Quantum Factoring and Search Algorithms

Quantum Information Seminar

Friday, Sep. 17, 1999

Ashok Muthukrishnan

Rochester Center for Quantum Information (RCQI)

---

§ 1. Quantum Algorithm for Factorisation

In 1994, at the 35th Annual Symposium on the Foundations of Computer Science in Los Alamitos, California, Peter Shor of Bell Laboratories announced his discovery of an algorithm for factorising numbers efficiently – one that would factorise a log N (= K) -digit number in O[poly(K)] steps, O[T(K)] meaning asymptotically less than T(K). However, the execution of the algorithm would require the coherent state evolution and measurement collapse of quantum mechanics. In classical complexity theory, the factorising problem is not at present known to be NP-hard, which means that an efficient algorithm for factorising does not imply the same for all problems in the NP class [see Gy79 for a discussion of complexity classes]. Nonetheless, Shor's algorithm has been hailed as a big acheivement for two reasons. One is that his finding is the first clear demonstration of a problem for which a quantum computer has a distinct advantage over a classical one. The second reason is the factorising problem itself, an efficient algorithm for which has rather disconcerting implications for present-day cryptography: the extreme difficulty of factorising large numbers (the best computer today would take about 428,000 years to factorise a 200-digit number) is precisely what secures the secrecy of most encoded messages transmitted today. This makes the feasibility of Shor's algorithm an urgent issue to secret government agencies and gives long-term impetus to the implementation of a quantum computer.

Shor showed that the factorising problem is polynomially equivalent to the problem of finding the period of a modular exponential function. It is the latter problem for which he presented a quantum algorithm. We start out by discussing the factorising problem and its link with cryptography in §1A. Shor's theorem relating factorising to period-finding is given without proof in §1B, while his period-finding quantum algorithm is generalised to apply to arbitrary functions in §1C. Finally in §1D, we discuss the efficiency of the period-finding algorithm.

1A. Factorising and cryptography

Before the advent of modern cryptography, secret messages were communicated using *one-time pads*, an example of which is

Actual word:	morning	evening	Monday	Tuesday	attack
Code word :	bat	glxt	king	button	figle

where there is a one-to-one correspondence between actual and code words which is known only to the sender and receiver of the secret message. The sending of the public message, "figle button bat" would be deciphered as "attack Tuesday morning," for example. It is immediately apparent that such a system of communicating can be secure only for a single use of a particular pad. In the trivial example above, a subsequent message like "figle button glxt," which deciphers to "attack Tuesday

evening," will be partially decipherable to the eavesdropper who knows that the first message was followed by an actual attack on Tuesday morning. Hence, the one-time pad is not secure for re-use. This means that the sender and the receiver would have to meet and share a new pad for every message, which is inefficient and unadvisable for security reasons. So it has long been apparent in cryptography that a durable system of secret communication has to have at least two properties,

1. Each person should have efficient, reusable algorithms for encoding and decoding messages.
2. Each person's decoding algorithm should *not* be obtainable from his or her encoding algorithm in any reasonable amount of time.

A system of cryptography that satisfies these two properties is known as a *public-key* system. The most commonly used public-key system today is the RSA cipher system [Rv78], which is based on the difficulty of factorising large integers. In 1977, R. Rivest, A. Shamir, and L. Addleman showed:

RSA Theorem: Let p, q be distinct primes, N = pq, and k = (p-1)(q-1). Let d be some number coprime with k. Let E be the inverse of d in Z_k (i.e. Ed = 1 mod k). Then, for any b < N, b^{Ed} = b mod N.

For the convenience of describing the use of this theorem in cryptography, we have used lower (upper) case variables for secret (public) information. Suppose Alice wants to send b < N = pq to Bob (where b is a string of digits that translates to a message), and Eve wants to eavesdrop on this message. In the RSA system, Alice and Bob arrange for a secret meeting at the start and agree on the primes p and q (and hence N and k) as well as d. In the transmission stage, Alice sends the coded message C = b^E mod N to Bob. Bob receives C and computes C^d = b^{Ed} = b mod N (using the RSA theorem in the last step), thus deciphering the message [see Hn90 for an example]. If Eve wants to find b from C, she needs d, for which she needs k, for which she needs p and q. So she is left with the task of factorising N, which is a formidable task (satisfying property 2 above), as we will see shortly. Note that the RSA system is re-usable, unlike a one-time pad – Alice could send as many as N messages before she has to meet Bob again to decide on a new p and q. Note also that for this system to be usable at all, Alice and Bob need to be able to generate large primes p and q and encode messages efficiently (property 1 above). Fortunately, the generation of primes is a much easier task than factorising and there are efficient algorithms for generating large primes [Pm81, Wg86]. Rabin's primality test [Rb80] indicates the primeness of a number with a fixed probability (0.75) that does not depend on the size of the number. Note that Fermat's Little Theorem, used in the proof of the RSA theorem [Hn90], says that $2^{p-1} \neq 1 \bmod p$ implies that p is not an odd prime. Although the converse statement is not true, the equality does imply a high chance that p is prime.

The RSA system has engendered interest in finding fast algorithms for factorising integers that are a product of two primes of approximately equal number of digits (which is the most difficult case to factorise). A naive approach to factorising N is to try dividing it by all the primes less than $\sqrt{N}$, which number $2N/\ln N$ approximately (by the prime number theorem). This is inefficient since

it would take O[exp(c K)] steps, where K = ln N and c is a constant. The same is true for all known deterministic factorising algorithms [Mc96]. The Quadratic Field Sieve (QFS) [Pm85, Sv87] and Number Field Sieve (NFS) [Le90] algorithms are more sophisticated probabilistic factorising algorithms, which are nonetheless inefficient, taking $O[\exp\{cK^{1/2}(\ln K)^{1/2}\}]$ and $O[\exp\{cK^{1/3}(\ln K)^{2/3}\}]$ steps respectively. In fact, there are no known classical algorithms at present that can factorise efficiently, even if they are allowed to be probabilistic. By contrast, Shor's probabilistic quantum algorithm [Sh94] factorises N in $O[K^3]$ steps, which is polynomial in the input size K = ln N.

In 1985, a Cray supercomputer took nearly 10 hours to factorise a 71-digit number into two primes using QFS techniques [Pm85]. This was followed in 1989 by a 100-digit number. In 1994, it took three months for computers running NFS methods to find the two factors of a 120-digit number [Dn94]. The present factorising record is held by the NFS methods for a 130-digit number [Cw96]. Work is in progress at present to factorise a 512-bit number (154 digits), which may cause the 512-bit RSA system to become obsolete in the near future [Ro95]. However, classical computing poses no serious threat to the RSA system, since the best classical factorising methods today would take approximately 428 millenia to factorise a 200-digit number (assuming GHz computing speed). For comparison, Shor's quantum algorithm would be able to factor the same number in a matter of days.

As noted earlier, quantum mechanics is indispensable in the execution of Shor's algorithm. A classical simulation of Shor's algorithm cannot hope to factorise even a 2-digit number in any reasonable amount of time [Ob97] – the exponential resources inherent in the superposition of quantum states allow for a parallelism that cannot be emulated classically. In addition to this parallelism, different computational paths in a quantum computer interfere as the system unitarily evolves to produce the final measurement probabilities. We will see that Shor's period-finding algorithm also requires a means for quantum entanglement [So35]. It has occured to some that classical wave optics – in which one also finds interference, superposition and unitary evolution – may offer a paradigm for imitating Shor's methods. A Young's N-slit interferometer [Cs96] and a Mach-Zender interferometer [Sm97] have both been shown capable of factorising integers. However, since classical wave optics lacks the essential feature of wavefunction collapse upon measurement, it is perhaps not surprising that we cannot perform efficient factorising in this context. Even if it is not practical, classical wave optics may offer some parallel intuition for the quantum processes that take place in the execution of Shor's period-finding algorithm and either lend evidence for the possibility of a classical factorising algorithm or else lend further support for the NP-hardness of factorsing. For example, the central step in Shor's period-finding algorithm is the Fourier-transform operation (Eq. 1.12 in §1C), which is also found in classical wave optics. However a quantum Fourier-transform would likely take place in an abstract Hilbert space rather than in real space and accounts for the impracticality of carrying out this operation classically [Be95].

