Proceedings of CONF-MLA 2024 Workshop: Securing the Future: Empowering Cyber Defense with Machine Learning and Deep Learning
DOI: 10.54254/2755-2721/110/2024MELB0122

An Overview of Regev's Quantum Factoring Algorithm and its Recent Developments

Rick Lan Chen

St Edmund Hall, University of Oxford, Oxford, United Kingdom

rick.chen@seh.ox.ac.uk

Abstract. Factoring large integers has long been a computationally difficult problem in classical computing, forming the foundation of widely used encryption methods like RSA. In 1994, Shor's quantum algorithm introduced a revolutionary method for factoring integers exponentially faster than classical algorithms, laying the groundwork for quantum cryptography. Despite numerous attempts to enhance Shor's algorithm over the last three decades, significant breakthroughs remained elusive until Regev's discovery in 2023. Regev introduced a novel, multi-dimensional version of the quantum factoring algorithm, achieving a substantial improvement by reducing the required number of quantum gates to $O(n^{3/2}\log n)$, compared to Shor's original $O(n^2 \log n)$ gate complexity. Although Regev's approach offers a significant speedup, it comes with increased qubit requirements and relies on an unproven number theory assumption. This paper presents an overview of Regev's factoring algorithm, including a review of Shor's work for context, followed by an examination of key recent developments and follow-up research. These include efforts to reduce the qubit count, improve error resilience, generalise Regev's algorithm to related problems, and validate the number theory assumption. This review serves as an accessible entry point for researchers interested in the rapidly evolving field of quantum computing and factoring algorithms.

Keywords: Quantum computing, quantum algorithm, quantum factoring, Shor's algorithm.

1. Introduction

The factorisation of large integers is a long-standing challenge in classical computing, and it underpins the security of widely utilised encryption methods, such as the RSA algorithm [1]. The classical difficulty of factoring large numbers ensures that RSA remains robust against conventional attacks. In 1994, Shor introduced a groundbreaking quantum algorithm that could factor large integers exponentially faster than any known classical algorithm, sparking significant interest in quantum cryptography and quantum computing [2]. Shor's algorithm, by harnessing quantum mechanics, posed a potential threat to RSA encryption by demonstrating an efficient means of factoring via quantum computation.

Since the inception of Shor's algorithm, researchers have made various attempts to enhance its efficiency, primarily aiming to reduce the number of quantum gates required. Despite this, substantial progress on Shor's original approach has proven elusive over the years. The problem of factoring large integers is thought to be hard classically. This difficulty establishes the basis of the RSA encryption algorithm [1]. When the number to be factorised is an n-bit number N, the original Shor's algorithm

DOI: 10.54254/2755-2721/110/2024MELB0122

uses $O(n^2 \log n)$ quantum gates [2]. Despite previous efforts, no one had managed to cut this down by a polynomial factor. Regev's result provided the first substantial speedup in nearly 30 years, reducing the number of quantum gates to $O(n^{3/2} \log n)$ [3].

To design good quantum algorithms, a lot of factors need to be considered. As a newly established result, a lot of improvements could be made on Regev's factoring algorithm [3]. For example, Regev's circuit required $O(n^{3/2})$ qubits, an increase from the $O(n)$ required in Shor's; an unproven number theory assumption was also crucial in the argument.

This paper provides a comprehensive overview of Regev's recent contributions to quantum factoring [3], framed within the context of Shor's pioneering work [2]. It presents an in-depth examination of Regev's methodology and its core innovations, such as the transition to higher-dimensional structures and the use of lattice-based techniques. Additionally, the paper highlights subsequent research developments, including improvements to qubit efficiency [4], error tolerance [4, 5], and extensions of Regev's algorithm to related problems like discrete logarithms [6]. This review aims to offer accessible insights for new researchers and experts alike, fostering further exploration in the promising domain of quantum factoring algorithms.

2. Regev's factoring algorithm

2.1. Review of Shor's factoring algorithm

Shor's key argument was to first reduce the problem of factoring an integer N to the problem of order finding: for a uniformly chosen $a \in \{1, \ldots, N-1\}$ such that a is coprime to N, find the smallest positive integer r such that $a^r = 1 \bmod N$ [1]. The order-finding algorithm then consists of two parts - the quantum procedure and the classical post-processing. The exponential speedup over classical methods is provided by the quantum procedure.

