Single Quantum Querying of a database

Terhal, B.M.; Smolin, J.A.

DOI
10.1103/PhysRevA.58.1822
Publication date
1998

Published in
Physical Review A

Single quantum querying of a database

Barbara M. Terhal[1,*] and John A. Smolin[2,†]
[1]Instituut voor Theoretische Fysica, Universiteit van Amsterdam, Valckenierstraat 65, 1018 XE Amsterdam, The Netherlands and Centrum voor Wiskunde en Informatica, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands
[2]IBM Research Division, T. J. Watson Research Center, Yorktown Heights, New York 10598

(Received 12 November 1997)

We present a class of fast quantum algorithms, based on Bernstein and Vazirani's parity problem, that retrieves the entire contents of a quantum database Y in a single query. The class includes binary search problems and coin-weighing problems. We compare the efficiency of these quantum algorithms with the classical algorithms that are bounded by the classical information-theoretic bound. We show the connection between classical algorithms based on several compression codes and our quantum-mechanical method. [S1050-2947(98)07909-8]

PACS number(s): 03.67.Lx, 89.70.+c

I. INTRODUCTION

Quantum computers have been shown recently to be able to solve certain problems faster than any known algorithm running on a classical computer [1–3]. These problems include factoring, which can be performed in polynomial time on a quantum computer [2], but is widely believed to be difficult on a classical computer, and database lookup, which is provably faster on a quantum computer [3]. Understanding the power of quantum algorithms and developing new algorithms is of major interest as the building of a quantum computer will require a huge investment.

In this paper we present quantum algorithms for binary search and coin-weighing problems in which the information in a quantum database is retrieved with a single query. These are applications of the Bernstein and Vazirani parity problem [4,5] and provide a strong illustration of the power of quantum computation and point out the limitations of classical information-theoretic bounds applied to quantum computers.

Information theory is a useful tool for analyzing the efficiency of classical algorithms. Problems involving information retrieval from a database are particularly amenable to such analysis. Consider this database search problem: we have a database Y that contains n items, of which a single one is marked. This database is represented as a bit string y of length n with Hamming weight 1 (y has exactly one "1"). One would like to locate the marked item in as few queries to the database as possible. The queries are bit strings x of length n such that the database returns the answer

$$a(x,y)=x\cdot y\equiv\left(\sum_{i=1}^{n} x_i y_i\right) \mod 2 \tag{1}$$

where x_i and y_i are the ith bits of x and y. A simple version of this problem is the case in which the allowed queries x have Hamming weight 1. The information retrieved by a single query $x_j=\delta_{ij}$ is small—it adds or eliminates item i from the set of possible marked items. It thus takes $n-1$ queries to locate the marked item in the worst case. A surprising result of Grover [3] is that a quantum-mechanical algorithm can be faster than this and find the marked item with high probability in $O(\sqrt{n})$ *quantum* queries, contrary to one's "classical" intuition. Grover's algorithm does not, however, violate the information-theoretic lower bound on the minimal number of queries M.

The information-theoretic lower bound [11] on M is given by the amount of information in the database divided by the maximal amount of information retrieved by a query that has A possible answers, i.e.,

$$M\geqslant\frac{H(Y)}{\log_2 A}, \tag{2}$$

where $H(Y)=-\Sigma_y p_y \log_2 p_y$, p_y is the probability for Y to contain y and $\Sigma_y p_y=1$.

A quantum algorithm employs a database that responds to superpositions of queries with superpositions of answers. The quantum database acts on two input registers: register X containing the query state $|x\rangle$ and register B, an output register of dimension A initially containing state $|b\rangle$. We define the operation of querying the database as

$$R_y:|x,b\rangle\rightarrow|x,[b+a(x,y)] \mod A\rangle, \tag{3}$$

where R_y is a classical reversible transformation that maps basis states to basis states (that is, a permutation matrix) depending on the contents of the database, and $a(x,y)$ is the answer to query x, given database state y. In a classical query only query basis states $|x\rangle$ are used and the output register B is initially set to $|0\rangle$. However, a quantum database is not restricted to working only on basis states but can handle arbitrary superpositions of inputs [6]. Because of this the information that is retrieved by a single quantum query is not bounded by $\log_2 A$. The relevant quantity in the quantum setting is the accessible information in the registers X and B (together called XB) and the internal state of the quantum computer Φ about the database Y. Together these are always in one of a set of pure states $\{|\psi_y\rangle,p_y\}_{y\in Y}$ and the accessible information on y is bounded by the Kholevo bound [7]

