Quantum Circuit Complexity[1]

Andrew Chi-Chih Yao

Department of Computer Science
Princeton University
Princeton, New Jersey 08544

Abstract

We study a complexity model of quantum circuits analogous to the standard (acyclic) Boolean circuit model. It is shown that any function computable in polynomial time by a quantum Turing machine has a polynomial-size quantum circuit. This result also enables us to construct a universal quantum computer which can simulate, with a polynomial factor slowdown, a broader class of quantum machines than that considered by Bernstein and Vazirani [BV93], thus answering an open question raised in [BV93]. We also develop a theory of quantum communication complexity, and use it as a tool to prove that the majority function does not have a linear-size quantum formula.

Keywords. Boolean circuit complexity, communication complexity, quantum communication complexity, quantum computation

AMS subject classifications. 68Q05, 68Q15

[1]This research was supported in part by the National Science Foundation under grant CCR-9301430.

1 Introduction

One of the most intriguing questions in computation theroy (see e.g. Feynman [Fe82]) is whether computing devices based on quantum theory can perform computations faster than the standard Turing machines. Deutsch proposed a Turing-like model [De85] for quantum computations, and constructed a universal quantum computer that can simulate any given quantum machine (but with a possible exponential slowdown). He subsequently considered a network-like model, called *quantum computational networks*, and established some of their basic properties [De89]. His discussions, however, centered mostly on the computability issue without regard to the *complexity* (i.e. cost) issue.

A significant step towards better understanding the complexity issue in the quantum Turing model was taken by Bernstein and Vazirani [BV93], who constructed an *efficient* universal quantum computer which can simulate a large class of quantum Turing machines with only a polynomial factor slowdown. In classical computation, Boolean circuit complexity has provided an important alternative framework than Turing complexity. It is thus of interest to develop an analogous quantum model to address the question whether quantum devices can perform computations faster than the classical Boolean devices.

A natural place to start is the framework of quantum computational networks as discussed in [De89]; these networks may be viewed as the quantum analog of conventional logical circuits (with feedback). In this paper, we single out the subclass of acyclic networks, and develop a complexity theory of quantum circuits analogous to the standard (acyclic) Boolean circuit model. We show that any function computable in polynomial time by a quantum Turing machine has a polynomial-size quantum circuit. This result, somewhat unexpectedly, also allows us to construct a universal quantum computer which can simulate, with a polynomial factor slowdown, a broader class of quantum machines than that considered by Bernstein and Vazirani [BV93], thus answering an open question raised in [BV93]. We also develop a theory of quantum communication complexity, and use it as a tool to prove that the majority function does not have a linear-size quantum formula.

For other developments on quantum complexity, see Berthiaume and Brassard [BB92] and Jozsa [Jo91]. Qquantum effects have also been studied in the context of cryptographic protocols by Wiesner, Bennett, Brassard, Crépeau, and others; for more information on this subject, see [Br93] for an up-to-date survey and the references in the recent paper [BCJL93]. For work in quantum systems from the perspective of information theory, see for example, Kholevo [Kh73] and Schumacher [Schu90].

2 Quantum Boolean Circuits

In Deutsch [De89], a quantum computation model different from that of quantum Turing machines was introduced. This is the quantum analog to the classical *sequential logical circuits*. In essence, some set of *elementary gates* is chosen as a *basis*, where each elementary gate is some ℓ-input ℓ-output device specified by a $2^\ell \times 2^\ell$ unitary matrix U. The function of the gate needs to be understood in the context of quantum computation (see [De89]). We summarize it briefly. Let $\mathbf{C}^d$ denote the vector space of d-tuples of complex numbers, equipped with an inner product $< u, v >= \sum_{1 \le i \le d} u_i^* v_i$ for $u, v \in \mathbf{C}^d$. The *length* of a vector u is given by $(< u, u >)^{1/2}$. We say that u, v are *orthogonal* if $< u, v >= 0$. Let $d = 2^\ell$. Identify each of the d natural unit vectors (those with a single 1 in one component and 0 in all other components) with one of the elements in $\{0,1\}^\ell$. The matrix U transforms any vector $u \in \mathbf{C}^d$ into another vector u' as follows. For an input $\alpha = \sum_{\tilde{x} \in \{0,1\}^\ell} c_{\tilde{x}}\ \tilde{x}$, the output is given by $\beta = \sum_{\tilde{x} \in \{0,1\}^\ell} c_{\tilde{x}}\ U_{\tilde{x},\tilde{y}}\ \tilde{y}$. In the above formulas, $\tilde{x}, \tilde{y}$ are interpreted as unit vectors in $\mathbf{C}^d$ (and not as an ℓ-tuple of numbers), and multiplications (by constants) and summations are with respect to operations in the vector space $\mathbf{C}^d$. By definition, a unitary matrix transforms mutually orthogonal unit vectors into mutually orthogonal unit vectors.

