Quantum Information and Computation, Vol. 1, No. 3 (2001) 26–32

DECOMPOSING FINITE ABELIAN GROUPS

KEVIN K. H. CHEUNG
Department of Combinatorics and Optimization, University of Waterloo
Waterloo, Ontario, N2L 3G1, Canada

MICHELE MOSCA
Department of Combinatorics and Optimization, University of Waterloo
Waterloo, Ontario, N2L 3G1, Canada

Received March 21, 2001
Revised September 20, 2001

This paper describes a quantum algorithm for efficiently decomposing finite Abelian groups into a product of cyclic groups. Such a decomposition is needed in order to apply the Abelian hidden subgroup algorithm. Such a decomposition (assuming the Generalized Riemann Hypothesis) also leads to an efficient algorithm for computing class numbers (known to be at least as difficult as factoring).

Keywords: Abelian groups, quantum, algorithm

Communicated by: R Cleve and J Watrous

1. Introduction

The work by Shor [1] on factoring and finding discrete logarithms over $\mathbb{Z}_n^*$ can be generalized to solve the Abelian Hidden Subgroup Problem (see for example [2, 3, 4]). These algorithms find the hidden subgroup of a function $f : G \to S$, where $G = \mathbb{Z}_{N_1} \times \cdots \mathbb{Z}_{N_l}$, for some integers $N_1, N_2, \ldots, N_l$. Any Abelian group G is isomorphic to such a product of cyclic groups. However, it is not always known how to find such an isomorphism efficiently. Consider for example the group $G = \mathbb{Z}_N^*$, the multiplicative group of integers modulo N. This is an Abelian group in which we can compute efficiently. Yet no known classical algorithm can efficiently find its decomposition into a product of cyclic groups. (By finding a decomposition of G into a product of cyclic groups, we mean finding an isomorphism between G and a product of cyclic groups.) However, to apply the well-known quantum algorithm for solving the Abelian hidden subgroup problem for a function $f : G \to S$, we *require* a decomposition of G into a product of cyclic groups. This problem is first pointed out in [4], where they also point out the remedy. This remedy was later sketched in more detail in [5], and used in the subsequent generalizations of Watrous [6]. This paper details fully the method sketched in [5].

Consider also the class group of an imaginary quadratic number field. In imaginary quadratic number fields we can efficiently find unique representatives of each group element (see Section 5.4.2 of [12]). This group is also Abelian, and finding its decomposition into a product of finite cyclic groups will give us the size of the group and therefore the class number of the imaginary quadratic number field. As Watrous [6] points out, assuming the Generalized

Riemann Hypothesis we can apply the algorithm in this paper and efficiently find these class numbers (a problem known to be at least as hard as factoring). If we consider real quadratic number fields then it is not known how to efficiently test the equality of two group elements (see Section 5.6 of [7] or [8]) and thus we cannot apply these methods.

2. Group Theory Preliminaries

Let G be an Abelian group. We will use multiplicative notation for the group G. Recall that G is said to be *cyclic* if there exists $a \in G$, such that $G = \{a^n \mid n \in \mathbb{Z}\}$. Here, we call a a *generator* of G. We call $H \subseteq G$ a *subgroup* of G if H is a group under the operation induced by G. In this case, we write $H \leq G$.

Let $a \in G$. If $a^n = e$ for some $n \in \mathbb{N}$, then a is said to have *finite order*. The smallest such n is called the *order* of a, denoted by $ord(a)$. It is easy to see that the elements in $\{e, a, a^2, ..., a^{n-1}\}$ form a subgroup. We call this subgroup, denoted by $\langle a \rangle$, the *the cyclic subgroup generated by a*.

If G_1, G_2 are subgroups of G such that $G_1 \cap G_2 = \{e\}$, the set $\{a_1 a_2 \mid a_1 \in G_1, a_2 \in G_2\}$, denoted by $G_1 \bullet G_2$, is called the *(internal) direct product* of G_1 and G_2. Clearly, $G_1 \bullet G_2 \leq G$. If there exist subgroups $G_1, ..., G_k$ of G such that $G = G_1 \bullet \cdots \bullet G_k$, we say that G can be expressed as (or decomposed into) a direct product of $G_1, ..., G_k$. We call $G_1 \bullet \cdots \bullet G_k$ a direct product representation of G.

Let G be a group and p be a prime number. Let $P \leq G$. Then P is called a *Sylow p-subgroup* of G if $|P| = p^\alpha$ for some $\alpha \in \mathbb{N}$ such that p^α divides $|G|$ but $p^{\alpha+1}$ does not.

