Odd orders in Shor’s factoring algorithm

Thomas Lawson

1 LTCI – Télécom ParisTech, 23 avenue d’Italie, 75013, Paris, France

(Dated: August 13, 2014)

Shor’s factoring algorithm (SFA) finds the prime factors of a number, \(N = p_1p_2\), exponentially faster than the best known classical algorithm. Responsible for the speed-up is a subroutine called the quantum order finding algorithm (QOFA) which calculates the order – the smallest integer, \(r\), satisfying \(a^r \mod N = 1\), where \(a\) is a randomly chosen integer coprime to \(N\) (in other words their greatest common divisor is one, \(\gcd(a, N) = 1\)). Given \(r\) the factors can usually be found: with probability not less than 1/2, they are given by \(p_1 = \gcd(a^{\frac{r}{2}} - 1, N)\) and \(p_2 = \gcd(a^{\frac{r}{2}} + 1, N)\). When the QOFA returns an odd value of \(r\) it is assumed the factors cannot be found (since \(a^{\frac{r}{2}}\) is not generally integer) and the QOFA is relaunched with a different value of \(a\). But a recent paper [E. Martin-Lopez et al., Nat Photon 6, 773 (2012)] (to which the author contributed) noted that the factors can sometimes be found from odd orders if the coprime is square. Here, we explain the role of odd orders in factoring. A natural question is “does considering odd values of \(r\) improve the probability of success of SFA?” We show that not discarding them – and using a second observation, that failure using a square coprime sometimes implies a new order finding relation – does, indeed, increase the probability of factoring, although only slightly. Even so, small improvements in the efficiency of the quantum algorithm may give substantial reductions in overall run-time, especially while quantum computations remain much harder to do than classical ones.

I. INTRODUCTION

The most famous application of quantum computers is Shor’s factoring algorithm (SFA), which promises to factor a number, \(N\), in time \(O((\log N)^3)\), much faster than the best known classical routines whose run time increases exponentially in the length of \(N\). SFA has been extensively studied theoretically, but it has not yet been convincingly demonstrated in the lab; the difficulty of controlling quantum systems means just a handful of experiments have been done, to test the basic principles [1–6]. These experiments are too simple to be of practical use but are, nonetheless, important. They have revealed previously unappreciated quirks of the algorithm, one of which – the role of odd orders – is the subject of this letter.

Much of SFA can be quickly done on a classical computer: the Euclidean algorithm lets one pick the coprime, \(a\), at random, and calculate the factors given \(r\). The part which is slow classically – and speeded-up by quantum mechanics – is the process at the heart of SFA: calculating \(r\). The quantum order finding algorithm (QOFA) uses phenomena such as quantum superposition and entanglement to calculate \(r\) efficiently. Even though in practice this is the hardest part of SFA to build, it is not the only source of failure. Sometimes, despite the QOFA finding \(r\) correctly, the classical algorithm does not return \(p_1\) and \(p_2\), either because it finds the trivial factors, 1 and \(N\), or because the QOFA returns an odd value of \(r\). In this case \(a^{\frac{r}{2}}\) is not generally integer and the QOFA is relaunched with a different value of \(a\) [7].

However, recently it was noticed [1] that the factors can sometimes be found from odd values of \(r\), if \(a\) is square. This is interesting for two reasons. First, it highlights a hitherto misunderstood aspect of SFA. Second, it suggests that it may be possible to reduce the number of failed runs in SFA. Although these are not fatal to the exponential speed-up, occurring with probability less than 1/2, reducing them cuts down on time-consuming quantum computations and could thus improve the run-time of SFA considerably.

In this letter we examine the effect of odd values of \(r\) on factoring and present two techniques for finding the order when \(r\) is odd.

II. ORDER FINDING

First, we review the classical part of SFA, showing where the factors come from.

By assumption, \(r\) is the smallest integer that respects \(a^r \mod N = 1\), or, equivalently,

\[ N | (a^{\frac{r}{2}} - 1)(a^{\frac{r}{2}} + 1), \]  

where \(|\) means divides. Assuming that \(a^{\frac{r}{2}}\) is integer, the factors can be found provided two conditions are met,

\[ N \not| a^{\frac{r}{2}} - 1 \]  
\[ N \not| a^{\frac{r}{2}} + 1, \]

where \(|\) means does not divide. If so, \(a^{\frac{r}{2}} \pm 1\) must each be divisible by one of the factors of \(N\) and, hence, the factors are \(p_1 = \gcd(a^{\frac{r}{2}} - 1, N)\) and \(p_2 = \gcd(a^{\frac{r}{2}} + 1, N)\).

