Exact Quantum Query Algorithm for Error Detection Code Verification

Alina Vasilieva

Faculty of Computing, University of Latvia
Raina bulv. 29, LV-1459, Riga, Latvia
alina.vasilieva@gmail.com

Abstract. Quantum algorithms can be analyzed in a query model to compute Boolean functions. Function input is provided in a black box, and the aim is to compute the function value using as few queries to the black box as possible. A repetition code is an error detection scheme that repeats each bit of the original message r times. After a message with redundant bits is transmitted via a communication channel, it must be verified. If the received message consists of r-size blocks of equal bits, the conclusion is that there were no errors. The verification procedure can be interpreted as an application of a query algorithm, where input is a message to be checked. Classically, for N-bit message, values of all N variables must be queried. We demonstrate an exact quantum algorithm that uses only $N/2$ queries. [1]

1 Introduction

Quantum computing is an exciting alternative way of computation, which is based on the laws of quantum mechanics. This branch of computer science is developing rapidly; various computational models exist, and this is a study of one of them.

Let $f(x_1, x_2, ..., x_N) : \{0,1\}^N \to \{0,1\}$ be a Boolean function. We consider the black box model (also known as the query model), where a black box contains the input $X = (x_1, x_2, ..., x_N)$ and can be accessed by querying x_i values. The goal is to compute the value of the function. The complexity of a query algorithm is measured by the number of questions it asks. The classical version of this model is known as decision trees [1]. This computational model is widely applicable in software engineering. For instance, a database can be considered a black box, and, to speed up application performance, the goal is to reduce the number of database queries.

Quantum query algorithms can solve certain problems faster than classical algorithms. The quantum query model differs from the quantum circuit model [2–4], and algorithm construction techniques for this model are less developed. The problem of quantum query algorithm construction is very non-trivial. Although

[1] This research is supported by the European Social Fund project Nr. 2009/0138/1DP/1.1.2.1.2/09/IPIA/VIAA/004, Nr. ESS2009/77

Mathematical and Engineering Methods in Computer Science (MEMICS), Znojmo, Czech Republic, 2009.
Petr Hliněný, Vashek Matyáš, Tomáš Vojnar (Eds.)
Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany
Digital Object Identifier: 10.4230/DROPS.MEMICS.2009.2343

there are many lower bound and upper bound estimations of quantum query algorithm complexity [2, 5–7], there are very few examples of original quantum query algorithms.

In this paper, we demonstrate an exact quantum query algorithm for resolving a specific problem. The task is to verify a codeword message that has been encoded using repetition code for detecting errors [8] and has been transmitted across a communication channel. Considered repetition code duplicates each bit of the message. The verification procedure can be considered as an application of a query algorithm, where the codeword to be checked is contained in a black box. To verify the message in the classical way, we would need to access all bits. That is, for a codeword of length N, all N queries to the black box would be required. We will demonstrate an exact quantum query algorithm that requires $N/2$ queries only.

An exact algorithm always produces a correct answer with 100% probability. Another variation is to use a bounded-error model, where an error margin of $1/3$ is allowed. It is well known that in the bounded-error model, a large difference between classical and quantum computation is possible. The complexity gap can be exponential as, for instance, in the case of Shor's algorithm [9]. Another famous example is Grover's search algorithm that achieves a quadratic speed up [10]. However, in certain types of computer software, we cannot allow even a small probability of error, for example, in spacecraft, aircraft, or medical software. For this reason, the development of exact algorithms is extremely important.

Regarding exact quantum algorithms, the maximum speedup achieved as of now is half the number of queries compared with a classical deterministic case [11]. The major open question is: is it possible to reduce the number of queries by more than 50%? In this paper, we present an algorithm that achieves a borderline gap of $N/2$ versus N.

2 Preliminaries

This section contains definitions and provides theoretical background on the subject.

2.1 2.1 Error Detection and Repetition Codes

In this article, we investigate a problem related to information transmission across a communication channel. The bit message is transmitted from a sender to a receiver. During that transfer, information may be corrupted. Because of the noise in a channel or adversary intervention some bits may disappear, or may be reverted, or even added. Various schemes exist to detect errors during transmission. In any case, a verification step is required after transmission. The received codeword is checked using defined rules and, as a result, a conclusion is made as to whether errors are present.

We consider a repetition error detection scheme known as repetition codes. A repetition code is a (r, n) coding scheme that repeats each n-bit block r times

[8]. Verification procedure for repetition code is the following - we need to check if in each group of r consecutive blocks of size n all blocks are equal.

