SCIENTIFIC REPORTS

OPEN

SUBJECT AREAS:
INFORMATION THEORY AND COMPUTATION
QUANTUM PHYSICS
MATHEMATICS
GENERAL PHYSICS

Received
7 December 2011

Accepted
20 January 2012

Published
10 February 2012

Correspondence and requests for materials should be addressed to X.H.P. (xhpeng@ustc.edu.cn) or J.F.D (dif@ustc.edu.cn)

An Efficient Exact Quantum Algorithm for the Integer Square-free Decomposition Problem

Jun Li[1], Xinhua Peng[1], Jiangfeng Du[1] & Dieter Suter[2]

[1]Hefei National Laboratory for Physical Sciences at Microscale and Department of Modern Physics, University of Science and Technology of China, Hefei, Anhui 230026, People's Republic of China, [2]Fakultät Physik, Technische Universität Dortmund, 44221 Dortmund, Germany.

Quantum computers are known to be qualitatively more powerful than classical computers, but so far only a small number of different algorithms have been discovered that actually use this potential. It would therefore be highly desirable to develop other types of quantum algorithms that widen the range of possible applications. Here we propose an efficient and exact quantum algorithm for finding the square-free part of a large integer - a problem for which no efficient classical algorithm exists. The algorithm relies on properties of Gauss sums and uses the quantum Fourier transform. We give an explicit quantum network for the algorithm. Our algorithm introduces new concepts and methods that have not been used in quantum information processing so far and may be applicable to a wider class of problems.

A fundamental tenet of classical computer science is based on the Church-Turing thesis, which asserts that any practically realizable computational device can be simulated by a universal computer known as the Turing machine[1]. However, this hypothesis implicitly relies on the laws of classical physics[2] and was challenged by Feynman[3] and others who suggested that computational devices behaving according to quantum mechanics could be qualitatively more powerful than classical computers. A first proof of this conjecture was given in 1993 by Bernstein and Vazirani[4]. They showed that a quantum mechanical Turing machine is capable of simulating other quantum mechanical systems in polynomial time, an exponential improvement in computational power over the classical Turing machine. Their proof did not give an actual fast quantum algorithm, but in the following year, Peter Shor came up with his famous factoring algorithm[5], which solves the integer factorization problem in polynomial time, exponentially faster than any known classical algorithms. The essential part of this algorithm is a solution of the order-finding problem, which can be formulated as a hidden subgroup problem (HSP)[6]. A hidden subgroup problem is like to find out the period of a given periodic function. The structure of the function's periodicity may be so complicated that it can not be easily determined by classical means. The importance of the HSP is that various instances (eg. Pell's equation, the principal ideal problem, unit group computing) and variants like the hidden shift problem and hidden nonlinear structures encompass most of the quantum algorithms found so far that are exponentially faster than their classical counterparts[7]. This relatively narrow range of existing fast quantum algorithms shows the urgent need for different types of quantum algorithms that will make other classes of problems accessible to efficient solutions.

Here we describe such a quantum algorithm that does not fall into the framework of HSP. It solves two number-theoretical problems in polynomial time, i.e., testing the square-freeness and computing the square-free part of a given integer. Compared to the known classical algorithms, this provides an exponential increase in computational efficiency. While these problems are related to the factorization problem solved by Shor, our algorithm relies on a different approach. Furthermore, while Shor's algorithm is probabilistic, the algorithm presented here is exact and its computational complexity is lower.

We consider a positive integer N with its unique prime factor decomposition $N = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$ (p_i are primes). N is called square-free if no prime factor occurs more than once, i.e., for all i ($i = 1, 2, \ldots, k$), $\alpha_i = 0$ or 1. An arbitrary positive integer can always be written as

$$N = r \cdot s^2, \tag{1}$$

where r is square-free, and this square-free decomposition is unique. Thus, usually r and s^2 are called the square-free part and the square part of N, respectively. The square-freeness testing problem corresponds to determining