A *computational network* is composed of elementary gates connected together by wires, with suitably chosen time delays as in the classical sequential circuits. The network has a set of external input wires and output wires. A computation is carried out by setting some of the input wires to variables, repetitions allowed, $x_1, x_2, \cdots, x_n$ (the rest set to constants $0, 1$), and designate some of the output wires as containing the output variables $y_1, y_2, \cdots, y_m$ to be sampled at a specified time. We will not give a detailed illustration of how such networks function, since we are mainly interested in a restricted class of networks which are analogs of acyclic Boolean circuits. ¿From now on, by *circuits* we mean acyclic circuits.

Let $\mathbf{\Phi}_m$ denote the set of all m-input m-output quantum gates. Deutsch showed [De89] that, for $n \ge 3$, any unitary transformation in $\mathbf{C}^{2^n}$ (as induced by n Boolean variables) can be computed by a computational network using $\mathbf{\Phi}_3$ as a basis, and with only n wires (initially each wire contains one distinct input variable). It turns out that one can show that the feedback loops can be avoided (as in classical sequential circuits), but at the price of adding additional wires (called *dummy wires*) which are set to constants (0 or 1) initially and take on the same constant values again at the output end. Note that the same phenomenon arose in reversible computing networks for the classical Boolean computation (Toffoli [To81]).

Theorem 1 Let $n \geq 1$. Any unitary transformation in in $\mathbf{C}^{2^n}$ (as induced by n Boolean variables) can be computed by a quantum Boolean circuit using $2^{O(n)}$ elementary gates from $\mathbf{\Phi}_3$, and with $O(n)$ auxiliary wires.

We use $\mathbf{\Phi}_3$ as the basis, and consider quantum Boolean circuits built from these gates. Since the circuits are acyclic, we don't need to specify the delay time for various gates and wires. For any quantum Boolean circuit K, with input variables $x_1, x_2, \cdots, x_n$ and output variables $y_1, y_2, \cdots, y_m$ (which is a subset of output wires), we associate with each input $\tilde{x} \in \{0,1\}^n$ a probability distribution $\rho_{\tilde{x}}$ over $\{0,1\}^m$. The probability is defined in the normal way for quantum computations. For input $\tilde{x}$, write the final quantum state v corresponding to all the output wires (not just the output variables y_i) as $v = \sum_{\tilde{y} \in \{0,1\}^m} v_{\tilde{y}}$, where $v_{\tilde{y}}$ is the projection of v when the output variables are set to the values $\tilde{y}$. Then $\rho_{\tilde{x}}(\tilde{y})$ is equal to the square of the length $\|v_{\tilde{y}}\|^2$. We say that $\{\rho_{\tilde{x}} \,|\, \tilde{x} \in \{0,1\}^n\}$ is the *distribution generated* by K.

The case $m = 1$ is of special interest, in which case the distribution is specified by a real number $p_{\tilde{x}} = \rho_{\tilde{x}}(1)$ for each $\tilde{x} \in \{0,1\}^n$. We say that a string $\tilde{x} \in \{0,1\}^n$ is *accepted* by the circuit K if $p_{\tilde{x}} > 2/3$, and rejected by K if $p_{\tilde{x}} < 1/3$. If every $\tilde{x} \in \{0,1\}^n$ is either accepted or rejected, we say that K *computes* the language $\{\tilde{x} \,|\, \tilde{x} \text{ is accepted by } K\}$.

The *size* of a quantum Boolean circuit is the number of elementary gates in the circuit, and the depth is the maximum length of any (directed) path from any input wire to any output wire. A circuit is a *formula* if every input wire is connected to a unique output variable y_i, and that the path connecting them is unique. Note that due to the unitary nature of quantum computation, the entire circuit cannot look like a forest. The definition here expresses the condition that, when one looks at only the part of circuit connected by directed paths to output variables, one sees a forest. (See Figure 1.)

