Reflections on Quantum Computing

Quantum Computing Based on Fixed Point Dynamics

In this rather speculative note three problems pertaining to the power and limits of quantum computing are posed and partially answered: (a) when are quantum speedups possible? (b) is fixed-point computing a better model for quantum computing? (c) can quantum computing trespass the Turing barrier?

CHRISTIAN S. CALUDE, MICHAEL J. DINNEEN, AND KARL SVOZIL

C.S. Calude and M.J. Dinneen are in the Computer Science Department, The University of Auckland, Private Bag 92109, Auckland, New Zealand. e-mail: cristian@cs.auckland.ac.nz and mjd@cs.auckland.ac.nz, respectively. K. Svozil is in the Institut für Theoretische Physik, University of Technology Vienna, Wiedner Hauptstraβe 8-10/136, A-1040 Vienna, Austria. e-mail: svozil@tph.tuwien.ac.at.

WHEN ARE QUANTUM SPEEDUPS POSSIBLE?

This section discusses the possibility that speedups in quantum computing can be achieved only for problems that have a *few* or even *unique* solutions [1]. For instance, this includes the computational complexity class UP [2]. Typical examples are Shor's quantum algorithm for prime factoring [3] and Grover's database search algorithm [4] for a single item satisfying a given condition in an unsorted database (see also Gruska [5]).

In quantum complexity, one popular class of problems is BQP, which is the set of decision problems that can be solved in polynomial time (on a quantum computer) so that the correct answer is obtained with probability at least $\frac{1}{2}$ on all instances. Both Shor's and Grover's problems are in BQP. The classical complexity of the primality problem is in NP $\cap$ co-NP and the unsorted database search problem is in P. However, Grover's quantum algorithm runs in time proportional to $\sqrt{n}$ for databases of size n, which is somewhat surprising because the classical lower bound is $\Omega(n)$ (for inputs of size n the algorithm runs in time at least proportional to n).

Problems that can be efficiently solved by utilizing quantum parallelism may belong to a complexity class that we might call "quasi-UP." This class is characterized by a solution space that is small (say, polynomial) with respect to the dimension of the Hilbert space (the exponent of the number of qubits) involved. Formally, quasi-UP is the set of languages decided by a polynomial-time quantum algorithm A that uses at most $f(n)$ qubits,[1] where any input of length n has at most $O(f(n))$ solutions (accepting computations). The class quasi-UP is very similar to the com-

[1]*Note that if* A *runs in polynomial-time, then* f(n) *is bounded by a polynomial.*

plexity class fewP $\subseteq$ NP that classifies problems with a fixed polynomial number of solutions per input size [2], so one can conjecture the following *Quasi-UP-thesis:*

The class fewP is a subset of the class BQP, which is a subset of quasi-UP.

We also suspect that neither NP nor BQP is a subset of the other. Here, BQP contains all of the bounded-error probabilistic polynomial-time problems (the class BPP), which potentially contains some co-NP problems not in NP. Also, most problems belonging to the class nondeterministic polynominal time NP are typically in another, dual regime: there, the number of conceivable solutions is large with respect to the number of bits involved to define the problem. Most NP-complete problems fall into this category, which is the primary reason we believe NP $\not\subseteq$ BQP.

We now give one simple example of how the quasi-UP-thesis can be applied. Suppose we are interested in the intensively studied traveling salesman problem (TSP) of finding the cheapest trip in cost through all of the nodes of a map. Suppose we know (as naturally suspected) that our problem instances (i.e., in the "real-world") have a unique best solution (or, at worst, a few equal optimal solutions). For these types of inputs, our restricted TSP problem is in fewP,[2] and thus we can expect to have an efficient quantum search algorithm.

Although we cannot give a direct proof of this quasi-UP-thesis, some informal arguments can be brought forward in its support. Efficient algorithms in quantum computing make use of the quantum parallelism. Yet in order to be able to extract a classically useful solution from the resulting quantum state, one has to extract the information by proper phase transformations and interference. And it is interference—the buildup of phases at points that indicate the problem solutions—we are mostly concerned about. Interference guarantees that the result of the quantum calculation can be effectively read out of the superposition of states.

A problem allowing only a single solution therefore has a better chance to be solvable by a quantum algorithm. In particular, in the interference phase the single solution could allow for a higher contrast[3] and thus a better detection efficiency than a situation that would allow for many solutions. Thus, for example, suppose we apply a quantum algorithm to an unsorted database problem that allows up to a fixed constant number of matches. Then our success with a quantum algorithm will probably deteriorate even though the probability of randomly findings a match increases. (Note this may be counter-intuitive from a classical perspective.)