whether $s = 1$. An additional problem consists in finding the square-free part r of N. These problems were listed as two unsolved open problems[8], since no efficient algorithm is currently known for either of them. Actually they may be no easier than the general problem of integer factorization[9]. It was found[10] that the factorization of $N = pq^2$ (p, q both prime) is almost as hard as the factorization of $N = pq$. This fact has been used in a proposed digital signature scheme called TSH-ESIGN, which is more efficient than any representative signature scheme such as elliptic curve and RSA based signature[10]. A concrete estimation of the lower bound of classical Boolean circuit complexity[11] showed that testing square-free numbers by unbounded fan-in circuits of bounded depth requires a superpolynomial size. On the other hand, the square-free part problem appears to be a representative of a larger class of computational problems. As an example, computing the ring of integers of an algebraic number field, one of the main tasks of computational algebraic number theory, reduces to it in deterministic polynomial time[12,13].

We now describe an efficient, exact quantum algorithm that solves both problems. It uses the Gauss sum[32–34], an important object which has been extensively investigated in mathematics (see supplementary information). Throughout this paper, we will assume that N is an odd integer (the case of even numbers can be trivially reduced to this case). The Gauss sum is defined as

$$G(a,\chi) = \sum_{m=0}^{N-1} \chi_N(m) e^{2\pi i a m/N},$$

where a is an integer and the function $\chi_N(m)$ represents the Jacobi symbol of m relative to N[14].

The evaluation of the Gauss sum is closely related to the square-freeness of N. Let notation (x, y) indicate the greatest common divider (GCD) of x and y. If N is square-free, then we have

$$G(a,\chi) = 0, \quad \forall (a, N) > 1. \tag{2}$$

Conversely[15], if N is not square-free

$$G(a, \chi) = 0, \quad \forall (a, N) = 1. \tag{3}$$

This remarkable fact suggests a dichotomy criterion for testing square-freeness, it represents the cornerstone of our algorithm.

Results

We present the algorithm first for the relatively simple case where $N = pq^2$ (p, q both prime) and subsequently generalize it. The algorithm consists of two parts, as illustrated in Fig. 1. In the first part, we generate the state

$$|\phi\rangle = \frac{1}{\sqrt{\varphi(N)}} \sum_{(m,N)=1} \chi(m, N)|m\rangle, \tag{4}$$

where the normalization coefficient $\varphi(N)$ represents Euler's function (number of integers smaller than N that are coprime to N). van Dam and Seroussi[16] proposed a general method for preparing such a superposition state. They gave the example of computing the Legendre symbol, which is a special case of the Jacobi symbol, which reduces to the Legendre symbol when N is prime. They also computed the Jacobi symbol for the case when the factorization of N is known. In our case, the factors of N are not known. Thus we would adopt another technique for computing the Jaocbi symbol[17], which we discuss in the following. The second part of the algorithm is to apply the quantum Fourier transform (QFT) to $|\phi\rangle$. The resulting state encodes the factors p and q of N, which can be retrieved by performing measurements on the qubits.

Now we discuss the details of the algorithm. Set $n = [\log N]$, the smallest integer for which $2^n \geq N$. We need two main registers A and B, both initialized to $|0\rangle^{\otimes n}$. Additional registers needed for storing auxiliary variables and constants are not represented explicitly for simplicity. The first part starts with a state uniformly superposed from 1 to $N - 1$, which is prepared just by an $N - 1$ dimensional Fourier transform on register A and a subsequent addition with 1

$$|0\rangle_A^{\otimes n}|0\rangle_B^{\otimes n} \to \frac{1}{\sqrt{N-1}} \sum_{m=1}^{N-1} |m\rangle_A|0\rangle_B^{\otimes n}.$$

Note that this Fourier transform is of order $N - 1$, and it was known[18] that the quantum fast Fourier transform can be made exact for arbitrary orders. Next we compute the greatest common divisor of m and N into register B

$$U_1 : \frac{1}{\sqrt{N-1}} \sum_{m=1}^{N-1} |m\rangle_A|0\rangle_B^{\otimes n} \to \frac{1}{\sqrt{N-1}} \sum_{m=1}^{N-1} |m\rangle_A|(M, N)\rangle_B. \tag{5}$$

Classically, the GCD problem can be efficiently solved by the classical Euclidean algorithm in quadratic polynomial time. In order not to involve the complicated division arithmetics of the Euclidean algorithm, we prefer to adopt the extended Euclidean algorithm[19]. The extended Euclidean algorithm can be directly generalized to a quantum GCD algorithm that operates on a superposition state with the same computational complexity (see supplementary information for the quantum network construction).

