二分決定グラフにおける量子計算能力と通信計算量の関連

佐々木勇也 [1]

東京大学情報理工学研究科コンピュータ科学専攻

概要

近年、量子・古典計算モデル間計算力差は計算量理論上の重要な問題となり、多数の量子・古典間計算力差が知られている。しかし、各計算力差間の関係は明らかになっていない。
本研究では、既に量子-古典間差が知られている通信計算問題を、古典的計算モデルである OBDD における問題に変換し、 OBDD 上の量子-古典間計算量差の存在を考察した。
結果として、多パーティ間通信計算理論由来の関数を対象に誤りなしの場合において、古典的には k パーティに対して $\log k - 1$ 以上の幅が要求される一方、幅 2 の量子 OBDD でその関数を計算するものが存在することを示した。
この結果は、量子計算においても OBDD の容量計算量と、通信計算量との間に何らかの関連が存在することを示唆するものである。

Sasaki Yuuya

On the relation of quantum OBDD size and communication complexity

Department of Computer Science, The University of Tokyo

Abstract

In recent years, one of the important problems on complexity theory is quantum-classical computational gap and many quantum-classical gaps have been found. However, how these results relate each other remains still to be unknown.
In this paper, we investigate the quantum-classical OBDD size complexity gap through a function which is already known to have quantum-classical communication complexity gap.
Consequently, by targeting a function from multi-party communication theory, we obtain: there exists a 2-width bound quantum OBDD with zero error such that classical OBDDs with zero error need at least $\log k - 1$ width where k is the number of parties.
This result suggests that there is also some relation between OBDD size and communication complexity in quantum case.

1 Introduction

With rapid increase in research of Quantum Computing, many examples of computational power gap has been found between quantum computational model and their classical counterpart. For instance, there are well known and important results such as Shor's factoring, discrete log[12] and Grover's quantum searching[4]. The above examples are all on time complexity, there are also other

results including Raz's 2-party quantum communication protocol[10] on communication complexity and Kondacs and Watrous' 2-way quantum finite state automaton which accepts non-regular language[6]. For more detail of quantum computations, we suggest seeing Gruska[5].

In this paper, we will focus on branching programs or more restricted model, Ordered Binary Decision Diagrams, which are another classical model obtained from extension of decision tree. There are some good reasons why branching programs are promising models of computation :

1. Some standard complexity classes like the class **L**(the class which a deterministic Turing Machine can recognize on log-space restriction) can even be directly characterized in terms of branching programs.

2. Their combinatorial structure and computation process are very simple.

3. It is easy to define reasonably-restricted branching program variants to classify specific problems.

4. If there is a Branching Program which computes some function, then there is a polynomial time bounded algorithm that makes a circuit which computes the same function.

It was shown by Nakanishi, Hamaguchi and Kashiwabara in [8] that there is computational power gap between bounded-width ordered quantum Branching Program and its counterpart in terms of bounded error.

However, it has been unknown how those computational power gap on time, communication, automata or branching program width complexity relate to each other.

On the other hand, there is a fact that the size of a Ordered Binary Decision Diagram which computes some function relates the communication complexity of that function closely (Babai,Nisan and Szegedy [1] seem to be the first who have explicitly used this relation to show lower bounds for branching programs). And also Buhrman, van Dam, Høyer and Tapp[2] proved that quantum multi-party communication protocol can achieve asymptotically smaller communication complexity in terms of zero error than that of classical case.

On the basis of those results, in this paper, we show another example of computational power gap between bounded-width quantum OBDD and its counterpart with zero-error. Specifically, we prove :

1. For any k, n $n \geq \log k$, there is 2-width bound quantum OBDD that computes $f : \{0,1\}^{nk} \to \{0,1\}, f(x_1, x_2, \ldots, x_k)$ =the n-th least significant bit of the sum of the x_i's with zero error.

2. There is no classical OBDD that computes same f with zero error, width smaller than k.

We prove 1 by simply constructing a 2-width bound quantum OBDD. In order to prove 2, we apply a technique from transformation of OBDD to multi-party communication protocol. Because the transformation relates directly the size (width) of the OBDD to the amount of information bits, the lower bound on the size of the OBDD is proved.

