Isomorphism and Factorization- Classical and Quantum Algorithms

Sebastian Dörn
Institut für Theoretische Informatik
Universität Ulm
89069 Ulm, Germany

Daniel Haase
Institut für Zahlentheorie und Wahrscheinlichkeitstheorie
Universität Ulm
89069 Ulm, Germany

Jacobo Torán
Institut für Theoretische Informatik
Universität Ulm
89069 Ulm, Germany

Fabian Wagner
Institut für Theoretische Informatik
Universität Ulm
89069 Ulm, Germany

{sebastian.doern,daniel.haase,jacobo.toran,fabian.wagner}@uni-ulm.de

1 Introduction

The integer factorization problem (IF) consists in given an integer n, finding a prime factor decomposition of n. Graph isomorphism (GI) is the problem to decide whether two given graphs are isomorphic, or in different words, whether there is a bijection between the nodes of both graphs respecting the adjacency relation. These are two well known natural problems with many applications and with a long history in the fields of mathematics and computer science. Moreover, their decisional versions are the best known examples of problems in the class NP that are neither known to be efficiently solvable (in the class P) nor hard for NP. They have an intermediate complexity and this lack of an exact classification has given much attention to both problems in the past.

(The decisional version of) IF is one of the few examples of problems in $\mathsf{NP} \cap \mathsf{CoNP}$ for which efficient algorithms (running in polynomial time in the length of the representation of n) are not known. It is believed to be a hard problem. In fact a considerable portion of modern cryptology relies in the supposition that IF cannot be efficiently solved. The best algorithm for IF runs in time $O(\exp(c \log(n)^{\frac{1}{3}} \log\log(n)^{\frac{2}{3}})$ for some constant c.

GI is not known to be in $\mathsf{NP} \cap \mathsf{CoNP}$ but in a probabilistic generalization of this complexity class. The best known algorithm, testing isomorphism of two unrestricted graphs with n nodes each, runs in time $O(\exp(c\sqrt{n \log(n)})$. On the other hand there are algorithms for the problem that work efficiently for "almost all graphs" and in fact a straightforward linear time algorithm can decide isomorphism for random graphs.

The best classical algorithms for IF and GI run therefore in exponential time in the input size. The situation is different in the field of quantum computation. In 1994 Shor gave an efficient quantum algorithm for factoring integers. His methods have been extended to more general algebraic problems and there is the hope that efficient quantum algorithms for graph isomorphism might also be developed.

We give in this chapter an overview of several attempts for obtaining efficient classical and quantum algorithms for IF and GI. In doing this we point out several similarities between both problems. Finally we review a result showing that IF and GI are in fact particular instances of a more general algebraic problem, the ring isomorphism problem.

2 Factorization of integers: classical algorithms

A natural number p is called a *prime number*, if it is divided by exactly two natural numbers (1 and p). The number 1 is not a prime. Any natural number $n \in \mathbb{N}$ admits a decomposition

$$n \;=\; p_1^{c_1} \cdots p_r^{c_r} \;.$$

into prime powers. The exponents $c_j \in \mathbb{N}$ are uniquely determined by n. A famous theorem of analytic number theory, the *prime number theorem* (see [43]), states that

$$\lim_{x\to\infty} \frac{\pi(x)}{x/\log(x)} \;=\; 1 \quad , \quad \pi(x) \;=\; \#\left\{p \text{ prime} \mid p \leq x\right\} \;,$$

i.e. the distribution of prime numbers among the natural numbers is asymptotically $\frac{x}{\log(x)}$. The *integer factorization* problem (IF) is defined as follows: given $n \in \mathbb{Z}$, find the primes p_j and exponents c_j of its prime factor decomposition.

No classical algorithm of polynomial complexity is known for this problem yet. Some important cryptographic systems (like RSA) draw their security from this fact. Factorization is a search problem. A decisional version of it (given $n, k \in \mathbb{Z}$ is there a prime factor of n grater than k) belongs clearly to the class $\mathsf{NP} \cap \mathsf{CoNP}$. Note that the inverse problem, i.e. the computation of n from its given factors, is extremely simple, it is therefore used in various public key systems and key exchange protocols which draw their security from mathematical problems which are difficult to solve, but easy to verify.

The naive approach to the factorization problem (brute force checking of all possible prime factors) needs at most $O(\sqrt{n})$ operations to compute the prime factorization of n (the smallest prime contained in n is bounded by $\sqrt{n}$ if n is composite). The prime number theorem tells us that we do not gain a substantial speedup by restricting the search to prime numbers (not taking into account the cost to compute them). We mention the following algorithms which have been proposed in the last century to compute the prime factorization:

- Factorization using reduced *quadratic forms*, the ideas going back to C. F. Gauß. Today it is known that the computation of generators of these forms is actually as hard as classical factorization, and not practicable for large numbers. This method gave rise to several number theoretical generalizations of the factorization problem, all known to be in $\mathsf{NP} \cap \mathsf{CoNP}$, but still without an efficient classical algorithm.

- *Pollard's ϱ-method*, using the birthday phenomenon to find pairs (x, y) with $\gcd(x - y, n) \neq 1$.

- The *elliptic curve method*, using the fact that non-invertible elements of $\mathbb{Z}/n\mathbb{Z}$ share a factor with n, and such elements can be found using the group law on elliptic curves defined over $\mathbb{Z}/n\mathbb{Z}$. This is a typical example of an algorithm using group structures to factor numbers.

- The various *sieve methods*, the most sophisticated algorithm for factoring numbers known today. They make heavy use of the number

theoretic background of the factorization problem, especially the theory of number fields.

The best known algorithm today to solve this problem is the *general number field sieve* (see [17, 10.5]) with expected running time

$$O\left(\exp\left(\log(n)^{\frac{1}{3}} \cdot \log(\log(n))^{\frac{2}{3}} \cdot (C + o(1))\right)\right)$$

for a constant $C \approx 1,922$. We note that until 2002, there was no efficient deterministic algorithm to check if a given number is actually a prime. This is due to the fact that the prime number theorem gives only an asymptotic distribution of the prime numbers. If we pick a small interval $[a, b]$ for very large a, the distribution of primes in it is not structured at all. The fine grained distribution is not known by today, it is connected to the famous Riemann Hypothesis which has been the field of research for many decades without a prove in sight.

3 Graph isomorphism: classical algorithms

The *graph isomorphism* problem (GI) consists in given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, decide whether there is a bijection $f : V_1 \to V_2$ respecting the adjacency relations of the graphs. In symbols, for every $u, v \in V_1$, $(u, v) \in V_1 \iff (f(u), f(v)) \in V_2$. GI is clearly in NP and its complement is contained in AM, a probabilistic generalization of NP [11]. GI is not believed to be hard for NP, however no polynomial time algorithms for the problem are known either.

The earliest significant algorithms for deciding isomorphism were restricticted to trees [29, 46]. They provided a canonical enumeration of the input graphs that could be computed in linear time. The same technique was used for the isomorphism of planar graphs [30]. Several years later this result could be extended to graphs of bounded genus [21, 36, 38].

Babai used in [10] for the first time a group theoretic approach to the graph isomorphism problem. He was able to prove that the problem restricted to colored graphs (isomorphisms have to preserve the colors) for which the color multiplicity is bounded, can be solved in random polynomial time. Based on this work, Furst, Hopcroft and Luks [20] developed polynomial time algorithms for several permutation group problems. They also were able to derandomize Babai's algorithm making it deterministic.