*Electronic address: terhal@phys.uva.nl
†Electronic address: smolin@watson.ibm.com

$$I_{\text{acc}}(\Phi XB) \leqslant S(\Phi XB), \tag{4}$$

where $S(\Phi XB) = -\mathrm{Tr}\rho_{\Phi XB}\log_2\rho_{\Phi XB}$ is the Von Neumann entropy of ΦXB and $\rho_{\Phi XB} = \Sigma_y p_y|\psi_y\rangle\langle\psi_y|$. In the case of a classical query, the Von Neumann entropy $S(\Phi XB)$ is strictly less than $\log_2\dim(B) = \log_2 A$, which gives rise to the classical bound (2). The quantum algorithms analyzed in this paper ''violate'' the classical information-theoretic bound by extracting extra information in the phases of the query register X. It is notable that Grover's Hamming weight 1 problem [3] has been proven optimal [8,9]; no quantum algorithm for this problem can violate the classical information-theoretic bound (2).

The quantum algorithms presented here are a major improvement on the classical algorithms in terms of computation time if the computation performed by the database is costly. All our algorithms make use of an interaction with the database of the form $a(x,y) = x \cdot y$. A direct implementation of such a database takes $O(\log_2 n)$ time using n Toffoli and n XOR gates in parallel. In general we will be given some algorithm that computes $a(x) = x \cdot y$ for any input $x \in \{0,1\}^n$ and that runs in some time $T(n)$. We will compare the running time of the quantum algorithms including this cost to the classical running time.

The parity problem and coin weighing

Bernstein and Vazirani [4,5] have given the first algorithm in which a single quantum query to the database is sufficient and a strong violation of the classical information-theoretic bound comes about. In their parity problem they consider a database Y that contains an arbitrary n-bit string y. The answer to queries represented by n-bit strings x to the database is the parity of the bits common to x and y given by $a(x,y) = x \cdot y$. Note that the problem is to determine y in its entirety, *not* to merely determine the parity of y. Bernstein and Vazirani have shown that y can be determined in only two queries to the database. But by preparing the output register B in an initial superposition $1/\sqrt{2}(|0\rangle - |1\rangle)$ [10], the algorithm can be simplified to comprise a single query. Their algorithm can be directly applied to the coin-weighing problem. Coin-weighing problems are a group of problems in which a set of defective coins is to be identified in a total set of coins [11]. Assume there are two types of coins, good and bad ones, and we can weigh arbitrary sets of coins with a spring scale (which gives the weight of the set of coins directly, as opposed to a balance that compares two sets of coins). All sets of coins are equiprobable. A set of n coins is represented as a bit string y of length n where $y_i = 1$ indicates that coin i is defective. A weighing can be represented by a query string x, where x_i specifies whether coin i is included in the set to be weighed. The result of a classical weighing is the Hamming weight of the bitwise product of x and y, $w_H(x \wedge y)$. For this problem the information-theoretic bound (2) gives

$$M \geqslant \frac{n}{\log_2(n+1)}. \tag{5}$$

This is close to what the best predetermined algorithm, which perfectly identifies the set of coins, can achieve [11],

$$\lim_{n\to\infty} M_{\text{pre}}(n) = \frac{2n}{\log_2(n)}. \tag{6}$$

If one has a spring scale capable of performing weighings in superposition, then one can use the Bernstein-Vazirani algorithm to identify the defective coins perfectly with a single weighing by using only the parity of the Hamming weight answer. We can compare the total running time of this quantum algorithm with that of the classical algorithm. The preprocessing and postprocessing of the register X of the query state and the preprocessing of the B register all consist of the Hadamard transforms on individual bits,

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{7}$$

which can be done in parallel. The total running time is then simply $2 + T(n)$. In the classical algorithm the database is queried at least [see Eq. 5)] $n/\log_2(n+1)$ times resulting in a total running time of at least $nT(n)/\log_2(n+1)$.