1B. Relation of factorisation to period-finding

The first part of Shor's factorising algorithm is the demonstration of the polynomial equivalence of factorising to finding the period of a modular exponential function. In 1994, he proved

Shor's Theorem : Let N be an odd composite number whose prime factorisation contains two or more distinct primes. Let n be chosen randomly such that $1 \le n \le N$. When n and N are coprime, let r be the period of the modular exponential function

$$f_N(x; n) = n^x \bmod N \tag{1-1}$$

When r is even, and $n^{r/2} \neq \pm 1 \bmod N$, then

$$N = PQ \quad \text{where} \quad P, Q = \gcd(N, n^{r/2} \pm 1) \tag{1-2}$$

Furthermore, the following is true of the joint probability :

$$\text{Prob}\ [\{\gcd(n, N) = 1\} \cup \{r \text{ is even}\} \cup \{n^{r/2} \neq \pm 1 \bmod N\}] \ \ge\ (2 \log N)^{-1} \tag{1-3}$$

where gcd stands for the greatest common divisor, gcd(n, N) = 1 being equivalent to the statement that n and N are coprime. Note that the factors P and Q need not be prime, unless N has only two factors (as in the RSA system). Also note that the theorem fails for a prime-powered N (i.e. when P = Q) The proof of the theorem [given in Sh94, Ek96] relies on the Chinese Remainder Theorem [see Hn90], and we will not elaborate on it here. Shor's theorem relates the period r of the function f_N to the factors P and Q of N. However, this relation applies only when n (which is chosen randomly) happens to be coprime with N, the period r of the function f_N happens to be even, and $n^{r/2} \pm 1$ does not vanish in Z_N. The lower bound of $(2 \log N)^{-1}$ on the joint probability of these three events being in our favor ensures the (probabilistic) polynomial equivalence of finding the period of f_N and factorising N, provided that we can evaluate the gcd efficiently. In this regard, Euclid's algorithm, which determines the gcd of two numbers efficiently, has been known since antiquity.

We can show that the modular exponential function, $f_N = n^x \bmod N$, is 1-1 within its period – a result that will be useful in §1C where we generalise the period-finding algorithm to arbitrary functions. To see this, note that if $n^x = n^y \bmod N$ with $0 \le x < y < r$, then $s \equiv y - x < r$ and $n^{y+s} = n^y n^s = n^x n^s = n^{x+s} = n^y$ in Z_N, but this implies that s is a period of f_N, in contradiction with our understanding that r is the smallest such period. Hence, no x, y satisfies both $0 \le x < y < r$ and $n^x = n^y \bmod N$, which is to say that the function f_N is 1-1 within one of its periods. Now since f_N is a modular function, its values are limited to 0, ..., N-1. Furthermore, since it is 1-1 within a period, it follows that its period can also be no bigger than N. To summarise, we have shown

Lemma: $f_N(x; n) = n^x \bmod N$ is periodic with period $r \le N$, and it is 1-1 within a period. (1-4)

To illustrate this result, consider n=33 and N=40. Then, for x = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ..., we get f_N = 1, 33, 9, 17, 1, 33, 9, 17, 1, 33, ... , which has the period $r = 4 \le 40 = N$.

1C. Quantum algorithm for period-finding

After reducing the factorising problem to finding the period of the function $f_N(x; n)$ in (1-1), Shor provided an efficient algorithm for doing the latter that is based on quantum mechanics [Sh94]. The period-finding problem is stated as

PERIOD: Given a function,

$$f: \quad \{0, 1, \ldots, q-1\} \quad \rightarrow \quad \{0, 1, \ldots, p-1\} \tag{1-5}$$

find its period $r \in Z^+$, if this exists.

The domain of f needs to be large enough to contain several periods for any algorithm to meaningfully establish its periodicity. We will see that Shor's algorithm needs atleast r periods in the domain (i.e. $q \geq r^2$) to efficiently resolve the period r. In the case of f_N, recall that $r \leq N$, so that we would choose $q \geq N^2$ when applying Shor's algorithm to this function. In general, not knowing how large the period r can be, means that we start with an arbtrary domain size q and keep increasing this until the algorithm finds a period with sufficiently high confidence. Note that having a large enough domain size ensures a nontrivial solution to the period-finding problem but has no bearing on the efficiency of any algorithm for the problem. Once chosen, the domain size determines the input size for that instance and the input will consist of the values of f over that domain. The domain of (1-5) can be stored, for example, using $\log_2 q$ binary digits, or $\log_{10} q$ decimal digits. So an efficient algorithm that finds the period of f on this domain would have to perform in a time that scales polynomially in log q. Although Shor's algorithm was intended originally for the function f_N (for its use in factorising N), we describe it in this section for a generic function f of the form (1-5), which can model any integral or real-valued function with appropriate shifting and sampling. However, we will need to assume (at the end of this section and the next) that the function f is one-to-one within each of its periods (as f_N was shown to be in (1-4)) in order to estimate the probability of success for the algorithm. We will find this assumption sufficient but not necessary for the algorithm to work.

An exhaustive (classical) approach for finding the period of f is to systematically examine its values to see if they repeat. This approach is not efficient, however, since it needs to evaluate f at $q^{1/2} \geq r$ points in the domain. Another approach for finding the period is to find the spacing of the frequency components in the Fourier transform of f. Although the classical Fast Fourier Transform (FFT) is an efficient algorithm, it would take atleast $(q/r) \geq q^{1/2}$ queries classically, or $q^{1/4}$ queries quantum mechanically [see §2], to determine where the Fourier transform is nonzero. Shor proposed doing the Fourier transform itself quantum mechanically (following on earlier results of Simon [Si94]) and using the probabilistic nature of quantum measurement to pick out the frequency components. His algorithm is probabilistically efficient in that it outputs the period r in a time that scales polynomially in log q with an arbitrarily high success rate. The algorithm simulates the

dependency of the function f by means of quantum entanglement [So35], and the main part of the algorithm is the requirement for a quantum mechanical discrete-Fourier-transform (DFT). In this section, we will give a description of the algorithm with the necessary assumptions about the quantum system needed to implement it. We will return to the DFT used in the algorithm in §1D and describe one method of efficiently implementing it using Boolean algebra – a method that is a reworking of the classical FFT algorithm in a quantum setting.

Let the quantum computer (QC) implementing the algorithm be described completely by two observables X and Y that commute. Let the eigenvalues of X and Y be enumerated by x = 0, 1, ..., q-1 and y = 0, 1, ..., p −1 respectively. We anticipate using X to represent the arguments of the function f, and using Y as the dependent observable that stores the functional values, the dependency acheived by entangling X and Y. Since X and Y form a complete set, the state of the QC can be expanded in an orthonormal basis, $\{|x, y\rangle\}_{x,y}$, composed of eigenstates of X and Y. In the quantum circuit model of computation, X and Y are composed of a direct product of $\log_2 q$ and $\log_2 p$ qubits respectively, since it is in this framework that the proof of the efficient implementability of the algorithm can appeal to the rules of Boolean logic. However, the validity of the algorithm per se does not require X and Y to be made up of qubits, nor even that the basis state, $|x, y\rangle$, be a product state of X and Y (which is of relevance to the entanglement of X and Y). Hence, we are using a general, and necessary, formalism here to describe the algorithm. We assume that the QC is initially in the ground state of X and Y,

$$|G\rangle \equiv |0, 0\rangle \tag{1-6}$$

where we have formally taken x = 0 = y as the ground state. Since X and Y form a complete set for the QC, we can evaluate the probability-density for $|G\rangle$ in the x,y representation,

$$P_G(x, y) \equiv \left|\langle x, y|G\rangle\right|^2 = \left|\langle x, y|0, 0\rangle\right|^2 = \delta_{x0}\,\delta_{y0} \tag{1-7}$$

where δ_{ij} is the Kroneker Delta function. (1-7) is shown graphically in figure 1, where $P_G(x, y)$ is represented by the saturation of the plotted points, unshaded implying P = 0. The graph shows that the probability density for $|G\rangle$ is nonvanishing only at one point (x = 0 = y), as expected. The next step is to put the QC in a uniform superposition of X eigenstates,

$$|U\rangle \equiv q^{-1/2} \sum_{x=0}^{q-1} |x, 0\rangle \tag{1-8}$$

whose probability-density is

$$P_U(x, y) \equiv \left|\langle x, y|U\rangle\right|^2 = q^{-1} \left| \sum_{x'=0}^{q-1} \langle x, y|x', 0\rangle \right|^2 = q^{-1}\,\delta_{y0} \tag{1-9}$$

where we have used the orthonormality of $\{|x,y\rangle\}_{x,y}$. (1-9) is shown graphically in figure 2.