Using involved results on the structure on permutation groups, a breakthrough was obtained by Luks in [37] when he gave a polynomial time algorithm for testing isomorphism of graphs of bounded degree. By providing a new degree reduction procedure and using Luks result, Zemlyachenco [47] managed to give a moderately exponential procedure of $\exp(n^{(\frac{2}{3}+o(1))})$ for

deciding isomorphism for unrestricted graph classes. Subsequent improvements of the bounded degree algorithm has brought this bound down to $\exp(c\sqrt{n \log n})$, (announced in [12]), which still is the algorithm for unrestricted graph isomorphism with the lowest worst case complexity.

There are several algorithms based on vertex-classification schemes that work well in practice. This is not surprising since it is known that trivial algorithms perform well on randomly generated graphs. Babai, Erdös and Selkow gave in [15] a straightforward linear time canonical labeling algorithm for GI, proving that it works well for almost all graphs.

The existing algorithms for graph isomorphism could roughly be divided into two main groups: those based on vertex classification methods and those constructing canonical labelings of the graphs.

- **Vertex classification methods:** A natural technique to restrict the search space when looking for an isomorphism is to divide the vertices of the input graphs in certain classes so that the vertices in one class in the first graph can only be mapped to vertices of the corresponding class in the second graph. Some ways to do this are to divide the vertices acording to their degree, the degree of their neighbors, the number vertices reachable by paths of a certain length, etc. This method can be used iteratively, refining the classifications of the vertices according to previous classifications.

- **Canonical labeling methods:** The idea here is to find canonical representatives for the different isomorphic graph classes. This problem is in principle harder than deciding isomorphism but in some known examples of restricted graph classes (like trees or planar graphs) the algorithms for isomorphism provide in fact canonical representatives. For unrestricted graphs the best known labeling algorithm works in $\exp(n^{\frac{1}{2}+o(1)})$ steps [12].

For more facts about the graph isomorphism problem and its structural complexity, we refer the reader to the textbook by Köbler, Schöning and Torán [34].

4 Quantum algorithms for integer factorization

The first efficient algorithm to factor integers on quantum computers was given by Peter W. Shor in [44]. His idea was to view the factorization problem as a period finding problem. Such problems can be solved on quantum computers using the quantum Fourier transformation as we well explain in the next section. Here we briefly show how to encode the factorization problem into a period finding problem.

Let N be the number to be factored and $a \in \{2, \ldots, N\}$ be chosen randomly. We first compute $\gcd(a, N)$ using Euclids Algorithm. If $\gcd(a, N) \neq$

1 the gcd is already a factor of N, otherwise we define the function $f(n) = a^n \bmod N$, a mapping from $\mathbb{Z}$ to $\mathbb{Z}/N\mathbb{Z}$. Its smallest period is called the *order* of $a \bmod N$ (because it is the order of the multiplicative group generated by a if multiplication is defined mod N). First, since $(\mathbb{Z}/N\mathbb{Z})^\times = \{a \bmod N \,:\, \gcd(a,N) = 1\}$ has at most $N-1$ elements ($0 \bmod N$ is never included in this set), the order of a is strictly less than N. We can compute the order as shown in the next section using $O(\log(N))$ measurements (it should be noted that already the computation of the gcd takes up to $\log(N)^3$ steps).

By definition of the order $a^n \equiv 1 \bmod N$, which means N is a multiple of $a^n - 1$. Suppose the order is even, then N is a multiple of $(a^{\frac{n}{2}} - 1)(a^{\frac{n}{2}} + 1)$, so we get a nontrivial factor of N by computing $\gcd(N, a^{\frac{n}{2}} + 1)$. Some elementary number theory shows that the probability of hitting a such that the order is odd is bounded by $\phi(N)2^{-m}$, where $\phi(N) = \#\{a \bmod N : \gcd(a,N) = 1\} < N$ is Eulers Totient function, and m is the number of prime powers in the prime factor decomposition of N. The total number of measurements needed to find a proper factor of N with given error probability ϵ is polynomial in $\log(N)$ and $\log(\epsilon^{-1})$, as was proved first by Shor.

4.1 The quantum Fourier transform and period finding

The concept of Fourier transformation is a fundamental tool in many areas of research. The quantum Fourier transform used in the field of quantum computation is, viewed by mathematics, the Fourier transform on the finite abelian group $\mathbb{Z}/n\mathbb{Z}$. In general, the *Fourier transform* of a finite abelian group G is given by

$$\hat{f}(\chi) \;=\; \sum_{g \in G} f(g)\bar{\chi}(g)$$

where $f : G \to \mathbb{C}$ is any function and $\chi : G \to \mathbb{C}^\times$ is a homomorphism of groups from G to the multiplicative group $\mathbb{C}^\times = \mathbb{C}\backslash\{0\}$, and $\bar{z}$ is the complex conjugate of $z \in \mathbb{C}$. For finite cyclic groups, $\chi(g)^n = \chi(g^n) = 1$ for the order $n = \#G$, so $\chi(g)$ is actually a root of unity, necessarily of the form $\chi_a(g) = \exp(2\pi i \frac{ag}{n})$ for some $a \in \mathbb{Z}$. We may identify G with $\mathbb{Z}/n\mathbb{Z}$ and each $a \in G$ with the homomorphism $\chi_a(g)$, and let

$$\sum_{g \in G} f(g)\bar{\chi}_a(g) \;=\; \sum_{g \in G} f(g)e^{-2\pi i \frac{ag}{n}}$$

be the Fourier transform of f at the value $a \in G$. The *quantum Fourier transform* (QFT) is the normalized Fourier transform on $\mathbb{Z}/n\mathbb{Z}$:

$$\hat{f}(a) \;=\; \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} f(j)e^{-2\pi i \frac{aj}{n}} \,.$$

The Fourier operator $\hat{}$ is actually an automorphism of the $\mathbb{C}$-space of functions $f : G \to \mathbb{C}$, the inverse transformation is given by

$$f(j) \;=\; \frac{1}{\sqrt{n}} \sum_{a=0}^{n-1} \hat{f}(a) e^{2\pi i \frac{aj}{n}} \,.$$

The most interesting property of this transformation is the translation law: Let $f(a+k)$ be the function $a \mapsto f(a+k)$, the function f shiftet by k, then its Fourier transform is

$$\widehat{f(a+k)} \;=\; \hat{f}(a) \cdot e^{2\pi i \frac{ak}{n}}$$

for any $k \in \mathbb{Z}$, that is shifts are mapped to complex prefactors by the Fourier transform. Suppose f is periodic: $f(a+p) = f(a)$ for some $p \in \mathbb{Z}$ and all $a \in \mathbb{Z}$, then we get

$$\hat{f}(a) \;=\; \widehat{f(a+p)} \;=\; \hat{f}(a) \cdot e^{2\pi i \frac{ap}{n}} \;\;\Rightarrow\;\; \hat{f}(a) \cdot \left(e^{2\pi i \frac{ap}{n}} - 1 \right) \;=\; 0 \;\forall a \in \mathbb{Z} \,,$$

which forces $\hat{f}(a)$ to be zero if a is not a multiple of $\frac{n}{p}$. The converse is also true, so the QFT defines an isomorphism

$$\left\{ \;\; f : \mathbb{Z}/n\mathbb{Z} \to \mathbb{C} \text{ of period } p \;\; \right\} \leftrightarrow \left\{ \;\; f : \mathbb{Z}/n\mathbb{Z} \to \mathbb{C} \text{ supported on } \frac{n}{p}\mathbb{Z} \;\; \right\}.$$

