.M., "Group Theoretic Algorithms and Graph Isomorphism" Springer Verlag (1982).

[25] Hu, T.C., "Integer Programming and Network Flows" Addison-Wesley (1969).

[26] Iliopoulos C.S. "Composition and Characters of finary quadratic forms" Warwick Univ., Theory of Comp. Rep. 37 (1981).

[27] Ilipoulos, C.S., "Worst-case complexity bounds on Algorithms for computing the canonical structure of infinite abelian groups and solving systems of linear Diophantine equations" Warwick Univ. Theory of Computation Rep. 50 (1983).

[28] Iliopoulos, C.S., "Worst-case complexity bounds on algorithms for computing the canonical structure of finite abelian groups and the Hermite and Smith normal forms of an integer matrix" in Theory of Computation, Warwick Univ. Rep. 49, (1983).

[29] Iliopoulos, C.S., "On the computation of the structure of an abelian group represented by a set of defining relations" Theory of Computation, Warwick Univ. Rep. 40 (revised) (1982).

[30] Iliopoulos, C.S., "Algorithms in the theory of integral binary quadratic forms" M.Sc. Thesis, Warwick Univ. (1981).

[31] Iliopoulos, C.S., "Analysis of an Algorithm for composition of binary quadratic forms" J. Algorithms 3, 157-159 (1982).

[32] Ja' Ja', J., "Computation of Algebraic Functions" in 22nd Annual Symposium on FOCS (1981).

[33] Kannan, R., Backem, A., "Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix" SIAM J. Computing, 8 (1979) pp. 499-507.

[34] Klyuyen, V.V., Kokovkin-Shchebak, N.I., "On the minimization of the number of arithmetic operations for the solution of linear Algebraic Systems of equations" (translated by G.J. Tee) Stanford Univ. Rep. 24 (1965).

[35] Knuth, D.E., "Seminumerical Algorithms" Addison-Wesley 1969.

[36] Lagarias, J.C., and Odlyzko, A.M., "Effective versions of the Chebotarev density theorem" in Algebraic Number Fields (A. Fröhlich, Ed.) Proc. 1973 Durham Symposium, Academic Press (1974).

[37] Lancaster, P., "Theory of Matrices", Academic Press (1969).

[38] Luks, E., "Isomorphism of Bounded Valence can be tested in Polynomial Time" Proc. in 21st IEEE symp. on FOCS (1980) pp. 42-49.

[39] McClellan, M.T., "The exact solutions of systems of linear Equations with Polynomial Coefficients" J. of ACM V.20, 4 (1973), pp.563-588.

[40] Miller, G.L., "Riemann's Hypothesis and tests for primality" J. Comput. System Sci. (13) (1976) 300-317.

[41] Montgomery, H.L., "Topics in Multiplicative Number Theory" Lecture Notes in Math. 227, Springer, Berlin (1971).

[42] Newman, M., "Integral Matrices" Academic Press New York (1972).

[43] Papadimitriou, C.H., Steiglitz, K., "Compinatorial Optimazation: Algorithms and Complexity" Prentice-Hall (1982).

[44] Sattler, J., Schnorr, C.P., "Generating random walks in groups (to appear).

[45] Schönhage, A., Strassen, V., "Schnelle Multiplication grosser Zachlen" computing 7 (1972) pp. 281-292.

[46] Schönhage, A., "Schnelle Berechnung von Kerren bruchentwicklungen" Acta Informatica 1 (1971), pp. 159-144.

[47] Shanks, D., "Class number, a theory of factorization and genera" in Proc. Symp. Pure Math., Vol. 20, AMS (1971), pp. 45-440.

[48] Smith, D.A., "A Basis algorithm for finitely generated abelian groups" in Math. Algor. V.1, (1966) pp. 13-66.

[49] Smith, H.J., "Report on the Theory of Numbers" Chelsea, New York, (1965).

[50] Sims, C.C. "The influence of computers in Algebra" Proc. of Symposia in Applied Mathematics, 20 (1974), pp. 13-30.

[51] Strassen, V. "Gaussian elimination is not Optimal". Numer. Math. 13, 354-356 (1969).

[52] Wielandt, H., "Finite permutation groups" Academic Press (1964).

[53] Williams, J.W.J., "Algorithm 232: Hempsort" Comm. ACM. 7:6 pp.347-348.

[54] Winogrand, S., "On the number of multiplications necessary to compute certain functions", Comm. Pure and Applied Maths 23, (1970) pp. 165-179.

[55] Zadeh A. and Pola, K.E., "Systems Theory", McGraw-Hill (1969).