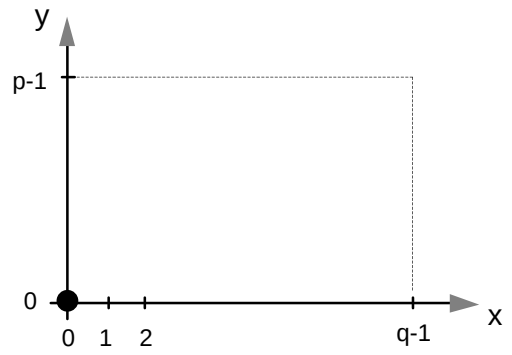

Figure 1: $P_G(x, y)$

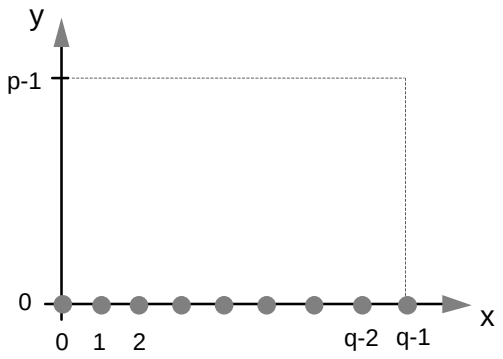

Figure 2: $P_U(x, y)$

The final stage in the preparation of the QC is to compute the function f using the X states as input and the Y states as output,

$$|F\rangle \equiv q^{-1/2} \sum_{x=0}^{q-1} |x, f(x)\rangle \tag{1-10}$$

We will see in §1D how $|F\rangle$ and $|U\rangle$ can be constructed efficiently within the framework of the qubit scheme for quantum computation. The probability density for $|F\rangle$ is

$$P_F(x, y) \equiv \left|\langle x, y|F\rangle\right|^2 = q^{-1} \left| \sum_{x'=0}^{q-1} \langle x, y|x', f(x')\rangle \right|^2 = q^{-1} \left|\langle x, y|\, x, f(x)\rangle\right|^2$$
$$= q^{-1}\, \delta_{y, f(x)} \tag{1-11}$$

using the orthonormality of $\{|x,y\rangle\}_{x,y}$. Figure 3 graphs (1-11) for a typical periodic function f.

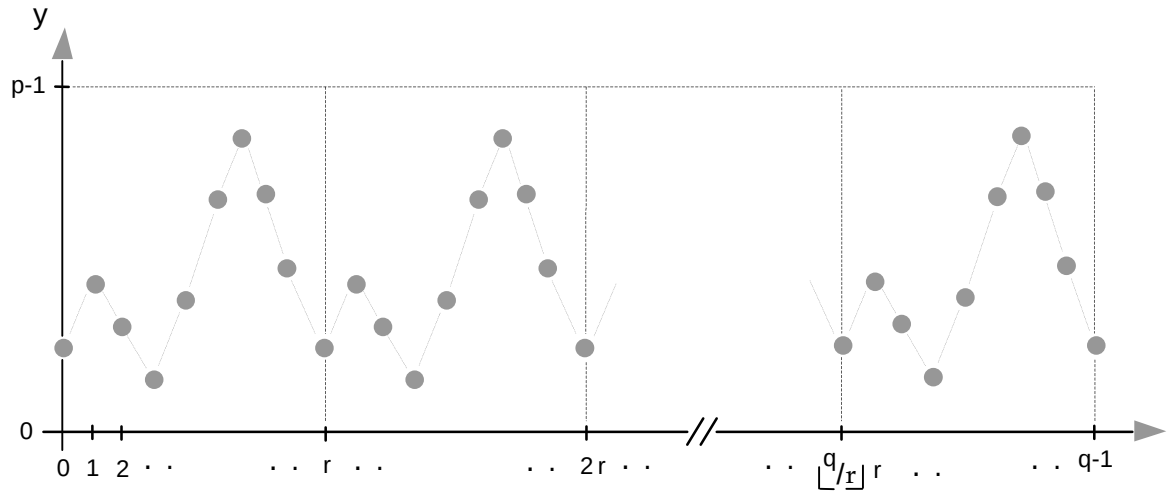

Figure 3: $P_F(x, y)$ for a typical periodic function f

The figure shows that $|F\rangle$ has a high degree of entanglement [So35], since a lack of entanglement corresponds to P(x, y) being a seperable function of x and y. The physical consequence of this entanglement is as follows. With the QC initially in the state $|F\rangle$, a measurement of X yield any of the values {0, ..., q-1} with uniform probability, and a measurement of Y would yield any of the values $\{f(x) \,|\, x = 0, ..., q-1\}$ with (roughly) uniform probability. However, if we first measure X in $|F\rangle$ and get x, the system has collapsed to $|x, f(x)\rangle$, and a subsequent measurement of Y would yield f(x) with probability 1. If, on the other hand, we first measure Y in $|F\rangle$ and get y, the system has collapsed to a state proportional to $\sum_{x_y} |x_y, y\rangle$, where x_y is any of at least $\lfloor q/r \rfloor$ solutions to $f(x_y) = y$ (on account of the periodicity of f), and a subsequent measurement of X will yield any of the x_y with (roughly) uniform probability. Thus, the entanglement of X and Y provides the means to represent the functional dependence of f quantum-mechanically.

Once we have the QC in the state $|F\rangle$, we are ready to perform the central step of the algorithm. We assume that the q-state unitary transform,

$$\mathrm{DFT}_q : \quad |x, y\rangle \quad \rightarrow \quad q^{-1/2} \sum_{k=0}^{q-1} e^{i 2\pi x k/q} \; |k, y\rangle \tag{1-12}$$

can be performed on the QC efficiently. DFT_q is unitary because the right side of (1-12) is also normalised, for any x and y. In §1D, we will discuss one way to realise DFT_q efficiently. Note that labeling (1-12) as a discrete-fourier-transform (DFT) is motivated by what it does to the probability-density of a state, such as $|F\rangle$, that is entangled in X and Y in such a way that Y can be construed as functionally dependent on X (this is known as maximal entanglement). Applying DFT_q to $|F\rangle$ gives

$$|F\rangle \quad \equiv \quad \mathrm{DFT}_q \, |F\rangle \quad = \quad q^{-1} \sum_{x=0}^{q-1} \sum_{k=0}^{q-1} e^{i 2\pi x k/q} \; |k, f(x)\rangle \tag{1-13}$$

which is normalised since $|F\rangle$ is normalised and DFT_q is unitary. The probability-density for $|F\rangle$ is

$$\begin{aligned} P_F(x, y) \quad &\equiv \quad \left| \langle x, y \,|\, F\rangle \right|^2 \quad = \quad q^{-2} \left| \sum_{x'=0}^{q-1} \sum_{k=0}^{q-1} e^{i 2\pi x' k/q} \; \langle x, y \,|\, k, f(x')\rangle \right|^2 \\ &= \quad q^{-2} \left| \sum_{x'=0}^{q-1} e^{i 2\pi x' x/q} \; \delta_{y, f(x')} \right|^2 \end{aligned} \tag{1-14}$$

using the orthonormality of $\{|x, y\rangle\}_{x, y}$ again. We confirm the normalisation of $P_F(x, y)$:

$$\sum_{x=0}^{q-1} \sum_{y=0}^{q-1} P_F(x, y) \quad = \quad q^{-2} \sum_{x'=0}^{q-1} \sum_{x''=0}^{q-1} \sum_{x=0}^{q-1} \sum_{y=0}^{q-1} e^{i 2\pi (x'-x'') x/q} \; \delta_{y, f(x')} \; \delta_{y, f(x'')}$$

$$= q^{-1} \sum_{x'=0}^{q-1} \sum_{x''=0}^{q-1} \delta_{f(x'),\, f(x'')} \sum_{x=0}^{q-1} q^{-1}\, e^{i\,2\pi\,(x'-x'')\,x/q} \sum_{y=0}^{q-1} \delta_{y,\, f(x')}$$

$$= q^{-1} \sum_{x'=0}^{q-1} \sum_{x''=0}^{q-1} \delta_{f(x'),\, f(x'')}\ (\delta_{x',\, x''})\ (1) \;=\; 1 \qquad (1\text{-}15)$$