We then take a measurement M_1 of register B. If the result is not 1, then it must be p or q or q^2 and clearly the algorithm already succeeds. However, it's highly possible that we would not obtain such results, and the algorithm continues. This is because the probability of obtaining $(m, N) = 1$ is $\varphi(N)/(N - 1) = (p - 1)(q - 1)/(pq - 1)$, which asymptotically approaches 1 for sufficiently large p and q. If M_1 results in 1, we get

$$\frac{1}{\sqrt{\varphi(N)}} \sum_{(m,N)=1} |m\rangle_A|1\rangle_B.$$

The next step is to obtain the state $|\phi\rangle$ as given in (4), i.e., we do the following unitary operation on register A

$$U_2 : \frac{1}{\sqrt{\varphi(N)}} \sum_{(m,N)=1} |m\rangle \to \frac{1}{\sqrt{\varphi(N)}} \sum_{(m,N)=1} \chi(m)|m\rangle, \tag{6}$$

where $\chi(m)$ are 1 or -1 as by the definition of Jacobi symbol, and register B is omitted. The key part of U_2 is to compute the Jacobi symbol $\chi(m)$ for all $(m, N) = 1$. Classically, the Jacobi symbol can be efficiently solved by many algorithms. There exists[20] a binary algorithm which has the advantage of lower complexity and easier

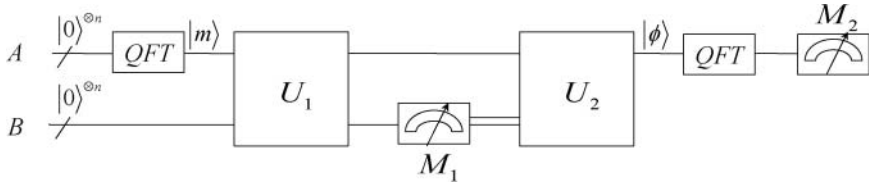

Figure 1 | Outline of quantum circuit for computing the square-free part for $N = pq^2$. The procedure denoted as Ω in the text consists of two main parts. In the first part, we generate the state $|\phi\rangle$; in the second part, we apply the quantum Fourier transform (QFT) to it. Single lines represent qubits, and boxes represent operations. Time runs from left to right. The transformation U_1 and U_2 are defined by Eq. (5) and Eq. (6). The meters M_1 and M_2 represent the measurements. The double lines coming from M_1 carry the classical bits, here the algorithm continues only if register B collapses to 1.

implementation on a binary computer. The binary algorithm can be seen as a variant of the extended Euclidean algorithm, and hence can also be extended to a quantum algorithm (see supplementary information for the quantum network construction).

As the last step of the algorithm, we take a Fourier transform on $|\phi\rangle$ and obtain

$$|\psi\rangle = \frac{1}{\sqrt{N_\varphi(N)}} \sum_{k=0}^{N-1} \left(\sum_{(m,N)=1} \chi(m) e^{2\pi i m k/N} \right) |k\rangle = \frac{1}{\sqrt{N_\varphi(N)}} \sum_{k=0}^{N-1} G(k,\chi)|k\rangle. \quad (7)$$

According to the properties (2) and (3) of the Gauss sum, all amplitudes vanish unless k shares a nontrivial common factor with N. If we perform a measurement M_2 on the register, it always collapses to a state $|k_0\rangle$, whose GCD with N is a non-trivial factor p or q of N. It therefore yields the complete decomposition of N.

We now determine the computational complexity of this algorithm. All the transformations involved in the algorithm, including the extended Euclidean algorithm for GCD and Jacobi symbol and QFT, require $O((\log N)^2)$ elementary gate operations[6]. Thus this algorithm has only a polynomial-time complexity.