It therefore appears to be not totally unreasonable to speculate that detection efficiency might drop linearly with the number of solutions, as for a problem with n items the contrast drops like $O(1/e^n)$. This could make most NP problems effectively intractable for quantum algorithms. Thus, the ability to enhance contrast and detector efficiency for problems in NP appears to be one of the most crucial steps a quantum algorithm has to cope with. The task here is formidable—to single out the most favorable solution out of a sea of possible but nonoptimal ones.

FIXED-POINT QUANTUM COMPUTATIONS

One radically new possibility for quantum computing would be an approach based on fixed-point computations. Stated differently, a quantum computation may arrive at results that are fixed-points of a unitary operator or, more precisely, eigenstates of unitary or Hermitian operators with eigenvalue 1.

One task that is impossible within the domain of classical computation but almost trivial for quantum algorithms is the solution of diagonalization problems, at least for small dimensions. Diagonalization is based on bit switches from true to false and vice versa. In order to solve this problem for a single qubit, it can be implemented by a unitary and self-adjoint *not*-operator

$$\hat{D} = not = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The cigenstates of $\hat{D}$ are

$$|I\rangle, |II\rangle = \frac{1}{\sqrt{2}}\left[\begin{pmatrix} 1 \\ 0 \end{pmatrix} \pm \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right], \qquad (1)$$

with the eigenvalues +1 and -1, respectively. The solution of the diagonalization problem is the eigenvector $|I|$ associated with the eigenvalue 1. This is a mixture or superposition of the classical bit states true and false.

Other, more general problems may be solvable by applying the fixed-point theorem of computability theory to quantum computations. The strategy is to assume a normal operator *A*, which, because of some yet unknown procedure, encodes an algorithm φ_A. Thereby, we seek an unknown eigenstate a_1 of A with eigenvalue 1. Let us assume that we start with a totally "unbiased" state 1, which, in matrix notation, is just represented by the unit matrix. Indeed, if we have information about the solution we may prepare the state in such a way that the outcome of the fixed-point is more likely. (In the extreme case we know the solution before the measurement and prepare the initial state to be exactly the fixed-point state. The outcome of the fixed-point state is thus certain.) As A is measured, a_1 is obtained if we observe the eigenvalue 1. We may now identify the fixed-point solution a_1 with the algorithm φ_A.

COMPUTING THE UNCOMPUTABLE?

One fundamental result of theoretical computer science is Turning's proof (in Ref. 6) that it is undecidable to determine whether a general computer program will halt or not. This is formally

[2]*Our problem is probably not NP-hard.*

[3]*The number of solutions seem to influence the relative minimal and maximal particle intensities and the width thereof, because they are observed in an interference pattern depicting the average number of clicks in a detector measuring the output of a quantum computer.*

known as the halting problem. We can restrict our attention to Turing machines, because they are equivalent in computational power to any "conventional" computer [7,8]. In what follows we present an attempt to trespass the Turing barrier. The method discussed might in principle allow us to "solve" the halting problem (for another proposal, see Mitchison and Josza, [9]). Thereby we are well aware of the fact that for all practical purposes [10] this goal will remain unreachable, at least within quantum computing.

Assume that it is possible to design a halting qubit, which indicates whether a computation has actually reached a state associated with a halting condition. Assume further that the halting qubit starts in its nonhalting state and, because the evolution is unitary, the buildup of the amplitude is continuous in time.

In such a case, the halting qubit acquires a halting component that is nonzero even in finite time. Therefore, a detection of a halting computation at small time scales is conceivable even if the associated classical computation lasts "very" long. The price to be paid is the "very small" amplitude and, associated with it, a correspondingly small chance of detection.

To be a little bit more precise, we need some rudiments of algorithmic information theory (see Chaitin [11,12], Calude [13,14]). We will work with programs with no input, which produce binary strings as outputs. For any n we denote by P_n a program of length n that halts and produces the longest string among all outputs produced by all programs of length n that eventually stop. We denote by $\Sigma(n)$ the length of the output produced by P_n. Here Σ is the busy beaver function [15,16]: it grows faster than every computable function of n. Let H be the program-size complexity, which is the length of the smallest universal program generating a particular binary string.

