Rapid solution of problems by quantum computation

By David Deutsch[1] and Richard Jozsa[2]†

[1] *Wolfson College, Oxford OX2 6UD, U.K.*

[2] *St Edmund Hall, Oxford OX1 4AR, U.K.*

A class of problems is described which can be solved more efficiently by quantum computation than by any classical or stochastic method. The quantum computation solves the problem with certainty in exponentially less time than any classical deterministic computation.

The operation of any computing machine is necessarily a physical process. Nevertheless, the standard mathematical theory which is used to study the possibilities and limitations of computing (e.g. based on Turing machines) disallows quantum mechanical effects, in particular the presence of coherent superpositions during the computational evolution. A suitable notion of a *quantum computer*, which, like the Turing machine, is idealized as functioning faultlessly and having an unlimited memory capacity, but which is able to exploit quantum effects in a programmable way, has been formulated by one of us (Deutsch 1985). Quantum computers cannot compute any function which is not turing-computable, but they do provide new modes of computation for many classes of problem. In this paper we demonstrate the importance of quantum processes for issues in computational complexity. We describe a problem which can be solved more efficiently by a quantum computer than by any classical computer. The quantum computer solves the problem with certainty in exponentially less time than any classical deterministic computer, and in somewhat less time than the expected time of any classical stochastic computer.

Let U_f be a device that computes a function $f: \boldsymbol{Z}_m \rightarrow \boldsymbol{Z}_n$. Given an input i, U_f will, after some time, output the value of $f(i)$. In general terms the class of computational task which we shall be considering involves being given U_f and then using it to determine some property $G[f]$ (that is, some function G of the sequence $f(0), f(1), \ldots, f(m-1)$) in the least possible time.

In the analysis of this type of task, it is often an excellent approximation that the internal workings of U_f are inaccessible, in which case U_f is known as an *oracle* for f. The approximation would be nearly exact if U_f were a new type of physical object with an unknown law of motion.

If U_f were simply a program for evaluating f on our computer, making the approximation is tantamount to assuming that there is no faster method of obtaining

† Present address: Département I.R.O., Université de Montréal, C.P. 6128 Succursale A, Montréal, Canada H3C 3J7.

Proc. R. Soc. Lond. A (1992) **439**, 553–558
Printed in Great Britain

$G[f]$ from the program U_f (e.g. by a textual analysis) than actually executing U_f to obtain sufficiently many values $f(i)$ to determine $G[f]$. It seems obvious that this is true for all properties G – obvious, but like $P \neq NP$, hard to prove.

If U_f were a ROM (read-only memory) containing a sequence of m integers from $\boldsymbol{Z}_n$, the approximation is that there is no faster way of obtaining $G[f]$ from U_f than reading from the ROM sufficiently many values $f(i)$ to determine $G[f]$. This is clearly not true in general – there could be physical ways of measuring $G[f]$ directly, like measuring the total spin if the values of values $f(i)$ were stored as individual spin values – but it is a good description in many realistic situations.

It is useful to classify computational tasks into evaluations of *functions* and solutions of *problems*. In the case of functions, the task is to obtain the unique output that is the specified function of the input. For example, U_f, as we have defined it, evaluates the function f. In the case of solving problems the task is to obtain any one output that has a specified property. For example, to find a factor of a given composite number is a problem. Finding the least prime factor is a function evaluation.

When a classical deterministic (Turing) computer solves a problem, it always does so by evaluating a function. For example, a factorization program will always find the same factor of a given input. Which factor it finds could be specified by an additional constraint, narrowing the task to a function evaluation. Therefore when solving problems a classical computer cannot help performing a harder computational task than the one it was set.

A stochastic computer (i.e. one containing a hardware random number generator) need not always evaluate functions because the course of its computation, and therefore its output, need not be uniquely determined by the input. However, this gives a stochastic computer little advantage over a Turing one in solving problems, for if every possible output of a stochastic computation has the specified property that solves the problem, what is the purpose of choosing numbers randomly in the course of the computation? One reason might be that there is a deterministic algorithm for solving the problem, which takes a parameter, and the running time depends on that parameter. If most values of the parameter give a short running time, but there are exceptional ones, which cannot easily be predicted, which give a long running time, it might be desirable to choose the parameter randomly if one wanted to reduce the expectation value of the running time.