If one of the conditions (2) or (3) is not met, a factor will not be found, except if it is equal to one, two or \(N\), in which case \(N\) is either even (and is, thus, easy to factor without a quantum computer) or one of the trivial factors has been found. For instance, the factor \(p_1 = \gcd(a^{\frac{r}{2}} - 1, N)\) divides \(a^{\frac{r}{2}} - 1\). If condition (3) is not satisfied (so that \(N(a^{\frac{r}{2}} + 1)\), \(p_1\) must also divide \(a^{\frac{r}{2}} + 1\), which is only possible if \(p_1 \leq 2\) since \(p_1\), \(a\) and \(r\)
are integers. Furthermore, the relation $N | a^{\frac{p_2}{2}} + 1$ implies $p_2 = N$, a trivial factor. (The same argument can be applied to the condition (2).) Hence, for any interesting $N$, we consider the conditions (2) and (3) necessary and sufficient for finding the factors.

The QOFA returns the trivial factors when these conditions are not satisfied. But the subject of this letter is the second cause of failure, odd values of $r$.

III. FACTORING WITH ODD ORDERS

Martin-Lopez et al. [1] considered factoring 21 with the coprime four giving order three. Despite the order being odd, the algorithm successfully returns the factors, $3 = \gcd(4^2 + 1, 21)$ and $7 = \gcd(4^2 - 1, 21)$, which are integer because the coprime is square.

Square coprimes do not always allow this trick, however. Take factoring 21 with coprime 16. The order three leads to the trivial factors $\gcd(16^2 + 1, 21) = 1$ and $\gcd(16^2 - 1, 21) = 21$ because the condition (2) is not met, $21 | 16^2 - 1$.

So how often are the factors found from odd orders? To calculate this we must know the effect of square coprimes.

We start with a definition. Let a square number, $b$, be written

$$b = a^{2m},$$

for positive integer $m$, in terms of a non-square root, $a$.

We define the order $s$ to be the smallest integer satisfying the order relation for coprime $b$,

$$b^s \mod N = 1,$$  

which, according to equation (4), can be written $a^{2ms} \mod N = 1$. Clearly $a$ has its own order, $r$,

$$a^r \mod N = 1.$$  

Since $r$ is optimal (there is no smaller integer satisfying equation (6)) we have $r | 2ms$. Without loss of generality we write $r = 2^n r_0$, where $r_0$ is odd, and hence,

$$s = x2^{n-m}r_0,$$

where $x$ is the smallest positive integer such that $s$ is integer. The value of $x$ depends on $n$ and $m$.

First, consider $n > m$, meaning that $r$ is even (since $m > 0$). In this case $x = 1$ and $s = 2^{n-m}r_0$ (which is even) and so the order finding relation,

$$b^{2^{n-m}r_0} \mod N = 1,$$

is identical to equation (6); here, order finding with $b$ is the same process as order finding with $a$.

Second, if $n = m$ (meaning $r$ is even) then, again, $x = 1$, the order finding relation for $b$ is identical to that for $a$. But this time $s$ is odd, $s = r_0$.

Finally, if $n < m$, then $x = 2^{m-n}$ and $s = r_0$ (so $s$ is odd). Equation (5) can be written in terms of $a$ and $r$,

$$a^{2^{m-n}} \mod N = 1.$$  

$2^{m-n-1}$ is integer, so the condition (2) is not satisfied, $N | a^{2^{m-n-1}} - 1$, and, thus, the factors are not found.

Using coprime $b$, SFA gives the factors only in the first case – when $s$ is even – and only if $a$ also gives the factors. Considering odd orders improves this slightly. When $n = m$, the factors, $b^{2^{m}} \pm 1$ are identical to those arising from the coprime $a$, $a^{2^{m-n}} \pm 1$, and so are found whenever they would have been found using $a$.

This explains the calculation in reference [1], where the coprime $b = 4$ gave order $s = 3$, equivalent to using $a = 2$ as the coprime, giving order $r = 6$ (here, $m = n = 1$).

