QUANTUM COMPUTING
AND SHOR'S FACTORING ALGORITHM

Yu. I. Manin

1. Introduction

Information processing (computing) is dynamical evolution of a highly organized physical system produced by technology (computer) or nature (brain). Initial state of this system is (determined by) its input, final state is the output. Physics describes nature in two complementary modes: classical and quantum. Up to the nineties, the basic mathematical models of computing, Turing machines, were classical objects, although the first suggestions to study quantum models date back at least to 1980.

Theoretical quantum computing entered its modern stage in 1994, when P. Shor ([Sh]) devised the first quantum algorithm showing that prime factorization can be done on quantum computers in polynomial time, that is, considerably faster than by any known classical algorithm. (P. Shor's work was inspired by the earlier work [Si] of D. Simon). Shor's paper gave a new boost to the subject. Another beautiful result due to L. Grover ([Gro]) is that the quantum search among N objects can be done in $c\sqrt{N}$ steps. A. Kitaev [Ki1] devised new quantum algorithms for computing stabilizers of abelian group actions; his work was preceded by that of D. Boneh and R. Lipton [BoL] who treated a more general problem by a modification of Shor's method (cf. also [Gri]). At least as important as the results themselves, are the tools invented by Shor, Grover, and Kitaev. In this talk, I briefly explain some of the basic ideas of quantum computing and review Shor's algorithm.

2. Quantum parallelism

In this section we will discuss the basics: how to use the superposition principle in order to accelerate (certain) classical computations.

2.1. Description of the problem. *Let N be a large number, $F : \{0, \dots, N-1\} \to \{0, \dots, N-1\}$ a function such that the computation of each particular value $F(x)$ is tractable, that is, can be done in time polynomial in $\log x$. We want to compute (to recognize) some property of the graph $(x, F(x))$, for example:*

(i) Find the least period r of F, i.e. the least residue $r \bmod N$ such that $F(x + r \bmod N) = F(x)$ for all x (the key step in the Factorization Problem.)

(ii) Find some x such that $F(x) = 1$ or establish that such x does not exist (Search Problem.)

As we already mentioned, the direct attack on such a problem consists in compiling the complete list of pairs $(x, F(x))$ and then applying to it an algorithm recognizing the property in question. Such a strategy requires at least exponential time (as a function of the bit size of N) since already the length of the list is N. Barring a theoretical breakthrough in understanding such problems (for example, a proof that $P = NP$), a practical response might be in exploiting the possibility of parallel computing, i.e. calculating simultaneously many – or even all – values of $F(x)$. This takes less time but uses (dis)proportionally more hardware.

A remarkable suggestion due to D. Deutsch (see [DeuJ], [Deu]) consists in using a quantum superposition of the classical states $|x\rangle$ as the replacement of the union of N classical registers, each in one of the initial states $|x\rangle$. To be more precise, here is a mathematical model formulated as the definition.

2.2. Quantum parallel processing: version I. *Keeping the notation above, assume moreover that $N = 2^n$ and that F is a bijective map (the set of all outputs is a permutation of the set of all inputs).*

(i) Quantum space of inputs/outputs is the 2^n–dimensional complex Hilbert space H_n with the orthonormal basis $|x\rangle$, $0 \le x \le N-1$. Vectors $|x\rangle$ are called classical states.

(ii) Quantum version of F is the unique unitary operator $U_F : H_n \to H_n$ such that $U_F|x\rangle = |F(x)\rangle$.

Quantum parallel computing of F is (a physical realization of) a system with the state space H_n and the evolution operator U_F.

Naively speaking, if we apply U_F to the initial state which is a superposition of all classical states with, say, equal amplitudes, we will get simultaneously all classical values of F (i.e. their superposition):

$$U_F\left(\frac{1}{\sqrt{N}}\sum|x\rangle\right) = \frac{1}{\sqrt{N}}\sum|F(x)\rangle. \tag{1}$$

We will now discuss various issues related to this definition, before passing to its more realistic modification.