A quantum computer (Deutsch 1985) is one in which quantum-mechanical interference can be harnessed to perform computations. Such a computation also need not necessarily evaluate functions when it is solving problems, because the state of its output might be a coherent superposition of states corresponding to different answers, each of which solves the problem. This allows quantum computers to solve problems by methods which are not available to any classical device.

Let us assume that, however U_f works, its operation is a coherent quantum-mechanical process. Of course all physical processes conform to this assumption at some sufficiently complete level of description, possibly including their environment. But we mean that U_f can conveniently be made part of the coherent computation of a quantum computer.

Let $\mathscr{H}_{mn}$ be a Hilbert space of dimension mn and let

$$\{|i,j\rangle\}\,(i \in \boldsymbol{Z}_m, j \in \boldsymbol{Z}_n) \tag{1}$$

be a fixed orthonormal basis in $\mathscr{H}_{mn}$. Suppose that U_f operates by accepting input

in any state $|k,0\rangle$ of the basis, representing the value k, and converting it to output in the state $|k,f(k)\rangle$, from which the value $f(k)$ can be read off with probability 1. More generally, we may suppose that U_f effects the unitary evolution

$$|i,j\rangle \overset{U_f}{\rightarrow} |i,j+f(i)\rangle, \tag{2}$$

where the addition in the expression $j+f(i)$ is performed modulo n. Then, by the linearity of quantum evolution, U_f will evolve the input state

$$m^{-\frac{1}{2}}(|0,0\rangle+\ldots+|m-1,0\rangle) \tag{3}$$

to the output state

$$m^{-\frac{1}{2}}(|0,f(0)\rangle+\ldots+|m-1,f(m-1)\rangle). \tag{4}$$

Thus, by running U_f only once, we have in some sense computed all m values of f, in superposition. Elementary quantum measurement theory shows that no quantum measurement applied to the system in the state (4) can be used to obtain more than one of the m values $f(0),\ldots,f(m-1)$. However, it is possible to extract some joint properties $G[f(0),\ldots,f(m-1)]$ of the m values, by measuring certain observables which are not diagonal in the basis (1). This is called the method of *computation by quantum parallelism* and is possible only with computers whose computations are coherent quantum processes. For examples see Deutsch (1985) and Jozsa (1991).

To date, all known computational tasks which can be performed more efficiently by quantum parallelism than by any classical method have the following two properties. Firstly, the answer is not obtained with certainty in a given time; that is, there is a certain probability that the program will report that it has failed, destroying the information about f, so that in general it has to be run repeatedly before the answer is obtained. Secondly, although on some occasions it runs faster than any classical algorithm, the quantum algorithm is on average no more efficient than a classical one. It can be shown (Deutsch 1985) that the second property must hold for at least one choice of input in the quantum computation of any function.

It is the purpose of this communication to describe a problem for which quantum parallelism gives a solution with certainty in a given time, and is absolutely more efficient than any classical or stochastic method.

The problem is as follows: Given a natural number N and an oracle U_f for a function f: $\boldsymbol{Z}_{2N}\rightarrow\boldsymbol{Z}_2$, find a true statement in the list:

(A) f is not a constant function (at 0 or 1);

(B) the sequence $f(0),\ldots,f(2N-1)$ of values of f does not contain exactly N zeros.

Note that for any f, at least one of (A) or (B) is always true. It may be that both are true, in which case either (A) or (B) is an acceptable solution. That is why the solution of this problem is not necessarily tantamount to the computation of a function. A stochastic or quantum algorithm for solving it may have the property that when (A) and (B) are both true, it returns either answer, randomly. But when only one of them is true, the algorithm must return that one with certainty.

Consider first the classical solution. We repeatedly run U_f to calculate values of f in some order, say $f(\Pi(0)), f(\Pi(1)), f(\Pi(2)),\ldots$, where Π is a permutation on $\boldsymbol{Z}_{2N}$. We continue until we have enough information to prove that (A) or (B) is true. This is always achieved in at most $N+1$ invocations of U_f, though many functions f will require fewer invocations. Representing a function f by the $2N$-sequence $f(\Pi(0)),\ldots,$ $f(\Pi(2N-1))$ of zeros and ones, we have the results of table 1.

