Classical Simulation Complexity of Quantum Branching Programs (extended abstract)

Farid Ablayev[*]

Abstract

We present classical simulation techniques for measure once quantum branching programs.

For bounded error syntactic quantum branching program of width w that computes a function with error δ we present a classical deterministic branching program of the same length and width at most $(1 + 2/(1 - 2\delta))^{2w}$ that computes the same function.

Second technique is a classical stochastic simulation technique for bounded error and unbounded error quantum branching programs. Our result is that it is possible stochastically-classically simulate quantum branching programs with the same length and almost the same width, but we lost bounded error acceptance property.

1 Introduction

Complexity of classical simulation of quantum computations for different models of computations were investigated in numerous papers [3, 5, 8, 12, 9].

Branching programs are important model of computations, because of their natural relationships to machines models (Turing machines, automata) and Circuit models. Different restricted models of branching programs are widely used for hardware verification and in numerous CAD applications. In the paper we present two classical simulation techniques for measure once quantum branching programs.

2 Definitions and Results

We start with definition of branching programs according to [11] (we call it *constructive* definition). Then we give an algebraic definition of branching programs and present results of the paper.

Definition 1 *A* branching program *(BP) on the variable set $X = \{x_1, \dots, x_n\}$ is a finite directed acyclic graph with one source node and sink nodes partitioned into two sets –* Accept *and* Reject*. Each non-sink node is labeled by a variable x_i and has two outgoing edges labeled* 0 *and* 1 *respectively. An input σ is* accepted *if and only if it induces a chain of transitions leading to a node in* Accept*, and the set of such inputs is the language accepted by the program.*

A branching program is *oblivious* if the nodes can be partitioned into levels $V_1, \dots, V_\ell$ and a level $V_{\ell+1}$ such that the nodes in $V_{\ell+1}$ are the sink nodes, nodes in each level V_j with $j \leq \ell$ have outgoing edges only to nodes in the next level V_{j+1}, and all nodes in a given level V_j query the same bit σ_{i_j} of the input.

[*]Work done in part while visiting Max-Plank Institute for Mathematics Bonn in 2007 Email: `ablayev@ksu.ru`

Definition 2 *The* size $Size(P)$ *of a branching program* P *is the number of its non-sink nodes. The* length $Length(P)$ *of branching program* P *is the maximum length of a path from the source to one of the sinks. The* width $Width(P)$ *of oblivious branching program* P *is the number* $Width(P) = \max_j |V_j|$.

Clearly we have that length of branching program corresponds to time of computation and width on a level j corresponds to a space that can be used on the step j of computation.

In this paper we deal with polynomial size branching programs. Recall that arbitrary branching program can be transformed to oblivious branching program (see for example [4]) with only polynomial increasing the size. So without loss of generality we consider only oblivious branching programs in this paper.

Now we give a definition of a *linear branching program* based on oblivious model. This definition is a generalization of the definition of quantum branching program presented in [2]. Deterministic, stochastic, and quantum oblivious branching programs are particular cases of linear branching programs. The model we consider is the influence of papers [4] and [5].

Let $\mathbf{V}^k$ be a k-dimensional vector space. We use $|\mu\rangle$ and $\langle\mu|$ to denote column vectors and row vectors of $\mathbf{V}^k$, respectively, and $\langle\mu_1 \mid \mu_2\rangle$ denotes the complex inner product. We write μ when it is not important whether μ is a column or a row vector.

Definition 3 (Linear branching program) *A Linear Branching Program (LBP)* P *over* $\mathbf{V}^k$ *is defined as*

$$\mathcal{P} = \langle T, |\mu_0\rangle, \text{Accept}\rangle \ ,$$

where $T = (T_1, \ldots, T_\ell)$ *is a sequence (of length* ℓ*) of instructions. Each instruction* T_j *is a triple* $T_j = \{i_j, M_j(0), M_j(1)\}$*, where* i_j *determines a variable* x_{i_j} *tested on the step* j*,* $M_j(0)$ *and* $M_j(1)$ *are* k*-dimensional linear transformations of the vector space* $\mathbf{V}^k$*. Vectors* $|\mu\rangle \in \mathbf{V}^k$ *are called states (state vectors) of* P*,* $|\mu_0\rangle \in \mathbf{V}^k$ *is the initial state of* P*, and* Accept $\subseteq \{1, \ldots, k\}$ *is the accepting set.*