For a general N with possibly many distinct prime factors (square-freeness of N is unknown), the procedure outlined above may not work. However, it can be generalized to include this case, and the generalized algorithm remains simple and efficient. We refer to the algorithm described above as Ω and discuss now the generalized algorithm, which includes Ω as a subroutine.

As we discussed, the algorithm Ω includes two measurements, M_1 and M_2. With a certain probability, M_1 yields a nontrivial factor of N. If this does not happen, we proceed to the second measurement M_2. Two possibilities will occur at M_2 due to the dichotomy property of Gauss sum (2, 3) : we obtain (*i*) a non-trivial factor of N if N is not square-free, or (*ii*) a result coprime to N, which signifies that N is definitely square-free. As a result, no matter whether Ω ends at M_1 or M_2, it either yields a non-trivial factor (say c) of N or determines that N is square-free. In the latter case, we have succeeded already, hence the algorithm finishes. In the former case, if the two parts c and N/c share a common factor $d = (c, N/c)$, we know that d^2 is a factor of the square part s^2 of N. We thus can split the problem of finding the square-free part r of N into two smaller problems: finding the square-free parts of c/d and $N/(cd)$. From the solutions of these subproblems, we find the corresponding parts of N as

$$r = R(N) = R(c/d)\cdot R(N/cd)$$
$$s^2 = S(N) = S(c/d)\cdot S(N/cd)\cdot d^2. \quad (8)$$

Here, $R(\cdot)$ and $S(\cdot)$ represent the square-free part and the square part of their argument, respectively. Clearly, this procedure can be iterated until all branches have determined that the arguments are square-free. Figure 2 illustrates this recursive procedure.

The execution time of the extended algorithm reaches a maximum when each execution of Ω yields just one factor, but clearly, the number of repetitions is still bounded by $O(\log N)$. Each execution of the subroutine Ω requires at most $O((\log N)^2)$ steps. The worst-case complexity of the extended algorithm is therefore $O((\log N)^3)$. Actually, we have a better estimation of how long it takes untill the