A second observation sometimes lets us retrieve the factors from a failed calculation. Factoring $N = 21$ with the coprime $b = 16$ fails because the condition (2) is not met. But this implies a new order finding relation, in terms of the coprime $a = 4$,

$$16^2 \mod 21 \equiv 4^3 \mod 21 = 1.$$  

We have recovered the calculation of reference [1] which, of course, does satisfy the two conditions and leads to the factors, $\gcd(4^2 \pm 1, 21)$. This works when $n < m$. Failing the condition (2), $N | a^{2^{m-n-1}} - 1$, implies an order finding relation for the coprime $\sqrt{b} = a^{2^{m-1}},$

$$a^{2^{m-n-1}} \mod N = 1.$$  

This process can be repeated; the factors are not found if $N | a^{2^{m-n-2}} - 1$, in which case we have recovered the order finding relation for the coprime $b^{\frac{1}{2}} = a^{2^{m-2}}$. After $m-n$ repetitions we arrive at equation (9), and the problem is reduced to order finding with the root, $a$.

IV. THE EFFECT ON EFFICIENCY

These two techniques – considering odd orders and collapsing the coprime to its root – sometimes allow the factors to be found for odd values of $s$ (where the QOFA is normally considered to have failed). Using them, order finding with the coprime $b$ will lead to the factors if and only if it would have using the coprime $a$. We now calculate the improvement in probability of factoring they bring.

Let us consider SFA – without our techniques – for factoring $N = p_1p_2$.

Let the coprime $c$, picked uniformly at random ($1 < c < N$), have order $t$,

$$c^t \mod N = 1.$$  

When $c$ is the (non-square) root we will consider coprime $c = a$ (giving order $r$), otherwise we will use $c = b$ (with order $s$).
SFA finds the factors – given $c$ and $t$ – with probability
\[
P(\text{factors}|c, t) = P(\text{factors}|a, r)P(c = a) + P(\text{factors}|b, s)P(c = b),
\]
\[
= P(\text{factors}|a, r)(P(c = a) + P(n > m)P(c = b)),
\]
where $P$ means probability ($P(c = a)$ is the probability that $c$ is non-square, for instance) and where we have used $P(\text{factors}|b, s) = P(\text{factors}|a, r)P(n > m)$ since the factors are found from $b$ only if $n > m$ and if they could have been found using the coprime $a$.

Using our techniques the algorithm succeeds whenever $a$ would have lead to the factors, i.e. with probability
\[
P_n(\text{factors}|c, t) = P(\text{factors}|a, r).
\]
We compare this to the original,
\[
\frac{P(\text{factors}|c, t)}{P_n(\text{factors}|c, t)} = P(c = a) + P(n > m)P(c = b).
\]
But,
\[
P(t \text{ even}) = P(n > 0)P(c = a) + P(s \text{ even})P(c = b)
\geq P(n > m)
\]
because $P(s \text{ even}) = P(n > m)$ and $P(n > 0) \geq P(n > m)$ since $m > 0$, meaning that
\[
P(\text{factors}|c, t) \leq P(c = a) + P(t \text{ even})P(c = b).
\]
The probability that $c$ is non-square is $P(c = a) = 1 - 1/\sqrt{N}$. This also defines $P(c = b)$ since $P(c = a) + P(c = b) = 1$. All that remains is to calculate the probability of $t$ being even. We assume that $p_i - 1 = 2q_i$, where $q_i$ are odd. This is one of the harder calculations for SFA since it leads to lots of odd orders, and hence one where our techniques help most. (It is also a particularly interesting one, since numbers of this form are among the hardest to factor using classical routines, such as that proposed in reference [8] and so are likely candidates for SFA.) In this case $P(t \text{ even}) = 3/4$, following the argument of reference [7]. For completeness we sketch the proof here.

\textbf{Proof} The Chinese remainder theorem (CRT) tell us that choosing $c$ uniformly at random from $1 < c < N$ is equivalent to randomly picking two integers, $c_1$ ($1 < c_1 < p_1$) and $c_2$ ($1 < c_2 < p_2$), where $c = c_1 \mod p_1$. Let $t_i$ be the order satisfying
\[
c_i^{t_i} \mod p_i = 1.
\]
According to the CRT this order finding relation is also satisfied by $t$, giving $t = \text{LCM}(t_1, t_2)$, where LCM means least common multiple. $t$ is odd only when both $t_1$ and $t_2$ are odd. How likely is this? Answering this is made easier by the fact that the multiplicative group $\mod p_i$ is cyclic. The elements of this group can be written in terms of a generator, $g$. Thus, $c_i = g^{k_i} \mod p_i$ for some integer $k$ ($1 \leq k \leq p_i - 1$). The order finding relation implies $g^{k_i} \mod p_i = 1$. But $g^{k_i - 1} \mod p_i = 1$, meaning that $p_i - 1 | k_i$. $p_i - 1$ is even and so, if $k$ is odd, $t_i$ must be even. Alternatively, if $k$ is even,
\[
g^{(p_i - 1)/2} \mod p_i = 1,
\]
so $t_i$ must be odd since it divides $(p_i - 1)/2 = q_i$. The probability that $t_i$ is odd is therefore the probability that $k$ is even, which is $1/2$ since $c_i$ is picked at random. Hence, $P(t \text{ even}) = 1 - P(t_1 \text{ odd})P(t_2 \text{ odd}) = 3/4$.