It is not apparent whether the graph of (1-14) depends critically on the particular choice of the function f (which corresponds to a particular instance of the period-finding problem). To make this clear, we look at the x-depenendce of the probability-density for fixed values of y. First note that due to the delta function in (1-14), P_F (x, y) vanishes if y is not an image of f. On the other hand, when y is an image, there are at least $\lfloor q/r \rfloor$ pre-images x_y' in {0, 1, ..., q-1} on account of the periodicity of f in this domain. To be precise, let $\{x_y^{(n)} \,|\, n = 1, ..., s \le r\}$ be the solutions of $y = f(x')$ that belong to {0, 1, ..., r-1}, the first period of the domain. Then the solutions x_y' to $y = f(x')$ on the whole domain are

$$x_y' \;=\; \{\, x_y^{(n)} + m\,r \;\big|\; n = 1, \ldots, s \le r;\quad m = 0, \ldots, \lfloor q/r \rfloor - 1 \,\} \qquad (1\text{-}16)$$

excepting those (at most r) solutions that belong to a fractional period of f at the end of the domain (when $q/r \notin Z$). Thus (1-14) can be written as

$$P_F\,(x, y) \;=\; \begin{cases} 0 & ;\quad y \text{ is not an image of } f \\ q^{-2}\ |S(x, y)|^2 & ;\quad y \text{ is an image of } f \end{cases} \qquad (1\text{-}17)$$

$$S(x, y) \;=\; \sum_{x_y'} e^{i\,2\pi\,x_y' x/q} \;\cong\; \sum_{n=1}^{s} e^{i\,2\pi\,x_y^{(n)} x/q} \sum_{m=0}^{\lfloor q/r \rfloor - 1} e^{i\,2\pi\,m\,r\,x/q}$$

where the approximation is due to the exclusion of the "boundary" solutions in (1-16). These excluded terms will cause a round-off error in S(x, y) that will be of negligible consequence to the final result. Now the summation over m above is a finite geometric series. Doing this sum and taking the square modulus of S(x, y), we can rewrite (1-17) as

$$P_F\,(x, y) \;=\; \begin{cases} 0 & ;\quad y \text{ is not an image of } f \\ q^{-2}\ P_1(x)\ P_2(x, y) & ;\quad y \text{ is an image of } f \end{cases} \qquad (1\text{-}18)$$

$$P_1(x) \cong \operatorname{cosec}^2[\pi\,(r/q)\,x]\ \sin^2[\pi \lfloor q/r \rfloor\,(r/q)\,x]\,; \qquad P_2(x, y) = \Big|\, \sum_{n=1}^{s} e^{i\,2\pi\,x_y^{(n)} x/q} \,\Big|^2$$

where again the approximation is due to the round-off in (1-16). Equation (1-18) is the most general expression that we can give for the probability-density of the transformed state, $|F\rangle$, for an arbitrary function f. Notice that of the two functions, P_1 and P_2, only the latter depends on the function f by

way of the $x_y^{(n)}$, which we defined to be the roots of y = f(x) between 0 and r. Now P_2 is the square of a sum of atmost r terms of unit modulus (since $0 \leq x_y^{(n)} \leq r$) and we can argue that for any x, y, it is of order unity for most integral or (sampled) real-valued functions f. However, it is possible to construct counter-examples where P_2 vanishes for some (or all) y. To avoid these troublesome cases, we assume henceforth that the function f is one-to-one within each of its periods, which we saw to be the case (1-4) for the function of interest, f_N. This assumption means that given some y in the range, there can be at most one pre-image x under f among any r consecutive numbers in the domain. This means that y = f(x) has atmost one solution in {0, 1, ..., r}, or s=1 in (1-18). Hence $P_2 = 1$, and we are left with

$$P_F(x, y) = q^{-2} P_1(x) \cong q^{-2} \operatorname{cosec}^2[\pi (r/q) x] \sin^2[\pi \lfloor q/r \rfloor (r/q) x] \qquad (1\text{-}19)$$

when y is an image, and zero otherwise. Note that (1-18) says that the probability density is *independent* of the functional form of f (i.e. independent of y) as long as it is 1-1 within its period.

We can now see why the DFT operation (1-12) merits its name. We know that Fourier-transforming a periodic function f produces discrete frequency components having different amplitudes (that depend on the functional form of f within a period) and which are spaced apart by the inverse of the period. In a sense, P_2 in (1-18) renders the former property while P_1 renders the latter. Our assumption that f is 1-1 within a period suppresses the P_2 contribution, which is to say that it effectively removes the dependence of the Fourier transform on the particular functional form of f. Hence the probability density P_F in (1-19) captures the periodicity of f but is not sensitive to the behavior of f within a period. This means that the period-finding algorithm is not restricted to a particular function f that is periodic but is generally applicable to a large class of instances where the functions satisfy the 1-1 property, or can be made to do so when appropriately sampled. For the purpose of factorisation, the functions $f_N(x; n)$ were shown to be 1-1 within a period in (1-4).

If P_F in (1-19) is plotted over the continuum x = [0, q-1], the squared cosecant (which has asymptotes at x = m (q/r) for m = 0, 1, ..., r-1) will modulate the squared sine (which has zeros at x = n (q/r) $\lfloor q/r \rfloor^{-1}$ for n = 0, 1, ..., q-1), with the combined function having the limiting value $q^{-2} \lfloor q/r \rfloor^2$ at the asymptotes. However, we are only interested in the probability at integer arguments of x since the quantum states are discrete. There are q such discrete values of the probability P_F in the domain {0, 1, ..., q-1}, and their values depend on whether q happens to be a multiple of r or not (something we do not know apriori). When q is (not) a multiple of r, P_F is equal to (slightly less than) $q^{-2} \lfloor q/r \rfloor^2$ at x = m $\lfloor q/r \rfloor$ for m = 0, 1, ..., r-1. For all other integers x in the domain, P_F is equal to (slightly more than) zero. We graph (1-19) in figure 4a(b) for the case when q is (is not) a multiple of r. Recall that the value of P_F in the graphs is represented by the saturation of the plotted points. Note that there are r (or zero) values of y where P_F is nonzero for any x (r =7 is depicted in figures 4a,b).

This reflects the fact that there are exactly r distinct images of f in any given period if f is 1-1 in that period. In figure 4a, there are exactly r^2 grid-points where P_F is nonzero and has the uniform value $q^{-2}\lfloor q/r \rfloor^2$. Since figure 4a represents the case when q is a multiple of r, we have for this case,

$$\sum_{x=0}^{q-1} \sum_{x=0}^{p-1} P_F(x, y) = r^2 \ (q^{-2}\lfloor q/r \rfloor^2) = 1$$

as expected. This is likewise true for the case of figure 4b also, although in this case there are qr grid-points where P_F is nonzero (and non-uniform).