2 Branching Programs and OBDDs

Branching Program is another non-uniform computational model,which generalize the concept of decision trees to decision graphs.

2.1 Branching Program and its variations

2.1.1 Deterministic Branching Program

Definition 1 A *branching program* (bp) on the variable set $\{x_1, x_2, \ldots x_n\}$ is directed acyclic graph with one source and sinks labeled by the constants 0 or 1 respectively. Each non-sink node is labeled by a variable x_i and has exactly two outgoing edges labeled by 0 or 1,respectively.

This graph represents a Boolean function $f : \{0,1\}^n \to \{0,1\}$ in the following way. To compute $f(a)$ for some input $a \in \{0,1\}^n$, start at the source node. For a non-sink node labeled by x_i check the value of this variable and follow the edge which is labeled by this value (this is

called "test of variable x_i "). Iterate this until a sink node is reached. The value of f on input a is uniquely determined and is called the *computational path for* a.

The *size* of a branching program G is the number of its nodes and is denoted by $|G|$.

The *depth* of a branching Program is the maximum length (number of edges) of a path from the source to one of the sinks.

The *width* of a branching program is defined as follows : for a node v of a branching program G, define $l(v)$ as the number of edges on the longest path from the source to v. If for each node v of G all paths from the source to v are of the same length, then we call G is *synchronous*.

For $i \geq 0$, let the *i-th level of* G be the set of all nodes v with $l(v) = i$. The *width* of a branching program is the maximum taken over the size of its levels.

2.1.2 Randomized Branching Program

A randomized branching program is probabilistic computational model defined as analogue to probabilistic circuit.

Definition 2 Let a deterministic branching program G with the following special properties be given:

1. G is defined on two disjoint sets of variables $X = \{x_1, \ldots, x_n\}$ and $Z = \{z_1, \ldots, z_r\}$
2. on each path from the source to a sink, each variable from Z is tested at most once.

By an obvious extension of the usual semantics for deterministic branching programs, G represents a function $g : \{0,1\}^n \times \{0,1\}^r \to \{0,1\}$.

We call G a *randomized branching program* if for all assignments $a = (a_1, \ldots, a_n) \in \{0,1\}^n$ to the variable in X it either holds that:

$$\Pr\{g(a,b) = 1\} > 1/2 or \Pr\{g(a,b) = 0\} > 1/2$$

where $b = (b_1, \ldots, b_r)$ is an assignment to the variables in Z chosen randomly according to the uniform distribution from $\{0,1\}^r$. We say that G as a randomized branching program represents the function $f : \{0,1\}^n \to \{0,1\}$ defined by

$$f(a) := \begin{cases} 1, & \text{if } \Pr\{g(a,b) = 1\} > 1/2 \\ 0, & \text{if } \Pr\{g(a,b) = 0\} > 1/2 \end{cases}$$

2.1.3 Probabilistic Branching Program

There is also known another probabilistic computational model based on branching program,called simply *probabilistic branching program.*

The randomized branching program has surely very simple structure and so has many good properties. It is difficult, however, to define a quantum computational model as its extension. Therefore, we need a new classical branching program with probabilistic computation process, equivalent for quantum branching model.

Nakanishi, Hamaguchi and Kashiwabara first introduced this model in [8] for comparability to quantum branching program.

Definition 3(Nakanishi et al.) A probabilistic branching program is acyclic directed graph with one source in-degree 0 node and two sink nodes labeled by 0 or 1 respectively. The *internal node* (non-sink node) is labeled by a variable x_i and has two types of outgoing edges, called 0-edges or 1-edges respectively. Each edge e has a weight on it $w(e)$ $(0 \leq w(e) \leq 1)$. let $E_0(v)$ and $E_1(v)$ be the set of 0-edges and the set of 1-edges of outgoing from node v. The weight on edges from each node satisfies following local probability condition:

$$\sum_{e \in E_0} w(e) = 1; \sum_{e \in E_1} w(e) = 1.$$

The computation of probabilistic branching program is as follows: Starting at the source, the value of labeled variable is tested at reached node. If this is 0(1) then an edge in $E_0(v)$ $(E_1(v))$ is chosen according to the probability distribution given as the weight of the

edges. The node pointed by the edge is activated as next node.