So if we want to compute the period of a function $f : \mathbb{Z}/n\mathbb{Z} \to \mathbb{C}$ which can be accessed only by evaluation, we compute $\hat{f}(a)$ for sufficiently many $a \in \mathbb{Z}/n\mathbb{Z}$ and calculate the greatest common divisor of those a for which $\hat{f}(a)$ does not vanish. Classically there is no speedup in finding the period this way, but computing a point in the support of a function is an easy task for a quantum computer, since this is what measurement actually does. Let us briefly note that the ket-notation $|k\rangle = |k\rangle(t)$ denotes the (column) vector whose k-th amplitude component is one. In the following we regard this as a function

$$|k\rangle : \{0, \ldots, n-1\} \;\to\; \mathbb{C} \;\;,\;\; t \;\mapsto\; \begin{cases} 1 & \text{if } t = k \\ 0 & \text{otherwise} \end{cases} .$$

The basic period finding algorithm is as follows: Let $f : \mathbb{Z}/n\mathbb{Z} \to \mathbb{C}$ be a function of period p. We assume the period is primitive, that is f has no period which is smaller then p. First, initialize two registers in the state

$$\psi_1 \;=\; \psi_1(t) \;=\; \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} |k\rangle |f(k)\rangle \,.$$

Measurement of the second register to a value y collapses the state to

$$\psi_2 \;=\; \frac{1}{\sqrt{\#\{k \in \mathbb{Z}/n\mathbb{Z} \;:\; f(k) = y\}}} \cdot \sum_{f(k)=y} |k\rangle |y\rangle \;=\; \frac{1}{\sqrt{p}} \cdot \sum_{j=0}^{p-1} |x + jp\rangle |y\rangle$$

since f is injective on the set $\{0,1,\ldots,p-1\}$ and $f(k)=y$ is equivalent to $k=x+jp$ for $x\in\{0,\ldots,p-1\}$ uniquely defined by y. Now we apply the QFT to this state, which transforms the p-periodic state to a $\frac{n}{p}$-periodic supported state:

$$\psi_3(t) \;=\; \frac{1}{\sqrt{p}}\sum_{j=0}^{p-1}\widehat{|x+jp\rangle}|y\rangle \;=\; \frac{1}{\sqrt{p}}\sum_{j=0}^{p-1}e^{2\pi i\frac{(x+jp)t}{n}}\widehat{|0\rangle}|y\rangle\;.$$

Since the equidistribution on all values is a function of period one, the Fourier transform of it is the singleton at 0. By the inversion formula the Fourier transform of $|0\rangle=|0\rangle(t)$ is the equidistribution $\sum|k\rangle(t)$. So actually we got the state

$$\psi_3(t) \;=\; \frac{1}{\sqrt{p}}\sum_{j=0}^{p-1}e^{2\pi i\frac{(x+jp)t}{n}}\frac{1}{\sqrt{n}}\sum_{k=0}^{n-1}|k\rangle|y\rangle\;.$$

Now if t is a multiple of $\frac{n}{p}$, we have

$$\frac{1}{\sqrt{p}}\sum_{j=0}^{p-1}e^{2\pi i\frac{(x+jp)t}{n}} \;=\; e^{2\pi i\frac{xt}{n}}\cdot\sqrt{p}\,,$$

while for other t the sum over the roots of unity $\exp(2\pi i\frac{(x+jp)t}{n})$ cancels to zero. So this state has support on the set $p^\perp=\{0,\frac{n}{p},2\frac{n}{p},\ldots,(p-1)\frac{n}{p}\}$, and is (up to complex phases) equidistributed. The equidistribution allows us to bound the number of measurements needed to infer the period p. The probability of measuring two adjacent points in the set $p^\perp$ depends on the length p of the period but not on the original length n of the register.

4.2 Generalization of the period finding algorithm

There are several generalizations of the period finding problem. The two most important are real periods, almost-periods, and hidden subgroups, of which we will explain the latter only. We briefly sketch the other generalizations: Periods of functions $f:\mathbb{R}\to S$ can be computed by approximation, that is using the period finding algorithm on the values of f at $\frac{1}{N},\frac{2}{N},\frac{3}{N},\ldots$ for sufficiently large N. The algorithm used is a modification of the integer period finding algorithm, which employs the same method. The complexity of this algorithm depends on the choice of N and the smallest period of f. An application of this technique can be found in [26], which gives an efficient algorithm to compute the regulator and the class number of quadratic number field (the complexity of this task is known to be at least as hard as integer factorization). Such algorithms can be generalized to functions $f:\mathbb{R}^n\to S$ having a period lattice, that is a discrete subgroup $L\subset\mathbb{R}^n$ such that $f(x+\lambda)=f(x)$ for any $x\in\mathbb{R}^n$ and $\lambda\in L$, which are approximated on a lattice of the form $\frac{1}{N}\mathbb{Z}^n$ for sufficiently large N. Applications occur

again in algebraic number theory, see [25] for example. We will not consider real functions here, since the effort to prove the probability bounds is rather lengthy. However the lattice finding problem is closely related to the natural generalization of the period problem for finite groups, which is in general called the *hidden subgroup problem* (HSP):

Definition 4.1 Given a group G and a function $f : G \to S$ into an arbitrary set S, find the subgroup $H \leq G$ such that f factorizes on H: $f(g+h) = f(g)$ for all $g \in G$ and $h \in H$, and $f(g) \neq f(g')$ whenever $g + H \neq g' + H$.

This problem has been studied for many types of groups G. Again the property of the Fourier transform of mapping functions with periods to functions with periodic supports can be used to find H in case of abelian groups G. This is done by extending the concept of Fourier transform to arbitrary finite abelian Groups. We need the concept of characters to introduce the Fourier transform:

Definition 4.2 A *character* of a finite abelian group G is a homomorphism $\chi : G \to \mathbb{C}^\times$.

As in the cyclic case, the Fourier transform of a function $f : G \to \mathbb{C}$ is

$$\chi \longmapsto \sum_{g \in G} f(g)\bar{\chi}(g)\,,$$

defined on characters. By identifying the elements of G with these characters we retrieve the usual notation of Fourier transforms defined on G itself. We briefly show how this identification is established: By the fundamental theorem for abelian groups, any finite abelian group is of the form

$$G \cong \bigoplus_{j=1}^{k} \mathbb{Z}/n_j\mathbb{Z}\,,$$

so any character of G is of the form

$$\chi(g) = \prod_{j=1}^{k} \chi_j(g_j)$$

where χ_j is a character on $\mathbb{Z}/n_j\mathbb{Z}$. The set of homomorhjphisms $\chi : G \to \mathbb{C}^\times$ is itself a group by multiplication, denoted by $\hat{G}$, and the above product representation shows $G \cong \hat{G}$, that is there is a bijection between elements of G and characters of G. Characters of the cyclic factors of G are always of the form $\chi^{(j)}(a) = \exp(2\pi i \frac{ab}{n_j})$, the bijection is then

$$\mathbb{Z}/n_j\mathbb{Z} \ni b \leftrightarrow \chi^{(j)} = \left[a \mapsto e^{2\pi i \frac{ab}{n_j}}\right] \in \widehat{\mathbb{Z}/n\mathbb{Z}}\,.$$

Characters of arbitrary abelian groups are (multiplicative) linear combinations of these cyclic characters. Since each $\mathbb{Z}/n\mathbb{Z}$ also carries a multiplicative structure, we can define a multiplication in the additive group G artificially by setting

$$g * h \;=\; (g_1, \ldots, g_k) * (h_1, \ldots, h_k) \;:=\; (g_1 h_1, \ldots, g_k h_k)\,,$$

and we obtain the identification of $a \in G$ with the character $\chi_a : g \mapsto \chi(a*g)$ for any character χ which is the product of generating characters of the cyclic factors. Thus we can define the Fourier transform taken on elements of G instead of characters:

Definition 4.3 The (normalized) *Fourier transform* on G by χ is defined as the transformation

$$\hat{f}(a) \;=\; \frac{1}{\sqrt{\#G}} \sum_{g \in G} f(g)\bar{\chi}(a * g)$$

for any function $f : G \to \mathbb{C}$.