(A) We put $N = 2^n$ above because we are imagining the respective classical system as an n–bit register: cf. the discussion of Boolean circuits. Every number $0 \le x \le N-1$ is written in the binary notation $x = \sum_i \epsilon_i 2^i$ and is identified with the pure (classical) state $|\epsilon_{n-1},\dots,\epsilon_0\rangle$ where $\epsilon_i = 0$ or 1 is the state of the i–th register. The quantum system H_1 is called *qubit.* We have $H_n = H_1^{\otimes n}$, $|\epsilon_{n-1},\dots,\epsilon_0\rangle = |\epsilon_{n-1}\rangle \otimes \dots \otimes |\epsilon_0\rangle$.

This conforms to the general principles of quantum mechanics. The Hilbert space of the union of systems can be identified with the tensor product of the Hilbert spaces of the subsystems. Accordingly, decomposable vectors correspond to the states of the compound for which one can say that the individual subsystems are in definite states.

(B) Pure quantum states, strictly speaking, are points of the projective space $P(H_n)$ that is, complex lines in H_n. Traditionally, one considers instead vectors of norm one. This leaves undetermined an overall phase factor $\exp i\varphi$. If we have two state vectors, individual phase factors have no objective meaning, but their quotient, that is the difference of their phases, does have one. This difference can be measured by observing effects of interference. This possibility is used for implementing efficient quantum algorithms.

(C) If a quantum system S is isolated, its dynamical evolution is described by the unitary operator $U(t) = \exp iHt$ where H is the Hamiltonian, t is time. Therefore one option to implement U_F physically is to design a device for which U_F would be a fixed time evolution operator. However, this seemingly contradicts many deeply rooted notions of the algorithm theory. For example, calculating $F(x)$ for different inputs x takes different times, and it would be highly artificial to try to equalize them already in the design.

Instead, one can try to implement U_F as the result of a sequence of carefully controlled by a classical computer brief interactions of S with environment (say, laser pulses). Mathematically speaking, U_F is represented as a product of some standard unitary operators $U_m \dots U_1$ each of which acts only on a small subset (two, three) of classical bits. These operators are called *quantum gates.*

The complexity of the respective quantum computation is determined by its length (the number m of the gates) and by the complexity of each of them. The latter point is a subtle one: continuous parameters, e.g. phase shifts, on which U_i may depend, makes the information content of each U_i potentially infinite and leads to a suspicion that a quantum computer will in fact perform an analog computation, only implemented in a fancy way. A very interesting discussion in [Ts], Lecture 9, convincingly refutes this viewpoint, by displaying those features of quantum computation which distinguish it from both analog and digital classical information processing. This discussion is based on the technique of fault tolerant computing using quantum codes for producing continuous variables highly protected from external noise.

(D) From the classical viewpoint, the requirement that F must be a permutation looks highly restrictive (for instance, in the search problem F takes only two values). Physically the reason for this requirement is that only such F extend to unitary operators ("quantum reversibility"). The standard way out consists in introducing *two* n–bit registers instead of one for keeping the value of the argument as well as that of the function. More precisely, if $F(|x\rangle)$ is an arbitrary function, we can replace it by the permutation $\widetilde{F}(|x,y\rangle) := |x, F(x) \oplus y\rangle$, where $\oplus$ is the Boolean (bitwise) sum. This involves no more than polynomial increase of the classical complexity, and the restriction of $\widetilde{F}$ to $y = 0$ produces the graph of F which we need anyway for the type of problems we are interested in.

In fact, in order to process a classical algorithm (sequence of Boolean gates) for computing F into the quantum one, we replace each classical gate by the respective reversible quantum gate, i.e. by the unitary operator corresponding to it tensored by the identical operator. Besides two registers for keeping $|x\rangle$ and $F(|x\rangle)$ this trick introduces as well extra qubits in which we are not particularly interested. The corresponding space and its content is sometimes referred to as "scratchpad", "garbage", etc. Besides ensuring reversibility, additional space and garbage can be introduced as well for considering functions $F : \{0,\dots,N-1\} \to \{0,\dots,M-1\}$ where N, M are not powers of two (then we extend them to the closest power of two). For more details, see the next section.

Notice that the choice of gate array (Boolean circuit) as the classical model of computation is essential in the following sense: a quantum routine cannot use conditional instructions. Indeed, to implement such an instruction we must observe the memory in the midst of calculation, but the observation generally will change its current quantum state.