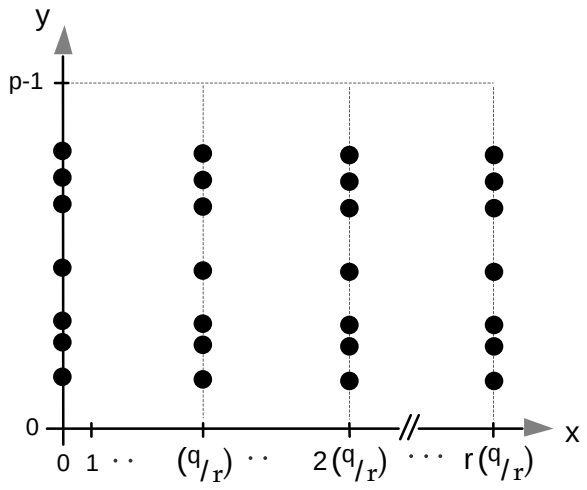

Fig 4 a -- $P_F(x,y)$ when q is a multiple of r

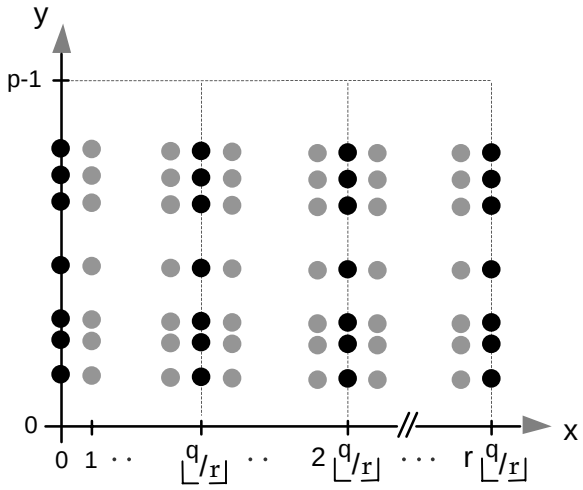

Fig 4b -- $P_F(x,y)$ when q is not a multiple of r

The final step of the period-finding algorithm is to measure the X observable of the QC when it has been brought to the state $|F\rangle$. From figures 4a,b, it is apparent that this measurement is most likely to yield one of the values, $x = \lfloor q/r \rfloor, 2\lfloor q/r \rfloor, 3\lfloor q/r \rfloor, \ldots, (r-1)\lfloor q/r \rfloor$, since the probability- density peaks at these values of x. Since the domain size q was chosen by us at the start of the algorithm, the final measured value for X allows us to infer the value of r, the period of the function. It is now apparent how the period-finding algorithm works in principle. Before considering issues of efficiency and implementation, we should emphasize why quantum mechanics is indispensable in the execution of the algorithm. As noted earlier, the Fourier transform can be efficiently carried out by classical means and does not by itself justify the use of quantum methods -- the constructive interference in the x-register to produce the peaks in figures 4a,b is in accordance with the Fourier analysis of periodic functions. The advantage in efficiency that the quantum implementation has is in the probabilistic measurement process. Rather than individual measurements made on each of the x-registers, a quantum measurement of the system observable X accesses the most likely values of X, which coincide with the desired solution values for the period-finding problem.

1D. Efficiency of the period-finding algorithm

There are three parts to demonstrating that the period-finding algorithm can be efficiently executed. First, we have to argue that the non-ideal (and more likely) final probability-density of figure 4b (where q is not a multiple of r) does not detract from the efficiency of the algorithm. Second, even supposing that only the desired values of X (i.e. multiples of $\lfloor q/r \rfloor$) are measured, the period r must be efficiently extractable from these measurements. The third issue is one already alluded to earlier -- that the preparation of states $|G\rangle$, $|U\rangle$, $|F\rangle$, and the DFT operation on the last of these have to be carried out efficiently. Recall that to be efficient, an algorithm has to perform in a time that scales polynomially in the input size of the problem which, in the case of period-finding, is the logarithm of the domain size q of the function. We will attend to each of the three issues briefly. First, in view of (1-15) and (1-19), the probability P of measuring X to be a multiple of $\lfloor q/r \rfloor$ is

$$P = \sum_{y} \sum_{m=0}^{r-1} P_F\left(m\lfloor q/r \rfloor, y\right) = r^2 \left(q^{-2} \lfloor q/r \rfloor^2 \right) = \lfloor q/r \rfloor^2 (q/r)^{-2}$$

Rewriting (q/r) as $\lfloor q/r \rfloor + h$, where $0 \le h \le 1$, we can write the denominator as

$$(q/r)^2 = \lfloor q/r \rfloor^2 + 2\, h \lfloor q/r \rfloor + h^2 \le \lfloor q/r \rfloor^2 + 3 \lfloor q/r \rfloor$$

using $h \le 1 \le \lfloor q/r \rfloor$. Also, since $r^2 \le q$ and $r \le \lfloor q/r \rfloor$, we have

$$P > \left[1 + {}^{3}/_{r} \right]^{-1} > \left[\log r \right]^{-1} > \left[\log q \right]^{-1} \qquad (1\text{-}20)$$

for sufficiently large r. Note that if $\lfloor q/r \rfloor = q/r$, we get $P = 1$ as expected for the case portrayed in figure 4a. Now (1-20) means that if the period-finding algorithm is iterated O(log q) times, we can expect about one such iteration to produce a multiple of $\lfloor q/r \rfloor$ for the measurement of X. We can make this statement more rigourous by using (1-20) to show that we can iterate the algorithm O(log q) times to achieve a probability P that is arbitrarily close to unity.

Measuring $x = m\lfloor q/r \rfloor$ for some integer m does not automatically tell us r, due to the presence of the $\lfloor\ \rfloor$ operation and due to our ignorance of the value of the integer m. Consider the simplified case of when q is a multiple of r. In this case, we can write $x/q = m/r$, and it is apparent that when r and m are coprime, r is the denominator of x/q when this fraction is reduced to simplest terms (i.e. when the numerator and denominator do not share any common factors). Since x is measured and q is known at the start, r can be determined in this way so long as it is coprime with m. For $m \le r$, the prime number theorem can be used to show [Ek96] that

$$[\text{Probability that } r \text{ is coprime with } m \le r] > \left[\log r \right]^{-1} > \left[\log q \right]^{-1} \qquad (1\text{-}21)$$

where we have again used the assumption that the domain size q is bigger than the square of the period r. So at least in the case when q is a multiple of r, (1-21) argues for the efficiency of extracting r from the measured values x. The case when q is not a multiple of r is a little more involved (it needs a theorem on continued fractions - see Ek96) but is based on the same kind of reasoning.

Results (1-20) and (1-21) argue for the efficiency of the period-finding algorithm in principle -- that the period of a function with domain size q can be found by iterating the steps of the algorithm O(log q) times. The implication for the efficiency of factorising follows immediately. When the function is $f_N = n^x \bmod N$, results (1-20) and (1-21), when used in conjunction with results (1-2), (1-3) and (1-4), show that N can be factored in O(log N) = O(log q) steps, having chosen $q \geq N^2$ ($\geq r^2$) at the start of the period-finding algorithm. The remaining issue is whether each iteration of the period-finding algorithm -- preparing the states and transforming them -- is performable in a time that scales as O(poly(log q)).