In the best case, assuming $p_i - 1 = 2q_i$, our two techniques improve the probability of success of SFA,
\[
\frac{P(\text{factors}|c, t)}{P_n(\text{factors}|c, t)} \leq 1 - \frac{1}{4\sqrt{N}}.
\]

\textbf{V. DISCUSSION}

Of course, since these techniques allow the factors to be found whenever the coprime $a$ would have given them, one might alternatively perform the QOFA on non-square coprimes. This was suggested by Leander [10], who used a different approach to deal with odd orders. Noting that they lead SFA to fail, he showed that they can be avoided completely by picking coprimes that are non-square under modular arithmetic (this can be efficiently checked by calculating the Jacobi symbol). This improves the efficiency of Shor’s algorithm significantly. Using non-square coprimes also agrees with the recommendation of Markov & Saeedi [10] based on numerical evidence, to use small, prime coprimes like $a = 2, 3$ and 5 to increase the efficiency of factoring. The schemes of Markov & Saeedi, and Leander have the further advantage of never picking a square and its root in different runs of the same calculation which, as we have shown, is a waste of resources: working through the coprimes in order, $a = 2, 3, 4, \ldots$, until the factors are found is certainly not efficient!

Even in the favourable case we consider, our techniques improve the success probability only very slightly, especially for large $N$. But this may understate the savings in practice, for two reasons. First, in the near future SFA will probably use small coprimes – which are more likely to be square – because encoding big coprimes in small experiments is hard.

Second, quantum subroutines are – and will probably remain for some time – much harder to implement than classical ones. This is especially true of quantum circuits that use young, imperfect technologies, which introduce their own errors and hold-ups, and may – in the case of photonics, for instance – need to be reconfigured each time the calculation changes. While this is the case small improvements in efficiency may give substantial savings in run-time.
Indeed, this idea has already inspired several techniques for reducing the workload of the QOFA by processing more information on classical computers. Some consider (as we do) cases when the QOFA has correctly returned the order, but the factors have, nonetheless, not been found [9, 10]. Others try to extract useful information from erroneous outputs of the QOFA [11, 12]. These techniques may make experimentalists’ lives easier in the future – and bring about a convincing demonstration of SFA all the more quickly.

Acknowledgments
Thanks to Frederic Grosshans, Marc Kaplan, Anthony Laing, Marc-Andre Lajoie, Enrique Martin-Lopez and Benjamin Smith for valuable discussions. Support is acknowledged from Digiteo and the City of Paris project CiQWii.

---

[1] E. Martin-Lopez, A. Laing, T. Lawson, R. Alvarez, X.-Q. Zhou, and J. L. O’Brien, Nat Photon 6, 773 (2012).
[2] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang, Nature 414, 883 (2001).
[3] C.-Y. Lu, D. E. Browne, T. Yang, and J.-W. Pan, Phys. Rev. Lett. 99, 250504 (2007).
[4] B. P. Lanyon, T. J. Weinhold, N. K. Langford, M. Barbieri, D. F. V. James, A. Gilchrist, and A. G. White, Phys. Rev. Lett. 99, 250505 (2007).
[5] A. Politi, J. C. F. Matthews, and J. L. O’Brien, Science 325, 1221 (2009).
[6] E. Lucero, R. Barends, Y. Chen, J. Kelly, M. Mariantoni, A. Megrant, P. O’Malley, D. Sank, A. Vainsencher, J. Wenner, T. White, Y. Yin, A. N. Cleland, and J. M. Martinis, Nat Phys 8, 719 (2012).
[7] M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information (Cambridge U.P., 2000).
[8] J. M. Pollard, Mathematical Proceedings of the Cambridge Philosophical Society 76, 3049 (1974).
[9] G. Leander, arXiv: 0208183 [quant-ph] (2002).
[10] L. L. Markov and M. Saeedi, Quantum Info. Comput. 12, 361 (2012).
[11] P. W. Shor, SIAM J. Sci. Statist. Comput. 26, 1484 (1997).
[12] E. Knill, “On shor’s quantum factor finding algorithm: Increasing the probability of success and tradeoffs involving the fourier transform modulus,” Tech. Report LAUR-95-3350, Los Alamos Natl. Lab (1995).