In the same vein, we must avoid copying instructions because the classical copying operator $|x\rangle \to |x\rangle \otimes |x\rangle$ is not linear. In particular, each output qubit from a quantum gate can be used only in one gate at the next step (if several gates are used parallelly): cloning is not allowed.

These examples show that the basics of quantum code writing will have a very distinct flavor.

We now pass to the problems posed by the input/output routines.

Input, or initialization, in principle can be implemented in the same way as a computation: we produce an input state starting e.g. from the classical state $|0\rangle$ and applying a sequence of basic unitary operators: see the next section. Output, however, involves an additional quantum mechanical notion: that of *observation.*

(E) The simplest model of observation of a quantum system with the Hilbert space H involves the choice of an orthonormal basis of H. Only elements of this basis $|\chi_i\rangle$ can appear as the results of observation. If our system is in some state $|\psi\rangle$ at the moment of observation, it will be observed in the state $|\chi_i\rangle$ with probability $|\langle\chi_i|\,\psi\rangle|^2$.

This means first of all that every quantum computation is inherently probabilistic. Observing (a part of) the quantum memory is not exactly the same as "printing the output". We must plan a series of runs of the same quantum program and the subsequent classical processing of the observed results, and we can hope only to get the desired answer with probability close to one.

Furthermore, this means that by implementing quantum parallelism simplemindedly as in (1), and then observing the memory as if it were the classical n–bit register, we will simply get some value $F(x)$ with probability $1/N$. This does not use the potential of the quantum parallelism. Therefore we formulate a corrected version of this notion, leaving more flexibility and stressing the additional tasks of the designer, each of which eventually contributes to the complexity estimate.

2.3. Quantum parallel processing: version II. *To solve efficiently a problem involving properties of the graph of a function F, we must design:*

(i) An auxiliary unitary operator U carrying the relevant information about the graph of F.

(ii) A computationally feasible realization of U with the help of standard quantum gates.

(iii) A computationally feasible realization of the input subroutine.

(iv) A computationally feasible classical algorithm processing the results of many runs of quantum computation.

All of this must be supplemented by the quantum error–correcting encoding, which we will not address here. In the next section, we will discuss some standard quantum subroutines.

3. Selected quantum subroutines

3.1. Initialization. Using the same conventions as in (1) and the subsequent comments, in particular, the identification $H_n = H_1^{\otimes n}$, we have

$$\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle = \frac{1}{\sqrt{N}} \sum_{\epsilon_i = 0,1} |\epsilon_{n-1} \dots \epsilon_0\rangle = \left(\frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \right)^{\otimes n} . \tag{2}$$

In other words,

$$\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle = U_1^{(n-1)} \dots U_1^{(0)} |0 \dots 0\rangle \tag{3}$$

where $U_1 : H_1 \to H_1$ is the unitary operator

$$|0\rangle \mapsto \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle), \ |1\rangle \mapsto \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle) ,$$

and $U_1^{(i)} = \mathrm{id} \otimes \dots \otimes U_1 \otimes \dots \otimes \mathrm{id}$ acts only on the i–th qubit.

Thus making the quantum gate U_1 act on each memory bit, one can in n steps initialize our register in the state which is the superposition of all 2^n classical states with equal weights.

3.2. Quantum computations of classical functions. Let $\mathcal{B}$ be a finite basis of classical gates containing one–bit identity and generating all Boolean circuits, and $F : \mathbf{F}_2^m \to \mathbf{F}_2^n$ a function. We will describe how to turn a Boolean circuit of length L calculating F into another Boolean circuit of comparable length consisting only of reversible gates, and calculating a modified function which however contains all information about the graph of F. Reversibility means that each step is a bijection (actually, an involution) and hence can be extended to the unitary operator, that is, quantum gate. For a gate f, define $\widetilde{f}(|x,y\rangle) = |x, f(x)+y\rangle$ as in 2.2(D) above.

3.2.1. Claim. *A Boolean circuit $\mathcal{S}$ of length L in the basis $\mathcal{B}$ can be processed into the reversible Boolean circuit $\widetilde{\mathcal{S}}$ of length $O((L+m+n)^2)$ calculating a permutation $H : \mathbf{F}_2^{m+n+L} \to \mathbf{F}_2^{m+n+L}$ with the following property:*

$$H(x,y,0) = (x, F(x)+y, 0) = (\widetilde{F}(x,y), 0).$$

Here x, y, z have sizes m, n, L respectively.