II. COMPRESSIVE ALGORITHMS

In this section we will consider modifications of this problem in which the information in the database $H(Y)$ is less than n bits. In these cases retrieving the data from the source can be viewed as a problem of data compression of a source Y. We will restrict ourselves here to the compression of the data from a single use of the source. In the classical case, each query to the database retrieves a single digit of the word into which the bit string y will be encoded. The minimal set of predetermined classical queries will serve to construct the single quantum query algorithm. We will use coding schemes that minimize the amount of pre/postprocessing in the single query quantum algorithm. A classically optimal encoding scheme (cf. [12]) has

$$H(Y) \leqslant \sum_i p_i l_i \leqslant H(Y) + 1, \tag{8}$$

where l_i is the length of the compacted y_i and p_i is the probability that the database contains bit string y_i. It is not guaranteed, however, that such an optimal encoding scheme can be implemented by the type of database interaction that one is given to use, namely expression (3).

In the following section we present single query quantum algorithms of which the construction is based on optimal classical encoding schemes, namely Huffman coding. In the section thereafter we consider more general types of databases and use a random coding scheme. Each of these schemes will require precomputation of a set of queries based on the encoding schemes. The time for this computation will not be counted in the total running time as the queries can be precomputed once and reused on subsequent problems.

A. Binary search problem

The first part of this section is included for pedagogical purposes. Binary search problems are defined as problems in which the database responds with one of two answers to the query. Here we look at such a search problem in which the queries have Hamming weight $n/2$. The database contains a bit string y with Hamming weight 1. Let us first look at the problem in which all these bit strings are equiprobable. We assume that n is an integer power of 2. For other n one simply extends the database size to the next higher power of 2.

Classically it is well known that the marked item can be found in $\log_2 n$ queries, which achieves the classical information-theoretic bound (2). The kth query, g_k, is a string of 2^{k-1} zeros alternating with a string of 2^{k-1} ones, where $k=1,\ldots,\log_2 n$, i.e.,

$$\begin{aligned} &g_1=01010101\ldots,\\ &g_2=00110011\ldots,\\ &g_3=00001111\ldots,\\ &\text{etc.}\end{aligned} \tag{9}$$

The result of query g_k is

$$z_k \equiv g_k \cdot y = a(g_k, y), \tag{10}$$

where z_k is the kth bit of the encoding z of y. Each y will have a different encoding z and thus z uniquely determines y. The g_k's are the generators of the group F of Walsh functions whose group multiplication rule is addition modulo 2. We can represent a Walsh function f_s as

$$f_s = \sum_{i=1}^{\log_2 n} g_i s_i \mod 2, \tag{11}$$

where s is an arbitrary bit string of length $\log_2 n$.

The quantum-mechanical algorithm that takes a single query is similar to the Bernstein-Vazirani algorithm. It makes use of superpositions of all the Walsh functions. We construct the query state

$$|\psi\rangle = \frac{1}{\sqrt{n}} \sum_s |s, f_s\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{12}$$

After one query the state becomes

$$|\psi_y\rangle = \frac{1}{\sqrt{n}} \sum_s (-1)^{a(f_s,y)} |s, f_s\rangle \otimes \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{13}$$

It can be shown that

$$\langle \psi_y | \psi_{y'} \rangle = \frac{1}{n} \sum_s (-1)^{a(f_s,y)+a(f_s,y')} = \delta_{yy'}. \tag{14}$$

We can write

$$a(f_s, y) = \sum_{k=1}^{\log_2 n} s_k a(g_k, y) \mod 2. \tag{15}$$

Using Eq. (10), it follows that $a(f_s,y) = s \cdot z$, and with this we can rewrite Eq. (14) as

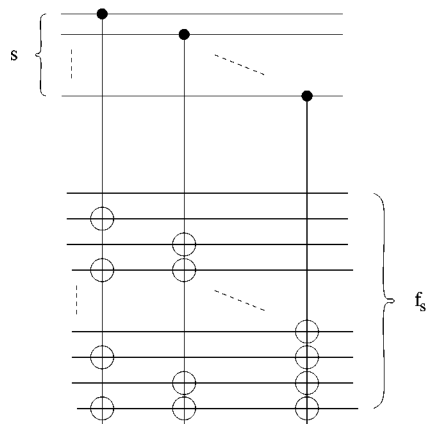

FIG. 1. ''Walsh'' Circuit.