We first quote three classical results without proof. The interested reader can refer to a standard text on group theory (e.g. [9]).

Theorem 1 *A finite Abelian group can be expressed as a direct product of its Sylow p-subgroups.*

Theorem 2 *Let K be a subgroup of $G = G_{p_1} \bullet \cdots \bullet G_{p_l}$ where G_{p_i} is a Sylow p_i-subgroup for $i = 1, ..., l$ and $p_1, ..., p_l$ are distinct primes. Then there exists $K_{p_i} \leq G_{p_i}$, $i = 1, ..., l$, such that $K = K_{p_1} \bullet \cdots \bullet K_{p_l}$.*

Theorem 3 *(Fundamental Theorem of Finite Abelian Groups). Any finite Abelian group can be expressed as a direct product of cyclic subgroups of prime power order.*

Once one has expressed a finite Abelian group G as a direct product of cyclic subgroups $G = \langle g_1 \rangle \bullet \cdots \bullet \langle g_l \rangle$, then one knows that G is isomorphic to the product of cyclic groups $\mathbb{Z}_{N_1} \times \cdots \mathbb{Z}_{N_l}$, where N_j is the order of g_j. The element $g \in G$ corresponds to the unique l-tuple $(x_1, x_2, \ldots, x_l)^T \in \mathbb{Z}_{N_1} \times \cdots \mathbb{Z}_{N_l}$ satisfying $g = g_1^{x_1} g_2^{x_2} \cdots g_l^{x_l}$. We will discuss the efficiency of establishing and computing this isomorphism at the end of Section 4.

3. Some Technical Results

Lemma 1 *Let q be a prime power, G a finite Abelian group having order dividing q, and $\{a_1, ..., a_m\}$ a generating set for G. Define the mapping $\tau : \mathbb{Z}_q^m \to G$ by $\tau(x_1, ..., x_m) =$*

$a_1^{x_1} \cdots a_m^{x_m}$. *Then* τ *is a surjective homomorphism, and thus* $\mathbb{Z}_q^m/K \cong Im(\tau) = G$, *where* K *denotes the kernel of* τ.

Proof. Mapping τ is well-defined since the order of each a_i divides q. It is surjective since every element in G can be written in the form $a_1^{x_1} \cdots a_m^{x_m}$ for some integers $x_1, ..., x_m$ satisfying that $0 \leq x_i < q$ for all $i = 1, ..., m$. It is a homomorphism since $\tau(\mathbf{x} + \mathbf{y}) = \tau(\mathbf{x})\tau(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^m$ $\square$.

A nonsingular integral matrix U is called *unimodular* if U has determinant ± 1. Clearly, U is unimodular if and only if U^{-1} is unimodular. More on unimodular matrices can be found in [10].

The following operations on a matrix are called *elementary (unimodular) column (row) operations*: 1. exchanging two columns (rows); 2. multiplying a column (row) by -1; 3. adding an integral multiple of one column (row) to another column (row).

For an integral $m \times n$ matrix A, size(A) is defined to be $mn + \sum_{i,j}(1 + \lceil \log_2(|(A_{ij})| + 1) \rceil)$.

Lemma 2 *For any integral matrix* A, *one can find in time polynomial in size(A) using elementary row and column operations unimodular matrices* U *and* V *such that* $UAV = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}$ *where* $D = Diag(d_1, ..., d_k)$ *with positive integers such that* $d_1|d_2|...|d_k$.

Proof. See [11] $\square$.

For a matrix M, let intcol(M) denote the set of vectors that can be obtained by taking integral linear combinations of columns of M.

Lemma 3 *Let* q *be a prime power and* k *a positive integer. Let* K *be any subgroup of* $\mathbb{Z}_q^k$. *Given any generating set for* K, *we can in time polynomial in* k, $\log_2(q)$, *and the size of the generating set, find elements* $y_1, ..., y_l \in \mathbb{Z}_q^k$, *so that* $\mathbb{Z}_q^k/K = \langle \mathbf{y}_1 + K \rangle \oplus \cdots \oplus \langle \mathbf{y}_l + K \rangle$.