For any language $L \subseteq \{0,1\}^n$, let $C_Q(L), D_Q(L)$ be the minimum circuit size, circuit depth for any quantum circuit computing L. Let $F_Q(L)$ be the minimum size of any quantum formula for computing L.

To illustrate how elementary gates from $\mathbf{\Phi}_3$ transform inputs, we consider an example. For each real number λ, let D_λ denote the 3-input 3-output elementary gate, with its associated unitary matrix given by

$$M_{a'b'c',abc} = \epsilon_{aa'}\epsilon_{bb'}[(1-ab)\epsilon_{cc'} + iabh_{cc'}],$$

where $\epsilon_{ij} = 1$ if $i = j$ and 0 otherwise, $h_{cc'} = \cos(\pi\lambda/2)$ if $c + c'$ is even and $-i\sin(\pi\lambda/2)$ if $c + c'$ is odd. (See Figure 2 for the matrix explicitly exihibited.) This family of gates was introduced by Deutsch [De89] as an extension of the Toffoli gates (Toffoli [To81]) for classical Boolean circuits. Just as Toffoli gates are complete for reversible (classical) Boolean circuits, Deutsch showed that the family D_λ are sufficient to implement all quantum connection networks in the sense that any single D_λ with λ irrational is universal in the sense that any computational network can be *approximated* by networks built from D_λ. For additional interesting members of $\mathbf{\Phi}_3$, see Deutsch [De89].

3 Relationships with Turing Machines

A quantum Boolean circuit K with n input variables is said to (n,t)-*simulate* a quantum Turing machine M, if the family of probability distributions $p_{\tilde{x}}$, $\tilde{x} \in \{0,1\}^n$ generated by K is identical to the distribution of the configuration of M after t steps with $\tilde{x}$ as input. For definiteness, the configuration is encoded as a list of the tape symbols from cell $-t$ to t, followed by the state and the position of the head, all naturally encoded as binary strings.

Theorem 2 Let M be a quantum Turing machine and n, t be positive integers. There exists a quantum Boolean circuit K of size $poly(n,t)$ that (n,t)-simulates Q.

Corollary If $L \in P$, then $C_Q(L_n) = O(n^k)$ for some fixed k. (L_n is the set of strings in L of length n.)

This is the quantum analog of the simulation of deterministic Turing machines by classical Boolean circuits (see Savage [Sa72], Schnorr [Schn76], Pippenger and Fischer [PF79]). The proof for the quantum version involves subtler arguments. A sketch of the main steps in the proof is given in Section 6.

4 A Universal Quantum Turing Machine

As noted in [BV93], the simulation of quantum machines is a nontrivial problem, and needs careful discussions even for the subclass of deterministic reversible machines (see [Be73] for discussions of such machines). In this section, we answer an open question about simulating quantum machines raised by Berstein and Vazirani [BV93]. In [BV93], it was shown that there is a univeral quantum Turing machine (with a polynomial slow-down) for the class of quantum Turing machines in which the read/write head must move either to the right or to the left at each step. It was asked whether there is a universal machine with only a polynomial slow-down when the head is not required to move. This is an interesting question, as it would be an uncomfortable situation in which one may produce quantum machines but cannot execute them as programs on a general computer efficiently, if the answer turns out to be negative. The next result gives a positive answer.

In this extended abstract, by quantum Turing machines we mean one-tape machines with its head allowed to move in each quantum step either to the right, or to the left, or stay in the same place. (For a formal specification, see [BV93].) Theorems 1 and 2 can be extended to the standard variations of this model. (This will be discussed in the complete paper.)

Theorem 3 There exists a universal quantum Turing machine that can simulate any given quantum Turing machine with only a polynomial slow-down.

The proof of Theorem 3 uses Theorem 2. Basically, the universal machine first constructs a quantum circuit K to simulate the given Turing machine, then follows the circuit diagram deterministically and uses quantum steps to simulate computation of successive elementary gates. One complication is that since a universal machine has only a finite set of transitions, one needs to perform approximate computations in the same way as was done in [BV93]. We omit the details in this extended abstract.

5 Quantum Communication Complexity

Interacting quantum machines can be defined in several ways. We will only introduce a special model here which can be used to prove lower bounds on circuit complexity. An *interacting pair* (M_1, M_2) of quantum Boolean circuits is a partition of a quantum Boolean circuit such that M_1 and M_2 have disjoint sets of input variables, and all the output variables are contained in one side. The *communication cost* of (M_1, M_2) is the number of wires passing between M_1 and M_2.