Proof. We will understand L here as the sum of sizes of the outputs of all gates involved in the description of $\mathcal{S}$. We first replace in $\mathcal{S}$ each gate f by its reversible counterpart $\widetilde{f}$. This involves inserting extra bits which we put side by side into a new register of total length L. The resulting subcircuit will calculate a permutation $K : \mathbf{F}_2^{m+L} \to \mathbf{F}_2^{m+L}$ such that $K(x,0) = (F(x), G(x))$ for some function G (garbage).

Now add to the memory one more register of size n keeping the variable y. Extend K to the permutation $\overline{K} : \mathbf{F}_2^{m+L+n} \to \mathbf{F}_2^{m+L+n}$ keeping y intact: $\overline{K} : (x,0,y) \mapsto (F(x), G(x), y)$. Clearly, $\overline{K}$ is calculated by the same boolean circuit as K, but with extended register.

Extend this circuit by the one adding the contents of the first and the third register: $(F(x), G(x), y) \mapsto (F(x), G(x), F(x)+y)$. Finally, build the last extension which calculates $\bar{K}^{-1}$ and consists of reversed gates calculating $\overline{K}$ in reverse order. This clears the middle register (scratchpad) and produces $(x, 0, F(x)+y)$. The whole circuit requires $O(L+m+n)$ gates if we allow to apply them to the not necessarily neighboring bits. Otherwise we must insert gates for local permutations which will replace this estimate by $O((L+m+n)^2)$.

3.3. Fast Fourier transform. Finding the least period of a function of one real variable can be done by calculating its Fourier transforms and looking at its maxima. The same strategy is applied by Shor in his solution of the factorization problem. We will show now that the discrete Fourier transform Φ_n is computationally easy (quantum polynomial time). We define $\Phi_n : H_n \to H_n$ by

$$\Phi_n(|x\rangle) = \frac{1}{\sqrt{N}} \sum_{c=0}^{N-1} |c\rangle \exp(2\pi i c x/N) \tag{4}$$

In fact, it is slightly easier to implement directly the operator

$$\Phi_n^t(|x\rangle) = \frac{1}{\sqrt{N}} \sum_{c=0}^{N-1} |c^t\rangle \exp(2\pi i c x/N). \tag{5}$$

where c^t is c read from the right to the left. The effects of the bit reversal can be then compensated at a later stage without difficulty.

Let $U_2^{(kj)} : H_n \to H_n$, $k < j$, be the quantum gate which acts on the pair of the k–th and j–th qubits in the following way: it multiplies $|11\rangle$ by $\exp(i\pi/2^{j-k})$ and leaves the remaining classical states $|00\rangle, |01\rangle, |10\rangle$ intact.

3.3.1. Lemma. *We have*

$$\Phi_n^t = \prod_{k=0}^{n-1} \left(U_1^{(k)} \prod_{j=k+1}^{n-1} U_2^{(kj)} \right). \tag{6}$$

By our rules of the game, (19) has polynomial length in the sense that it involves only $O(n^2)$ gates. However, implementation of $U_2^{(kj)}$ requires controlling variable phase factors which tend to 1 as $k-j$ grows. Moreover, arbitrary pairs of qubits must allow quantum mechanical coupling so that for large n the interaction between qubits must be non–local. The contribution of these complications to the notion of complexity cannot be estimated without going into the details of physical arrangement. Therefore I will add a few words to this effect.

The implementation of quantum register suggested in [CZ] consists of a collection of ions (charged atoms) in a linear harmonic trap (optical cavity). Two of the electronic states of each ion are denoted $|0\rangle$ and $|1\rangle$ and represent a qubit. Laser pulses transmitted to the cavity through the optical fibers and controlled by the classical computer are used to implement gates and read out. The Coulomb repulsion keeps ions apart (spatial selectivity) which allows to prepare each ion separately in any superposition of $|0\rangle$ and $|1\rangle$ by timing the laser pulse properly and preparing its phase carefully. The same Coulomb repulsion allows for collective excitations of the whole cluster whose quanta are called phonons. Such excitations are produced by laser pulses as well under appropriate resonance conditions. The resulting resonance selectivity combined with the spatial selectivity implements a controlled entanglement of the ions that can be used in order to simulate two and three bit gates. For a detailed and lucid mathematical explanation, see [Ts], Lecture 8.