Proof. (Adapted from Algorithm 4.1.3 in [12].) In this paper, we assume that the elements of $\mathbb{Z}_q^k$ are column tuples. Clearly, $\mathbf{e}_1, ..., \mathbf{e}_k$ generate $\mathbb{Z}_q^k$ where $\mathbf{e}_i$ is the ith column of $\mathbb{I}_k$, the $k \times k$ identity matrix. Hence, $\mathbf{e}_1 + K, ..., \mathbf{e}_k + K$ generate $\mathbb{Z}_q^k/K$. Let $\mathbf{x} = (x_1, ..., x_k)^T \in \mathbb{Z}^k$. Note that $\sum_{j=1}^k x_j(\mathbf{e}_j + K) = K$ if and only if $\mathbf{x} \in$ intcol(M) where $M = [q\mathbb{I}_k|A]$ with A being a matrix whose columns generate K. (Here, the entries of A are treated as elements of $\mathbb{Z}$ rather than $\mathbb{Z}_q$.)

By Lemma 2, we can find in time polynomial in size(M) (i.e. polynomial in k, $\log_2(q)$, and size(A)) unimodular matrices U and V such that $U^{-1}MV = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}$ where D is a diagonal matrix with diagonal entries $d_1, ..., d_m$ such that $d_1|d_2|...|d_m$. Since V is unimodular, we see that $(x_1, ..., x_k)^T \in$ intcol(M) if and only if $(x_1, ..., x_k)^T \in$ intcol(MV). Hence, $\sum_{j=1}^k x_j(\mathbf{e}_j + K) = K$ if and only if $(x_1, ..., x_k)^T \in$ intcol(MV).

For each $i = 1, ..., k$, set $\mathbf{a}_i = (U_{1i}, ..., U_{ki})^T$. Then,

$$\begin{aligned}\sum_{j=1}^{k} x_j(\mathbf{a}_j + K) = K \quad &\Leftrightarrow \quad \sum_{j=1}^{k}(x_1U_{j1} + x_2U_{j2} + \cdots + x_kU_{jk})(\mathbf{e}_j + K) = K \\ &\Leftrightarrow \quad U(x_1, ..., x_k)^T \in \text{intcol}(MV) \\ &\Leftrightarrow \quad (x_1, ..., x_k)^T \in \text{intcol}(U^{-1}MV).\end{aligned}$$

Since $U^{-1}MV = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix}$, we see that $\sum_{j=1}^{k} x_j(\mathbf{a}_j + K) = K$ if and only if $d_i | x_i$ for $i = 1, ..., m$, and $x_i = 0$ for $i = m+1, ..., k$. Note that we must have $m = k$. Otherwise, $\mathbf{a}_k + K$ will have infinite order, contradicting the fact that $\mathbb{Z}_q^k/K$ is a finite group.

Note that if $d_j = 1$, then $\mathbf{a}_j + K = K$. Let r be the smallest index such that $d_r > 1$. Set $\mathbf{y}_i = \mathbf{a}_{i+r-1}$ for $i = 1, .., l$ where $l = m - r + 1$. It is clear that $(\mathbf{y}_1 + K), ..., (\mathbf{y}_l + K)$ still generate $\mathbb{Z}_q^k/K$ and $\mathbf{y}_i + K$ has order d_{i+r-1} for $i = 1, ..., l$. Therefore, if $0 \leq v_i < \text{ord}(\mathbf{y}_i + K)$, then $v_1(\mathbf{y}_1 + K) + \cdots + v_l(\mathbf{y}_l + K) = K$ implies that $v_i = 0$ for all i. Hence, $\mathbb{Z}_q^k/K = \langle \mathbf{y}_1 + K \rangle \oplus \cdots \oplus \langle \mathbf{y}_l + K \rangle$ □.

Lemma 4 *There exists a quantum algorithm that given integers q and k, and a function f defined on $\mathbb{Z}_q^k$ fulfilling the hidden subgroup promise with respect to some subgroup K (here, $K = \{\mathbf{s} \in \mathbb{Z}_q^k \mid f(\mathbf{x}) = f(\mathbf{x}+\mathbf{s})$ for all $\mathbf{x} \in \mathbb{Z}_q^k\}$), outputs a generating set for K with probability $1 - \epsilon$ in time polynomial in k, $\log(1/\epsilon)$, $\log(q)$, and in time required to evaluate f.*

Proof. See, for instance, [4] □.

Theorem 4 *Let a be an element of a group G. The order r of a can be found in bounded-error quantum polynomial time.*

Proof. See [13] (or [4]) □.

Using Theorem 4, one can deduce the result by Shor [1].

Theorem 5 *Factoring can be solved in bounded-error quantum polynomial time.*

4. Decomposing Abelian Groups

By Theorem 3, we know that we can decompose a finite Abelian group into a direct product of cyclic groups of prime power order. This problem was discussed briefly in [5]. We will make four reasonable assumptions on the group G. These assumptions will be sufficient for our algorithm to work.