Now we have to show that the Fourier transform on G provides the needed properties. They follow from the known *orthogonality relation* for χ:

$$\sum_{g \in G} \chi(g) \;=\; \begin{cases} \#G & \text{if} \quad \chi = 1 \quad \text{is the trivial character} \\ 0 & \text{if} \quad \chi \neq 1 \end{cases} .$$

The inversion formula for the general Fourier transform is

$$f(g) \;=\; \frac{1}{\sqrt{\#G}} \sum_{a \in G} \hat{f}(a)\chi(a * g)\,.$$

This is proved by

$$\begin{aligned} \frac{1}{\sqrt{\#G}} \sum_{a \in G} \hat{f}(a)\chi(a * g) \;&=\; \frac{1}{\#G} \sum_{a \in G} \sum_{b \in G} f(b)\bar{\chi}(b * a)\chi(a * g) \\ &=\; \frac{1}{\#G} \sum_{b \in G} f(b) \sum_{a \in G} \chi(a * (g - b))\,. \end{aligned}$$

By the orthogonality relation the last sum selects the value $b = g$ and removes the normalization factor, and the value of the expression is $f(g)$ as asserted. So the Fourier transformation on G is still an automorphism of functions from G to $\mathbb{C}$, by the normalization factor it is also a unitary transformation. It maps functions with period subgroup $H \leq G$ to functions having support in some group $H^{\perp}$, which is now more complex then the set

$\frac{n}{p}\mathbb{Z}/n\mathbb{Z}$ in the cyclic case. We use the equation

$$\begin{aligned}\hat{f}(a) &= \frac{1}{\sqrt{\#G}}\sum_{g\in G} f(g)\bar{\chi}(g*a) \;=\; \frac{1}{\sqrt{\#G}}\sum_{g\in G} f(g+h)\bar{\chi}(g*a)\\ &= \frac{1}{\sqrt{\#G}}\sum_{g\in G} f(h)\bar{\chi}((g-h)*a) \;=\; \hat{f}(a)\cdot\chi(h*a)\end{aligned}$$

which is true for all $a \in G$ if and only if $\hat{f}(a) = 0$ or $\chi(h * a) = 1$ for all $h \in H$, that is if $\hat{f}$ vanishes outside the *dual group* of H:

$$H^{\perp} \;=\; \{a \in G \;:\; \forall h \in H :\; \chi(h*a) = 1\}\;.$$

We have proven

Theorem 4.4 *The Fourier operator $\hat{}$ gives an isomorphism between these $\mathbb{C}$-spaces:*

- *The set of functions f with $f(g+h) = f(g)$ for all $g \in G$, $h \in H$.*
- *The set of functions f which vanish outside $H^{\perp}$.*

Finding the subgroup H therefore amounts to two tasks:

- Computation of $H^{\perp}$ by the period finding algorithm (generalized to $k+1$ registers). This is again done by measuring the $(k+1)$-th register and applying the Fourier transform on G to the first k registers, and measurement of enough elements from $H^{\perp}$ to infer generators or coefficients for $H^{\perp}$.
- Computation of H from $H^{\perp}$. Note that we actually do not want to compute the set H, but its generators, or equivalently coefficients d_j such that

  $$H \;=\; \bigoplus_{j=1}^{k} \mathbb{Z}/d_j\mathbb{Z}\,.$$

  This is not difficult if the corresponding coefficients for $H^{\perp}$ are known. Important applications of the hidden subgroup problem, for example the class number problem, content themselves with the determinant of H, which is the product of the coefficients. There is an efficient quantum algorithm for group decomposition, that is an algorithm to compute the coefficients d_j given generators of H.

We note that both group decomposition and hidden subgroup computation admit still no efficient classical algorithms. We conclude by stating the complexity of the abelian hidden subgroup problem. Assume the following preconditions:

- Elements of G can be represented uniquely as binary string, and it is possible to recognize a representation classically in a number of steps polynomial in the representation length.
- It is possible to compute (classically) the representation of the sum and the inverse of elements using a number of steps polynomial in the length of their representation.
- Both group operations and the evaluation of f can be implemented in a quantum circuit.

Then we have

Theorem 4.5 *There is a quantum algorithm with the following properties: Given a finite abelian group G (represented by generators of cyclic factors of prime power) and $f : G \to S$ satisfying 4.1, and some error probability ϵ, it computes a set of elements of G along with coefficients (or equivalently generator relations) for the subgroup H' generated by those elements, such that*

- *the probability that $H' = H$ is the period subgroup of f is $\geq 1 - \epsilon$,*
- *the number of measurements performed is bounded by a polynomial in $\log(\epsilon^{-1})$ and the input length for the group G.*

The proof uses the crucial fact that the Fourier transform of a H-period function is not only supported by $H^{\perp}$, but that function values are equidistributed among this set (up to complex phases). The Fourier theory of non-abelian groups is far more complicated, and we do not have the correspondence of group elements to characters implicitly used in the definition of the operation $*$. By now there is no general quantum algorithm known to compute H in case of non-abelian groups. There are some works on special group types, like dihedral groups or solvable groups, but these cover still a very small portion of all non-abelian groups.

5 Quantum approach to graph isomorphism

5.1 The hidden subgroup problem and graph isomorphism

We observe that the graph isomorphism problem can be solved with the help of the hidden subgroup problem. Let $G = (V, E)$ be a graph with vertex set $V = \{1, \ldots n\}$ and consider the symmetric group S_n of permutations over n elements. For a permutation $\pi \in S_n$, $\pi(G)$ is the graph resulting from permuting the labels of the vertices in G according to π. The set of automorphisms of G, $\mathrm{Aut}(G) = \{\pi \in S_n \mid \pi(G) = G\}$ is clearly a subgroup

of S_n. Consider the function f_G acting on S_n defined as $f_G(\pi) = \pi(G)$. Observe that for every $\sigma \in S_n$ and $\pi \in \mathrm{Aut}(G)$,

$$f_G(\sigma \cdot \pi) = \sigma \cdot \pi(G) = \sigma(G) = f_G(\sigma).$$

Moreover f_G has different values for the different cosets of $\mathrm{Aut}(G)$ in S_n:

$$\sigma_1 \mathrm{Aut}(G) \neq \sigma_2 \mathrm{Aut}(G) \Rightarrow \sigma_1(G) \neq \sigma_2(G) \Rightarrow f_G(\sigma_1) \neq f_G(\sigma_2).$$

In other words, $\mathrm{Aut}(G)$ is the hidden subgroup in S_n defined by f_G. With a generating set for $\mathrm{Aut}(G)$ it is possible to efficiently compute the order of the subgroup, $|\mathrm{Aut}(G)|$. With this one can decide the graph isomorphism problem in the following way: Let G_1 and G_2 be the input graphs and consider the graph $G_1 \cup G_2$ defined by the vertices and edges of both G_1 and G_2. It is not hard to see that if G_1 and G_2 are not isomorphic then $|\mathrm{Aut}(G_1 \cup G_2)| = |\mathrm{Aut}(G_1)| \cdot |\mathrm{Aut}(G_2)|$. On the other hand if the graphs are isomorphic then $|\mathrm{Aut}(G_1 \cup G_2)| = 2|\mathrm{Aut}(G_1)| \cdot |\mathrm{Aut}(G_2)|$ (we have to count in this case the automorphisms interchanging the vertices of G_1 and G_2).