Another recent suggestion ([GeC]) is to use a single molecule as a quantum register, representing qubits by nuclear spins of individual atoms, and using interactions through chemical bonds in order to perform multiple bit logic. The classical technique of nuclear magnetic resonance developed since the 1940's and allowing to work with many molecules simultaneously provides the start up technology for this project.

3.4. Quantum search. All the subroutines described up to now boiled down to some identities in the unitary groups involving products of not too many operators acting on subspaces of small dimension. They did not involve output subroutines and therefore did not "compute" anything in the traditional sense of the word. We will now describe the beautiful quantum search algorithm due to L. Grover which produces a new identity of this type, but also demonstrates the effect of observation and the way one can use quantum entanglement in order to exploit the potential of quantum parallelism.

We will treat only the simplest version. Let $F:\mathbf{F}_2^n\to\{0,1\}$ be a function taking the value 1 at exactly one point x_0. We want to compute x_0. We assume that F is computable in polynomial time, or else that its values are given by an oracle. Classical search for x_0 requires on the average about $N/2$ evaluations of F where $N=2^n$.

In the quantum version, we will assume that we have a quantum Boolean circuit (or quantum oracle) calculating the unitary operator $H_n\to H_n$

$$I_F:\,|x\rangle\mapsto e^{\pi iF(x)}|x\rangle.$$

In other words, I_F is the reflection inverting the sign of $|x_0\rangle$ and leaving the remaining classical states intact.

Moreover, we put $J=-I_\delta$, where $\delta:\mathbf{F}_2^n\to\{0,1\}$ takes the value 1 only at 0, and $V=U_1^{(n-1)}\dots U_1^{(0)}$, as in (16).

3.4.1. Claim. *(i) The real plane in H_n spanned by the uniform superposition ξ of all classical states (15) and by $|x_0\rangle$ is invariant with respect to $T:=VJVI_F$.*

(ii) T restricted to this plane is the rotation (from ξ to $|x_0\rangle$) by the angle φ_N where

$$\cos\varphi_N=1-\frac{2}{N},\ \sin\varphi_N=2\,\frac{\sqrt{N-1}}{N}.$$

The check is straightforward.

Now, φ_N is close to $\dfrac{2}{\sqrt{N}}$, and for the initial angle φ between ξ and $|x_0\rangle$ we have

$$\cos\varphi=-\frac{1}{\sqrt{N}}.$$

Hence in $[\varphi/\varphi_N]\approx\dfrac{\pi\sqrt{N}}{4}$ applications of T to ξ we will get the state very close to $|x_0\rangle$. Stopping the iteration of T after as many steps and measuring the outcome in the basis of classical states, we will obtain $|x_0\rangle$ with probability very close to one.

One application of T replaces in the quantum search one evaluation of F. Thus, thanks to quantum parallelism, we achieve a polynomial speed–up in comparison with the classical search. The case when F takes value 1 at several points and we only want to find one of them, can be also treated similarly and leads to the same speed–up. Thus the relevant NP–complete problem still takes exponential time to solve even with this quantum algorithm.

4. Shor's factoring algorithm

4.1. Notation. Let M be a number to be factored. We will assume that it is odd and is not a power of a prime number.

Denote by N the size of the basic memory register we will be using (not counting scratchpad). Its bit size n will be about twice that of M. More precisely, choose $M^2 < N = 2^n < 2M^2$. Finally, let $1 < t < M$ be a random parameter with $\gcd(t, M) = 1$. This condition can be checked classically in time polynomial in n.

Below we will describe one run of Shor's algorithm, in which t (and of course, M, N) is fixed. Generally, polynomially many runs will be required, in which the value of t can remain the same or be chosen anew. This is needed in order to gather statistics. Shor's algorithm is probabilistic one, with two sources of randomness that must be clearly distinguished. One is built into the classical probabilistic reduction of factoring to the finding of the period of a function. Another stems from the necessity to observe quantum memory which, too, produces random results.

More precise estimates than given here show that a quantum computer which can store about $3n$ qubits can find a factor of M in time of order n^3 with probability close to 1 : see [BCDP]. To the contrary, it is widely believed that no recursive function of the type $M \mapsto$ *a proper factor of* M belongs to PF.