1. We have a unique binary representation for each element of G, and this representation has length in $O(\log(|G|))$.

2. Using the binary representation, for any $a \in G$, we can efficiently construct a quantum network for implementing $U_a : |y\rangle \rightarrow |ay\rangle$.

3. We can efficiently find a generating set for G. That is, in time polynomial in $\log(|G|)$ and $\log(1/\epsilon)$, we can find elements $a_1, a_2, \ldots, a_k$ such that, with probability at least $1-\epsilon$, the elements generate G.

4. The orders of the generators are of prime power order.

The first assumption is essential for quantum interference to occur, since the several computational paths leading to the same group element will interfere quantumly if and only if the group element is represented by the same physical state (i.e. string) regardless of which computational path is taken. So, for example, this algorithm might not work for groups where we do not know how to test the equality of two group elements.

To meet the third assumption, it suffices to have an upper bound of 2^m on the size of the groups we work with for some $m \in \Theta(\log |G|)$ and that we can efficiently sample elements of G almost uniformly at random. (If we do not have such a bound, we can easily devise a procedure that tries an increasing sequence of values for m and still has expected running time in $O(\text{poly} \log |G|)$). Let H be a proper subgroup of G. Then there are at least two cosets of H. If we randomly sample an element x from G, then with probability at least $1/2$, the subgroup spanned by x and H will have size at least twice that of H. Hence, it takes an expected number of at most $(1/(1/2))m = 2m$ samples to obtain a generating set $\mathcal{F}$ for G and therefore $2m + c\sqrt{m}$ samples will find a generating set with probability in $1-\epsilon^c$ for some $\epsilon \in (0,1)$ (by a Chernoff bound.)

We may also assume that the orders of the elements are a prime power. Let a be an element in $\mathcal{F}$ with order rs where $(r,s) = 1$, $r \neq 1$ and $s \neq 1$. Note that r and s can be determined efficiently as a result of Theorem 4 and Theorem 5. By the Euclidean algorithm, we can find integers α, β such that $\alpha r + \beta s = 1$. Thus $(a^r)^\alpha (a^s)^\beta = a$. Hence, replacing a with a^r and a^s still leaves us with a generating set. We repeat this procedure until each element in $\mathcal{F}$ has prime power order.

In order to satisfy the second assumption, note that if we know the order r of a, we can efficiently compute $a^{-1} = a^{r-1}$ and therefore efficiently perform the uncomputation needed to implement U_a reversibly.

By Theorem 1, we have $G = G_{p_1} \bullet \cdots \bullet G_{p_l}$ where p_i is a prime for all $i = 1, ..., l$ and G_{p_i} is a Sylow p_i-subgroup of G. Let S_j be the set of all the elements in $\mathcal{F}$ having order a power of the prime p_j. For $a \in S_j$, let K_a denote the (cyclic) subgroup generated by a. By Theorem 2, we have $K_a = K_{p_1} \bullet \cdots \bullet K_{p_l}$ where $K_{p_i} \leq G_{p_i}$ for all $i = 1, ..., l$. Since $|K_a|$ is a power of p_j, we must have $K_a \leq G_{p_j}$. Thus $S_j \subset G_{p_j}$. Since $\mathcal{F}$ generates G, S_j generates G_{p_j}. Hence, we can first find the direct product representation for each of the Sylow p-subgroups of G and then take the direct product of these subgroups to obtain a direct product representation of G. We may therefore assume without loss of generality that G has prime power order. (It is not necessary to do so, but the analysis is simpler, and implementations would require smaller quantum registers.)

Theorem 6 *There is a quantum algorithm with the following properties. Given any finite Abelian group G of prime power order that satisfies assumptions* 1,2,3 *and* 4, *and an $\epsilon > 0$, the algorithm runs in time polynomial in* $\log(|G|)$ *and* $\log(1/\epsilon)$ *and outputs elements $g_1, ..., g_l \in G$ such that with probability at least $1-\epsilon$, $G = \langle g_1 \rangle \bullet \cdots \bullet \langle g_l \rangle$.*

Proof. The algorithm is as follows.

1. Find a generating set $\{a_1, ..., a_k\}$ of the group G. This can be done efficiently by assumption 3.

2. Compute the maximum order $q = p^r$ of the elements $a_1, ..., a_k$. This can be done efficiently by Theorem 4.

3. Define $f : \mathbb{Z}_q^k \to G$ by mapping $\mathbf{x} = (x_1, ..., x_k)^T$ to $f(\mathbf{x}) = a_1^{x_1} \cdots a_k^{x_k}$. Apply Lemma 4 to find a generating set for the hidden subgroup K of $\mathbb{Z}_q^k$ as defined by the function f. This can be done with error probability at most ϵ in time polynomial in $\log(|G|)$ and $\log(1/\epsilon)$.