Hence, given a large number of random fs, the average number of invocations of U_f required to solve the problem for each f is

$$\frac{N+1}{2^{N-1}} + \sum_{n=2}^{N} n(\tfrac{1}{2})^{n-1} = 3 - \frac{1}{2^{N-1}}, \tag{5}$$

i.e. approximately three invocations for large N. If we are exceptionally unlucky, or if the fs are not presented randomly, but perversely by someone who knows what algorithm we are going to use, we shall require $N+1$ invocations. With a classical stochastic computer we can choose the permutation Π randomly, a process which requires $O(\ln(N))$ steps on average, and can thereby expect to solve the problem in approximately three invocations. though again in unlucky cases this may rise to $N+1$ invocations, plus an overhead of $O(N\ln(N))$ steps.

Now we present a method of solution using quantum parallelism. Let S be the unitary operation defined by

$$S|i,j\rangle = (-1)^j|i,j\rangle. \tag{6}$$

This operation can be performed by a quantum computer (cf. Deutsch 1985) in a fixed number of steps, independent of N and f. The state

$$|\phi\rangle = \frac{1}{\sqrt{(2N)}} \sum_{i=0}^{2N-1} |i,0\rangle \tag{7}$$

can be prepared, starting with the 'blank' input $|0,0\rangle$, in $O(\ln(N))$ steps, independently of f. For example, if $2N$ is a power of two, this could be done by applying the elementary one-bit transformation

$$|x\rangle \to \tfrac{1}{\sqrt{2}}(|x\rangle + (-1)^x|1-x\rangle) \quad (x \in \boldsymbol{Z}_2) \tag{8}$$

successively to each of the $\log_2(2N)$ bits that hold the value i in (7).

Given a quantum oracle U_f, apply the three operations U_f, S, U_f successively to the memory locations prepared in the state $|\phi\rangle$. Then from (1), (6) and (7) the evolution is

$$\begin{aligned} |\phi\rangle &\xrightarrow{U_f} \frac{1}{\sqrt{(2N)}} \sum_{i=0}^{2N-1} |i,f(i)\rangle \\ &\xrightarrow{S} \frac{1}{\sqrt{(2N)}} \sum_{i=0}^{2N-1} (-1)^{f(i)}|i,f(i)\rangle \\ &\xrightarrow{U_f} \frac{1}{\sqrt{(2N)}} \sum_{i=0}^{2N-1} (-1)^{f(i)}|i,0\rangle \equiv |\psi\rangle. \end{aligned} \tag{9}$$

The magnitude of the inner product

$$|\langle\phi|\psi\rangle| = \frac{1}{2N}\left|\sum_{i=0}^{2N-1} (-1)^{f(i)}\right| \tag{10}$$

is zero when statement (B) is false, and unity when statement (A) is false. Therefore if, after performing the operations in (9), we measure the projection observable $|\phi\rangle\langle\phi|$, and the outcome is 0, we can be sure that $|\psi\rangle$ was not parallel to $|\phi\rangle$, and hence that (A) is true. And if the outcome is 1, we can be sure that $|\psi\rangle$ was not

Table 1

number, n, of invocations of U_f	form of sequence decided	number of decided sequences, k	probability of solving problem in n invocations, $2^{-2n}k$
1	no sequences decided	0	0
2	0 1 ... } (A) true 1 0 ... }	2^{2N-2} $+2^{2N-2}$	$\frac{1}{2}$
3	0 0 1 ... } (A) true 1 1 0 ... }	2^{2N-3} $+2^{2N-3}$	$\frac{1}{4}$
...	...	...	...
N	0 0 ... 0 1 } (A) true 1 1 ... 1 0 }	2^{2N-N} $+2^{2N-N}$	$\frac{1}{2^{N-1}}$
$N+1$	0 0 ... 0 0 (B) true	2^{N-1}	
	0 0 ... 0 1 (A) true	$+2^{N-1}$	$\frac{1}{2^{N-1}}$
	1 1 ... 1 0 (A) true	$+2^{N-1}$	
	1 1 ... 1 1 (B) true	$+2^{N-1}$	
		total: 2^{2N}	total: 1

orthogonal to $|\phi\rangle$, and hence that (B) is true. The outcome must be either 0 or 1, because those are the only eigenvalues of any projection observable. Therefore the procedure cannot fail to establish the truth of either (A) or (B).