4.2. Classical algorithm. Put

$$r := \min \{\rho \,|\, t^\rho \equiv 1 \bmod M\}$$

which is the least period of $F : a \mapsto t^a \bmod M$.

4.2.1. Claim. *If one can efficiently calculate r as a function of t, one can find a proper divisor of M in polynomial in $\log_2 M$ time with probability $\geq 1 - M^{-m}$ for any fixed m.*

Assume that for a given t the period r satisfies

$$r \equiv 0 \bmod 2, \;\; t^{r/2} \not\equiv -1 \bmod M$$

Then $\gcd(t^{r/2} + 1, M)$ is a proper divisor of M. Notice that gcd is computable in polynomial time.

The probability that this condition holds is $\geq 1 - \dfrac{1}{2^{k-1}}$ where k is the number of different odd prime divisors of M, hence $\geq \dfrac{1}{2}$ in our case. Therefore we will find a good t with probability $\geq 1 - M^{-m}$ in $O(\log M)$ tries. The longest calculation in one try is that of $t^{r/2}$. The usual squaring method takes polynomial time as well.

4.3. Quantum algorithm calculating r. Here we describe one run of the quantum algorithm which purports to compute r, given M, N, t. We will use the working register that can keep a pair consisting of a variable $0 \leq a \leq N - 1$ and the respective value of the function $t^a \bmod M$. One more register will serve as the scratchpad needed to compute $|a, t^a \bmod M\rangle$ reversibly. When this calculation is completed, the content of the scratchpad will be reversibly erased: cf. 3.2.1. In the remaining part of the computation the scratchpad will not be used anymore, we can decouple it, and forget about it.

The quantum computation consists of four steps, three of which were described in sec. 3:

(i) Partial initialization produces from $|0,0\rangle$ the superposition

$$\frac{1}{\sqrt{N}} \sum_{a=0}^{N-1} |a, 0\rangle.$$

(ii) Reversible calculation of F processes this state into

$$\frac{1}{\sqrt{N}} \sum_{a=0}^{N-1} |a, t^a \bmod M\rangle.$$

(iii) Partial Fourier transform furnishes then

$$\frac{1}{N} \sum_{a=0}^{N-1} \sum_{c=0}^{N-1} \exp\left(2\pi i a c/N\right) |c, t^a \bmod M\rangle.$$

(iv) The last step is the observation of this state with respect to the system of classical states $|c, m \bmod M\rangle$. This step produces some concrete output

$$|c, t^k \bmod M\rangle \tag{7}$$

with probability

$$\left| \frac{1}{N} \sum_{a:\, t^a \equiv t^k \bmod M} \exp\left(2\pi i a c/N\right) \right|^2 . \tag{8}$$

The remaining part of the run is assigned to the classical computer and consists of the following steps.

(A) *Find the best approximation (in lowest terms) to* $\frac{c}{N}$ *with denominator* $r' < M < \sqrt{N}$*:*

$$\left| \frac{c}{N} - \frac{d'}{r'} \right| < \frac{1}{2N}. \tag{9}$$

As we will see below, we may hope that r' will coincide with r in at least one run among at most polynomially many. Hence we try r' in the role of r right away:

(B) *If* $r' \equiv 0 \bmod 2$, *calculate* $\gcd(t^{r'/2} \pm 1, M)$.

If r' is odd, or if r' is even, but we did not get a proper divisor of M, repeat the run $O(\log\log M)$ times with the same t. In case of failure, change t and start a new run.

4.3.1. Justification. We will now show that, given t, from the observed values of $|c, t^k \bmod M\rangle$ in $O(\log\log M)$ runs we can find the correct value of r with probability close to 1.

Let us call the observed value of c *good*, if

$$\exists\, l \in \left[-\frac{r}{2}, \frac{r}{2}\right],\ rc \equiv l \bmod N.$$

In this case there exists such d that

$$-\frac{r}{2} \leq rc - dN = l \leq \frac{r}{2}$$

so that

$$\left|\frac{c}{N} - \frac{d}{r}\right| < \frac{1}{2N}.$$

Hence if c is good, than r' found from (22) in fact divides r.