The efficient realisation of states $|G\rangle$, $|U\rangle$, $|F\rangle$ (Eqs. 1-6, 1-8, 1-10), and the DFT (1-12) is apparent in the paradigm of the quantum circuit model, in which arbitrary unitary transforms are built up from a set of two-qubit and one-qubit logic gates. In this framework, we represent the q-dimensional basis of X as an array of $\log_2 q$ qubits and the p-dimensional basis of Y as an array of $\log_2 p$ qubits. The state $|G\rangle$ corresponds to when all the qubits are in the ground state. When each qubit is rotated by $\pi/4$ (corresponding to a quarter of a Rabi oscillation if this is carried out optically), the state $|U\rangle$ results. The state $|F\rangle$ may be made by decomposing f into arithmetic operations to be carried out on the binary digits of X. In the case of $f_N = n^x \bmod N$, we need to raise n to the xth power mod N (recall that $x < q < N^2$). When x is a power of two for example, this function can be evaluated efficiently by repeatedly squaring the value in the x-register and storing the final result in the y-register. Note that these operations are carried out simultaneously on all elements of the superposition comprising the quantum computer. Finally, it can also be shown that the DFT operation can be performed efficiently on a quantum system. The classical Fast Fourier Transform algorithm [Ku81] can be expressed in terms of the one-bit and two-bit unitary transforms [Cp94],

$$A_j = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad B_{jk} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\pi 2^{j-k}} \end{bmatrix} \tag{1-22}$$

where A_j (known as the one-bit Walsh-Hadamaard transform) acts on the jth qubit, and B_{jk} acts on qubits j and k. Specifically, if $Q = \log_2 q$ is an integer, the following sequence of $Q(Q+1)/2$ elementary operations (read left to right)

$$(A_{Q-1})\,(B_{Q-2,Q-1}\,A_{Q-2})\,(B_{Q-3,Q-2}\,B_{Q-3,Q-1}\,A_{Q-3})\ \ldots\ (B_{0,1}\,B_{0,2}\ldots B_{0,Q-1}\,A_0) \tag{1-23}$$

accomplishes DFT_q on a Q-qubit system.

§ 2. Quantum Algorithm for Database Search

Isolating a desired item from an unsorted list of items is the goal of a quantum algorithm developed by Lov Grover of Bell Laboratories last year. His algorithm – described in §2B – accomplishes this task with more than 50% success in a time that scales as the square root of the number of list items, while any classical simulation has to examine at least half the items on the list for equivalent success. The quantum algorithm for database searching is less sophisticated than the factorising algorithm of §1, but it has given insights into the kind of problems that are amenable to speed-up by quantum computation. We describe database search and its uses in §2A. The universality of a database search is evident in the use of the search algorithm to solve a number of related problems, described in §2C.

2A. Database search and its uses

An unsorted database contains N items of which one item is sought. It is assumed that a single query – in unit time – can determine whether a given item matches the sought item. The input size of the problem is K = cN, where the constant c is determined by the size of the largest item on the list. Since the items are in random order, a classical algorithm would take at least N/2 queries to extract the sought item with a probability of success exceeding 0.5, and indeed we may expect any classical algorithm to take as much time, given that the list has no structure. Hence, Grover's quantum algorithm, which amounts to an unsorted database search in $O(\sqrt{N})$ time with the same success probability, is a result that cannot be emulated classically. Recall that the same cannot be said at present for Shor's algorithm -- that is, factorising may yet succumb to efficient classical methods.

Database search is nominally applicable to storage retrieval – that is, retrieving items from the computer memory – and this was the original motivation for Grover's result. However, we have to be clear about what storage retrieval entails for a quantum computer. On a classical computer, the sought item could be a file name, for example, and designate the memory location of the contents of the file. In this sense, item retrieval implies a determination of the file location in the memory. Storage retrieval in a quantum computer is meaningful in the context of entanglement [So35]. Consider the maximally entangled state, $\Sigma_j |j\rangle|\alpha_j\rangle$, regarded as a quantum memory of states α_j whose locations are marked by the enumerators j. A quantum search for the state $|i\rangle$ would leave the system in the state $|i\rangle|\alpha_i\rangle$, which corresponds to a retrieval of the information α_i .

Database search cannot also model the search for the solution of a general computation problem. Consider an instance of a search problem with input size K, and let the solution S_α for this instance belong to a set of N = exp(K) possible outcomes of a computation, S_1, S_2, ..., S_α, ..., S_N. In this context, the computation can be regarded as a nondeterministic simulation of the problem with

exponential time complexity. In the classical case, a search for S_α cannot be done in less than N steps, while a quantum search using Grover's algorithm can retrieve S_α in just $\sqrt{N}$ steps. Note that the quantum speed-up is only quadratic and the problem remains intractable, taking exp(K/2) steps. Also note that in order to apply database search to a quantum computation, the solution space needs to be entangled with another degree of freedom to distinguish S_α (which is not known prior to the search) from the non-solutions. For example, the state of the system after the simulation could be $\Sigma_j |S_j\rangle|0\rangle$, where $|0\rangle$ is an independent marker bit, and this system is then evolved, in accordance with the problem constraints, to the state $|S_\alpha\rangle|1\rangle + \Sigma_{j\neq\alpha}|S_j\rangle|0\rangle$, whence the solution S_α is marked for the database search query by the value 1 of the marker bit. The link between the search algorithm and algorithmic models for problems in general has spurred arguments on the ultimate limits of what a quantum computer can do, giving rise to the new field of quantum complexity theory.

A cryptographic application of database search is to the problem of password decryption: the standard 56-bit encryption scheme can be broken in 10^8 rather than $2^{56} \sim 10^{16}$ steps, for example. There are other problems of relevance to computer science and mathematics for which the quantum search algorithm has been useful, and we will give some examples in §2C. It has been shown that, just as O(N) is a lower bound on the time complexity of a classical search, $O(\sqrt{N})$ is a lower bound on the time complexity of a quantum search – that is, Grover's quantum algorithm searches optimally. To be precise, it was shown [By96] that

> Quantum search theorem: Let S be any set of N strings, and M be any oracle quantum machine with bounded error probability. Let $y \in S$ be a randomly and uniformly chosen element from S and let O be the oracle with $O(y) = 1$ and $O(x \neq y) = 0$. Then the expected number of times M must query O in order to determine y with probability at least 0.5 is at least $\lfloor \sqrt{N} \sin(\pi/8) \rfloor$.

An *oracle*, which is used in relativised complexity theory, can be regarded as a function that distinguishes the solution of a problem from all non-solutions, and does so in unit time.

2B. The quantum search algorithm

Although entanglement was seen to be pervasive in the application of database search, it is not a part of the quantum search routine itself. Grover showed [Gr97a] that given a uniform superposition of N quantum states, his algorithm can be implemented to put more than half of the population of the system into a chosen state using $O(\sqrt{N})$ elementary operations. An elementary operation in quantum computing can be defined as a unitary transformation that either involves at most two bits, or that takes place in a subspace of the N-dimensional Hilbert space whose dimensionality is independent of N. The elementary operations needed in this algorithm are the one-bit *Walsh-Hadamaard* transform $\hat{A}_j$ acting on the jth bit,

$$\begin{matrix} & |0\rangle_j & |1\rangle_j \\ \hat{A}_j = \frac{1}{\sqrt{2}} & \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \end{matrix} \tag{2-1}$$

which was introduced in (1-22) in the context of the discrete fourier transform, and the one-dimensional *sign-flip* operation $\hat{S}_n$,

$$\hat{S}_n |m\rangle = \begin{cases} -|m\rangle & ; \quad m = n \\ |m\rangle & ; \quad m \neq n \end{cases} \tag{2-2}$$

which affects only the nth dimension of the computational Hilbert space. The general Walsh Hadamaard transform $\hat{W}$ acting within the N-dimensional space spanned by $J = \log_2 N$ bits is defined

$$\hat{W} |m\rangle = N^{-1/2} \sum_{n=0}^{N-1} (-1)^{\vec{n} \bullet \vec{m}} |n\rangle \tag{2-3}$$

where $\vec{n}, \vec{m}$ are the binary forms of n, m and $\vec{n} \bullet \vec{m}$ is their bitwise dot product. $\hat{W}$ can be constructed using J one-bit Walsh Hadamaard transforms acting on all the bits,

$$\hat{W} = \hat{A}_1 \hat{A}_2 \ldots \hat{A}_J \tag{2-4}$$

The Walsh Hadamaard transform helps construct the main transform used in Grover' s algorithm,

$$\hat{D} |m\rangle = -|m\rangle + \frac{2}{N} \sum_{n=0}^{N-1} |n\rangle \tag{2-5}$$

which is called a *diffusion* transform and which accomplishes an "inversion-about-average" on the state amplitudes since

$$\hat{D} \sum_{n=0}^{N-1} c_n |n\rangle = \sum_{n=0}^{N-1} \left(-c_n + \frac{2}{N} \sum_{n'=0}^{N-1} c_{n'} \right) |n\rangle = \sum_{n=0}^{N-1} c'_n |n\rangle \tag{2-6}$$

where $c'_n = -c_n + 2\langle c \rangle = \langle c \rangle + [\langle c \rangle - c_n]$. Moreover, $\hat{D}$ is unitary,

$$\Sigma_n c'_n c'^*_n = \Sigma_n c_n c_n^* = 1 \tag{2-7}$$

and implementable in $O(\log_2 N)$ steps since it is easily shown [Gr97a] that,

$$\hat{D} = \hat{W}(-\hat{S}_0)\hat{W} \tag{2-8}$$

If all the state amplitudes are equal to the average – as is the case for a uniform superposition – the diffusion transform leaves the system unchanged. Otherwise, it inverts the real (and imaginary) parts of each state amplitude about the average in the sense described above. Although the state amplitudes

are generally complex, they can be considered real for the purpose of the algorithm – since the transforms are real – provided the intial state amplitudes are real.