$$\langle \psi_y | \psi_{y'} \rangle = \frac{1}{n} \sum_s (-1)^{s\cdot z + s \cdot z'} = \delta_{zz'} = \delta_{yy'}. \tag{16}$$

As all states $|\psi_y\rangle$ and $|\psi_{y'}\rangle$ are orthogonal, they can be distinguished by a measurement and no further queries to the database are required.

What are the transformations that are required for preprocessing and postprocessing of this single query? The preparation of the queries takes $1+(n\log_2 n)/2$ steps. Register s is prepared in superposition using parallel one-bit Hadamard transforms and used as input to the circuit shown in Fig. 1. The circuit in Fig. 1 uses multibit XOR's, which we have counted as being in series. The same sequence in reverse is used as the postprocessing. The total running time is thus $2+n\log_2 n+T(n)$. In the classical case the queries are also prepared using the circuit in Fig. 1, but the multibit XOR's can be done in parallel, and the total time is $\log_2 n+T(n)\log_2 n$. Note that in the Cirac-Zoller ion-trap model [13] of quantum computation, a multibit XOR gate can be done in parallel by using the ''bus phonon'' modes. Such quantum computers run our algorithm in time $2+\log_2 n+T(n)$.

The straightforward Bernstein-Vazirani algorithm to retrieve y and subsequent compaction of y to z would have taken time $2+T(n)$ for the algorithm plus $n\log_2 n$ time steps for the compression, which amounts to the same running time as this direct compression.

If we generalize this problem to databases that have unequal probabilities assigned to different y's, a scheme based on Huffman coding [12] is sometimes more efficient in terms of pre/postprocessing. We assume that the probability distribution, or $H(Y)$, is known beforehand.

Huffman coding is a fixed-to-variable-length encoding of a source, in our case the database, which is optimal in the sense that it minimizes the average codeword length $\Sigma_i p_i l_i \leq H(Y)+1$ with $H(Y)$ the entropy of the source. The encoding prescribes a set of queries that play the role of the Walsh generators in the equal probabilities case. This is illustrated with an example in Fig. 2. (In fact, the Huffman construction results in Walsh queries in the equal-probability case.)

Classically, instead of querying with the Walsh generators, one can use these Huffman queries, until the marked item has been found. The optimality of the Huffman code assures that the expected number of queries is minimized. Choose the set of queries that will take the place of the

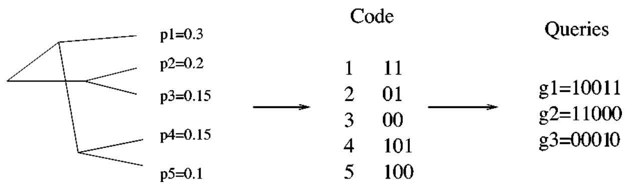

FIG. 2. Example of a Huffman code.

Walsh generators in the following way. Select a set of m queries, the first m queries that are used in the classical case, such that the probability of not finding the marked item after these m queries is very small. The value of m will depend on the probability distribution. If

$$m \leqslant \lceil \log_2 n \rceil, \tag{17}$$

this Huffman scheme can be more efficient than a Walsh scheme.

Note that this requirement is not necessarily satisfied for all probability distributions. For example, for the distribution $p_1 = 1/10, p_i = 9/10n, i = 2, \ldots, n$, the length of $n-1$ encoded words will be about $\lceil \log_2 n \rceil$. Choosing a number of queries that is less than the length of these words will result in a high probability of error.

This set of m queries will take the place of the Walsh generators in our quantum algorithm. The circuit that implements the Huffman queries will be as in Fig. 1 but with a different pattern of XOR's corresponding the Huffman queries. All the database states that gave rise to distinct codewords after these m classical queries will give rise to distinct $|\psi_y\rangle$ in our quantum algorithm.