For the reduction to order-finding to work, there are two constraints [1]. Firstly, r is asked to be even, or equivalently $a = b^2 \bmod N$ for some b; secondly, $b^r \notin \{-1, 1\} \bmod N$. The first condition implies $(b^r - 1)(b^r + 1) = 0 \bmod N$, where the second condition then implies neither bracket is divisible by N. Thus, the greatest common divisors $\gcd(N, b^r - 1)$ and $\gcd(N, b^r + 1)$ give non-trivial factors of N. For a uniformly chosen a, the two constraints are satisfied with good probability. By sampling different values of a, the reduction argument will work within a reasonable number of trials.

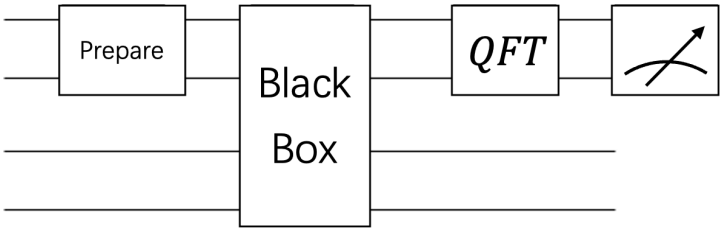

Figure 1. A circuit diagram displaying the overall structure of the quantum procedure in both Shor's and Regev's algorithm, with state preparation and Quantum Fourier Transform (QFT). The specific implementation for each step is different for the two algorithms. (Photo credit: Original).

The order-finding problem is a case of the structure of the quantum procedure is similar to other famous quantum algorithms, such as Deutsch-Jozsa and Simon [6]. They all fall into the category of Hidden Subgroup Problems [1, 6]. Figure 1 shows a sketch for the quantum circuit of Shor's algorithm. For Shor's algorithm, the black-box has the ability to compute $a^z \bmod N$ for any $1 \leq z \leq N$, a process called modular exponentiation. The quantum procedure can be broken into three parts [1]. First, the first register is prepared in the state $|0\rangle$, then passed through a Hadamard gate to create the superposition state $2^{-n/2} \sum_{z=0}^{2^n-1} |z\rangle$. Next, with the second register set to $|0\rangle$, the two registers are

DOI: 10.54254/2755-2721/110/2024MELB0122

passed through the black-box; this stores the result of $a^z \bmod N$ into the second register, resulting in the state $2^{-n/2}\sum_{z=0}^{2^n-1}|z\rangle\,|a^z \bmod N\rangle$. Finally, QFT is applied to the first register, after which the first register is measured.

The quantum procedure outputs (after dividing by N) an estimation to $v = c/r$, where c is a random number in $0,1,\cdots,r-1$, determined by the measurement process at the end [1]. When the estimation is close enough to c/r, the Continued Fraction Algorithm can be used to recover c/r from the estimation [1]. Note that this is the classical-post processing - computations are done classically rather than quantumly. At the end, the order r can be retrieved from c/r provided c is coprime to r. The two requirements that the estimation is accurate enough, and that c is coprime to r, can be shown to happen with good probability.

Since building quantum devices is much harder than building classical computers, the priority is to try to reduce the amount of quantum resources needed. Of course, this is no use if the classical post-processing part cannot be done efficiently; this is not an issue in the case of the Continued Fraction Algorithm [1]. The black-box is where the $O(n^2 \log n)$ gates in Shor's algorithm comes from. Modular exponentiation is typically done via repeated squaring, outlined below:

1. Set the register to 1.
2. Write the exponent z in binary, $z_1 z_2 \ldots z_n$.
3. For each j from 1 to n, do:
   (i). Square the register;
   (ii). If $z_j = 1$, multiply the register by a; do nothing if $z_j = 0$.

In general, all intermediate results need to be stored using n bits; therefore, this process involves $O(n)$multiplications between n-bit integers. Using fast multiplication [7], each multiplication costs $O(n \log n)$ gates, giving $O(n^2 \log n)$ gates in total.

2.2. Overview of Regev's factoring algorithm

To reduce the number of gates used in the black-box, Regev applied two key ideas: moving to higher dimensions and keeping the numbers stored small [3].