Because of this observation, efficient algorithms for HSP would imply the existence of efficient algorithms for GI. But the symmetric group S_n needed here is non-abelian and therefore the methods explained in the previous section cannot be applied. There have been several attempts to extend the algorithms for HSP from abelian to non-abelian groups [22, 31] but the solved cases are not sufficient for solving GI. Hallgren, Russel and Ta-Shma show in [28] how to solve the HSP efficiently in the cases where the hidden subgroup is normal. Observe that this extends the results presented in the previous section since every subgroup of an abelian group is normal. Bacon, Childs and van Dam [13] have proposed a new approach giving efficient quantum algorithms for various semi-direct product groups. Kuperberg [35] developed a sieve algorithm for the hidden subgroup problem in the dihedral group D_n with running time $2^{O(\sqrt{n})}$. Based on this result a subexponential time algorithm for solving HSP on direct product groups was presented in [4]

Recently some negative results pointing to the impossibility of obtaining efficient quantum algorithms for the HSP have been published. In [27] it is shown that strong Fourier sampling is insufficient to efficiently resolve the HSP on certain non-abelian groups and that multiregister Fourier sampling over $\Omega(\log |G|)$ registers is required to distinguish subgroups of certain groups, including the symmetric group. A good overview of these results can be seen in [5]

5.2 The quantum query model and graph isomorphism

The difficulty of obtaining non-trivial upper or lower bounds for the graph isomorphism problem on classical or quantum computers motivates the study of more restricted models, in which it is possible to establish differences between both computing paradigms.

We consider here the quantum query model, a basic restricted model of quantum computation. In the query model, the input $x_1, \ldots, x_N$ is contained in a black box or oracle and can be accessed by queries. In a query we give a position i as input to the black box and it outputs x_i. The goal is to compute a boolean function $f : \{0,1\}^N \to \{0,1\}$ on the input bits $x = (x_1, \ldots, x_N)$ minimizing the number of queries. The classical version of this model is known as decision tree. We can consider the query complexity of a concrete boolean function in trying to show that the quantum model presents advantages over the classical model.

The quantum query model was explicitly introduced by Beals et al. [14]. Unlike the classical case, the power of quantum parallelism can be used in order to make queries in superposition. The state of the computation is represented by $|i, b, z\rangle$, where i is the query register, b is the answer register, and z is the working register. A quantum computation with T queries is a sequence of unitary transformations

$$U_0 \to O_x \to U_1 \to O_x \to \ldots \to U_{T-1} \to O_x \to U_T,$$

where each U_j is a unitary transformation that does not depend on the input x, and O_x are query (oracle) transformations. The oracle transformation O_x can be defined as $O_x : |i, b, z\rangle \to |i, b \oplus x_i, z\rangle$. The computation consists of the following three steps:

1. Go into the initial state $|0\rangle$.
2. Apply the transformation $U_T O_x \cdots O_x U_0$.
3. Measure the final state.

The result of the computation is the rightmost bit of the state obtained by the measurement. The quantum computation determines f with bounded error, if for every x, the probability that the result of the computation equals $f(x_1, \ldots, x_N)$ is at least $1 - \epsilon$, for some fixed $\epsilon < 1/2$. In the query model of computation each query counts as one computation step but all other computation steps are free.

We consider upper and lower bounds for the number queries needed to compute a boolean function stored in the black-box. If the black-box contains N positions, then trivially N queries are sufficient. But in some cases less quantum queries are needed. Grover [23] showed that in order to compute the OR function of N inputs $(x_1, \ldots, x_N)$, $O(\sqrt{N})$ quantum queries are sufficient. This supposes a quadratic speed-up over the number of classical queries for the same problem.

The idea that with quantum queries we could search more efficiently in an unordered search space than with classical procedures gave initially some hope for efficient quantum solution of NP problems. For example we can consider than in a problem like graph isomorphism, each oracle position x_i

encodes a 0 or a 1 depending on whether the i-th bijection is an isomorphism between two given graphs. Computing the OR of these bits is equivalent to solve the isomorphism problem. Since for graphs with n nodes there are $n!$ possible bijections, a naive application of Grover's method would compute the problem with $O(\sqrt{n!})$ queries, which is still a very large number of steps. Some other methods have been proposed trying to improve the efficiency of the search. One of these methods is the quantum walk.

5.3 Quantum walks and the fix-automorphism problem

Quantum walks are the quantum counterpart of Markov chains and random walks. We present here some facts on quantum walks and their connection to isomorphism problems. A discrete quantum walk is a way of formulating local quantum dynamics on a graph. The walk takes discrete steps between neighbouring vertices and is a sequence of unitary transformations. We present a recent scheme for quantum search, based on any ergodic Markov chain, given by Magniez et al. [40]. We use then this tool for the development of a quantum algorithm for a special isomorphism problem.

Aharonov et al. [3] introduced quantum walks on graphs. They showed how fast quantum walks spread and proved lower bounds on the possible speed up by quantum walks for general graphs. Ambainis [7] constructed a fundamental quantum walk algorithm for the element distinctness problem. This was the first quantum walk algorithm that went beyond the capability of Grover search. Magniez et al. [41] have used Ambainis algorithm for finding a triangle in a graph. Szegedy [45] generalized the element distinctness algorithm of Ambainis to an arbitrary graph by using Markov chains. He showed that for a class of Markov chains, quantum walk algorithms are quadratically faster than the corresponding classical algorithms. Buhrman and Špalek [16] constructed a quantum algorithms for matrix multiplication and its verification. Recently, Magniez et al. [40] developed a new scheme for quantum search based on any ergodic Markov chain. Their work generalizes previous results by Ambainis [7] and Szegedy [45]. They extend the class of possible Markov chains and improve the quantum complexity. Dörn and Thierauf [18,19] presented the first application of this new quantum random walk technique for testing the associativity of a multiplication table.

Let $P = (p_{xy})$ be the transition matrix of an ergodic symmetric Markov chain on the state space X. Let $M \subseteq X$ be a set of marked states. Assume that the search algorithms use a data structure D that associates some data $D(x)$ with every state $x \in X$. From $D(x)$, we would like to determine if $x \in M$. When operating on D, we consider the following three types of costs:

- *Setup cost* s: The worst case cost to compute $D(x)$, for $x \in X$.
- *Update cost* u: The worst case cost for transition from x to y, and

update $D(x)$ to $D(y)$.

- *Checking cost* c: The worst case cost for checking if $x \in M$ by using $D(x)$.

Theorem 5.1 [40] *Let $\delta > 0$ be the eigenvalue gap of a ergodic Markov chain P and let $\frac{|M|}{|X|} \geq \epsilon$. Then there is a quantum algorithm that determines if M is empty or finds an element of M with cost*

$$s + \frac{1}{\sqrt{\epsilon}} \left(\frac{1}{\sqrt{\delta}} u + c \right).$$

In the most practical applications (see [7, 41]) the quantum walk takes place on the Johnson graph $J(n, r)$, which is defined as follows: the vertices are subsets of $\{1, \ldots, n\}$ of size r and two vertices are connected iff they differ in exactly one number. It is well known, that the spectral gap δ of $J(n, r)$ is $\Theta(1/r)$ for $1 \leq r \leq \frac{n}{2}$.

Now we consider the *fix-automorphism* problem as an application of the quantum walk search procedure. We have given a graph $G = (V, E)$ with n vertices represented as adjacency matrix and an integer $k < n$. One has to decide whether G has an automorphism which moves at most k vertices of G.