2.1.4 Quantum Branching Program

Finally, we introduce the focus of this work.

Definition 4 (Nakanishi et al.) A *quantum branching program* is an extension of probabilistic branching program, and its structure is same as a probabilistic branching program except for edge weights. The weight of each edge has a complex number $w(e)$ $(0 \leq ||w(e)||^2 \leq 1)$. The weight on edges from each node satisfies following local probability condition:

$$\sum_{e \in E_0} ||w(e)||^2 = 1; \sum_{e \in E_1} ||w(e)||^2 = 1.$$

And also satisfies the global probability condition (see below).

The edge weight represents the amplitude with which, currently in the node v, the edge will be followed in the next step.

Nodes of quantum branching program G are divided into the three sets of *accepting set* (Q_{acc}or, Q_1), *the rejecting set* (Q_{rej}or, Q_0) and *the non-halting set* (Q_{non}). The configurations of P are identified with the nodes in $Q = (Q_{acc} \cup Q_{rej} \cup Q_{non})$. A superposition of a G is any element of $l_2(Q)$. For each $q \in Q$, $|q\rangle$ denotes the unit vector that takes value 1 at q and 0 elsewhere.

Let define a *transition function* $\delta : (Q \times \{0,1\} \times Q) \to \mathbf{C}$ as follows:

$$\delta(v, a, v') = w(e)$$

where $w(e)$ is the weight of the a-edge ($a = 0 or 1$) from node v to v'. If the a-edge does not exist, then $\delta(v, a, v') = 0$.

Definition of *time evolution operator* is as follows:

$$U^{\delta}_{\mathbf{x}}|v\rangle = \sum_{v' \in Q} \delta(v, \mathbf{x}(*v), v')|v'\rangle$$

where $\mathbf{x}$ denotes the input of G and $\mathbf{x}(v)$ denotes the assigned value in $\mathbf{x}$ to the labeled variable of the node v. If the time evolution operator is unitary, then we say that quantum branching program is well formed, i.e, the branching program is valid in terms of the quantum theory.

An observable $\boldsymbol{O}$ to be $E_{acc} \oplus E_{rej} \oplus E_{non}$,where

$$E_{acc} = span\{|v\rangle | v \in Q_{acc}\},$$

$$E_{rej} = span\{|v\rangle | v \in Q_{rej}\},$$

$$E_{non} = span\{|v\rangle | v \in Q_{non}\}.$$

The computation of quantum branching program is as follows: The initial state is the source node $|v_s\rangle$. At each step, the time evolution operator is applied to the state $|\psi_i\rangle$,that is,$|\psi_{i+1}\rangle = U^{\delta}_{\mathbf{x}}|\psi_i\rangle$. Next,$|\psi_{i+1}\rangle$ is observed with respect to $E_{acc} \oplus E_{rej} \oplus E_{non}$. Note that this observation causes the quantum state $|\psi_{i+1}\rangle$ to be projected onto the subspace compatible with the observation. Let the outcomes of an observation be "accept","reject" and "non-halting" corresponding to E_{acc} , E_{rej} and E_{non} respectively. Until "accept" or "reject" is observed, applying the time evolution operator and observation is repeated.

2.2 OBDD

Definition 5 An OBDD is a special branching program that satisfies following condition:

Let $k \in \mathbf{N}$. A *read k-times branching program* is a branching program with the restriction that on each path from the source to a sink each variable is allowed to appear at most k times as the label of a node.

Let π be a permutation of the set $\{1, 2, \ldots n\}$. A π-ordered branching program on the variable set $\{x_1, x_2, \ldots x_n\}$ is a branching program with the following additional *ordering restriction*: For each edge leading from a node labeled

by some variable x_i, to a node labeled x_j,it must hold that $\pi(i) \leq \pi(j)$. The permutation π is called *variable-ordering* of the branching program. We frequently will describe a variable ordering simply by an ordered list of the variables (e.g., "$x_1, x_2, \ldots x_n$" for $\pi = id$).

A *π-Ordered Binary Decision Diagram* (OBDD) is a read-once branching program with variable ordering π restriction.