Firstly, Shor's algorithm is simply generalised to having multiple variables. Instead of choosing one number $a = b^2 \bmod N$, Regev chose d numbers $a_1, a_2, \ldots, a_d$ with $a_i = b_i^2 \bmod N$ [*3*]. The black-box is generalised to compute $\prod_{i=1}^{d} a_i^{z_i} \bmod N$ for $z \in \mathbb{Z}^d$. This is done via repeated squaring again:

1. Set the register to 1,
2. For each exponent z_i, Write z_i in binary, $z_{i1}\, z_{i2} \ldots z_{in}$.
3. For each j from 1 to n, do:
   (i). Square the register;
   (ii). Compute $\prod_{i=1}^{d} a_i^{z_{ij}} \bmod N$;
   (iii). Multiply the register by $\prod_{i=1}^{d} a_i^{z_{ij}} \bmod N$.

He then worked in multi-dimensional structures called lattices. Lattice is a mathematical concept that already had wide applications in classical computing like lattice cryptography [8, 9]. Regev defined the two lattices [3]:

$$\mathcal{L} = \{(z_1, z_2, \ldots, z_d) \in \mathbb{Z}^d \mid \prod_{i=1}^{d} a_i^{z_i} = 1 \bmod N\} \tag{1}$$

$$\mathcal{L}_0 = \{(z_1, z_2, \ldots, z_d) \in \mathbb{Z}^d \mid \prod_{i=1}^{d} b_i^{z_i} \in \{-1,1\} \bmod N\} \tag{2}$$

Figure 2 visualises one possibility for the two lattices in the case $d = 2$. Throughout this section, it is very helpful to try the case $d = 1$, which often directly corresponds to Shor's algorithm. Table 1 compares various aspects of Regev's algorithm to Shor's algorithm. In this way, Regev reduced the factoring problem to a problem of finding elements in $\mathcal{L} \setminus \mathcal{L}_0$.

DOI: 10.54254/2755-2721/110/2024MELB0122

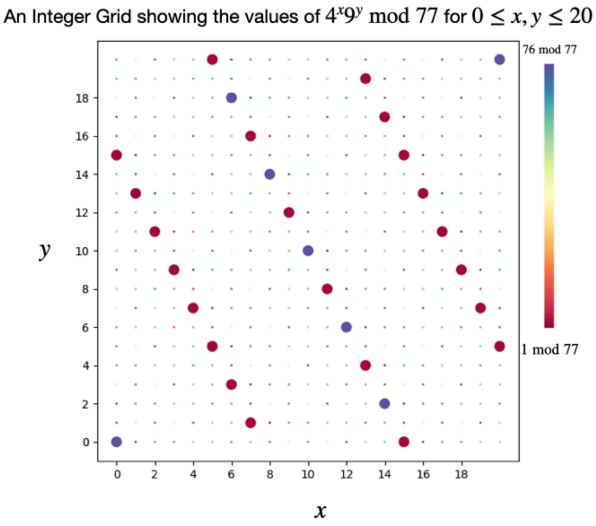

Figure 2. Modular Arithmetic Visualisation (Photo credit: Original).

Table 1. A table comparing various aspects of Shor's algorithm and Regev's algorithm.

Stage	Regev's	Shor's
Initialisation of States (proportional to)	$\sum_{z\in\{-D/2,\ldots,D/2-1\}^n} \rho_s(z)\|z\rangle$	$\sum_{z\in\{0,\ldots,z^n-1\}}\|z\rangle$
Computation of Black-Box	$f(z)=\prod_{i=1}^{d} a_i^{z_s} \mathrm{mod} N$	$f(z)=a^z \mathrm{mod} N$
Lattice / Hidden Subgroup	$\mathscr{L}$	$r\mathbb{Z}$
Dual Lattice	$\mathscr{L}^*$	$\frac{1}{r}\mathbb{Z}$
Output of Quantum Procedure	approximation of some $v \in \mathscr{L}^*/Z^d$	approximation of c/r for some c
Repeat Quantum Procedure	repeat $d+4$ times to get $\{v_i\}$ generating the dual lattice	repeat constant number of times to get c coprime to r
Recover Lattice from the Dual	Forming the lattice $\mathscr{L}'$ and applying the LLL Algorithm	Continued Fraction Algorithm

Figure 2 demonstrates this in two dimensions. In fact, a pigeon-hole argument shows that there exist a non-zero vector $z \in \mathscr{L}$ with $|z_i| \leq 2^{\frac{n}{d}}$ [3]. Fewer squaring operations are required to get to this exponent, which reduces the number of squaring operations in the black-box [7]. For Regev's argument, the exponents actually have to go up to D, a number greater than $2^{\frac{n}{d}}$ [3, 10].