Assume that any program that halts requires a running time at least proportional to the length of its output. If an n-bit program p halts, then the time t it takes to halt satisfies $H(t) \leqslant n + c$. So if p has run for time T without halting and T has the property that if $t \geqslant T$, then $H(t) > n + c$, then p will never halt. This shows that the running times of the programs in the sequence $P_1, P_2, \ldots P_n, \ldots$ grow faster than any computable function.

We are now ready to present the argument. Let us assume the halting qubit is represented by

$$|Halt\rangle = c_h(t)|h\rangle + c_n(t)|n\rangle,$$

where $|h\rangle$, $|n\rangle$ represent the halting state and nonhalting state and $c_h(t)$, $c_n(t)$ are time-dependent amplitudes thereof, respectively.

Initially, let $|c_h(t)| = |c_n(t)| - 1 = 0$. As a worst-case scenario derived from the above analysis, for a linear buildup of the amplitude we obtain

$$|c_h(t)|^2 \propto (\Sigma(H(n) + O(1)))^{-1}.$$

The setup of a detection of $|Halt\rangle$ is a simple transmission measurement of the halting qubit. Although the buildup may be very slow, there is a nonvanishing chance to obtain a solution of the halting problem in finite time.[4] Of course, the solution is probabilistic (one can argue that all mathematical proofs or computer programs are ultimately probabilistic, see Davis [17], De Millo, Lipton, Perlis [18]), but goes beyond the capability of any classical computation: even the best probabilistic algorithms are not able to achieve this computational power (by a classical result [19], probabilistic algorithms are equivalent to Turing machines).

Let us finally notice that by virtue of the same information-theoretic argument, the possibility of time-travel (see, Nahin [20]) would not solve the halting problem, unless one could travel back and forth in time at a pace exceeding the growth of any computable function.

[4]*One referee suggested that this might be also true for a classical probabilistic machine whose transitions are governed by a Poisson process in continuous time.*

ACKNOWLEDGMENTS
We thank Greg Chaitin, Garry Tee, Marius Zimand, and the anonymous referee for criticism and encouragement.

REFERENCES
1. Gottlob, G. Private communication to K. Svozil, 1998.
2. Johnson, D.S. In Handbook of Theoretical Computer Science, Vol. A; van Leeuwen, J., Ed.; Elsevier: Amsterdam, 1990, p 69.
3. Shor, P.W. Algorithms for quantum computation: discrete log and factoring, Proceedings of the 35th IEEE Annual Symposium on Foundations of Computer Science, 1994, p 124.
4. Grover, L.K. A fast quantum mechanical algorithm for database search. Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, 1996, p 212.
5. Gruska, J. Quantum Computing; McGraw-Hill: London, 1999.
6. Turing, A.M. Proc London Math Soc Series 2, 1936–1937, 42, 230, 544.
7. Calude, C. Theories of Computational Complexity; North-Holland: Amsterdam, 1988.
8. Barrow, J.D. Impossibility—The Limits of Science and the Science of Limits; Oxford University Press: Oxford, 1998.
9. Mitchison, G.; Josza, R. Counterfactual computation, quant-ph/9907007.
10. Bell, J.S. Speakable and Unspeakable in Quantum Mechanics; Cambridge University Press: Cambridge, 1987.
11. Chaitin, G.J. Information, Randomness and Incompleteness, Papers on Algorithmic Information Theory; World Scientific: Singapore, 1987. (2nd ed., 1990).
12. Chaitin, G.J. The Unknowable; Springer-Verlag: Singapore, 1999.
13. Calude, C. Information and Randomness—An Algorithmic Perspective; Springer-Verlag: Berlin, 1994.
14. Calude, C.S.; Casti, J.; Dinneen, M.J. (eds.). Unconventional Models of Computation; Springer-Verlag: Singapore, 1998.
15. Rado, T. Bell System Technical J 1962, 41, 877.
16. Chaitin, G.J.; Arslanov, A.; Calude, C. EATCS Bull. 1995, 57, 198.
17. Davis, P.J. Amer. Math. Monthly 1972, 79, 252.
18. De Millo, R.; Lipton, R.; Perlis, A. Comm. ACM 1979, 22, 271.
19. De Leeuw, K.; Moore, E.F.; Shannon, C.E.; Shapiro, N. In Automata Studies; Shannon, C.E.; McCarthy, J., Eds.; Princeton University Press: Princeton, NJ, 1956, p 183.
20. Nahin, P.J. Time Machines; Springer-Verlag: New York, 1999.