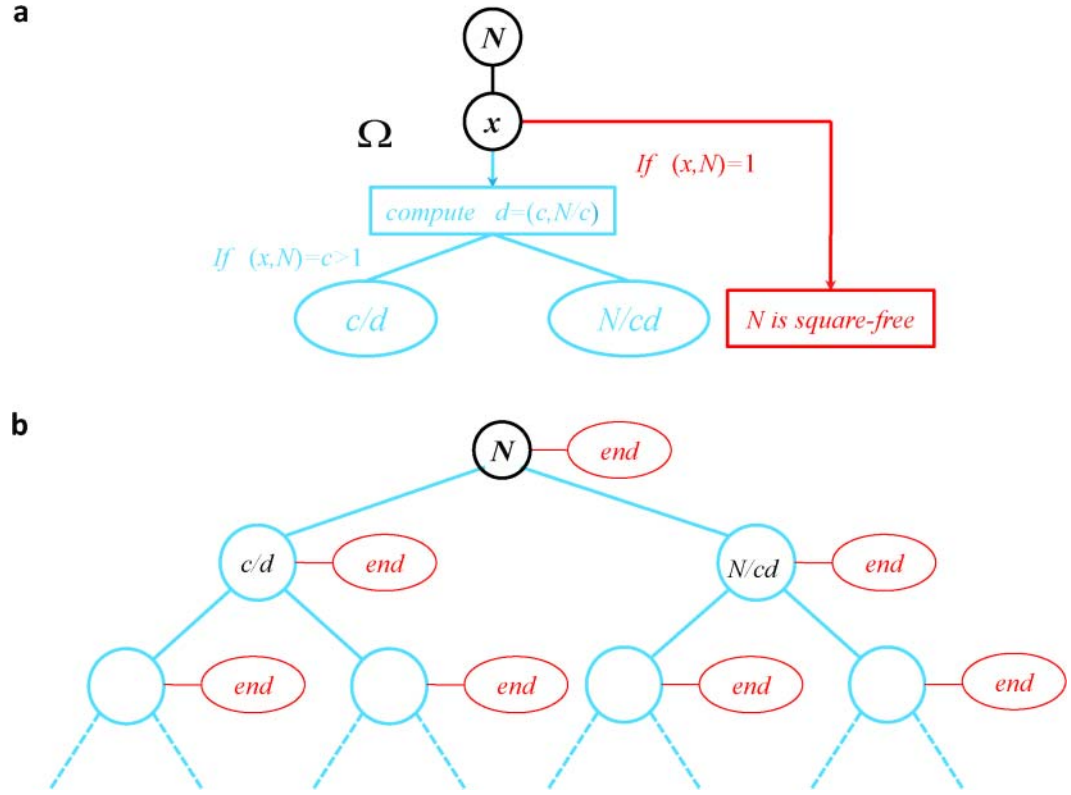

Figure 2 | Schematic flow chart of the recursive quantum algorithm for computing the squrare-free part of an arbitrary odd integer *N*. (a) Possible outcomes of applying the algorithm Ω on an arbitrary odd integer N: either return a factor c or else ensure that N is a square-free number with the square-free part $r = N$. If a factor $c > 1$ is returned, and N is tested to be not square-free, then the problem is converted to two smaller sub-problems for c/d and N/cd where d is the greatest common divisor of c and N/c.This serves as the subroutine of the recursive quantum algorithm. (b) Recursive algorithm for a general N. Different colors are used to designate two different outcomes after applying the subroutine Ω. The red color denotes that number is square-free, then this branch terminates. The blue color denotes the other outcome; in this case, the algorithm proceeds to the next step of recursion. The Ω operation needs to be performed at most log N times to solve this problem.

algorithm succeeds. This is by virtue of the observation that M_2 yields the square part with high probability, and calculations show that the algorithm will finish with high probability in just $O((\log N)^2(\log\log N)^2)$ (see methods).

Discussion

Classically, finding the square-free part of an integer is believed to be very difficult. It was argued[10] that the best method known for its solution is through factorization. The fastest classical algorithm for factorization would be the number field sieve[21], which requires $O(\exp(c(\log N)^{1/3}(\log\log N)^{2/3}))$ steps. Thus the quantum algorithm presented here offers an exponential speed-up over the classical algorithm. A feasible alternative to our algorithm would be to use Shor's algorithm to obtain the complete decomposition of N also in polynomial time. Application of Shor's algorithm yields, with some probability, two divisors of N in time $O((\log N)^2\log\log N\log\log\log N)$[6]. Like our algorithm, Shor's algorithm would thus also be applied repetitively, with the number of iterations bounded by $O(\log N)$. The overall computational complexity using Shor's algorithm would be $O((\log N)^3\log\log N\log\log\log N)$. We further remark that achieving complete factorization through Shor's algorithm raises more subtleness. A necessary part of complete factorization is primality test, however Shor's algorithm fails to recognize a prime number with probability 1, this of course increases algorithmic complexity[30,31]. Figure 3 compares the computational costs of the three algorithms described above, clearly showing the increase in computational efficiency by the algorithm presented here.

Our algorithm relies on the mathematical properties of the Gauss sums. The possibility of using the periodicity properties of Gauss sums for factorization was suggested earlier[22,23] and the feasibility of this approach was demonstrated in various physical systems including nuclear magnetic resoance[24–26], cold atoms[27] and superconducting circuits[28]. However, these schemes did not use the specific properties of quantum mechanical systems. They can be implemented in classical as well as in quantum systems and the scaling properties are therefore not superior to other classical algorithms[32,33]. In contrast, the algorithm that we have described in this paper relies on quantum superpositions and is both efficient and exact in solving the square-free part computation problem, even demonstrates advantages over Shor's approach. In Shor's algorithm, the major cost comes from the modular exponentiation operation, while Gauss sums can be generated through $O((\log N)^2)$ modular square operation. In our algorithm, we have noticed that Gauss sum evaluations are closely related to the factorization of N. While we have not found such an algorithm so far, it may thus be possible to develop a quantum algorithm on the basis of Gauss sums that solves integer factorization.