However, the total number of multiplications remains $O(n)$. The products $\prod_{i=1}^{d} a_i^{z_{ij}} \mod N$ require in general d multiplications, which is a factor not present in Shor's algorithm (the case $d = 1$).

Fortunately, Regev's second ingredient does the trick. In Shor's algorithm, the number a was chosen randomly [1]. However, in Regev's algorithm, the numbers b_i (and so a_i) were deliberately chosen to be fixed small numbers, more precisely $O(\log d)$-bit integers [3]. The idea is that while the total number of multiplications was not reduced, the number of gates can be reduced by distributing and reordering the multiplications, in a way that maximises the number of small-number multiplications. Using a binary tree method, the products $\prod_{i=1}^{d} a_i^{z_{ij}} \mod N$ can be computed by only $O\left(d\log^3 d\right)$ gates. As explained later, choosing $d = \sqrt{n}$ turns out to be optimal [10]. In this way, Regev managed to construct a black-box using only $O\left(n^{3/2}\log n\right)$ gates.

However, asking b_i to be small came at a cost. Although it was shown that there are vectors in $\mathscr{L}$ of norm at most $2^{O(n/d)}$, they are not necessarily helpful for factoring. Recall that the reduction

DOI: 10.54254/2755-2721/110/2024MELB0122

argument works for only vectors in $\mathscr{L} \setminus \mathscr{L}_0$. Regev had to rely on a number theory assumption: that there exists a vector in $\mathscr{L} \setminus \mathscr{L}_0$ of norm at most $T = 2^{O(n/d)}$ [3]. This would be true if the numbers b_i were sampled uniformly instead of asked to be small. Regev's algorithm works unconditionally only if the assumption holds.

The quantum procedure in Regev's algorithm is now described. The overall structure of the quantum procedure is very similar to Shor's, as shown in Figure 1. However, the specific implementations are more sophisticated due to the added dimensions [3].

Instead of initialising the first register to a uniform superposition of states, Regev used a Gaussian superposition: $\sum_{z \in \left\{-\frac{D}{2}, \ldots, \frac{D}{2}-1\right\}^d} \rho_R(z)|z\rangle$, where D is as defined before, and ρ_R is a Discrete Gaussian distributions of some radius parameter R [3]. Discrete Gaussian distributions already had applications in classical lattice problems before Regev's paper [8]. Importantly, Gaussian distributions are computationally nice, for example the Fourier transform of a Gaussian distribution is (up to a multiplicative constant) Gaussian. These properties heavily simplified the calculations in Regev's proofs [3].

After the improved black-box, a multi-dimensional version of QFT is applied to the first register. Formally, the dual lattice for a given lattice $\mathscr{L}$ is defined as

$$\mathscr{L}^* = \{w \in \mathbb{R}^d \mid \forall z \in \mathscr{L}, \langle w, z\rangle \in \mathbb{Z}\}, \tag{3}$$

which is another lattice of the same dimension [3, 9]. Table 1 highlights how Shor's algorithm gives a demonstration of the dual lattice in the case $d = 1$. The element $v \in \mathscr{L}^*$ is chosen uniformly from a list of possibilities, just as in the case of Shor's.

The classical post-processing is then responsible for recovering an element in $\mathscr{L} \setminus \mathscr{L}_0$ from the estimation to $\mathscr{L}^*$ [3]. This requires a more sophisticated argument than in the case of Shor's. Firstly, the entire quantum procedure is repeated $d + 4$ times. To be able to gain enough information about $\mathscr{L}^*$, at least d runs of the quantum procedure is needed as $\mathscr{L}^*$ is d-dimensional. Since the elements from $\mathscr{L}^*$ are chosen randomly, there are an additional 4 runs to make sure the entire lattice can be generated with probability at least $1/2$ [3, 11]. This gives a noisy sample of the dual lattice. Secondly, by cleverly constructing a new lattice $\mathscr{L}'$, Regev was able to recover certain vectors in the original lattice from the noisy sample of the dual lattice.