Theorem 5.2 *The quantum query complexity of the fix-automorphism problem is $O(n^{2k/(k+1)})$.*

Proof. We apply the quantum walk search scheme of Theorem 5.1. To do so, we construct a Markov chain and a data base for checking if a vertex of the chain is marked.

Let $G = (V, E)$ be the input graph with n vertices represented as adjacency matrix. Let U be a subset of vertices of G of size r. We will determine r later. Our quantum walk takes place on the Johnson graphs $J(n, r)$. The data base of the quantum walk is the induced subgraph on the set of vertices U, denoted by $G[U]$, has U as its vertex-set, and it contains every edge of G whose endpoints are in U. The marked vertices of $J(n, r)$ correspond to subsets $U \subset V$, such that $G[U]$ contains an automorphism which moves at most k vertices of G. In every step of our walk we exchange one vertex of U.

Now we determine the quantum query setup, update and checking cost. The setup cost to determine $G[U]$ is $O(r^2)$ and the update cost is $O(r)$. Checking if $G[U]$ contains such a fixed automorphism needs no queries, since we require only the data base for checking if the vertex U is marked.

If there is at least one automorphism which moves at most k vertices, then there are at least $\binom{n-k}{r-k}$ marked vertices of the Johnson graph. Therefore we have

$$\epsilon \geq \frac{|M|}{|X|} \geq \frac{\binom{n-k}{r-k}}{\binom{n}{r}} \geq \Omega\left(\left(\frac{r}{n} \right)^k \right).$$

Then the quantum query complexity of the fix-automorphism problem is

$$O(r^2 + \left(\frac{n}{r}\right)^{k/2} \cdot r^{1.5}),$$

which is minimized for $r = n^{k/(k+1)}$. □

6 Reductions of integer factorization and graph isomorphism to ring isomorphism

We review in this section a result from Kayal and Saxena [32] showing that GI and IF are instances of a more general question: the ring isomorphism problem. Ring isomorphism has received attention in the last years in connection with the efficient primality test algorithm from [2]. Other applications of this problem can be seen in [9] and [1].

Definition 6.1 A finite *ring* with identity element 1 is a triple $(R, +, \cdot)$, where R is a finite set such that $(R, +)$ is a commutative group with identity element 0 and $(R, \cdot)$ is a semigroup with identity element 1, such that multiplication distributes over addition. The *characteristic* of a ring R is defined to be the smallest number of times one must add the ring's multiplicative identity element 1 to itself to get the additive identity element 0. Let I be a ideal of R, the *factor ring* is the ring $R/I = \{a + I : a \in R\}$ together with the operations $(a + I) + (b + I) = (a + b) + I$ and $(a + I)(b + I) = ab + I$.

The *polynomial ring* $R[X]$ is the ring of all polynomials in a variable X with coefficients in the ring R.

Let n be the characteristic of the ring. The complexity of the problems involving finite rings depends on the representation used to specify the ring. We will use the following representation models of a ring:

- *Table representation:* A ring R is given as a list of all the elements of the ring and their addition and multiplication tables.
- *Basis representation:* A ring R is given by m *basis elements* $b_1, \ldots, b_m$ and the additive group can be expressed as

$$(R, +) = \bigoplus_{i+1}^{m} \mathbb{Z}_{n_i} b_i,$$

with $n_i | n$ for each i. The multiplication in R is given by specifying the product of each pair of basis elements as an integer linear combination of the basis elements: $b_i \cdot b_j = \sum_{k=1}^{m} a_{ij,k} b_k$ for $1 \leq i, j \leq m$ with $a_{ij,k} \in \mathbb{Z}_n$.

- *Polynomial representation:* A ring R is given by

$$R = \mathbb{Z}_n[Y_1, \ldots, Y_m]/(f_1(Y_1, \ldots, Y_m), \ldots, f_k(Y_1, \ldots, Y_m)),$$

where $Y_1, \ldots, Y_m$ are basis elements and $(f_1(Y_1, \ldots, Y_m), \ldots, f_k(Y_1, \ldots, Y_m))$ is an ideal generated by the polynomials $f_1, \ldots, f_k$.

The table representation has size $O(|R|^2)$, which is a highly redundant representation. The size of the basis representation is $O(m^3)$, where m is the number of basis elements. This is in general exponentially smaller than the size of the Ring $|R| = \Pi_{i=1}^{m} n_i$. Often the polynomial representation is exponentially more succinct than the basis representation. For example $\mathbb{Z}_2[Y_1, \ldots, Y_m]/(Y_1^2, \ldots, Y_m^2)$ has 2^m basis elements and so the basis representation would require $\Omega(2^{3m})$ space.

Since the polynomial representation is of smaller size, for clarity of exposition, we will use it here to express the rings. However, for complexity issues this representation is too succinct and we will consider the rings given as input to the problems given in basis representation. For a polynomial representation, say $R = \mathbb{Z}_n[Y_1, \ldots, Y_t]/\mathcal{I}$ an automorphism or isomorphism ϕ will be specified by a set of t polynomials $p_1, \ldots, p_t$ with $\phi(Y_i) = p_i(Y_1, \ldots, Y_t)$.

Definition 6.2 An *automorphism* of ring R is a bijective map $\phi : R \mapsto R$ such that for all $x, y \in R$, $\phi(x+y) = \phi(x) + \phi(y)$ and $\phi(x \cdot y) = \phi(x) \cdot \phi(y)$. An *isomorphism* between two rings R_1, R_2 is a bijective map $\phi : R_1 \mapsto R_2$ such that for all $x, y \in R_1$, $\phi(x+y) = \phi(x) + \phi(y)$ and $\phi(x \cdot y) = \phi(x) \cdot \phi(y)$.

We define some ring automorphism and isomorphism problems:

- The *ring automorphism* problem (RA) consists in given a ring R, decide whether there is a non-trivial automorphism for R.
- The *finding ring automorphism* problem (FRA) consists in given a ring R, find a non-trivial automorphism of R.
- The *ring isomorphism* problem (RI) consists in given two rings R_1, R_2, decide whether there is an isomorphism between both rings.

All rings are given in its basis representation.

6.1 Factoring integers and finding ring automorphisms

We discuss the complexity of finding automorphisms in a ring and present a result from Kayal and Saxena [32] showing that this problem is at least as hard as factoring integers. Let $\leq_m^P$ denote a polynomial time many-one reduction or transformation between problems.

Theorem 6.3 [9] IF $\leq_{\mathrm{m}}^{\mathrm{P}}$ FRA

The quadratic and number field sieve methods can be easily viewed as trying to find a non-obvious automorphism in a ring. Both methods aim to find two numbers u and v in $\mathbb{Z}_n$ such that $u^2 = v^2$ and $u \neq \pm v$ in $\mathbb{Z}_n$, where n is an odd square-free composite number to be factored. We will encode this in a ring such that finding its ring automorphisms gives us u and v.