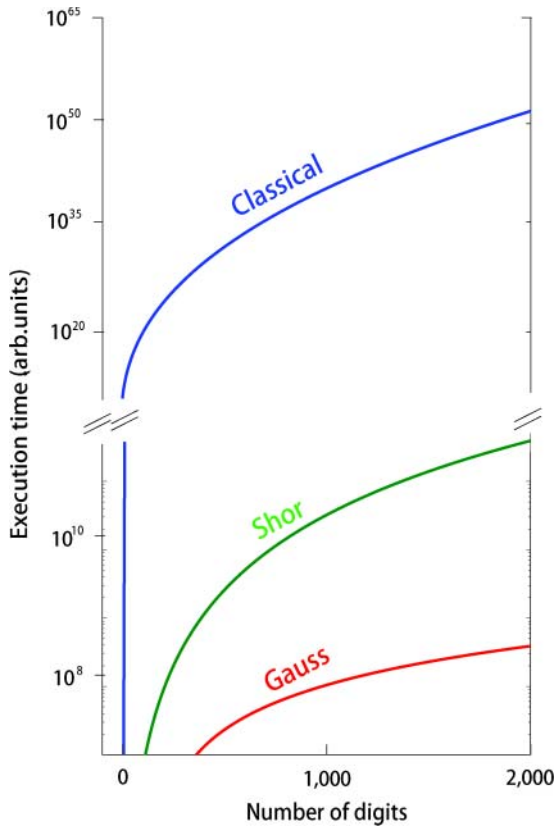

Figure 3 | Comparison between the computational costs of the three algorithms discussed in the text. Both quantum algorithms offer exponential speedup over the classical methods. For hundreds of digits, our algorithm is almost two orders of magnitude faster than the Shor's algorithm.

Methods

Realization of U_1. U_1 is to compute the greatest common divisor of m and N. Classically, the GCD problem can be efficiently solved based on the famous extended Euclidean algorithm. There is a variant of this algorithm, called the binary GCD algorithm, which can be more conveniently performed on a binary computer. We adopt this method here, and succeed in finding a quantum network that performs the binary GCD algorithm on a quantum superposition state (see supplementary information for details).

Realization of U_2. In Fig. 1, the operation U_2 is realized through the following steps

(0).	$\sum_{(m,N)=1}\|m\rangle_A\|1\rangle_B$	initial state
(1).	$\rightarrow \sum_{(m,N)=1}\|m\rangle_A\|\chi(m)\rangle_B$	apply the operation of Jacobi symbol computation
(2).	$\rightarrow \sum_{(m,N)=1} e^{i\pi(\chi(m)-1)/2}\|m\rangle_A\|\chi(m)\rangle_B$	apply conditional phase shifts
(3).	$\rightarrow \sum_{(m,N)=1}\chi(m)\|m\rangle_A\|1\rangle_B$	apply step (1) in reverse order
(4).	$=\|\varphi\rangle_A\|1\rangle_B$,	

where we use the phase kickback trick[16] and the identity $e^{i\pi(\chi(m)-1)/2}=\chi(m)$. The computation of Jacobi symbol can be implemented by binary Jaocbi algorithm (see supplementary information for details).

Complexity estimation of the algorithm. In the following, we discuss the algorithm complexity for a general case N. To do this, we slightly change the algorithm presented in the text. Our analysis is based on the finding: if at the measurement M_2 we obtain a result whose common divisor with N is a square number, then the common divisor must be the square part s^2 of N; and the probability of this case is larger than $(\varphi(N)/N)^2$ (see supplementary information for proofs). Hence the algorithm can be altered in the way that if any branch of the algorithm proceeds to M_2 and results in a square number, then that branch terminates.