Analogous to the standard notion of communication complexity (see [Ya79]), the *quantum communication complexity* of a function $f(\tilde{x}, \tilde{y})$ is defined to be the minimum communication cost of any interacting pair of quantum Boolean circuit for computing f with $\tilde{x}$, $\tilde{y}$ being the respective inputs to M_1, M_2. It is possible to generalize this concept to other models, such as the multi-party case with shared variables (Chandra, Furst and Lipton [CFL83]) and the communication complexity for relations (Karchmer and Wigderson [KW90]). One can also define quantum communication complexity with no error allowed, or with quantum help bits, etc. We discuss these matters further in the complete paper.

The determination of communication complexity is more difficult in the quantum case, we discuss here only one result here. It will be applied to prove a lower bound result about quantum formula size.

Let $\tilde{x} = (x_1, x_2, \cdots, x_n)$, $\tilde{y} = (y_1, y_2, \cdots, y_n)$ be n-tuples of Boolean variables. Let $f(\tilde{x}, \tilde{y}) = 1$ if there are at least n 1's among the $2n$ arguments, and 0 otherwise.

Theorem 4 The quantum communication complexity of f is $\geq \Omega(\log\log n)$.

A proof of Theorem 4 is given in Section 7. Let MAJ_n be the majority function of n variables. We show that MAJ_n have no linear-size quantum formula.

Theorem 5 $F_Q(MAJ_n)/n \to \infty$.

To prove Theorem 5, we reduce the problem to one of communication complexity (using a Ramsey-type argument similar to those used by Hodes and Specker [HS68]), and then apply Theorem 4. We omit the proofs here.

6 Proof of Theorem 2

Let M be a quantum Turing machine with alphabet set Σ, set of states Q, and transitional coeffiecients $\delta(q, a, \tau, q', a')$ with $\tau \in \{\leftarrow, \circ, \rightarrow\}$; the symbols $\leftarrow, \rightarrow, \circ$ are interpreted as moving to the left, to the right, and staying stationary. As is in the notation of [BV93], δ is the amplitude of M to change state to q', print a' and move according to τ, if the machine is currently in state q and reading tape symbol a.

We construct a quantum circuit which is the concatenation of T identical subcircuits. Each subcircuit, denoted by K, performs one step of the simulation.

The encoding for the configuration can be chosen differently from the one specified in Section 3. As long as it is polynomial-time equivalent to the required format, one can add an encoding and decoding unit to the front and back ends of the solution to obtain the required final network.

For our solution, we use $\ell = O(2 + \lceil \log_2(|Q| + 1) \rceil + \lceil \log_2 |\Sigma| \rceil)$ wires for each of the $2t + 1$ cells (numbered from 0 to $2t$ instead of from $-t$ to t). The current values of the wires for cell i will be denoted by s_i, q_i, a_i, where $s_i \in \{0, 1, 2, 3\}$ (two wires), $q_i \in Q \cup \{\emptyset\}$ ($\lceil \log_2(|Q| + 1) \rceil$ wires) and $a_i \in \Sigma$ ($\lceil \log_2 |\Sigma| \rceil$ wires). The variable s_i takes on value 0 when the head is not at cell i, value 1 when the head is at cell i and has not been actively invoked in the simulation, and 2 when the head has been used in the simulation and is now at cell i.

The subcircuit K is constructed as follows. The basic building block is a circuit G with 3ℓ wires. We build K by cascading $2t-1$ units of G, each shifting right by ℓ wires, and at the end, adding a circuit I whose function is to change all s_i with values 2 to 1 and 1 to 2. Denote the i-th unit of G by G_i. (See Figure 3.)

Clearly, I is unitary, and can be constructed with $O(t)$ elementary gates. We now describe how to construct the unitary G.

The central idea is as follows. Think of G as having 3ℓ inputs describing the contents of three consecutive cells (including the information whether the head is there). We want G to transform the contents of these cells if the head is at the middle cell and the simulated step has not occurred (i.e. $s_i = 1$ if cell i is the middle cell), according to how the simulated machine would transform the contents. The obvious first try for designing G would be to let G do nothing when $s_i \neq 1$. This would not work since some linear combinations of configurations with $s_i \neq 1$ can lead to the same output as when $s_i = 1$, and G would not be unitary. The idea is for G to leave all the *realizable* linear combinations of configurations with $s_i \neq 1$ untouched, but allowed to alter the values of wires for situations that do not arise in any computation. This turns out to give enough freedom for a unitary G to exist (and constructible).