In this article, we examine verification of the (2,1) repetition code. The verification process can be expressed naturally as a computing Boolean function in a query model. We assume that the codeword to be checked is located in a black box. We define the Boolean function to be computed by the query algorithm as follows.

Definition 1. *The Boolean function $VERIFY_N(X)$, where $N = 2k$, $X = (x_1, x_2, ..., x_{2k})$ is defined to have a value of "1" Iff variables are equal by pairs:*

$$VERIFY_{2k}(X) = \begin{cases} 1, \ if \ x_1 = x_2 \ \& \ x_3 = x_4 \ \& \ x_5 = x_6 \ \& \ ... \ \& \ x_{2k-1} = x_{2k} \\ 0, \ otherwise \end{cases}$$

2.2 Classical Decision Trees

The classical version of the query model is known as decision trees [1]. A black box contains the input $X = (x_1, x_2, ..., x_N)$ and can be accessed by querying x_i values. The algorithm must be able to determine the value of a function correctly for arbitrary input. The complexity of the algorithm is measured by the number of queries on the worst-case input. For more details, see the survey by Buhrman and de Wolf [1].

Definition 2. *[1] The deterministic complexity of a function f, denoted by D(f), is the maximum number of questions that must be asked on any input by a deterministic algorithm for f.*

Definition 3. *[1] The sensitivity $s_x(f)$ of f on input $X = (x_1, x_2, ..., x_N)$ is the number of variables x_i with the following property: $f(x_1, .., x_i, .., x_N) \neq f(x_1, .., 1 - x_i, .., x_N)$. The sensitivity of f is $s(f) = max_x s_x(f)$.*

It has been proved that $D(f) \geq s(f)$ [1].

Theorem 1. $D(VERIFY_N) = N$.

Proof. Check function sensitivity on any accepting input, for instance, on $X =$ 1111..11. Inversion of any bit will invert the function value, because a pair of bits with different values will appear. $s(VERIFY_N) = N \Rightarrow D(VERIFY_N) = N$.

2.3 Quantum Computing

This section briefly outlines the basic notions of quantum computing that are necessary to define the computational model used in this paper. For more details, see the textbooks by Nielsen and Chuang [3] and Kaye et al. [4].

An n-dimensional quantum pure state is a unit vector in a Hilbert space. Let $|0\rangle, |1\rangle, ..., |n-1\rangle$ be an orthonormal basis for $\mathbb{C}^n$. Then, any state can be expressed as $|\psi\rangle = \sum_{i=0}^{n-1} a_i |i\rangle$ for some $a_i \in \mathbb{C}$. Since the norm of $|\psi\rangle$ is 1, we

have $\sum_{i=0}^{n-1} |a_i|^2 = 1$. States $|0\rangle, |1\rangle, ..., |n-1\rangle$ are called *basis states*. Any state of the form $\sum_{i=0}^{n-1} a_i |i\rangle$ is called a *superposition* of basis states. The coefficient a_i is called an *amplitude* of $|i\rangle$. The state of a system can be changed by applying *unitary transformation*. Unitary transformation U is a linear transformation on $\mathbb{C}^n$ that maps vector of unit norm to vector of unit norm. The *transpose* of a $m \times n$ matrix A is the $n \times m$ matrix $A^T_{i,j} = A_{j,i}$ for $1 \leq i \leq n$, $1 \leq j \leq n$. We denote the *tensor product* of two matrices by $A \otimes B$.

The simplest case of quantum *measurement* is used in our model. It is the full measurement in the computation basis. Performing this measurement on a state $|\psi\rangle = a_0 |0\rangle + ... + a_{n-1} |n-1\rangle$ gives the outcome i with probability $|a_i|^2$. The measurement changes the state of the system to $|i\rangle$ and destroys the original state.

2.4 Quantum Query Model

The quantum query model is the quantum counterpart of decision trees and is intended for computing Boolean functions. For a detailed description, see the survey by Ambainis [6] and textbooks by Kaye, Laflamme, Mosca [4] and de Wolf [2]. A quantum computation with T queries is a sequence of unitary transformations:

$$U_0 \rightarrow Q_0 \rightarrow U_1 \rightarrow Q_1 \rightarrow ... \rightarrow U_{T-1} \rightarrow Q_{T-1} \rightarrow U_T.$$

U_i's can be arbitrary unitary transformations that do not depend on the input bits. Q_i's are query transformations. A computation starts in the initial state $\left|\vec{0}\right\rangle$. Then we apply $U_0, Q_0, ..., Q_{T-1}, U_T$ and measure the final state.