We consider the ring $R = \mathbb{Z}_n[Y]/(Y^2 - 1)$, which has an obvious non-trivial automorphism mapping Y onto $-Y$. The problem is to find another non-trivial automorphism $\phi(R) \neq \pm Y$. Agrawal and Saxena give the following proof:

Proof. Let $\phi(Y) = aY + b$, by definition of the factor ring R we have $\phi(Y^2 - 1) = 0$. Observe that $\phi(a + b) = \phi(a) + \phi(b)$, $\phi(ab) = \phi(a)\phi(b)$, and that $Y^2 - 1 \equiv 0$. Thus we have

$$\begin{aligned}\phi(Y^2 - 1) &= (aY + b)^2 - 1 = a^2Y^2 + 2abY + b^2 - 1 \\ &= a^2(Y^2 - 1 + 1) + 2abY + b^2 - 1 = a^2 + b^2 - 1 + 2abY = 0.\end{aligned}$$

This gives $ab = 0$ and $a^2 + b^2 = 1 \in \mathbb{Z}_n$. Notice that a and n are relatively prime, that is $(a, n) = 1$, since otherwise

$$\phi\left(\frac{n}{(a,n)}Y\right) = \frac{n}{(a,n)}(aY + b) = \frac{a}{(a,n)}nY + \frac{n}{(a,n)}b = \phi\left(\frac{n}{(a,n)}b\right),$$

because $\frac{a}{(a,n)}nY \equiv 0 \mod n$. Therefore, $b = 0$ and $a^2 = 1$. By assumption, $a \neq \pm 1$ and so $u = a$ and $v = 1$. Conversely given u and v with $u^2 = v^2$, $u \neq \pm v$ in $\mathbb{Z}_n$ we get $\phi(Y) = \frac{u}{v}Y$ as an automorphism of R. □

As shown in [32], factoring integers can be reduced to a number of questions about automorphisms and isomorphisms of rings, that is counting the number of automorphisms of ring $\mathbb{Z}_n[Y]/(Y^2)$ or finding the isomorphisms between rings $\mathbb{Z}_n[Y]/(Y^2 - a^2)$ and $\mathbb{Z}_n[Y]/(Y^2 - 1)$ for a randomly chosen $a \in \mathbb{Z}_n$. But for RA a polynomial time algorithm is known [32]. That means, some automorphisms are easy to compute and some others not.

6.2 Graph isomorphism and ring isomorphism

An interesting fact is that there also is a connection between the graph isomorphism and the ring isomorphism problems.

Theorem 6.4 [32] GI $\leq_{\mathrm{m}}^{\mathrm{P}}$ RI

We present the reduction from [9].

Proof. Let $G = (V, E)$ be a simple graph on n vertices. Then define polynomial p_G as

$$p_G(x_1, \ldots, x_n) = \sum_{(i,j)\in E} x_i \cdot x_j,$$

and define ideal $\mathcal{I}_G$ as

$$\mathcal{I}_G(x_1,\ldots,x_n) = (p_G(x_1,\ldots,x_n), \{x_i^2\}, \{x_i x_j x_k\}_{1\le i,j,k\le n}).$$

Observe, that for a polynom ring with ideal $\mathcal{I}_G$ in basis representation the number of basis elements is bounded by $O(n^2)$ since any combination of three variables are zero. In detail, Agrawal and Saxena proved the following: Let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be simple graphs over n vertices and let F_q be a field of odd characteristic. Then G_1 is isomorphic to G_2 iff

- either both graphs contain a clique of size m (K_m) and $n-m$ isolated vertices (D_{n-m}), each (in this case isomorphism testing is trivial),
- or the rings $R_1 = F_q[Y_1,\ldots,Y_n]/\mathcal{I}_{G_1}(Y_1,\ldots,Y_n)$ and $R_2 = F_q[Z_1,\ldots,Z_n]/\mathcal{I}_{G_2}(Z_1,\ldots,Z_n)$ are isomorphic.

R_1 and R_2 are polynomial rings with polynomials of degree at most two. If π is an isomorphism mapping G_1 onto G_2 then an isomorphism between both rings can be found as $\phi : R_1 \mapsto R_2$ with $\phi(Y_i) = Z_{\phi(i)}$, since $\phi(p_{G_1}(Y_1,\ldots,Y_n)) = p_{G_2}(Z_1,\ldots,Z_n)$.

We show now, that there is no further isomorphism. Suppose, that $G_1 \cong G_2$ and that G_2 is not of the form $K_m \cup D_{n-m}$. Let $\phi : R_1 \mapsto R_2$ be an isomorphism with

$$\phi(Y_i) = \alpha_i + \sum_{1\le j\le n} \beta_{i,j} Z_j + \sum_{1\le j<k\le n} \gamma_{i,j,k} Z_j Z_k.$$

We will show now, which values for $\alpha_i, \beta_{i,j}$ and $\gamma_{i,j,k}$ in $\phi(Y_i)$ may occur. For any isomorphism ϕ on rings it must hold that $\phi(0) = 0$. In R_1, $Y_i^2 = 0$ and in R_2, $Z_i^2 = 0$ for any value of i it follows, that

$$\phi(Y_i^2) = (\phi(Y_i))^2 = \alpha_i^2 + (\text{higher degree terms}) = 0.$$

Thus $\alpha_i = 0$. Again looking at the same equation:

$$\phi(Y_i^2) = (\phi(Y_i))^2 = 2 \sum_{1\le j<k\le n} \beta_{i,j}\beta_{j,k} Z_j Z_k = 0.$$

The other terms disappeared since $\alpha_i = 0$ or they become of degree greater two.

If more than one $\beta_{i,j}$ is non-zero, then we must have $\sum_{j,k\in J, j<k} \beta_{i,j}\beta_{i,k} Z_j Z_k$ divisible by $p_{G_2}(Z_1,\ldots,Z_n)$ with J the set of non-zero indices. Since p_{G_2} is also homogeneous polynomial of degree two, it must be a constant multiple of the above expression implying that $G_2 = K_{|J|} \cup D_{n-|J|}$. This is not possible by assumption. Therefore, at most one $\beta_{i,j}$ is non-zero.

If all $\beta_{i,j}$ are zero, then $\phi(Y_i, Y_l) = 0$ for all i, l which is not possible. Hence, exactly one $\beta_{i,j}$ is non-zero. Define $\pi(i) = j$ where j is the index with

$\beta_{i,j}$ non-zero. We now prove, that π is not surjective. Suppose $\pi(i) = \pi(l)$ for $i \neq l$. Then $\phi(Y_i Y_l) = Z_{\pi(i)} Z_{\pi(l)} = 0$. This is not possible. Hence π is a permutation on $[1, n]$. Now consider $\phi(p_{G_1}(Y_1, \ldots, Y_n))$, then it follows that

$$0 = \phi(p_{G_1}(Y_1, \ldots, Y_n)) = \sum_{(i,j) \in E_1} \phi(Y_i)\phi(Y_j) = \sum_{(i,j) \in E_1} \beta_{i,\pi(i)} \beta_{j,\pi(j)} Z_{\pi(i)} Z_{\pi(j)}$$

The last expression must be divisible by p_{G_2}. This gives $\beta_{i,\pi(i)} = \beta_{j,\pi(j)}$ for all i, j and implies that the expression is a constant multiple of p_{G_2} or equivalently, that G_1 is isomorphic to G_2. □

Notice, that the rings R_1 and R_2 constructed above have lots of automorphisms. For example, $Y_i \mapsto Y_i + Y_1 Y_2$ is a non-trivial automorphism of R_1. Thus, automorphisms of G_1 do not directly correspond to automorphisms of R_1.

References

[1] M. Agrawal, *Rings and integer lattices in computer science,* Lecture Notes, the 2007 Barbados Workshop on Computational Complexity. `http://www.cs.mcgill.ca/ denis/barbados07/barbados2007.ps`, 2007.

[2] M. Agrawal, N. Kayal, N. Saxena *PRIMES is in P*, Annals of Mathematics, Second Series, vol. 160: pages 781-793, 2004.

[3] A. Aharonov, A. Ambainis, J. Kempe, U. Vazirani, *Quantum walks on graphs*, Proceedings of Symposium on Theory of Computing (STOC): pages 50-59, 2001.

[4] G. Alagic, C. Moore, A. Russell, *Quantum algorithms for Simon's problem over general groups*, Proc. SODA 2007: pages 1217–1224, 2007.