3 Communication Complexity Approach

The main subject of communication complexity theory is the analysis of the simple communication game, 2-party k-round communication. In this section, we focus on its natural modification, k-party communication case.

3.1 Multi-party Communication Complexity

Definition 6 Let f be a k-variable Boolean function whose inputs are n-bit binary strings(that is, $f : \{0,1\}^{nk} \to \{0,1\}$). There are k parties, denoted by $P_1, \ldots P_k$, where party P_i holds input data $\boldsymbol{x}_i (i - 1, \ldots, k)$, and each party has unlimited computational power. Initially, party P_i only knows $\boldsymbol{x}_i$, so, to evaluate f, the parties have to communicate among each other. The communication is done by broadcasting classical bits, where, each time, a party broadcasts one bit to everybody, on the total cost of one bit of communication.

We are interested the minimum number of bits required to be broadcasted in the worst-case for every party to know the value of f. This number called the *communication complexity* of f and is denoted $C(f, k, n)$.

A *multi-party communication protocol* is an algorithm specifying which party is the next to communicate and determining the message which this party will broadcast given its input and the message received so far. We consider only *one round* case, that is , each party broadcast its message at most once (but may broadcast multiple bits at once).

3.2 Transformation of OBDD to Communication Protocol

OBDDs and communication protocol are closely related, especially a OBDD can be seen as a communication protocol. Consequently, the size lower-bound for the OBDD of f and communication complexity of same f is related.

To put this technique intuitively, the known proofs of lower bounds on the size of OBDDs are all based on the fact that a large amount of information has to be exchanged across a suitably chosen cut in the graph in order to evaluate the given function. Results from communication complexity theory are then used to get lower bounds on the necessary amount of information.

3.2.1 OBDD as Multi-party Communication Protocol

Lemma 1 Let G be a randomized OBDD which represents f with zero-error. Then it holds that

$$\lceil \log(width(G)) \rceil \geq \frac{1}{k} C(f, k, n).$$

Proof We construct a randomized protocol for f from G.

First, we partition the randomized OBDD G into k sets of nodes G_i such that:

1. edges run only from G_i to G_j with $i \leq j$.
2. G_i contains only random node or node which is labeled by $x_{i,l}$ for $1 \leq i \leq k, 1 \leq l \leq n$.

Then, we identify sets of nodes (cuts) which separate different G_i. For $i = 1, \ldots k$, define C_i as the set of source nodes of G_i. Naturally, C_i is also sink of G_{i-1}.

Now, we are ready to describe the protocol P for f. Each party P_i obtains x_i as inputs. All party use the graph G as an "oracle". Let $v_0 \in C_0$ be the source of G. Let $i \in \{1, \ldots, k\}$. The i-th party P_i follows a path in If P_i encounters probabilistic variable, it locally choses a value (0 or 1) at random and precedes to the corresponding successor. If P_i encounters a

node labeled by $X_{i,l}$, then P_i tests the variable and activates correct successor. P_i broadcasts the number of the node $v_i \in C_i$ reached in this way. So, P_i must broadcast $\lceil C_i \rceil$.

Then, P_{i+1} begins its computation.

Since G is a OBDD, $width(G) \geq max_{1 \leq i \leq k}\{C_i\}$.

On the other hand, the sum of broadcasts in protocol P is $\sum_i \lceil \log C_i \rceil \geq C(f,k,n)$. To minimize the maximum value of the set bounded by its sum, we must make each element of the set have same value.

Therefore, $\lceil \log(width(G)) \rceil \geq \frac{1}{k} C(f,k,n)$

4 Computational Gap between bw-Quantum-OBDD and its classical counterpart

In this section, we use the technique from previous section to show that bw-quantum OBDD with zero error can compute some function which randomized OBDD with zero error needs $\log k$ width. We define the function f_k that first introduced in Buhrman, van Dam, Høyer and Tapp[2]. It is known that this function has quantum-classical separation by logarithmic factor in terms of the number of parties.