We use the following definition of query transformation: if an input is a state $|\psi\rangle = \sum_i a_i |i\rangle$, then an output is $|\phi\rangle = \sum_i (-1)^{x_{k_i}} a_i |i\rangle$, where we can arbitrarily choose a variable assignment x_{k_i} for each basis state $|i\rangle$.

Each quantum basis state corresponds to the algorithm's output. We assign a value of a function to each output. The probability of obtaining result $j \in \{0, 1\}$ after executing an algorithm on an input X equals the sum of squared modulus of all amplitudes, which correspond to outputs with value j.

Definition 4. *[1] A quantum query algorithm computes f exactly if the output equals f(x) with a probability p=1, for all* $x \in \{0, 1\}$. *The complexity is denoted by* $Q_E(f)$.

3 Computing $VERIFY_N$ in a Quantum Query Model

In this section, we present the results of designing an exact quantum query algorithm for Boolean function $VERIFY_N(X)$. We start from the case of four variables and then show how to extend the algorithm to verify N-bit codewords. We have used a combinatorial approach to determine the structure of the algorithm, and have used *Mathematica* [14] software to verify its correctness. In our approach, we have tried to employ the full power of quantum parallelism, also known as computing in a superposition.

3.1 Exact Quantum Query Algorithm for $VERIFY_4$

To familiarize the reader with the quantum query model and to build a base for extension, we demonstrate an algorithm for verification of 4-bit codewords. The algorithm flow is presented in Fig. 1.

Theorem 2. *There exists an exact quantum query algorithm Q1 that computes the Boolean function $VERIFY_4(X)$ using two queries: $Q_E(Q1) = 2$.*

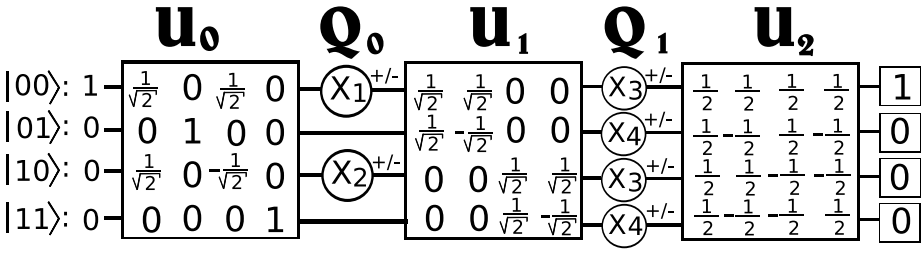

Fig. 1. Exact quantum query algorithm Q1 for computing $VERIFY_4$

The algorithm uses a 2-qubit quantum system. Each horizontal line corresponds to the amplitude of the basis state. Computation starts with amplitude distribution $|START\rangle = (1,0,0,0)^T$. Three large rectangles correspond to the 4×4 unitary matrices U_0, U_1 and U_2. Two vertical layers of circles specify the queried variable order for queries Q_0 and Q_1. Finally, four small squares at the end of each horizontal line define the assigned function value for each basis state.

We demonstrate an example of computational flow for accepting input X=1100:

$$|final\rangle = U_2Q_1U_1Q_0U_0\,(1,0,0,0)^T = U_2Q_1U_1Q_0\left(\tfrac{1}{\sqrt{2}},0,\tfrac{1}{\sqrt{2}},0\right)^T =$$
$$= U_2Q_1U_1\left(-\tfrac{1}{\sqrt{2}},0,-\tfrac{1}{\sqrt{2}},0\right)^T = U_2Q_1\left(-\tfrac{1}{2},-\tfrac{1}{2},-\tfrac{1}{2},-\tfrac{1}{2}\right)^T =$$
$$= U_2\left(-\tfrac{1}{2},-\tfrac{1}{2},-\tfrac{1}{2},-\tfrac{1}{2}\right)^T = \textbf{(-1, 0, 0, 0)} \overset{Measure}{\Rightarrow} [ACCEPT : f(1100) = 1]$$

3.2 Exact Quantum Query Algorithm for $VERIFY_N$

This section describes a generalized algorithm for computing the Boolean function $VERIFY_N$. In the previous section, we demonstrated in detail the first algorithm in the sequence. Now, we will show how to extend this approach to verify codewords of length N.

