Nonuniform Quantum Complexity:
Bounded-error Quantum Polynomial-size Branching Programs and their Computational Power

Yuuya Sasaki
Email: y_sasaki@is.s.u-tokyo.ac.jp
Fax: +81-3-5800-6933

Department of Computer Science, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan

Abstract

In this paper, we investigate the complexity class of *Bounded-error Quantum Polynomial-size Branching Programs*(BQP-BP) in the case of *Measurement-Once*. Ablayev et al. have showed that in the case of *Bounded-width branching programs*, quantum models are equivalent to classical models up to *Polynomial-size, Bounded or Exact error bound*. However, because their technique requires exponential increase of branching program width, it is still unknown whether there is computational power gap between Quantum and Classical BPs when the width is not bounded. Based on the emulation by *nonuniform Probabilistic Turing machine*, we show that in the case of Bounded-error bound, the Bounded-error nonuniform Probabilistic Turing machine with Logarithmic-space bound can emulate the given quantum branching program with exponentially many computation time, therefore, BQP-$BP \subseteq BPL/\mathrm{poly}$.
Key words.: quantum computation, nonuniform complexity class, Branching Program, BQP-BP.

1 Introduction

In the classical settings, Branching Programs(BPs) are very useful data structure for representing Boolean functions with many applications such as hardware verification, model checking, and other CAD applications. But also, BPs can be seen as one of the nonuniform computational models. This option has useful feature that first, we can handle BPs rather easily from their simple structure and second, because of their natural relationships to Turing machine or Circuit models, we can incorporate results of BPs into other popular models easily.

Accordingly, it is interesting to answer the question whether there is computational power gap between quantumized BPs and classical BPs for we can expect that same relationships will stand for other quantumized models.

There are two different quantum branching program model, defined by Nakanishi et al. [4] and Ablayev et al. [1], separately. While the former is graph oriented, measurement-many model, the latter is based on matrix application and measurement-once model. Later, Ablayev et al. showed that in the case of *Bounded-width BPs*, quantum models are equivalent to Classical models up to *Polynomial-size, Bounded or Exact error bound* [2], i.e. Bw-BQP-$BP = Bw$-P-$BP = \mathrm{NC}^1$.

In this paper, we use the latter definition of quantum branching program to show that without bounded-width condition, BQP-BP is weaker than nonuniform Bounded-error Probabilistic Turing machine with Logarithmic-space bound, BPL/poly. This latter model is a bit stronger than the classical counterpart of quantum branching program, Polynomial-size Branching Programs, so in the case of unbounded-width, there is still possibility of computational power increse by quantumization.

2 Preliminaries

Throughout this work, we are concerned with *nonuniform* models of computation. A nonuniform model of computation describes a sequence of Boolean functions by a sequence of representations, one for each input length. Circuit and Boolean Formula are the most popular model of this type, however, we use the next two models of computation to show the main theorem.

Definition 1: A *nonuniform Turing machine* (also called *advice-taking Turing machine*) is a multitape Turing machine with two read-only tapes, input and advice tape and one ordinary working tape. On input $\mathbf{x} \in \{0,1\}^n$, this tape is automatically loaded with the "advice" $A(n)$ before computation begins, where $A : \mathbb{N} \to \{0,1\}^*$ is an arbitrary function.

By changing the mode of computation, we can easily obtain definitions of nonuniform Turing machines, such as *nonuniform randomized Turing machine*.

Definition 2 : If X is a class of languages defined in terms of resource-bounded Turing machines, then $X/P\mathit{oly}$ is the class of languages defined by nonuniform Turing machines with the same resource bound and an advice function $A : \mathbb{N} \to \{0,1\}^*$ with $A(n) = n^{\mathcal{O}(1)}$.

We can incorporate the complexity class result of ordinary uniform Turing machines to that of nonuniform Turing machines by the straightforward way. In the case of Unbounded-error Quantum Log-space bound Turing machine (QL/poly), Watrous showed that this model is equivalent to Unbounded-error Probbilistic Log-space bound Turing machine (PL/poly) through the simulation of quantum computation by random sampling [6], so same relationship stands also in the case of nonuniform models. We extended this unbounded-error case proof technique to bounded error case.

Remark 1 [6] : $QL/\mathrm{poly} = PL/\mathrm{poly}$

Definition 3 : A *branching program* (BP) on the input variable $\mathbf{x} = x_1 x_2 \ldots x_n$ 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 the path traversed i n this process 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.

Definition 4 : *P-BP* is the class of language recognized by family of polynomial size BPs, where for any $L \in$ *P-BP*, for any $n \in \mathbb{Z}$, there exists polynomial size branching program P_n that computes the characteristic function of $L_n = L \cap \{0,1\}^n$.

We can easily extend the above definition of BP by changing the mode of computation to another such as randomized computation. For example, we can obtain the complexity class *BPP-BP* defined as the class of language recognized by family of polynomial size randomized branching program with bounded-error, where randomized BP is ordinary BP with internal coin-flipping node. However, this is known that randomization does not increase the power of branching programs. For more detail of this well-known derandomization, see Sauerhoff [5].

Note that when we consider *the configuration graph* of nonuniform Turing machine, we obtain the branching program recognizes same Boolean function. Actually,

Theorem 1[Cobham/ Púdlak and Žák] :
P-BP = *L*/P*oly*

This is famous result of branching program complexity class, and for more detail, see monograph of Meinel [3].