Let us formalize the above conditions. We write down the conditions for the $i+1$st unit G (with wires from cells $i-1, i, i+1$). Let H denote the subspace of $\mathbf{C}^{2^{3\ell}}$ spanned by three types of vectors:

(i) $|s_{i-1}, q_{i-1}, a_{i-1}, s_i, q_i, a_i, s_{i+1}, q_{i+1}, a_{i+1} >$
where $s_i \neq 1$ and none of s_{i-1}, s_i, s_{i+1} is equal to 2;

(ii) $v_{q_{i-1},a_{i-1},a_i,a_{i+1}}$ for all possible values of these parameters, where

$$v_{q_{i-1},a_{i-1},a_i,a_{i+1}} =$$
$$\sum_{q',a'} \delta(q_{i-1}, a_{i-1}, \circ, q', a')|2, q', a', 0, \emptyset, a_i, 0, \emptyset, a_{i+1} > +$$
$$\sum_{q',a'} \delta(q_{i-1}, a_{i-1}, \rightarrow, q', a')|0, \emptyset, a', 2, q', a_i, 0, \emptyset, a_{i+1} > ;$$

(iii) $u_{q_{i-2},a_{i-2},a',a_{i-1},a_i}$ for all possible values of these parameters, where

$$u_{q_{i-2},a_{i-2},a',a_{i-1},a_i} =$$
$$\sum_{q'} \delta(q_{i-2}, a_{i-2}, \rightarrow, q', a')|2, q', a_{i-1}, 0, \emptyset, a_i, 0, \emptyset, a_{i+1} > .$$

Type (i) vectors and their linear combinations are distinct from, and in fact orthogonal to, any possible resulted vector when the Turing machine takes a step with head at cell i. Type (ii) vectors are vectors resulted when the Turing machine takes a step with head at cell $i-1$ and, afterwards, with the head resting at cell $i-1$ or i. Type (iii) vectors are vectors resulted when the Turing machine takes a step with head at cell $i-2$ and with the head resting at cell $i-1$. ¿From the viewpoint of G, the only input configurations are linear combinations of two kinds of vectors: those with $s_i = 1$, and those from H. Clearly, these two kinds of vectors are orthogonal. The next lemma states the crucial property that, for an input w with $s_i = 1$ (a vector of the former kind), the execution of one step of the simulated Turing machine takes w to w' which will still be orthogonal to H. Write w as $|0, \emptyset, a_{i-1}, 1, q_i, a_i, 0, \emptyset, a_{i+1} >$.

Lemma 1 For all possible values of $a_{i-1}, q_i, a_i, a_{i+1}$, the following vectors are mutually orthogonal unit vectors and are orthogonal to the subspace H:

$$\sum_{q',a'} \delta(q_i, a_i, \leftarrow, q', a')|2, q', a_{i-1}, 0, \emptyset, a', 0, \emptyset, a_{i+1} >$$
$$+ \sum_{q',a'} \delta(q_i, a_i, \circ, q', a')|0, \emptyset, a_{i-1}, 2, q', a', 0, \emptyset, a_{i+1} >$$
$$+ \sum_{q',a'} \delta(q_i, a_i, \rightarrow, q', a')|0, \emptyset, a_{i-1}, 0, \emptyset, a', 2, q', a_{i+1} > .$$

Proof By a careful check of the unitarity constraints on the quantum Turing machine M. Details omitted from this abstract. □

We put the following requirements on G:

(a) For each $v \in H$, $G(v) = v$.

(b) $G|0, \emptyset, a_{i-1}, 1, q_i, a_i, 0, \emptyset, a_{i+1} >=$
$$\sum_{q',a'} \delta(q_i, a_i, \leftarrow, q', a')|2, q', a_{i-1}, 0, \emptyset, a', 0, \emptyset, a_{i+1} >$$
$$+ \sum_{q',a'} \delta(q_i, a_i, \circ, q', a')|0, \emptyset, a_{i-1}, 2, q', a', 0, \emptyset, a_{i+1} >$$
$$+ \sum_{q',a'} \delta(q_i, a_i, \rightarrow, q', a')|0, \emptyset, a_{i-1}, 0, \emptyset, a', 2, q', a_{i+1} > .$$

Lemma 2 There exists a unitary G satisfying the above requirements. Furthermore, the matrix entries of G are rational functions of entries of transitional coefficients of the simulated Turing machine.