All that was left to do was to find a short vector in the original lattice. This can be done via lattice reduction [12]. The LLL algorithm [12] provides a way to approximate the length of the shortest vector. When the lattice is d-dimensional, this length is approximated to within a factor of 2^d. This agrees with the intuition that it is harder to recover information from higher-dimensional lattices. Thus, while having more dimensions seemingly improves the black-box by only contributing a factor of $2^{\frac{n}{d}}$, the advantage starts to fade for large d due to the factor of 2^d [3, 10]. This effectively fixes the optimal choice $d = \sqrt{n}$.

By applying the LLL algorithm (which runs in polynomial time) in place of the Continued Fraction Algorithm, the classical post-processing is able to recover a short vector from $\mathscr{L} \setminus \mathscr{L}_0$, which can then be used to factor N.

Like in Shor's algorithm, several steps in Regev's algorithm are not guaranteed to work [3]. However, overall the algorithm fails with bounded probability, and it is easy to check whether the algorithm has succeeded. Thus, the probability of failure can be made arbitrarily small by just repeating the algorithm a constant number of times.

2.3. Discussion

Several remarks are in order.

Firstly, the total number of gates being applied is still $O\left(n^{3/2} \log n\right) \cdot \left(\sqrt{n} + 4\right) = O\left(n^2 \log n\right)$, due to the repetition [3]. However, having only $O\left(n^{3/2} \log n\right)$ gates in each run of the circuit is better for several reasons [10]. Importantly, we expect quantum error correction to work better for smaller

DOI: 10.54254/2755-2721/110/2024MELB0122

circuits, and that fewer errors would occur. On the other hand, the $(\sqrt{n}+4)$ repetitions can be run in parallel, saving overall runtime.

Secondly, the total number of qubits used in Regev's algorithm was $O(n^{3/2})$ - for Shor's it was only $O(n)$ [4]. Fortunately, this has been reduced by Ragavan and Vaikuntanathan, making practical implementations of Regev's algorithm much more likely [4]. Ragavan and Vaikuntanathan also modified the classical post-processing part to make it more noise-robust.

Altogether, it must be said that the analysis was asymptotic. Hidden constants like the constant C in $R = 2^{C\sqrt{n}}$ can make Regev's algorithm less effective for factoring small numbers [3].

One potential investigation is to replace the Gaussian superposition with a uniform superposition, reverting to Shor's approach [10].

In 2024, Pilatte managed to prove the number theory assumption for a modified version of Regev's algorithm [13]. In the proof, the numbers b_i were chosen to be $O(d \log d)$-bit numbers instead of $O(\log d)$. This was not ideal for practical applications, as the increased size in b_i also increased the circuit size. For Regev's algorithm to be both unconditionally correct and practical, a proof for a stronger result is needed.

Lastly, Ekerå and Gärtner [5] extended Regev's factoring algorithm to solve the discrete logarithm, a closely related problem.

3. Further developments

3.1. Reducing the number of qubits

To analyse the number of qubits used in a quantum circuit, often one starts from the corresponding classical circuit. The key characteristics in quantum computing is that computations are reversible, something not required for classical circuits. To convert a classical circuit with G gates and S bits into a quantum one, irreversible gates are replaced with reversible ones, and ancilla qubits are added to keep information around. The details can be found in [4].

The problem with Regev's circuit originates from the black-box yet again. The first register uses $\log D^{\mathrm{d}} = O(n)$ qubits and the second register uses n qubits; each fast multiplication circuit uses $O(n \log n)$ ancilla qubits [7]. This apparently results in $O(n \log n)$ qubits. Unfortunately, while modular multiplication can be done in-place, modular squaring cannot [4]. Here, in-place means that the outcome has to be stored in the input register, whereas out-of-place means storing the outcome in a new register. It is difficult to make modular squaring reversible and in-place. The square root of a number modulo N is not necessarily unique, and to find even one of them is hard. Thus, for each squaring operation done in the black-box, the result needs to be stored in a separate register. Since there are $O(\sqrt{n})$ squaring operations and each result occupies $O(n)$ qubits, at least $O(n^{3/2})$ qubits are needed [4].