4.1 Previous result from quantum multiparty communication theory

Definition 7 (Buhrman et al.) There are k parties, where party P_i obtains input data as $x_i \in \{0, \ldots, 2^{n-1}\}$. We say that an input $\boldsymbol{x} = (x_1, = ldots, x_k)$ is *valid* if it satisfies that

$$(\sum_{i=1}^{k} x_i) \bmod 2^{n-1} = 0.$$

Let $f_k : \{0, \ldots, 2^{n-1}\}^k \rightarrow \{0,1\}$ denote the Boolean function on the valid inputs defined by

$$f_k(\boldsymbol{x}) = \frac{1}{2^{n-1}}[(\sum_{i=1}^{k} x_i) \bmod 2^n].$$

The function f_k can be viewed as computing the n-th least significant bit of the sum of the x_i's.

Theorem 1 (Buhrman et al.)

$$C(f_k, k, n) = k \log k - k$$

for $n \geq \log k$.

4.2 Bw-Quantum-OBDD that computes f_k

Theorem 2 Quantum OBDD with zero error can compute the function f_k with 2-width bound.

Proof We construct a bw-quantum OBDD that computes f_k for any n, k. We define the set of node Q as follows:

$$Q := \{v_{source}, v_{sink0}, v_{sink1}\}$$

$$\cup \{v_{(i,j),down}, v_{(i,j),up} | 1 \leq i \leq k, 1 \leq j \leq n\}$$

$$\cup \{v_{(k+1,1),down}, v_{(k+1,1),up}\}$$

The labeled variable to the node $v_{(i,j),down}$ is $x_{i,j}$. All other nodes are dummy node which activate their successor(s) without testing.

We define the 0-set (Q_0) and the 1-set (Q_1) as follows:

$$Q_0 = \{v_0\}, Q_1 = \{v_1\}$$

.

The sets of edges E_d, E_1, E_0 are defined as follows:

$$E_d = \{(v_{source}, v_{(1,1),down}), (v_{source}, v_{(1,1),up}),$$

$$(v_{(i,j),up}, v_{(i,j+1),up}), (v_{(i,n),up}, v_{(i+1,1),up}),$$

$$(v_{(k+1,1),up}, v_{sink0}), (v_{(k+1,1),up}, v_{sink1}),$$

$$(v_{(k+1,1),down}, v_{sink0}), (v_{(k+1,1),down}, v_{sink1})$$

$$| 1 \leq i \leq k, 1 \leq j \leq n-1\},$$

$$E_0 = \{(v_{(i,j),up}, v_{(i,j+1),up}), (v_{(i,n),up}, v_{(i+1,1),up})$$

$$| 1 \leq i \leq k, 1 \leq j \leq n-1\}$$

$$E_1 = \{(v_{(i,j),up}, v_{(i,j+1),up}), (v_{(i,n),up}, v_{(i+1,1),up})$$

$$|1 \leq i \leq k, 1 \leq j \leq n-1\}$$

And, amplitude of edges are as follows:

$$a(v_{source}, v_{(1,1),down}) = a(v_{source}, v_{(1,1),up}) = \frac{1}{\sqrt{2}},$$

$$a(v_{(i,j),down}, v_{(i,j+1),down})$$

$$= a(v_{(i,n),down}, v_{(i+1,1),down}) =$$

$$\begin{cases} 1, & \text{if labeled variable is 0} \\ e^{\frac{2\pi\sqrt{-1}}{2^{n-1}}2^j}, & \text{if labeled variable is 1} \end{cases}$$

$$a(v_{(k+1,1),down}, v_{sink0}) = a(v_{(k+1,1),down}, v_{sink1})$$

$$= a(v_{(k+1,1),up}, v_{sink0}) = \frac{1}{\sqrt{2}},$$

$$a(v_{(k+1,1),up}, v_{sink1}) = -\frac{1}{\sqrt{2}}.$$

Others are all 1.

It is straight forward to see that this OBDD has 2-width and is well formed.