Each iteration of the algorithm consists of a sign flip of the target state, followed by a diffusion. This effectively transfers (or "diffuses") population from all the states to the target state. Starting with a uniform superposition of the N states, O($\sqrt{N}$) iterations of the sign-flip and diffusion leaves the system with more than half of the population in the target state, thus favouring it for measurement. With a probability greater than 0.5, the measurement projects the system onto the target state, completing a successful quantum search. To be precise, Grover showed

Grover's theorem: (2-9)

If $|\psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} |n\rangle$, and $|\psi_M\rangle = [\hat{D}\ \hat{S}_\alpha]^M |\psi_0\rangle$, then $\exists\ M < \sqrt{N}$ such that $|\langle\alpha|\psi_M\rangle|^2 > 0.5$.

The target state in the theorem is labeled $|\alpha\rangle$ and the the result of $\sqrt{N}$ iterations of $\hat{D}$ and $\hat{S}_\alpha$ puts more than half of the population of the system in this state. Recall that $\hat{D}$ and $\hat{S}_\alpha$ are implementable in $\log N$ steps (eqs 2-1, 2-2, 2-4, 2-8). It is relatively straightforward to prove the theorem in the large-N approximation. We can write the state of the system after m iterations as

$$|\psi_m\rangle = \Sigma_n\, c_n(m)\, |n\rangle \qquad (2\text{-}10)$$

and the sign-flip transform operating on this state (in the m+1th iteration) produces

$$\hat{S}_\alpha\ |\psi_m\rangle = \Sigma_n\, \{c_n(m) - 2\,\delta_{n\alpha}\, c_n(m)\}\ |n\rangle \qquad (2\text{-}11)$$

with a diffusion transform completing the m+1th iteration,

$$|\psi_{m+1}\rangle = \hat{D}\ \hat{S}_\alpha\ |\psi_m\rangle = \Sigma_n \{c_n(m) - 2\,\delta_{n\alpha}\, c_n(m)\}\ \{-|n\rangle + (2/N) \Sigma_{n'} |n'\rangle\} \qquad (2\text{-}12)$$

Expanding (2-12) and projecting onto the target state gives a recursive relation for the target state amplitude,

$$\langle\alpha|\psi_{m+1}\rangle = c_\alpha(m+1) = (1-\tfrac{4}{N})\, c_\alpha(m) + 2\,\langle c\rangle \cong c_\alpha(m) + 2\,\langle c(m)\rangle \qquad (2\text{-}13)$$

where we have dropped the 1/N factor assuming that N » 1. The average of all the state amplitudes is intially $N^{-1/2}$ for the uniform superposition state. While $\hat{D}$ leaves the average unchanged (2-6), $\hat{S}_\alpha$ changes it by $2\,N^{-1}\, c_\alpha(m)$ every iteration, or roughly by m/N for m iterations. As long as the number of iterations is much less than $\sqrt{N}$, this average remains $N^{-1/2}$ to a good approximation, and

$$c_\alpha(m+1) \cong c_\alpha(m) + \frac{2}{\sqrt{N}} \tag{2-14}$$

or converting this to a non-recursive form,

$$c_\alpha(M) \cong \frac{2M+1}{\sqrt{N}} \tag{2-15}$$

from which it is apparent that $M > \sqrt{N/8} - 1/2$ is sufficient for $|c_\alpha(M)|^2 > 0.5$. The exact calculation [By96] yields

$$c_\alpha(M) = \sin\left[(2M+1)\ \sin^{-1}(N^{-1/2})\right] \tag{2-16}$$

which reduces to (2-15) under the approximations made. (2-16) shows that the final target state probability is a nonlinear function of M, For example, a doubling of the number of iterations does not result in a doubling of the success probability. Indeed, the final target state probability is sinusoidal in M, with zero probability recuring for arbitrarily high M.

The implementation of the quantum search algorithm presented above is relatively easier than that of the factorising algorithm of §1. A proof-of-principle demonstration of quantum search would require the sign-flip (2-2) and diffusion (2-5) transforms acting within an N-dimensional Hilbert space. A generalisation of Grover's search algorithm that allows for an arbitrary initial state amplitude distribution [Bi98] shows that the final target state probability satisfies the problem constraints while being sensitive only to the standard deviation of the initial state amplitude distribution. This result relaxes the constraints placed on the preparation of the intial state for the algorithm.

2C. Related problems and algorithms

The quantum search algorithm is not classically intuitive since it seems to accomplish a search of randomly placed items without examining every item. Its success seems to depend both on the diffusion of sign-dependent state amplitudes as well as on the interpretation of the square modulus of these amplitudes as the measurement probability. Indeed, the square-root time complexity is a direct mathematical consequence of the state amplitude being the square-root of the probability. Since classical probabilities are positive definite, this rules out such a search in this regime. The quantum search algorithm also highlights the role played by the Walsh Hadamaard (WH) transform – vis a vis the diffusion transform – and it is instructive to ask how sensitive the search is to the special properties of this transform. In this regard, it was shown that the quadratic speed-up of the search is not restricted to the use of the WH transform. For a general unitary transform U, the square of the matrix element, $|U_{fi}(t)|^2$, gives the probability of reaching state f at time t having started out in state i

initially. It was shown [Gr97e], using methods similar to that used in the search algorithm, that it takes only $O(|U_{fi}|)$ iterations of U – rather than the $O(|U_{fi}|^{-2})$ that would be expected – to place more than half the population of the system in state f, having started out in state i. An application of this to the case when i denotes a uniform superposition of states and f denotes one of the eigenstates shows that U need not be the WH transform for a quantum search. However, this is at the cost of requiring a sign-flip of the f amplitude in a basis that includes the uniform superposition state i.

Another issue to consider in regard to a database search is whether the search methods can be extended to the case when the sought item appears more than once in the database. It has been shown [By96] that the quantum search algorithm can be extended to search an N-item database for t marked items using $(N/t)^{1/2}$ queries, and that this can be done even if t is unknown. Incidentally, the problem of determining t – the problem of quantum counting – has yet to be addressed successfully. The issue of searching a database which has been sorted already – envisioned as an oracle which can give a one- bit answer to any query, however complicated – has been studied [Te97] in the quantum regime. Classically, the number of queries needed to do such a binary search is not less than log N for an N-item database – for example, each query could ask whether the target item is located before or after a randomly chosen item, and the answer eliminates one of these two segments of the database in each iteration of the search. A quantum algorithm for this problem has been shown possible [Gr97d] with only *one* query to the database, although the preparing and processing of this query is space consuming. Nonetheless, this result is grossly in violation of the classical information-theoretic lower bound on the number of queries [Te97], namely log N.

While the efficiency of the quantum search is quantified by counting the number of elementary steps – conceived of as gate operations or low-dimensional transforms – the overall quantum evolution from the intial superposition to the final target state is a continuous process unbroken by intermediate measurements. It has been shown [Fa97] that this efficiency can also be quantified without partitioning the algorithm into iterations, and that the time taken to evolve into the target state scales as $\sqrt{N}$ for an N-state system.