Proof The requirements state that all the vectors in the subspace H remain fixed by G, and that a set of unit vectors mutually orthogonal and orthogonal to H are transformed by G into unit vectors that are mutually orthogonal and orthogonal to H. Such G exists and can be found by solving a set of linear equations. □

By Theorem 1, G can be implemented as a quantum Boolean circuit using $2^{O(\ell)}$ elementary gates. We have thus specified how K is built as an $O(t2^{O(\ell)})$-size quantum Boolean circuit. It remains to prove that K correctly simulates one step of the operation of the given quantum Turing machine M.

It suffices to prove that K correctly simulates one step of M when the head is at cell i for $1 \leq i \leq 2t-1$. For each Turing machine (pure) configuration ψ, let $\eta(\psi)$ denote the corresponding unit vector $|s_0, q_0, a_0, s_1, q_1, a_1, \cdots, s_{2t}, q_{2t}, a_{2t} > \; \in \mathbf{C}^{(2t+1)\ell}$.

Let ψ_0 be any (pure) configuration of M with head at some cell $1 \leq i \leq 2t-1$. Let $\psi_0 \rightarrow \sum_\psi c_\psi \psi$ after one step of execution by M. We show that, for input $\eta(\psi_0)$ to K, the output of K is $\sum_\psi c_\psi \eta(\psi)$.

Let $\eta(\psi_0) = k_0, k_1, \cdots, k_{2t-1}$, where k_i is the vector in $\mathbf{C}^{2^{(2t+1)\ell}}$ corresponding to the wire values in K after G_i has just been passed by. We would like to show that k_{2t-1} is essentially equal to $\sum_\psi c_\psi \eta(\psi)$ (except that the values of s_j would be 2 when they should be 1).

Clearly, for $j = 1, 2, \cdots, i-1$, the 3-cell segments input to G_j belongs to H (in fact type (i)), and hence no modifications of wire values take place. Thus, $k_j = k_0$ for $0 \leq j \leq i-1$. At G_i, since $s_i = 1$, k_i is obtained from k_{i-1} according to item (b) in the requirements for G (see the paragraph before Lemma 2). This is almost $\sum_\psi c_\psi \eta(\psi)$, except that the values of s_j would be 2 when they should be 1. We only need to show that this vector remains the same through the rest of the G units (i.e. $G_{i+1}, \cdots, G_{2t-1}$).

At G_{i+1}, we can calculate k_{i+1} as follows. Write $k_i = k_i' + k_i''$, where k_i' is the portion with the head at cell $i-1$ and k_i'' is the portion with the head at cell i and $i+1$. We can examine how G_{i+1} modifies k_i' and k_i'' separately and add the resulted vectors. It is easy to see that the 3-cell segment of k_i' input to G_{i+1} is a vector in H (in fact a linear combination of vectors of type (i)), and hence k_i' will not be changed by G_{i+1}. It is also easy to see that the 3-cell segment of k_i'' input to G_{i+2} is a vector in H of type (ii), and hence k_i'' will also not be changed by G_{i+1}. We conclude the $k_{i+1} = k_i$. A similar argument shows that G_{i+2} does not change its input in any way and hence $k_{i+2} = k_{i+1} = k_i$.

Note that k_i is a linear combination of vectors of the form $|s_0, q_0, a_0, s_1, q_1, a_1, \cdots, s_{2t}, q_{2t}, a_{2t}) > \in \mathbf{C}^{2^{(2t+1)\ell}}$ with $s_j = 2$ for some $j \in \{i-1, i, i+1\}$ and all other $s_r = 0$. It follows that, by induction, each G_j ($j > i+2$) sees only 3-cell segments belonging to H (type (i) vectors), and hence $k_j = k_i$. This completes the proof of Theorem 2.

7 Proof of Theorem 4

Let (M_1, M_2) be a pair of interacting quantum Boolean circuit that computes f with error probability less than 1/3. We will show that $t \geq \Omega(\log\log n)$, where t is the number of wires crossing between M_1 and M_2.