Denote $P(\cdot)$ as the probability of obtaining the square part of its argument by application of Ω. Let p_k denotes the probability that the algorithm succeeds at the k-th iteration step. Obviously

$$p_1 = P(N),$$

where $P(N)\geq(\varphi(N)/N)^2$. If Ω does not succeed at the first step, and suppose we have obtained c and N/c and $d=(c, N/c)$, then

$$\begin{aligned} p_2 &= (1-P(N))P\left(\frac{c}{d}\right)P\left(\frac{N}{cd}\right) \\ &\geq (1-P(N))\left(\frac{\varphi(c/d)}{c/d}\frac{\varphi(N/cd)}{N/cd}\right)^2 \\ &\geq (1-P(N))\left(\frac{\varphi(N)}{N}\right)^2 \\ &= (1-P(N))P(N). \end{aligned}$$

Here, the second inequality is valid because of a basic property of the Euler function

$$\varphi(mn)=\varphi(m)\varphi(n)\frac{\gcd(m,\,n)}{\varphi(\gcd(m,\,n))}.$$

Analogously, we will have

$$p_3 \geq (1-P(N))^2 P(N)$$
$$\cdots$$
$$p_k \geq (1-P(N))^{k-1} P(N).$$

Therefore, after k steps, the probability that the algorithm still does not succeed is

$$Q \leq 1=\sum_{i=1}^{k}(1-P(N))^{i-1}P(N)=(1-P(N))^k \leq \left(1-\left(\frac{\varphi(N)}{N}\right)^2\right)^k.$$

According to the inequality (Theorem 8.8.7[29])

$$\frac{\phi(N)}{N}>\frac{1}{e^{\gamma}\log\log N+\frac{3}{\log\log N}},$$

where $\gamma = 0.5772\ldots$ is the Euler-Mascheroni constant, and for a large N, $\varphi(N)/N > 1/(2 \log\log N)$.

So we have

$$Q<\left(1-\left(\frac{1}{2\log\log N}\right)^2\right)^k. \quad (9)$$

When $k = O((\log\log N)^2)$, $Q \to 0$, this means, the algorithm doesn't need to go for $k = O(\log N)$ times, but would finish with high probability in $O((\log\log N)^2)$ steps.