Then, we introduce the quantum analogue of branching program. At first glance, this definition differs from that of classical case. Actually, we can assume without loss of generality, all branching programs are reversible, leveled (for any path, the level of that node is same) and oblivious (for any level, all nodes in the same level test same variable), up to a polynomial increase of size. Therefore, we assume that for any branching program, each level is represented by permutation over paths.

Definition 5 [1] : A measurement-once leveled oblivious quantum branching program is defined as follows. For the sake of simplicity, we call this branching program as quantum branching program.

On the condition of Width $w(n) = \mathcal{O}(poly(n))$, Depth $d(n) = \mathcal{O}(poly(n))$, error-bound $\epsilon > 0$, Quantum branching program Q computes n-input function $f_n : \{0,1\}^n \to \{0,1\}$is iff

$$Q = ((k_j, U_{0,j}, U_{1,j})_{j=1}^{d(n)}, |init\rangle, |acc\rangle),$$

where $U_{\sigma,j}(\sigma = 0,1)$ is unitary transform over $w(n)$-dimensional Hilbert Space $\mathcal{H}^{w(n)}$, $|init\rangle, |acc\rangle \in \mathcal{H}^{w(n)}$ and $poly(n)$ is polynomial to n. Then, for any input $\mathbf{x} \in \{0,1\}^n$

$$\begin{aligned} |\langle acc| \quad U_{x_{k_{d(n)}},d(n)} U_{x_{k_{d(n)-1}},d(n)-1} \cdots U_{x_{k_1},1} |init\rangle|^2 \\ \geq 1/2 + \epsilon, (f_n(\mathbf{x}) = 1) \\ \leq 1/2 - \epsilon, (f_n(\mathbf{x}) = 0) \end{aligned}$$

3 BQP-$BP \subseteq BPL/$poly

We already have seen that graph based classical branching program can be assumed as permutation matrix based model. Note that it stands vice versa.

Lemma 1 : Quantum branching program Q can be represented by directed acyclic graph consists of test, Walsh-Hadamard, $\pi/4-$Phase-shift and Measurement node.

Therefore, without loss of generarity, we can assume that for any input $\mathbf{x} \in \{0,1\}^n$, the amplitude of each computational path of Q is in the form of $\frac{a}{\sqrt{2}^b} e^{\frac{c\pi\sqrt{-1}}{4}}$ where $a, b \in \mathbb{Z}$ and $c \in \mathbb{Z}_8$.

However, we can strengthen the restriction on quantum branching programs more.

Lemma 2 : The computational power of quantum branching program is not weakened if we restrict the BP such that for any input $\mathbf{x}, \mathbf{x}' \in \{0,1\}$, the differnce between the execution of BP for input $\mathbf{x}$ and that for $\mathbf{x}'$ is only in their conditional phase-shifting factor.

From these two lemmas, we can imply that if there exists quantum branching program, we have a datastructure representing the amplitudes of the original quantum BP, which is based on switching Directed Acyclic Graphs.

Lemma 3 : From given quantum branching program, we can construct leveled Directed Acyclic Graphs (DAGs) with 1 source and 1 sink $G_{\frac{c\pi}{4}}(V, E; s, t)$ and edge label $l_1 : E \to \{1, \ldots, n\}, l_2 : E \to \{0,1\}$ for $c = 1, \ldots, 7$. These DAGs meet following condition:

For any input $\mathbf{x} \in \{0,1\}^n$,$c \in 1, \ldots, 7$, let $p := sv_1v_2 \ldots v_{d-1}t$ be a path between the source and the sink that satisfies for all $i, j = 1, \ldots, d-1$, $x_{l_1(v_iv_j)} = l_2(v_iv_j)$. Then, the total sum of success amplitudes on original quantum branching program is equal to $\sum_{c=1}^{7} a_c e^{\frac{k\pi\sqrt{-1}}{4}}$, where a_c is the total number of path p on DAG $G_{\frac{c\pi}{4}}(V, E; s, t)$.

From the above lemma, we prove:
Theorem 2 : If there exist quantum branching programs Qs that computes function f, then there exist bounded-error probabilistic Turing machine with Logarithmic-space bound that computes f on input $\mathbf{x} \in \{0,1\}^n$ where the advice is DAGs $G_{\frac{c\pi}{4}}(V, E; s, t)$ in Lemma 3. This Turing machine runs exponential time.

References

[1] F. Ablayev, A. Gainutdinova and M. Karpinski. On computational Power of quantum branching programs. *FCT 2001*, LNCS 2138, pp.59-70, 2001.

[2] F. Ablayev, C. Moore and C. Pollet. Quantum and Stochastic Branching Programs of Bounded Width. arXiv: quant-ph/0201139, 2002.

[3] C. Meinel. Modified Branching Programs and Their Computational Power. LNCS 370,1989.

[4] 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* LNCS 1858, pp.467-476, 2000.

[5] M. Sauerhoff. Complexity theoretical results for randomized branching programs. PhD.thesis,Univ. of Dortmund,`http://ls2-www.infomatik.uni-dortmund.de/sauerhoff`, 1999.

[6] J. Watrous. On Quantum and Classical Space-bounded Processes with Algebraic Transition Amplitudes. *FOCS*, pp.341-351, 1999.