Without loss of generality, we can assume that the t cross wires go alternately from one machine to the other, with the first wire being from M_1 to M_2, and the last from M_2 to M_1. The last wire carries the result of the computation, with the answer being 1 if the the wire is in state $|1>$. Let $\Delta = \{|0>, |1>\}$. By definition, Δ is a computational basis for the signal space of every wire, and the Hilbert space of the circuit is the direct product of these signal spaces.

Let M_1 and M_2 contain $k+1$ and ℓ wires, respectively. The Hilbert space of the circuit can be regarded as the direct product of three Hilbert spaces H_1, H_2, and H_3, where H_1 and H_2 come from the wires in M_1 and M_2, and H_3 is the signal space of one wire which goes from M_1 to M_2 and back t times. Clearly, H_1 and H_2 have dimensions 2^k and 2^ℓ, and H_3 has dimension 2.

Let $\vec{e} = (e_1, e_2, \cdots, e_t) \in \Delta^t$, where e_i denotes the state of the i-th cross wire. For any input $x \in \{0,1\}^n$ to M_1, let $a_{x,\vec{e}} \in H_1$ be the output state of M_1 obtained from the input state as follows: at cross wire # 1, project the current state $s'_0 \in H_1 \times H_3$ to $s_1 \in H_1$ by restricting the component of s'_0 in H_3 to e_1; then at cross wire # 2, with s_1 having evolved within M_1 to state s'_1, force the # 2 cross wire state to be e_2, i.e. make the state of the circuit on the M_1 side $s_2 = s'_1 \otimes e_2$; following the circuit to the point of cross wire # 3, project the current state $s'_2 \in H_1 \times H_3$ (s_2 having evolved into s'_2) to $s_3 \in H_1$ by restricting the component of s'_2 in H_3 to e_3; $\cdots$, etc. In a similar way, for any $y \in \{0,1\}^n$, let $b_{y,\vec{e}} \in H_2$ be the output state of M_2 obtained by the circuit from input y.

It is clear that, for input (x, y) to the circuit (M_1, M_2), the output state is equal to

$$\sum_{\vec{e}=(e_1,\cdots,e_t)\in\Delta^t} a_{x,\vec{e}} \otimes b_{y,\vec{e}} \otimes e_t.$$

Thus, the probability of the circuit accepting input (x, y) is

$$p_{x,y} = \|\sum_{\vec{e}\in E} a_{x,\vec{e}} \otimes b_{y,\vec{e}}\|^2,$$

where $E = \Delta^{t-1} \times \{|1>\}$.

The idea of the proof is to show that, if t is not large enough, then there will be two $y, y' \in \{0,1\}^n$ with different number of 1's in them, say n_1 and n_2, but with *similar* features in $b_{y,\vec{e}}, b_{y',\vec{e}}$ such that $p_{x,y} \approx p_{x,y'}$ for all x. This leads to a contradiction if we select an x with its number of 1's between $n - n_1$ and $n - n_2$, since the circuit should accept exactly one of the pairs (x, y), (x, y'). We now make it precise.

For every $e, e' \in E$, let $\hat{a}_{x,e,e'} =< a_{x,e}, a_{x,e'} >$, and $\hat{b}_{y,e,e'} =< b_{y,e}, b_{y,e'} >$.

Lemma 3 $p_{x,y} = \sum_{e,e' \in E} \hat{a}_{x,e,e'} \hat{b}_{y,e,e'}$ for all x, y.

Proof Omitted. □

For each y, define the *feature vector* of y by

$$v_y = ((m, m') \,|\, e, e' \in E),$$

where $m = \lfloor Re(\hat{b}_{y,e,e'})(\log_2 n)^3 \rfloor$ and $m' = \lfloor Im(\hat{b}_{y,e,e'})(\log_2 n)^3 \rfloor$. Clearly, there are at most $(((\log_2 n)^3) + 1)^{2E^2}$ distinct possible feature vectors.

Assume that $t < (\log_2 \log_2 n - \log_2 \log_2 \log_2 n - 10)/2$. We will derive a contradiction (for large n). Clearly, $E = 2^{t-1} < (\log_2 n / 20 \log_2 \log_2 n)^{1/2}$. Thus, there are at most $((\log_2 n)^3 + 1)^{2E^2} < n$ different feature vectors. It follows that there are two y, y' with different number of 1's, say $n_1 > n_2$, but with $v_y = v_{y'}$.