The measurement of $|\phi\rangle\langle\phi|$ can be performed in $O(\ln N)$ steps, by first performing the inverse of the transformation which prepared $|\phi\rangle$ from a blank input $|0,0\rangle$, and then measuring the observable $|0,0\rangle\langle 0,0|$, which is simply a matter of measuring each bit independently. The oracle U_f is invoked exactly twice in (9), and no other invocations are required. This is a clear improvement over the average of $3-2^{-N+1}$ invocations required by the best classical or stochastic method, and a vast improvement over the worse case ($N+1$ invocations) for either of those methods. Note that the problem is solved on each occasion with certainty.

It is interesting to compare the computational complexity of this problem relative with classical and quantum computers. In the classical case, polynomial equivalence class complexity theory (Garey & Johnson 1979) is based on deterministic (DTM) and non-deterministic (NDTM) Turing machine models. We first note the result (referred to as (*)) that for any classical solution of our problem, using a DTM, there exists a function $f: \boldsymbol{Z}_{2N} \to \boldsymbol{Z}_2$ which requires at least $N+1$ invocations of the oracle. To see this, suppose that a DTM can solve the problem for every such f using only $M \leqslant N$ invocations. Let f_c be a constant function so that statement (A) is false and the machine must conclude that statement (B) is true. Then for any M invocations, for inputs chosen in any way whatsoever, there exists a function g which agrees with f_c at all M choices, and has exactly N zero values. Since, by assumption, the M values constitute the only information that the DTM has about the function, it cannot distinguish U_{f_c} from U_g, i.e. it cannot conclude that statement (B) is true. The same argument applies to NDTMs, showing that the decision problem of whether B is true or not, is not in the class NP (though the corresponding problem for A is in NP but not in P).

To assess the complexity of the problem consider first an idealized situation in which the oracle is deemed to deliver its result in one computational step, and not to contribute to the size of the problem's input data. Then the problem is specified

by giving N, which has size $O(\ln N)$. Hence by (*), *exponential* time is required for its solution. The quantum solution, requiring only two invocations, and a time of $O(\ln N))$ to set up the input state, solves the problem in *polynomial* time. Thus the problem is in QP, the quantum analogue of the class P.

If we wish, more realistically, to model the oracle's size and running time, then we could assume the oracle size, for general f, to be $O(N)$, this being the size of the oracle which simply contains a ROM list of the function values. Also for any f there exists an oracle which operates in a time of $O(\ln N)$: e.g. to look up $f(k)$ it could traverse a binary tree following the binary expansion of k. From these estimates, and (*), it follows that any classical computer requires at least polynomial time, whereas the quantum computer requires only logarithmic time, again providing an exponential saving.

If we restrict the input fs to a class of functions whose oracles have size less than $p(\ln N)$, where p is a fixed polynomial unknown to the solver of the problem, then the restricted problem requires exponential time in the classical case and only polynomial time in the quantum case. That is because for any given N this condition does not, from the solver's point of view, exclude any function $f: \boldsymbol{Z}_{2N} \to \boldsymbol{Z}_2$, so by the same argument that we used for the general problem, there cannot be a less-than-exponential classical solution even for the restricted problem.

References

Deutsch, D. 1985 *Proc. R. Soc. Lond.* A **400**, 97–117.

Deutsch, D. 1986 In *Quantum concepts in space and time* (ed. C. J. Isham & R. Penrose). Oxford University Press.

Garey, M. R. & Johnson, D. S. 1979 *Computers and intractability*. New York: W. H. Freeman.

Jozsa, R. 1991 *Proc. R. Soc. Lond.* A **435**, 563–574.

Received 27 April 1992; accepted 15 July 1992