To avoid this, Ragavan and Vaikuntanathan exploited the fact that in-place, reversible modular multiplications are easier to implement than modular squaring [4]. The idea is to use Fibonacci numbers instead of the powers of 2, something that appeared in the work of Kaliski [14]. In the following, every state is reduced modulo N, so the notation is omitted. To start, two registers are initialised to $|a\rangle|a\rangle$. Afterwards, repeatedly multiply the two registers together and store the result in the appropriate register. An example run would be $|a\rangle|a\rangle \mapsto |a^2\rangle|a\rangle \mapsto |a^2\rangle|a^3\rangle \mapsto |a^5\rangle|a^3\rangle$. This generates the Fibonacci sequence, which grows slightly slower than the powers of 2, but still gets to the desired power of a very efficiently. At the expense of an additional register, all squaring operations have been converted to multiplications.

It can be proven by induction that every positive integer has such a decomposition [15], which can be easily found by a greedy algorithm: repeatedly finding the largest Fibonacci number smaller than the current number, then subtracting off this Fibonacci number from the current number. This decomposition can be written as $z_i = \sum_{j=1}^{K} z_{ij} F_j$, where $z_{ij} \in \{0,1\}$, F_j is the jth Fibonacci number, and K is the largest such that $F_K \leq D$. By using Binet's formula, it can be shown that $K = O(\sqrt{n})$.

DOI: 10.54254/2755-2721/110/2024MELB0122

This decomposition is used to compute $\prod_{i=1}^{d} a_i^{z_i} \bmod N$ [4]:

$$\prod_{i=1}^{d} a_i^{z_i} = \prod_{i=1}^{d} \prod_{j=1}^{K} a_i^{z_{ij} F_j} = \prod_{j=1}^{K} \left(\prod_{i=1}^{d} a_i^{z_{ij}}\right)^{F_j} = \prod_{j=1}^{K} c_j^{F_j} \quad (4)$$

defining the numbers c_j. Note that each c_j can be computed exactly as in Regev's original algorithm [3], the only difference being the individual z_{ij} might not agree. This means the small numbers a_i can again be exploited to compute each c_j efficiently. The whole product can then be computed using the idea of Fibonacci exponentiation.

By analysing other multiplication algorithms, the number of qubits can be further reduced to $O(n)$, though at the expense of having more gates [4].

3.2. Results on noise tolerance

For Regev's original algorithm [3], all $\sqrt{n} + 4$ runs of the quantum procedure need to be successful for the classical post-processing. This is not ideal in practice as quantum systems are exposed to noise from the environment, making some or all samples corrupted. Ragavan and Vaikuntanathan also managed to improve on noise-tolerance by showing that under certain assumptions on the noise, a modified version of Regev's classical post-processing can work even when a fraction of the sample is corrupted [4]. This involves running an algorithm that filters out the corrupted samples.

By definition, if v_i is in the dual lattice, then $\langle u, v_i \rangle$ is an integer for any vector u in the lattice. A key observation in Regev's classical post-processing was that for any u not in the lattice, one can expect that there is some v_i in the dual lattice such that $\langle u, v_i \rangle$ is not close to any integer [3, 9]. This gave the ability to recognise which vectors u are in the lattice and which are not.

Assume for simplicity that now that corrupted samples are uniformly chosen vector in $[0,1]^d$, with probability of corruption p. The key idea is that if all the samples v_i from the dual lattice are not corrupted, then by a pigeon-hole argument, a small-number linear combination of v_i gives a vector in $\mathbb{Z}^d$ [*4*]. This implies that the linear combination of the noisy samples w_i gives a vector close to $\mathbb{Z}^d$. However, even one corrupted sample would mean that any linear combination of the samples is uniformly distributed in $[0,1]^d$, and thus unlikely to be close to $\mathbb{Z}^d$. Ragavan and Vaikuntanathan generalised the case of uniformly distributed error by intuitively abstracting the property that the error distribution is 'well-spread' with respect to the lattice. This gave a way to detect corrupted samples under a mild assumption on the error distribution.