Now call c *very good* if $r' = r$.

Estimating the exponential sum (21), we can easily check that the probability to observe a good c is $\geq \dfrac{1}{3r^2}$. On the other hand, there are $r\varphi(r)$ states $|c, t^k \bmod M\rangle$ with very good c. Thus to find a very good c with high probability, $O(r^2 \log r)$ runs will suffice.

Bibliography

[BCDP] D. Beckman, A. N. Chari, Sr. Devabhaktuni, J. Preskill. *Efficient networks for quantum computing.* Phys. Rev. A, 54:2 (1996), 1034–1063.

[Ben1] P. Benioff. *The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines.* J. Stat. Phys., 22 (1980), 563–591.

[Ben2] P. Benioff. *Quantum mechanical Hamiltonian models of Turing machines that dissipate no energy.* Phys. Rev. Lett., 48 (1980), 1581–1585.

[BoL] D. Boneh, R. Lipton. *Quantum cryptoanalysis of hidden linear functions.* Proc. of Advances in Cryptology — CRYPTO '95, Springer LN in Computer Science, vol. 963 (1995), 424–437.

[CZ] J. Cirac, P. Zoller. *Quantum computation with cold trapped ions.* Phys. Rev. Lett., 74:20 (1995), 4091–4094.

[Deu] D. Deutsch. *Quantum theory, the Church–Turing principle and the universal quantum computer.* Proc. R. Soc. Lond. A 400 (1985), 97–117.

[DeuJ] D. Deutsch, R. Jozsa. *Rapid solutions of problems by quantum computation.* Proc. Roy. Soc. London, Ser. A, 449 (1992), 553–558.

[Fe1] R. Feynman. *Simulating physics with computers.* Int. J. of Theor. Phys., 21 (1982), 467–488.

[Fe2] R. Feynman. *Quantum mechanical computers.* Found. Phys., 16 (1986), 507–531.

[Fr1] M. Freedman. *Topological views on computational complexity.* In Proc. ICM Berlin 1998, vol. II, 453–464.

[Fr2] M. Freedman. *Limit, logic, and computation.* Proc. Nat. Ac. Sci. USA, 95 (1998), 95–97.

[Fr3] M. Freedman. *P/NP, and the quantum field computer.* Proc. Nat. Ac. Sci. USA, 95 (1998), 98–101.

[GaJ] M. Garey, D. Johnson. *Computers and Intractability: A Guide to the Theory of NP–Completeness.* W. H. Freeman and Co., San–Francisco, 1979.

[GeC] N. Gershenfield, I. Chuang. *Bulk spin–resonance quantum computation.* Science 275 (1997), 350–355.

[Gri] D. Grigoriev. *Testing the shift–equivalence of polynomials using quantum mechanics.* In: Manin's Festschrift, Journ. of Math. Sci., 82:1 (1996), 3184–3193.

[Gro] L. K. Grover. *Quantum mechanics helps in searching for a needle in a haystack.* Phys. Rev. Lett. 79 (1997), 325–328.

[Ki1] A. Kitaev. *Quantum computations: algorithms and error correction.* Russian Math. Surveys, 52:6 (1997), 53–112.

[Ki2] A. Kitaev. *Classical and quantum computations.* Lecture notes, Independent University, Moscow, 1998.

[Ma1] Yu. Manin. *A Course in Mathematical Logic.* Springer Verlag, 1977, pp. xiii+286.

[Ma2] Yu. Manin. *Computable and uncomputable (in Russian).* Moscow, Sovetskoye Radio, 1980.

[Po] R. P. Poplavskii. *Thermodynamical models of information processing (in Russian).* Uspekhi Fizicheskikh Nauk, 115:3 (1975), 465–501.

[Sa] A. Salomaa. *Computation and Automata.* Cambridge UP, 1985.

[Sh] P. W. Shor. *Polynomial–time algorithms for prime factorization and discrete logarithms on a quantum computer.* SIAM J. Comput., 26:5 (1997), 1484–1509.

[Si] D. Simon. *On the power of quantum computation.* Proc. of the 35th Ann. Symp. on Foundations of Comp. Sci. (1994), 116–123.

[Ts] B. Tsirelson. *Quantum information processing.* Lecture notes, Tel–Aviv University, 1997.