Using Lemma 3, we have for any x

$$\begin{aligned} |p_{x,y} - p_{x,y'}| &\leq \sum_{e,e' \in E} \hat{a}_{x,e,e'} |\hat{b}_{y,e,e'} - \hat{b}_{y',e,e'}| \\ &\leq \sum_{e,e' \in E} \hat{a}_{x,e,e'} 2(\log_2 n)^{-3} \\ &\leq 2E^2/(\log_2 n)^3 \\ &\leq (\log_2 n)^{-1}. \end{aligned}$$

Let $x \in \{0,1\}^n$ be a string with its number of 1's being in the interval $[n - n_1, n - n_2]$. Then one of $p_{x,y}, p_{x,y'}$ should be less than 1/3 and the other greater than 2/3, since exactly one of the pairs (x, y), (x, y') is accepted by the circuit. This is a contradiction. This proves the theorem.

8 Conclusions

We have initiated a study of Boolean circuit and communication complexity in the quantum computation context. It is hoped that this line of investigation leads to interesting new mathematical questions, and perhaps sheds light on other aspects of quantum computation such as the quantum Turing machine model. The results presented here seem to be encouraging.

References

[Be73] C. Bennett, "Logical reversibility of computation," *IBM J. Res. Develop.* **17** (1973), 525-532.

[BB92] A. Berthiaume and G. Brassard, "The quantum challenge to structural complexity theory," *Proceedings of 7th IEEE Conference on Structure in Complexity Theory*, 1992.

[BV93] E. Bernstein and U. Vazirani, "Quantum complexity theory," *Proceedings of 1993 ACM Symposium on Theory of Computing*, 1993.

[Br93] G. Brassard, "Cryptology column - quantum cryptology: a bibliography," *Sigact News*, vol. 24, no. 3, 1993, 16-20.

[BCJL93] G. Brassard, C. Crépeau, R. Jozsa, and D. Langlois, "A quantum bit commitment scheme provably unbreakable by both parties," *Proceedings of 1993 IEEE Symposium on Foundations of Computer Science*, 1993.

[CFL83] A. Chandra, M. Furst, and R. Lipton, "Multi-party protocols," *Proceedings of 1983 ACM Symposium on Theory of Computing* (1983), 94-99.

[De85] D. Deutsch, "Quantum theory, the Church-Turing principle and the universal quantum computer," *Proceedings of the Royal Society of London*, Volume **A400** (1985), 97-117.

[De89] D. Deutsch, "Quantum computational networks," *Proceedings of the Royal Society of London*, Volume **A425** (1989), 73-90.

[Fe82] R. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics* **21** (1982), 467-488.

[HS68] L. Hodes and E. Specker, "Lengths of formulas and elimination of quantifiers I," in *Contributions to Mathematical Logic*, edited by H. Schmidt, K Schutte and H. Thiele, North-Holland (1968), 175-188.

[Jo91] R. Jozsa, "Characterizing classes of functions computable by quantum parallelism," *Proceedings of the Royal Society of London* **A435** (1991), 563-574.

[KW90] M. Karchmer and A. Wigderson, "Monotone circuits for connectivity require super-logarithmic depth," *SIAM Journal on Discrete Mathematics* **3** (1990), 255-265.

[Kh73] A. Kholevo, "Bounds for the quantity of information transmitted by a quantum communication channel," *Problemy Peredachi Informatsii* **9** (1973), 3-11. English translation of the journal by IEEE under the title *Problems of Information Transfer*.

[PF79] N. Pippenger and M. Fischer, "Relations among complexity measures," *Journal of ACM* **26** (1979), 361-381.

[Sa72] J. Savage, "Computational work and time on finite functions," *Journal of ACM* **19** (1972), 660-674.

[Schn76] C. Schnorr, "The network complexity and Turing machine complexity of finite functions," *Acta Informatica* **7** (1976), 95-107.

[Schu90] B. Schumacher, "Information from quantum measurements," in *Complexity, Entropy, and the Physics of Information*, Santa Fe Institute Studies in the Sciences of Complexity, Volume VIII, edited by W. Zurek, Addison-Wesley, 1990, 29-38.

[To81] T. Toffoli, "Bicontinuous extensions of invertible combinatorial functions," *Mathematical Systems Theory* **14** (1981), 13-23.

[Ya79] A. Yao, "Some questions on the complexity of distributive computing," *Proc. 1979 STOC*, 1979, 209-213.