Regev used the fact that $\sqrt{n} + 4$ samples of the dual lattice are enough to generate it with probability at least $1/2$ [11]. The case of Ragavan and Vaikuntanathan is different, since they first obtained $\alpha\sqrt{n}$ samples from the quantum procedure, for some $\alpha > 1$, then selected a subset of γd samples, which are assumed to be uncorrupted by running the error detection algorithm [4]. Thus, a stronger version of the fact [16] was applied – that provided γ is close enough to α, any such subset of γd samples generate the dual lattice with probability at least $1/2$ [4]. Finally, by assuming the error probability p is small enough, there are likely enough uncorrupted samples to make up a subset of γd, completing the argument. Once the subset of uncorrupted samples is obtained, Regev's classical post-processing can be applied with little modifications – the central argument is still running LLL to obtain a short basis for the original lattice.

Independently, Ekerå and Gärtner also discussed error-robustness of Regev's algorithm [5]. However, this relied on assumptions on the structure of the lattice and the error distribution, in particular the number theory assumption that the lattice $\mathscr{L}$ has a short basis, which is stronger than Regev's number theory assumption. As commented by Ragavan and Vaikuntanathan [4], this should be seen as a distinct contribution to noise-tolerance of Regev's algorithm.

3.3. Extension to discrete logarithm

In Shor's original paper, the quantum factoring algorithm was immediately extended to solve the discrete logarithm problem [2]. Like order-finding, this problem can be state in general groups. For

DOI: 10.54254/2755-2721/110/2024MELB0122

simplicity, only the version in $\mathbb{Z}_p^* = 1, \cdots, p-1$ is stated here: given $g \in \mathbb{Z}_p^*$ and $x = g^e$ for some unknown e, determine $e = \log_g x$ [1]. Note that r - the order of g - is assumed to be known, which can be found by first running the order-finding quantum algorithm [2].

Shor's approach was to essentially apply two copies of the order-finding circuit to two separate registers, one responsible for exponentiating g and the other responsible for exponentiating x [1]. For two integers a and b, the black-box computes $g^a x^b \bmod N$ and stores the result in a third register. At the end, QFT is applied to each of the first two registers, then the results are measured. This obtains two approximations, one to c/r and one to ec/r. The Continued Fraction Algorithm can then be applied to recover these two fractions, after which e can be found.

Ekerå and Gärtner modified Regev's algorithm to solve the discrete logarithm problem using lattices, which achieved a similar reduction in circuit size [5]. There were two approaches, one with pre-computation and one without. The one without pre-computations (other than finding r) is presented here.

To be able to exponentiate x and g, which are not necessarily small, the small-number restriction in Regev's algorithm must be relaxed a bit [5]. Notice that not all number b_i need to be small – there can be a constant number of them being large, without affecting the asymptotic size of the circuit. Thus, Ekerå and Gärtner proposed to look at the lattice

$$\mathcal{L}_{x,g} = \{(z_1, z_2, \ldots, z_d) \in \mathbb{Z}^d \mid \mathrm{x}^{z_{d-1}} \mathrm{g}^{z_d} \prod_{i=1}^{d-2} a_i^{z_i} = 1 \bmod N\}, \quad (5)$$

where a_i are $O(\log d)$-bit integers as before. The black-box then computes this new product modulo N, where the product of a_i can be computed efficiently as in Regev's algorithm, and $\mathrm{x}^{z_{d-1}}$ and g^{z_d} are computed using the typical repeated squaring. The number of qubits is also subject to the optimisation of Ragavan and Vaikuntanathan [4].

They then used the assumption that the lattice $\mathcal{L}_{x,g}$ has a short basis, a stronger assumption than the number theory assumption of Regev [5]. Fortunately, a version of the assumption has been proven by Pilatte, again under the relaxation that a_i are slightly larger numbers [13]. Under this assumption, the number e can then be retrieved. The quantum procedure and classical post-processing work the same as in Regev's algorithm. Using the LLL algorithm, a short basis for $\mathcal{L}_{x,g}$ can be recovered. Finally, this basis can be used to find $(0, \ldots, 0, 1, -e) \in \mathcal{L}_{x,g}$ and therefore e, which just follows from solving a linear system of equations.

The change of bases formula for logarithm can be exploited to further optimise the algorithm [5]. By using $\mathrm{e} = \log_g x = \log_{g'} x / \log_{g'} g$, the number g can be replaced by a small number g', and $\log_{g'} g$ can be pre-computed. Less resources are then needed to compute $\log_{g'} x$ as the large number g has been eliminated.