If we query only once, the total running time will be $2 + mn + T(n)$ if we are willing to accept a small chance of error. A classical algorithm that uses the same Huffman queries and has the same probability of error takes $m + mT(n)$ time. Thus for some probability distributions, m can be significantly smaller than $\log_2 n$ and the algorithm is faster than a straightforward search with the Walsh queries.

B. Random coding

The binary search and the coin-weighing problem are special cases of a more general problem in which we have a database Y that contains k arbitrary base A strings of length n. Here we restrict ourselves to databases that contain k equally probable strings.

The queries x are all possible base A strings of length n, the elements of $(\mathbb{Z}_A)^n$. The database returns the answer

$$a(x,y) = \sum_i^n x_i y_i \mod A \equiv x \cdot y. \tag{18}$$

The information $H(Y)$ is equal to $\log_A k$. A classical predetermined algorithm to determine y with high probability makes use of $m = \log_A k + l$ random strings, where l is a relatively small integer. Pick m linearly independent random base A strings of length n; these are the queries g_i, $i = 1, \ldots, m$. Similarly to Eq. (10) we define the encoding as

$$z_k = g_k \cdot y, \tag{19}$$

where z_k is the kth digit of z. The g_i's are used to compress the string y of length n to the codewords z of length m. What is the probability that the codeword z determines y uniquely? The probability that two base A strings of length n are mapped onto the same codeword is equal to $(1/A)^m$. Thus, the probability of a collision with y is

$$p_{\text{col}} = 1 - \left[1 - \left(\frac{1}{A}\right)^m\right]^{k-1}. \tag{20}$$

For small 2^{-l} we approximate

$$p_{\text{col}} = 1 - \left(1 - \frac{2^{-l}}{k}\right)^{k-1} \sim 2^{-l}(1 - 1/k) + O(2^{-2l}). \tag{21}$$

This probability can be made arbitrarily small for only a relatively small l. Thus $O(\log_A k)$ random g_i's are sufficient to retrieve the information with arbitrarily low probability of error. It is clear that for negative l the length of the codewords is not sufficiently large to avoid collisions. A codeword length of $O(\log_A k)$ is thus necessary as well as sufficient. If the contents of the database are to be determined with certainty, the codeword length m must be made larger. A code with no collisions and with codeword length $O(2\log_A k)$ always exists (cf., the discussion of the birthday problem [14]).

Our quantum algorithm to determine the contents of the database in a single query with high probability makes use of this classical random coding construction. A set of random linearly independent strings g_i, $i = 1, \ldots, m$, is the set of generators of a group C_A. The multiplication rule for this group is a digitwise addition modulo A and the identity element is the string 0. Members of C_A can be written as

$$c(s) \in C_A \Rightarrow c(s)_k = \sum_i (g_i)_k s_i \mod A \tag{22}$$

with $c(s)_k$ the kth digit of a group element $c(s)$ and s is a base A string of length m. Due to the linear independence of the generators, C_A is a subgroup of $(\mathbb{Z}_A)^n$ with A^m elements.

In the quantum algorithm we construct a state

$$|\psi\rangle = \frac{1}{\sqrt{A^m}} \sum_s |s, c(s)\rangle \otimes \frac{1}{\sqrt{A}} \sum_{b=0}^{A-1} \omega_A^b |b\rangle, \tag{23}$$

with $\omega_A = e^{i2\pi/A}$. The query results in the state

$$|\psi_y\rangle = \frac{1}{\sqrt{A^m}} \sum_s \omega_A^{-a(c(s),y)} |s, c(s)\rangle \otimes \frac{1}{\sqrt{A}} \sum_{b=0}^{A-1} \omega_A^b |b\rangle. \tag{24}$$

We can write, using the encoding z of y defined in Eq. (19),

$$a(c(s),y) = s \cdot z. \tag{25}$$

Thus we have

$$\langle \psi_y | \psi_{y'} \rangle = \frac{1}{A^m} \sum_s \omega_A^{s\cdot(z-z')} = \delta_{zz'}. \tag{26}$$

If two different strings y and y' are mapped onto a different codeword, they are thus distinguishable by a measurement. The probability that this occurs [Eq. (21)] can be made arbitrarily small just as in the classical case since the encoding is the same. In order to measure, we reverse the preparation steps and then we perform a Fourier transform over $(\mathbb{Z}_A)^m$,

$$H_A:|s\rangle \rightarrow \frac{1}{\sqrt{A^m}}\sum_z \omega_A^{s\cdot z}|z\rangle. \tag{27}$$

A measurement in the query basis determines z and, with high probability, y.