4. Apply Lemma 3 to obtain elements $\mathbf{y}_1, ..., \mathbf{y}_l \in \mathbb{Z}_q^k/K$ so that $\mathbb{Z}_q^k/K = \langle \mathbf{y}_1 + K \rangle \oplus \cdots \oplus \langle \mathbf{y}_l + K \rangle$. This can be done in deterministic polynomial time.

5. Compute $g_1 = f(\mathbf{y}_1), ..., g_l = f(\mathbf{y}_l)$. By Lemma 1, $G = \langle g_1 \rangle \bullet \cdots \bullet \langle g_l \rangle$. This can be done in deterministic polynomial time.

6. Output the set $\{g_1, ..., g_l\}$.

□.

By Theorem 4, we can find the order N_j of g_j for $j = 1, 2, \ldots, l$, in bounded-error quantum polynomial time. The group G is therefore isomorphic to $\mathbb{Z}_{N_1} \times \cdots \mathbb{Z}_{N_l}$, with the element g corresponding to the unique l-tuple $(x_1, x_2, \ldots, x_l)^T$ satisfying $g = g_1^{x_1} g_2^{x_2} \cdots g_l^{x_l}$. We can efficiently compute $g_1^{x_1} g_2^{x_2} \cdots g_l^{x_l}$ given $(x_1, x_2, \ldots, x_l)^T$, and this suffices for many purposes. If we wish to map efficiently $g = g_1^{x_1} g_2^{x_2} \cdots g_l^{x_l}$ to $(x_1, x_2, \ldots, x_l)^T$, then we can apply Lemma 4 to find the hidden subgroup of the function $f : \mathbb{Z}_q^{l+1} \to G$ that maps $(x_0, x_1, \ldots, x_l)^T$ to $g^{-x_0} g_1^{x_1} g_2^{x_2} \cdots g_l^{x_l}$.

Acknowledgements

The authors thank the anonymous referees for helpful suggestions on improving the exposition of the paper. We are also very grateful to Edlyn Teske for help with the class group results and for making other helpful suggestions.

Research of the first author was supported by NSERC PGSB. Research of the second author was supported by NSERC, MITACS, ORDCF, and PREA.

References

1. P. W Shor (1997), *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*, SIAM J. Comput., 26, pp. 1484-1509.
2. U. Vazirani (1997), UC Berkeley Course CS294-2 Quantum Computation Fall 1997. Lecture notes last downloaded on June 21, 2001 from `http://www.cs.berkeley.edu/~vazirani/qc.html`.
3. P. Høyer (1999), *Conjugated operators in quantum algorithms*, Phys. Rev. A, 59, pp. 3280–3289.

4. M. Mosca and A. Ekert (1999), *The hidden subgroup problem and eigenvalue estimation on a quantum computer*, in C. P. Williams, editor, Proceedings of the 1st NASA International Conference on Quantum Computing and Quantum Communication, Palm Springs, USA, Lecture Notes in Computer Science, 1509, pp. 174–188, Springer (Berlin).
5. M. Mosca (1999), *Quantum Computer Algorithms*, D. Phil Thesis, University of Oxford.
6. J. Watrous (2001), *Quantum algorithms for solvable groups*, Proceedings of the 33rd ACM Symposium on Theory of Computing, pp. 60-67.
7. H. Cohen (1993), *A Course in Computational Algebraic Number Theory*, Springer-Verlag (Berlin).
8. M. Jacobson (1999), *Subexponential Class Group Computation in Quadratic Orders*, Doctoral Thesis, TU Darmstadt.
9. J. A. Beachy and W. D. Blair (1990), *Abstract algebra with a concrete introduction*, Prentice-Hall Inc. (New Jersey).
10. A. Schrijver (1986), *Theory of Linear and Integer Programming*, Wiley and Sons (England).
11. R. Kannan and A. Bachem (1979), *Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix*, SIAM J. Comput., 8, pp. 499–507.
12. H. Cohen (1991), *Advanced Topics in Computational Number Theory*, Springer-Verlag (New York).
13. D. Boneh and R. J. Lipton (1995), *Quantum cryptanalysis of hidden linear functions (extended abstract)*, in D. Coppersmith, editor, Advances in Cryptology: CRYPTO 95, Lecture Notes on Computer Science, 963, pp. 424–437, Springer (Berlin).