4. Conclusion

This paper has provided a comprehensive overview of Regev's recent advancements in quantum factoring algorithms, building on Shor's foundational work. By introducing multi-dimensional structures and utilising lattice-based techniques, Regev has significantly reduced the gate complexity for quantum factoring. This advancement represents a major step forward in quantum computing, offering potential efficiency improvements over Shor's algorithm despite the increased qubit requirements and reliance on an unproven number theory assumption. In addition, recent developments by Ragavan and Vaikuntanathan, Ekerå and Gärtner, and Pilatte have further refined Regev's approach, focusing on reducing qubit counts, enhancing noise tolerance, validating the number theory assumption, and extending the algorithm to other problems, like discrete logarithms. These contributions collectively push the boundaries of quantum computing, making the practical implementation of quantum factoring more feasible. Looking ahead, several intriguing research directions remain. Future work could explore alternative initialisation strategies, such as replacing the Gaussian superposition with uniform superpositions, to streamline the algorithm further. Additionally, improving the proof of Regev's number theory assumption for smaller numbers or finding ways to strengthen Pilatte's results on correctness

DOI: 10.54254/2755-2721/110/2024MELB0122

could bolster the algorithm's practical utility. Exploring other optimisations for error tolerance and robustness will also be crucial as quantum hardware continues to evolve. By addressing these challenges, researchers can continue to refine Regev's algorithm, paving the way for broader applications and advancements in quantum cryptography and computation.

References

[1] Nielsen MA, Chuang IL 2000 Quantum Computation and Quantum Information Cambridge University Press
[2] Shor PW 1994 Algorithms for quantum computation: Discrete logarithms and factoring 35th Annual Symposium on Foundations of Computer Science Santa Fe, New Mexico, USA IEEE Computer Society pp 124–134
[3] Regev O 2023 An Efficient Quantum Factoring Algorithm arXiv:2308.06572 [quant-ph] [online] Available: https://arxiv.org/abs/2308.06572
[4] Ragavan S, Vaikuntanathan V 2024 Space-Efficient and Noise-Robust Quantum Factoring arXiv:2310.00899 [quant-ph] [online] Available: https://arxiv.org/abs/2310.00899
[5] Ekerå M, Gärtner J 2024 Extending Regev's Factoring Algorithm to Compute Discrete Logarithms Post-Quantum Cryptography Springer Nature Switzerland pp 211–242 ISBN 9783031627460 DOI 10.1007/978-3-031-62746-0_10 [online] Available: http://dx.doi.org/10.1007/978-3-031-62746-0_10
[6] Kaye P, Laflamme R, Mosca M 2007 An Introduction to Quantum Computing Oxford University Press
[7] Harvey D, van der Hoeven J 2021 Integer multiplication in time O(n log n) Annals of Mathematics 2(193) 563–617
[8] Aharonov D, Regev O 2005 Lattice Problems in NP coNP [online] Available: https://cims.nyu.edu/~regev/papers/cvpconp.pdf
[9] Kiebert M 2024 Oded Regev's Quantum Factoring Algorithm [online] Available: https://vdwetering.name/pdfs/thesis-midas.pdf
[10] Regev O 2024 Presentation at Simons Institute: An Efficient Quantum Factoring Algorithm — Quantum Colloquium [online] Available: https://www.youtube.com/watch?v=Uzn93GjAfRg)
[11] Pomerance C 2001 The expected number of random elements to generate a finite abelian group Periodica Mathematica Hungarica 43(1-2) 191–198
[12] Regev O 2004 Lecture 2: LLL Algorithm [online] Available: https://cims.nyu.edu/~regev/teaching/lattices_fall_2004/ln/lll.pdf
[13] Pilatte C 2024 Unconditional correctness of recent quantum algorithms for factoring and computing discrete logarithms [online] Available: https://arxiv.org/pdf/2404.16450v1
[14] Kaliski BS Jr 2017 Targeted Fibonacci exponentiation arXiv preprint arXiv:1711.02491
[15] Zeckendorf E 1972 Representations of natural numbers by a sum of Fibonacci numbers and Lucas numbers Bulletin of the Royal Society of Sciences of Liege 179–182
[16] Acciaro V 1996 The probability of generating some common families of finite groups Utilitas Mathematica 243–254