This OBDD computes the function following way:

1. First, it branches out in the state :

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|v_{(1,1),up}\rangle +$$

$$|v_{(1,1),down}\rangle).$$

2. Next, the down row is applied a phase changing transformation:

$$|v_{(i,1),down}\rangle \rightarrow e^{\frac{2\pi\sqrt{-1}}{2^n}\boldsymbol{x}_i}|v_{(i+1,1),down}\rangle.$$

3. Finally, after $nk+1$ step, it is in state

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|v_{(k+1,1),up}\rangle +$$

$$e^{\frac{2\pi\sqrt{-1}}{2^{n-1}}\sum_{i=1}^{k} x_i}|v_{(k+1,1),down}\rangle).$$

Since, $(x_1, \ldots, x_k)$ satisfies the condition $(\sum_{i=1}^{k} x_i) \bmod 2^{n-1} = 0$,

$$e^{\frac{2\pi\sqrt{-1}}{2^{n-1}}\sum_{i=1}^{k} x_i} = (-1)^{f(\boldsymbol{x})}.$$

Consequently,

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|v_{(k+1,1),up}\rangle +$$

$$(-1)^{f(\boldsymbol{x})}|v_{(k+1,1),down}\rangle)$$

In the next step, the resulting state is

$$|v_{sink\ f(\boldsymbol{x})}\rangle.$$

Therefore, correct $f(\boldsymbol{x})$ is obtained without error.

4.3 Width of Randomized-OBDD that computes f_k

Theorem 3 There is no bw-randomized OBDD with zero error.

Proof From Lemma 1 and Theorem 1 , $\lceil \log(width(G)) \rceil \geq \log k - 1$ where G is randomized OBDD that computes f_k with zero error.

5 Conclusion

At the end of this work, we summarize the achievement of the previous section and comment on some open problems.

We showed that from relation of classical multi-party communication protocol and classical OBDD, there exists bw-quantum OBDD with zero error which can compute the function no bw-classical OBDD cannot. Especially, to prove the upper bound, we found such quantum OBDD from ad hoc method.

The structure of this quantum OBDD is closely related to quantum multi-party protocol of Buhrman et al.

Therefore, this result suggests that there also exists some relation between multi-party protocols and OBDDs in quantum case.

However, the exact transformation of quantum multi-party protocol and quantum OBDD is still remain unknown.

On the other hand, their remains also open how quantum branching programs correspond to other quantum computational model such as quantum circuit or quantum (non-uniform) Turing Machine.

参考文献

[1] L.Babai, N.Nisan and M.Szegedy: Multi-party Protocols, Pseudorandom Generators for Logspace, and Time-Space Trade-offs, Journal of computer Science, 45: pp.204-232, 1992.

[2] H.Buhrman, W.van Dam, P.Høyer and A.Tapp: Multiparty Quantum Communication Complexity, quant-ph/ 9710054, 1997.

[3] D.A.Barrington: Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1, Journal of Computer and System Sciences, 38: pp.150-164, 1989.

[4] L.Grover: A fast quantum mechanical algorithm for database search, Proc. 28th Symp. on the Theory of Computing, pp.212-219, 1996.

[5] J.Gruska: Quantum Computing, 439pp., McGraw-Hill, 1999.

[6] A.Kondacs and J.Watrous:On the Power of Quantum Finite State Automata, Proc. 38th Symp. on Foundation of Computer Science, pp.66-75, 1997.

[7] C.Meinel: Modified branching programs and their computational power, LNCS 370, Springer-Verlag, Berlin, 1989.

[8] M.Nakanishi, K.Hamaguchi and T.Kashiwabara: Ordered Quantum Branching Programs Are More Powerful than Ordered Probabilistic Branching Programs under a Bounded-Width Restriction, COCOON 2000, LNCS 1858, pp.467-476, 2000.

[9] P.Pudlák and S.Žák: Space complexity of computations, Technical report, Univ.Prague, 1983.

[10] R.Raz: Exponential Separation of Quantum and Classical Communication Complexity, Proc. 31st Symp. on the Theory of Computing, pp.358-367, 1999.

[11] M.Sauerhoff: Complexity theoretical results for randomized branching programs, PH.D Theses, Univ. of Dortmund, Shaker, 1999
`http://ls2-www.infomatik.uni-dortmund.de/sauerhoff`

[12] P.Shor: Algorithms for quantum computation: discrete logarithms and factoring, Proc. 35th Symp. on Foundation of Computer Science, pp.124-134, 1994.