The applicability of the search paradigm to any computational problem is borne out by the use of methods similar to those in the quantum search algorithm to solve related problems in computer science and mathematics. It has been shown [Gr96] that the median of N numbers can be isolated with a precision $\varepsilon > 0$ – that is, the numbers less than or greater than the answer should be bounded by $(1-\varepsilon)$ (N/2) with large probability – in $O(\varepsilon^{-1})$ queries using quantum methods, rather than the $O(\varepsilon^{-2})$ needed by classical methods. A similar result has been established for the mean of N numbers [Gr97b,c]. A more direct parallel of the quantum search algorithm is the determination of the minimum value of a set of N numbers in $\sqrt{N}$ time [Dr96], as opposed to the N time that is classically feasible.

It was noted that Shor's algorithm is of interest to cryptography because it deals with the factorising problem. Another problem of cryptographic interest is the *collision* problem. A collision of a function $F: X \mapsto Y$ consists of two distinct domain values that have the same image under F, and the problem consists of determining such collisions, given the images of F. Some of the cryptographic protocols for encryption and decryption of information depend for their security on the collisions of certain so-called hash functions remaining unknown to eavesdroppers. It was shown [Br97], using a direct application of the quantum search algorithm, that a collision for an r-to-one function – where each image has r distinct pre-images – can be determined in $(N/r)^{1/3}$ queries rather than the $(N/r)^{1/2}$ queries needed classically. A space-time trade-off, mentioned earlier in the context of the search algorithm, can also be used to further reduce the number of queries in the collision problem.

References

[Be95] C.H. Bennett, "Quantum Information and Computation," *Physics Today*, October 1995, 24-30.

[Bi98] D. Biron, O.Biham, E. Biham, M. Grassi, D.A. Lidar, "Generalized Grover Search Algorithm for Arbitrary Initial Amplitude Distribution," http: //xxx.lanl.gov/abs/quant-ph/9801066.

[Br97] G. Brassard, P. Hoyer, A. Tapp, "Quantum Algorithm for the Collision Problem," http: //xxx.lanl.gov/abs/quant-ph/9705002.

[By96] M. Boyer, G. Brassard, P.Hoyer, A. Tapp, "Tight bounds on quantum searching," http: //xxx.lanl.gov/abs/quant-ph/9605034.

[Cp94] D. Coppersmith, "An Approximate Fourier Transform useful in Quantum Factoring," *IBM Research Report No. RC 19642* (1994), T.J. Watson Research Center, Yorktown Heights, N.Y., U.S.A.

[Cs96] J.F. Clauser, J.P. Dowling, "Factoring integers with Yound's N-slit interferometer," *Phys. Rev. A*, 53 (1996), 4587-90.

[Cw96] J. Cowie, B. Dodson, R.M. Elkenbracht-Huizing, A.K. Lenstra, P.L. Montgomery, J. Zayer, "A Worldwide number field sieve factoring record: on to 512 bits," *Advances in Cryptology - ASIACRYPT 1996 Int'l Conference on Theory and App. of Cryptology and Info. Security*, 382-94.

[Dn94] T. Denny, B. Dodson, A.K. Lenstra, M.S. Manasse, "On the factorisation of RSA-120," *Advances in Cryptology - CRYPTO 1993 - 13th Annual Int'l Cryptology Conference Proceedings*, 166-74.

[Dr96] C. Durr, P. Hoyer, "A Quantum Algorithm for Finding the Minimum," http: //xxx.lanl.gov/abs/quant-ph/907014.

[Ek96] A. Ekert, "Quantum Computation and Shor's Factorising Algorithm," *Rev. Mod. Phys.*, 68 (1996), 733-53.

[Fa97] E. Farhi, S. Gutmann, "Analog analoge of digital quantum computation," http: //xxx.lanl.gov/ abs/quant-ph/9612026.

[Gr96] L.K. Grover, "A fast quantum mechanical algorithm for estimating the median," http: //xxx.lanl.gov/ abs/quant-ph/9607024.

[Gr97a] L.K. Grover, "Quantum Mechanics helps in Searching for a Needle in a Haystack," *Phys. Rev. Lett.*, 79 (1997), 325-8.

[Gr97b] L.K. Grover, "Quantum Telecomputation," http: //xxx.lanl.gov/abs/quant-ph/9704012.

[Gr97c] L.K. Grover, "A framework for fast quantum mechanical algorithms," http: //xxx.lanl.gov/abs/quant-ph/9711043.

[Gr97d] L.K. Grover, "Quantum Computers Can Search Arbitrarily Large Databases by a Single Query," *Phys. Rev. Lett.*, 79 (1997), 4708-12.

[Gr97e] L.K. Grover, "Quantum computers can search rapidly by using almost any transformations," http: //xxx.lanl.gov/ abs/quant-ph/9712011.

[Gy79] M.R. Garey, D.S. Johnson, *Computers and Intractability -- A Guide to the Theory of NP-Completeness*, W.H.Freeman and Company, New York, 1979.

[Hn90] T.W. Hungerford, *Abstract Algebra - An Introduction* (Ch. 1, 12, 13), Saunders College Publishing (Holt, Rinehart and Winston, Inc.), Philadelphia, 1990.

[Ku81] D. E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, Addison-Wesley, 1981.

[Le90] A.K. Lenstra, H.W. Lenstra Jr., M.S. Manasse, J.M. Pollard, "The number field sieve," *Proc. 22nd Annual ACM Symp. on Theory of Computing* (1990), 564-72.

[Ob97] K.M. Obenlan, A.M. Despain, "Models to reduce the complexity of simulating a quantum computing," http: //xxx.lanl.gov/abs/quant-ph/9712004.

[Pm81] C. Pomerance, "The Search for Prime Numbers," *Scientific American*, 247 (6), Dec. 1982, 136-47.

[Pm85] C. Pomerance, "The quadratic sieve factoring algorithm," *Advances in Cryptology: Proc. EuroCrypt 1984 -- A Workshop on the Theory and Application of Cryptographic Techniques*, 169-82.

[Rb80] M.O. Rabin, "Probabilistic algorithm for testing primality," *J. Num. Theory*, 12 (1980), 128-38.

[Ro95] M.J.B. Robshaw, "Security estimates for 512-bit RSA," *1995 WESCON Conference*, Cat No. 95CH35791), 409-12.

[Rv78] R.L. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the A.C.M.*, 21 (1978), 120-6.

[Sh94] P. Shor, "Algorithms for quantum computation: discrete log and factoring," *Proc. 35th Annual Symp. on Found. of Computer Science* (1994), IEEE Computer Society, Los Alamitos, 124-34.

[Si94] D. Simon, "On the power of quantum computation," *Proc. 35th Annual IEEE Symp. Found. of Computer Science*, 1994, 116-123.

[Sm97] J. Summhammer, "Factoring and Fourier transformation with a Mach-Zender interferometer, http: //xxx.lanl.gov/abs/quant-ph/9708047.

[So35] Entanglement is a correlation between quantum observables that has no classical analog. E. Shrödinger called it "*the* characteristic trait of quantum mechanics" [*Proc. Cambridge Philo. Soc.* 31 (1935), 555]. An entangled pure state for two observables A and B is $|\psi\rangle = \Sigma_{a,b}\, c\,(a,b)\,|a\rangle_A |b\rangle_B$ where c(a,b) is not separable in a and b. For a physically motivated measure of the degree of entanglement of a quantum system, see Be96, Po96 and Wo97.

[Sv87] R.D. Silverman, "The multiple polynomial quadratic sieve," Math. Comp., 48 (1987), 329-39.

[Te97] B.M. Terhal, J.A. Smolin, "Superfast quantum algorithms for coin weighing and binary search problems," http: //xxx.lanl.gov/abs/quant-ph/9705041.

[Wg86] S. Wagon, "Primality Testing," *The Mathematical Intelligencer*, 8 (3) (1986), 58-61.