The circuit used to implement the random coding is similar to that in Fig. 1 but the XOR's are replaced by summation base A operators,

$$\oplus_A(a,b)=(a+b) \mod A, \tag{28}$$

and their locations are according to the random queries.

The total quantum running time is $2+mn+T(n)$ if we take the basic unit of time to be an operation on an A-dimensional Hilbert space. The classical time using the same random codewords is $m+mT(n)$. Since m is less than n, this algorithm is better than the direct coin-weighing algorithm provided we are willing to tolerate a small chance of error bounded by p_{col}.

III. DISCUSSION

We have discussed the complexity of our quantum algorithms compared to a classical setup and shown that the quantum algorithms are faster in situations in which $T(n) > O(n)$. In problems where querying the database would occur repeatedly, a bigger (real) separation between the classical computation time and the quantum computation time could be achieved (see [5] for an instance of such a problem).

It is noteworthy that in the binary search problem in the classical case only the generators of the Walsh functions are required, while the quantum algorithm needs all the Walsh functions to achieve this speedup. It would be interesting to find out whether any speedup is possible if the database only responds to queries that are the generators.

We have chosen the quantum database R_y as defined in expression (3) to make a fair comparison with the classical setting. A unitary U_y could easily become more powerful, as was pointed out in [15]. At its most general, a quantum database could be defined by an arbitrary unitary transformation acting on an input register and a hidden quantum state (the database). This has no good classical analog and might be worthwhile to explore.

We would like to thank Charles H. Bennett, David P. DiVincenzo, and Markus Grassl for helpful discussions, and the U.S. Army Research Office and the Institute for Scientific Interchange, Italy, for financial support. B.M.T. would like to thank Bernard Nienhuis and Paul Vitanyi for advice and encouragement.

[1] D. Simon, in *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science* (IEEE Computer Society Press, Los Alamitos, CA, 1994), p. 116.
[2] P. W. Shor, in *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science* (Ref. [1]), p. 124.
[3] L. K. Grover, in *Proceedings of the 28th Annual ACM Symposium on Theory of Computing* (ACM, New York, 1996), pp. 212–219.
[4] E. Bernstein and U. Vazirani, in *Proceedings of the 25th Annual ACM Symposium on Theory of Computing* (ACM, New York, 1993), pp. 11–20.
[5] E. Bernstein and U. Vazirani, SIAM J. Comput. **26**, 1411 (1997).
[6] One could have a nonunitary quantum database where the X register is always reset to $|0\rangle$ to ensure it gives no information on Y. This would not matter to a classical query where all the information is in B, but severely cripples the quantum algorithm. It is not known if such a database will allow any improvement over classical algorithms.
[7] A. S. Kholevo, Probl. Peredachi Inf. **9**, 3 (1973). This paper has appeared in an English translation in Probl. Inf. Transm. **9**, 177 (1973).
[8] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, SIAM J. Comput. **26**, 1510 (1997).
[9] M. Boyer, G. Brassard, P. Høyer, and A. Tapp, Fortsch. Phys. **46**, 493 (1998).
[10] A. Barenco *et al.*, Phys. Rev. A **52**, 3457 (1995).
[11] See Martin Aigner, *Combinatorial Search*, Wiley-Teubner Series in Computer Science (Teubner, Stuttgart, 1988).
[12] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, Wiley Series in Telecommunications (Wiley, New York, 1991).
[13] J. I. Cirac and P. Zoller, Phys. Rev. Lett. **74**, 4091 (1995); see S. Braunstein and J. A. Smolin, Phys. Rev. A **55**, 945 (1997) for a discussion.
[14] W. Feller, *An Introduction to Probability Theory and its Applications*, Wiley Mathematical Statistics Series (Wiley, New York, 1950), Vol. I, p. 29.
[15] J. Machta, Physics Department, University of Massachusetts at Amherst, 1996, e-print quant-ph/9805022.