Theorem 3. *The Boolean function $VERIFY_N(X)$ can be computed by an exact quantum query algorithm using N/2 queries: $Q_E(VERIFY_N) = N/2$.*

We introduce an algorithm that will construct all required transformation matrices for a specified N. Then obtained transformations must be applied to the initial state in a specified order.

The algorithm is described in Table 1. The algorithm was implemented using *Mathematica* software, and its correctness was verified by a computer program.

Table 1. Exact quantum query algorithm for computing $VERIFY_N$

1. Setup
Boolean function to be computed: $VERIFY_N = (x_1, x_2, ..., x_N)$. Number of queries: $T = N/2$. Number of qubits: T. Number of amplitudes (dimension of Hilbert space): $K = 2^T = 2^{N/2}$.
2. Algorithm structure construction

FOR (i=1 to T) {
STEP 1: Calculate a set of indices:
$\|IND\| = 2^i; IND = \{ind_1, ind_2, ..., ind_{2^i}\};$
$IND = \left\{j \cdot \frac{K}{2^i} + 1 \middle\| j \in \left\{0, 1, ..., (2^i - 1)\right\}\right\}$
STEP 2: Construct matrices U_i and Q_i :
Initialize U_i with the identity matrix $U_i = I_K$
Initialize Q_i with the identity matrix $Q_i = I_K$
index=1;
WHILE(index < 2^i) {
t1=IND[index] // $index^{th}$ element from the set IND
t2=IND[index+1]
Replace elements of U_i and Q_i:
$U_{t1,t1} = U_{t1,t2} = U_{t2,t1} = \frac{1}{\sqrt{2}}$
$U_{t2,t2} = -\frac{1}{\sqrt{2}}$
$Q_{t1,t1} = (-1)^{X_{2i-1}}$
$Q_{t2,t2} = (-1)^{X_{2i}}$
index = index + 2;
}
}
STEP 3: Final transformation - $U_{FINAL} = H^{\otimes T}$, where $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$.
STEP 4: Initial state - $\|START\rangle = (1, 0, 0, ..., 0)^T$
STEP 5: Measurement - the only accepting state is $\left\|\vec{0}\right\rangle = \|000...0\rangle$.

3. Algorithm application
Execute the algorithm on input X by applying a constructed unitary and query transformations in the following order: $\|START\rangle \rightarrow U_1 \rightarrow Q_1 \rightarrow ... \rightarrow Q_T \rightarrow U_T \rightarrow U_{FINAL} \rightarrow [Measure]$.

3.3 Algorithm Analysis

To improve intuition and understanding, general algorithm for verification of N-bit codeword can be visualized as an abstract tree (see Fig. 2). We start at the top with state vector that has exactly one amplitude initialized to a=1 .

Queries and unitary transformations are formed and combined in such a way, that if values of function variables are equal by pairs, then in the final state

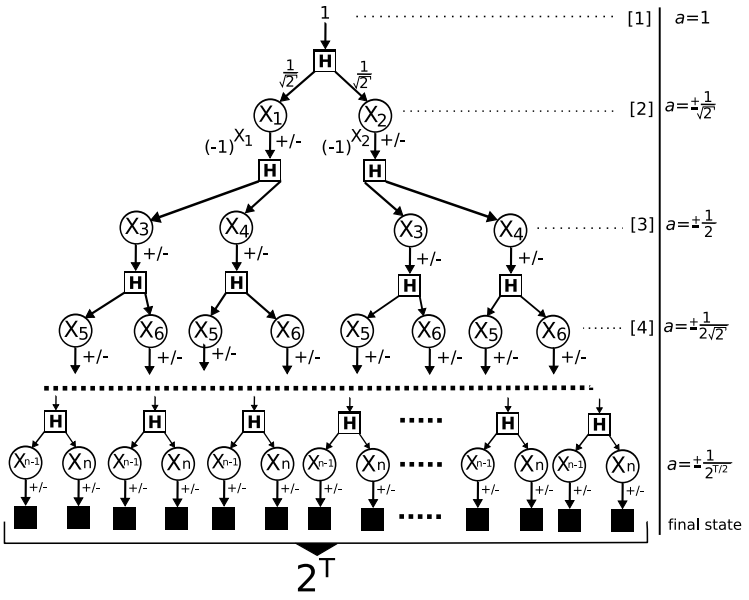

Fig. 2. Visualization of the quantum query algorithm as an abstract tree

vector signs of all amplitudes will be identical. At the same time, the first row of matrix U_{FINAL} consists of equal elements $+\frac{1}{2^{T/2}}$. It means that application of U_{FINAL} will join together all amplitudes and results in the state vector with $a = 1$ in the first position. So, the measurement will output the state $\left|\overrightarrow{0}\right\rangle$ with 100% probability. This is the accepting state $\Rightarrow VERIFY_N(X) = 1$.