[5] G. Alagic, A. Russell *Quantum computing and the hunt for the hidden symmetry*, Bulletin of the EATCS, October 2007.

[6] A. Ambainis, *Quantum walks and their algorithmic applications*, International Journal of Quantum Information 1: pages 507-518, 2003.

[7] A. Ambainis, *Quantum walk algorithm for element distinctness*, Proceedings of Symposium on Foundations of Computer Science (FOCS): pages 22-31, 2004.

[8] S. Aaronson, Y. Shi, *Quantum lower bounds for the collision and the element distinctness problems*, Journal of ACM 51: pages 595-605, 2004.

[9] M. Agrawal, N. Saxena, *Automorphisms of Finite Rings and Applications to Complexity of Problems*, Proceedings of Symposium on Theoretical Aspects of Computer Science (STACS): pages 1-17, 2005.

[10] L. Babai, *Monte-Carlo algorithms in graph isomorphism testing,* Preprint, University Toronto, 1979.

[11] L. Babai, *Trading group theory for randomnes* Proceedings of the 17th ACM Symposium on Theory of Computing (STOC): pages 424-429, 1985.

[12] L. Babai, E. Luks, *Canonical labeling of graphs* Proceedings of the 15th ACM Symposium on Theory of Computing (STOC): pages 171–183, 1983.

[13] D. Bacon, A. Childs, W. van Dam, *From optimal measurement to efficient quantum algorithms for the hidden subgroup problem over semidirect product groups,* Proceedings of the 46th IEEE Symposium on Foundations of Computer Science (FOCS): pages 469-478, 2007.

[14] R. Beals, H. Buhrman, R. Cleve, M. Mosca, R. de Wolf, *Quantum lower bounds by polynomials*, Journal of ACM 48: pages 778-797, 2001.

[15] L. Babai, P. Erdös, S.M. Selkow, *Random graph isomorphism*, SIAM J. Comp. 9, No. 3: pages 628-635, 1980.

[16] H. Buhrman, R. Špalek, *Quantum Verification of Matrix Products*, Proceedings of Symposium on Discrete Algorithms (SODA): pages 880-889, 2006.

[17] H. Cohen, *A Course in Computational Algebraic Number Theory*, GTM 138, Springer Verlag, 1993.

[18] S. Dörn, T. Thierauf, *The Quantum Query Complexity of Algebraic Properties*, Proceedings of Symposium on Fundamentals of Computation Theory (FCT): pages 250-260, 2007.

[19] S. Dörn, T. Thierauf, *The Quantum Complexity of Group Testing*, Proceedings of Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM): pages 506-518, 2008.

[20] M. Furst, J. Hopcroft, E. Luks, *Polynomial-time algorithms for permutation groups,* Proceedings of Symposium on Foundations of Computer Science (FOCS): pages 36-41, 1980.

[21] I. S. Filotti, J. N. Mayer, *A polynomial-time algorithm for determining the isomorphism of graphs of fixed genus,* Proceedings of Symposium Theory of Computing (STOC): pages 236-243, 1980.

[22] M. Grigni, L. Schulman, V. Vazirani, *Quantum mechanical algorithms for the nonabelian hidden subgroup problem*, Proceedings of Symposium on Theory of Computing (STOC): pages 68-74, 2001.

[23] L. Grover, *A fast mechanical algorithm for database search*, Proceedings of Symposium on Theory of Computing (STOC): pages 212-219, 1996.

[24] D. Haase, *Polynomiale Inversion der Distanzfunktion total reeller Zahlkörper*, Diploma Thesis, Karlsruhe 2003.

[25] D. Haase, M. Maier, *Quantum Algorithms in Number Fields, Progress of Physics 54*, Nr. 8-10: pages 866-881, 2006.

[26] S. Hallgren, *Polynomial-Time Quantum Algorithms for Pell's Equation and the Principal Ideal Problem*, Proceedings of Symposium on Theory of Computing, Montreal Canada, May 2002.

[27] S. Hallgren, C. Moore, M. Rötteler, A. Russell, P. Sen, *Limitations of quantum coset states for graph isomorphism*, Proceedings of Symposium on Theory of Computing (STOC): pages 604-617, 2006.

[28] S. Hallgren, A. Russel, A. Ta-shma, *Normal subgroup reconstruction and quantum computing using the group representation,* Proceedings of Symposium on Theory of Computing (STOC): pages 627-635, 2000.

[29] J. E. Hopcroft, R. E. Tarjan, *Dividing a graph into triconnected components,* SIAM Journal on computing, 2, No. 3: pages 136-158, 2005.

[30] J. E. Hopcroft, J. K. Wong, *Linear time algorithm for isomorphism of planar graphs* Proceedings of Symposium on Theory of Computing (STOC): pages 172-184, 1974.

[31] G. Ivanyos, F. Magnier, M. Santha, *Efficient quantum algorithms for some instances of the non-abelian hidden subgroup problem,* Proceedings of Symposium on Parallel Algorithms and Architechtures: pages 263-270, 2001.

[32] N. Kayal, N. Saxena, *On the Ring Isomorphism and Automorphism Problems*, Proceedings of Conference on Computational Complexity (CCC): pages 2-12, 2005.

[33] J. Kempe, N. Shenvi, K.B. Whaley, *Quantum Random-Walk Search Algorithm*, Physical Review Letters A, Vol. 67 (5), 2003.

[34] J. Köbler, U. Schöning, J. Torán, *The Graph Isomorphism Problem - Its Structural Complexity*, Birkhäuser, 1993.

[35] G. Kuperberg, *A subexponential time quantum algorithm for the dihedral hidden subgroup problem,* SIAM Journal on Computing 35 (1): pages 170-188, 2005.

[36] R. M. Lipton, *The beacon set approach to graph isomorphism,* SIAM J. Comput. 9, 1980.

[37] E. Luks, *Isomorphism of graphs of bounded valence can be tested in polynomial time,* J. Comp. Sci. 25: pages 42-65, 1982.

[38] G. L. Miller, *Isomorphism testing for graphs of bounded genus,* Proceedings of Symposium on Theory of Computing (STOC): pages 225-235, 1980.

[39] F. Magniez, A. Nayak, *Quantum complexity of testing group commutativity*, Proceedings of International Colloquium on Automata, Languages and Programming (ICALP): pages 1312-1324, 2005.

[40] F. Magniez, A. Nayak, J. Roland, M. Santha, *Search via Quantum Walk*, Proceedings of Symposium on Theory of Computing (STOC): pages 575-584, 2007.

[41] F. Magniez, M. Santha, M. Szegedy, *Quantum Algorithms for the Triangle Problem*, Proceedings of Symposium on Discrete Algorithms (SODA): pages 1109-1117, 2005.

[42] M.A. Nielsen, I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2003.

[43] W. Schwarz, *Einführung in die Methoden und Ergebnisse der Primzahltheorie*, Hochschultaschenbcher-Verlag, 1969.

[44] P. W. Shor, *Polynomial-time algorithms for factorization and discrete logarithms on a quantum computer*, SIAM Journal on Computing 26: pages 1484-1509, 1997.

[45] M. Szegedy, *Quantum speed-up of Markov chain based algorithms*, Proceedings of Symposium on Foundations of Computer Science (FOCS): pages 32-41, 2004.

[46] V. N. Zemlyachenko, *Canonical numbering of trees (Russian),* Proc. Seminar on Comb. Anal. at Moscow State Univ., Moscow 1970.

[47] V. Zemlyachenko, N. Kornienko, R. Tyshkevich, *Graph isomorphism problem*, The Theory of Computation I, Notes Sci. Sem LOMI 118, 1982.