1. Turing, A. M. On computable numbers, with an application to the entscheidungsproblem. *Proc. Lond. Math. Soc.* (2) **42**, 230–265 (1936).
2. Landauer, R. Information is physical. *Phys. Today* **44**, 23 (1991).
3. Feynman, R. P. Quantum mechanical computers. *Found. Phys.* **16**, 507–531 (1986).
4. Bernstein, E. & Vazirani, U. Quantum complexity theory. *Proc. 25th ACM Symp. on Theory of Computing*, 11–20 (1993).
5. Shor, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* **26**, 1484–1509 (1997).
6. Nielsen, M. A. & Chuang, I. L. *Quantum Computation and Quantum Information* (Cambridge Univ. Press, Cambridge, 2000).
7. Childs, A. M. & van Dam, W. Quantum algorithms for algebraic problems. *Rev. Mod. Phys.* **82**, 1–52 (2010).
8. Adleman, L. M. & Mccurley, K. S. Open problems in number-theoretic complexity. *Lect. Notes Comput. Sci.* **877**, 291–322 (1994).
9. Buchmann, J. A. & Lenstra, H. W. Approximating rings of integers in number fields. *J. Théor. Nombres Bordeaux* **6**, 221–260 (1994).
10. Okamoto, T. & Uchiyama, S. A new public-key cryptosystem as secure as factoring. *Lect. Notes Comput. Sci.* **1403**, 308–318 (1998).
11. Bernasconi, A. & Shparlinski, I. Circuit complexity of testing squarefree numbers. *Lect. Notes Comput. Sci.* **1563**, 47–56 (1999).
12. Chistov, A. L. The complexity of constructing the ring of integers of a global field. *Dokl. Akad. Nauk. SSSR* **306**, 1063–1067 (1989).
13. Lenstra, H. W. Algorithms in algebraic number theory. *Bull. Amer. Math. Soc.* **26**, 211–244 (1992).
14. Ireland, K. & Rosen, M. *A Classical Introduction to Modern Number Theory* (Springer-Verlag, New York, 1990).
15. Cohen, H. *Number Theory, Volume I: Tools and Diophantine Equations* Corollary 2.1.46. (Springer-Verlag, New York, 2007).
16. van Dam, W. & Seroussi, G. Efficient quantum algorithms for estimating Gauss sums. arXiv: quant-ph/0207131 (2002).
17. van Dam, W., Hallgren, S. & Ip, L. Quantum algorithms for some hidden shift problems. *SIAM J. Comput.* **36**, 763–778 (2006).
18. Mosca, M. & Zalka, C. Exact Quantum Fourier Transforms and Discrete Logarithm Algorithms. *INT J QUANTUM INF.* **2**, 91C100 (2004).
19. Koshy, T. *Elementary number theory with applications* (Elsevier, 2007).
20. Shallit, J. O. & Sorenson, J. P. A binary algorithm for the Jacobi symbol. *ACM SIGSAM Bulletin* **27**, 4C11. (1993).
21. Crandall, R. & Pomerance, C. *Prime Numbers: A Computational Perspective* (Springer, New York, ed. 2 2001).
22. Merkel, W. *et al.* Factorization of numbers with physical systems. *Fortschr. Phys.* **54**, 856–865 (2006).
23. Merkel, W. *et al.* Chirped pulses, Gauss sums and the factorization of numbers. *Int. J. Mod. Phys. B* **20** 1893–1916 (2006).
24. Mehring, M. *et al.* NMR experiment factors numbers with Gauss sums. *Phys. Rev. Lett.* **98**, 120502 (2007).
25. Mahesh, T. S., Rajendran, N., Peng, X. & Suter, D. Factorizing numbers with the Gauss sum technique: NMR implementations. *Phys. Rev. A* **75**, 062303 (2007).
26. Peng, X. & Suter, D. NMR implementation of factoring large numbers with Gauss sums: Suppression of ghost factors. *Europhys. Lett.* **84**, 40006 (2008).
27. Gilowski, M. *et al.* Gauss sum factorization with cold atoms. *Phys. Rev. Lett.* **100**, 030201 (2008).
28. Ng, H. T. & Franco Nori. Quantum phase measurement and Gauss sum factorization of large integers in a superconducting circuit. *Phys. Rev. A* **82**, 042317 (2010).
29. Bach, E. & Shallit, J. *Algorithmic Number Theory, Vol. 1: Efficient Algorithms* (MIT Press, Cambridge, 1996).
30. Buhrman, H. A short note on Shor's factoring algorithm. ACM SIGACT News. **27**, 89–90 (1996).
31. Chau, H. F. & Lo, H. K. Primality test via quantum factorization. arXiv: quant-ph/9508005 (1996).
32. Wölk, S. *et al.* Factorization of numbers with Gauss sums: I. Mathematical background. New J. Phys. 13, 103007 (2011).
33. Merkel, W. *et al.* Factorization of numbers with Gauss sums: II. Suggestions for implementation with chirped laser pulses. New J. Phys. 13, 103008 (2011).
34. Wölk, S. & Schleich, W. P. Factorization of numbers with Gauss sums: III. Algorithms with entanglement. New J. Phys. 14, 013049 (2012).

Acknowledgements

This work was supported by the Chinese Academy Of Sciences and National Natural Science Foundation of China through grant no. 10975124, and by the DFG through grant Su 192/19-1. The authors also thank W. Schleich who introduced us to the fascinating properties of Gauss sums.

Author Contributions Statement

All authors reviewed the manuscript. XP conceived the study; XP and JL designed the procedure, performed the analysis, and contributed mainly to the overall writing; DS brought the concept of Gauss sum to us and revised the paper and the figures; All authors contributed to the writing and reviewing of the paper.

Additional information

Supplementary information accompanies this paper at http://www.nature.com/scientificreports

Competing financial interests: The authors declare no competing financial interests.

How to cite this article: Li, J., Peng, X., Du, J. & Suter, D. An Efficient Exact Quantum Algorithm for the Integer Square-free Decomposition Problem. *Sci. Rep.* **2**, 260; DOI:10.1038/srep00260 (2012).