If algorithm is executed on rejecting input, i.e., there is at least one pair of variables with different values, then after all T queries number of $+\frac{1}{2^{T/2}}$ and $-\frac{1}{2^{T/2}}$ amplitudes in state vector will be equal. This is provided by the algorithm structure. After multiplication with U_{FINAL} the value of the first amplitude will be zero, so there is no probability to obtain $\left|\overrightarrow{0}\right\rangle$ state after the measurement.

4 Application for a String Equality Problem

Described quantum algorithm can be adapted for solving such computational problem as testing if two binary strings are equal. This is a well-known task, which can be used as a subroutine in various algorithms.

Quantum algorithm for the Boolean function $VERIFY_N$ checks whether variables are equal by pairs, i.e., $x_1 = x_2$ & $x_3 = x_4$ & $x_5 = x_6$ & ... & $x_{2k-1} = x_{2k}$. On the other hand, we can consider that our algorithm is checking whether two binary strings, $Y = x_1x_3x_5...x_{2k-1}$ and $Z = x_2x_4x_6...x_{2k}$, are equal. Therefore, presented quantum algorithm can be easily used not only to verify repetition codes, but also for checking the equality of binary strings.

5 Conclusion

In this paper, we investigated the verification of error detection codes. We have represented the verification procedure as an application of a query algorithm to an input codeword contained in a black box. We have presented an exact quantum query algorithm, which allows verifying a codeword of length N using only $N/2$ queries to the black box. This algorithm saves exactly half the number of queries comparing to the classical case. This result repeats the largest difference between classical deterministic and quantum exact algorithm complexity for a total Boolean function known today in this model.

We see many possibilities for future research in the area of quantum query algorithm design. The most significant open question still remains: is it possible to increase exact algorithm performance more than two times using quantum tools? We believe that it may be possible. Next, there are many computational tasks waiting for efficient solution in a quantum setting. Regarding the verification of repetition codes, we would like to be able to verify not only (2,1) code, but also an arbitrary (r, n) code. Another fundamental goal is to develop a framework for building efficient ad-hoc quantum query algorithms for arbitrary Boolean functions.

References

1. H. Buhrman and R. de Wolf: Complexity Measures and Decision Tree Complexity: A Survey. Theoretical Computer Science, v. 288(1): 21-43 (2002).
2. R. de Wolf: Quantum Computing and Communication Complexity. University of Amsterdam (2001).
3. M. Nielsen, I. Chuang: Quantum Computation and Quantum Information. Cambridge University Press (2000).
4. P.Kaye, R.Laflamme, M.Mosca: An Introduction to Quantum Computing. Oxford (2007).
5. A.Ambainis: Quantum query algorithms and lower bounds (survey article). In Proceedings of FOTFS III, Trends on Logic, vol. 23 (2004), pp. 15-32.
6. A.Ambainis and R. de Wolf: Average-case quantum query complexity. Journal of Physics A 34, pp. 6741-6754 (2001).
7. A.Ambainis: Polynomial degree vs. quantum query complexity. Journal of Computer and System Sciences 72, pp. 220-238 (2006).
8. T. M. Cover and J. A. Thomas: Elements of Information Theory. pp. 209-212, Wiley-Interscience, (1991).
9. P. W. Shor: Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing, 26(5):1484-1509 (1997).
10. L. Grover: A fast quantum mechanical algorithm for database search. In Proceedings of 28th STOC'96, pp. 212. -219 (1996).
11. A.Ambainis. Personal communication, April 2009.
12. D. Deutsch and R. Jozsa: Rapid solutions of problems by quantum computation. In Proceedings of the Royal Society of London, volume A 439, pp. 553-558 (1992).
13. R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca: Quantum algorithms revisited. In Proceedings of the Royal Society of London, volume A 454, pp. 339-354 (1998).
14. Wolfram Research, Mathematica, http://www.wolfram.com/