According to the definition 2 it is natural to define the $Width(P)$ *of linear BP as the* dimension *of state space* $\mathbf{V}^k$*. Further for LBP* P *it is natural do define its size as* $Size(P) = Width(P)Length(P)$*.*

We define a computation on P *with an input* $\sigma = \sigma_1 \ldots \sigma_n \in \{0,1\}^n$ *as follows:*

1. *A computation of* P *starts from the initial state* $|\mu_0\rangle$*;*
2. *The* j*'th step of computation of* P *applies instruction* T_j*: program* P *queries a variable* x_{i_j}*, and applies the transition matrix* $M_j(\sigma_{i_j})$ *to the current state* μ *to obtain the state* $\mu' = M_j(\sigma_{i_j})\mu$*;*
3. *The final state (i.e., the state after step* ℓ*) is*

$$|\mu(\sigma)\rangle = \prod_{j=\ell}^{1} M_j(\sigma_{i_j})|\mu_0\rangle \ .$$

Now oblivious deterministic, stochastic, and quantum branching programs can be presented as follows:

Deterministic branching programs. A *deterministic* branching program is a linear branching program over a vector space $\mathbf{R}^k$. A state μ of such a program is a Boolean vector with exactly one 1. The transition matrices M have exactly one 1 in each column.

Stochastic branching programs. The concept of deterministic branching programs naturally generalizes to stochastic branching programs (SBP), by letting μ be a probability distribution, and by letting the M_j be *stochastic* matrices, i.e., matrices with non-negative entries where each column sums to 1.

For a deterministic and stochastic branching program P, for an input $\sigma \in \{0,1\}^n$ we define the acceptance probability of σ as follows

$$\Pr^P(\sigma) = Pr(\mu(\sigma)) = \sum_{i \in \text{Accept}} |\langle i \mid \mu(\sigma)\rangle| = \|\Pi_{\text{Accept}}\mu(\sigma)\|_1 \quad . \tag{1}$$

Here $|i\rangle$ is the basis vector with support on the node i (unit vector with value 1 at i and 0 elsewhere), and Π_{Accept} is a projection operator on the *accepting subspace* $\text{span}\{|i\rangle : i \in \text{Accept}\}$.

Quantum branching programs. We define a *quantum* branching program (QBP) as a linear branching program over a Hilbert space $\mathbf{C}^k$. The μ for such a program are complex state vectors with $\|\mu\|_2 = 1$, and the M_j are complex-valued unitary matrices. For a quantum branching program P, for an input $\sigma \in \{0,1\}^n$ we define the acceptance probability of σ as follows

$$\Pr^P(\sigma) = Pr(\mu(\sigma)) = \sum_{i \in \text{Accept}} |\langle i \mid \mu(\sigma)\rangle|^2 = \|\Pi_{\text{Accept}}\mu(\sigma)\|_2^2 , \tag{2}$$

that is, the probability that if we measure $\mu(\sigma)$, we will observe it in the accepting subspace. Note that this is a "measure-once" model analogous to the model of quantum finite automata in [6], in which the system evolves unitarily except for a single measurement at the end.

Notice that in contrast to algebraic definition of quantum and stochastic BPs one can define these models in a (so called) *constructive* form. See for example book [11] for constructive definition of SBP and the paper [8] for constructive definition of QBP.

Acceptance criteria. We say that a LBP P computes a Boolean function f *with unbounded error* if $\Pr^P(\sigma) > 1/2$ if $f(\sigma) = 1$ and $\Pr^P(\sigma) \leq 1/2$ if $f(\sigma) = 0$. *We say that P computes f with threshold* $1/2$.

We say that a LBP P computes a Boolean function f *with bounded error* if there is some $\epsilon > 0$ such that $\Pr^P(\sigma) \geq 1/2 + \epsilon$ if $f(\sigma) = 1$ and $\Pr^P(\sigma) \leq 1/2 - \epsilon$ if $f(\sigma) = 0$. *We say that P computes f with error $\delta = 1/2 - \epsilon$* (with margin ϵ).

2.1 Deterministic Simulations of Stochastic and Quantum Branching Programs

Syntactic Stochastic and Quantum Programs For unbounded and bounded error stochastic and quantum branching programs we define two subsets $\mathcal{A}$ and $\mathcal{R}$ of the set $\mathcal{F}$ of sink state vectors (consistent and inconsistent) as follows. For unbounded error programs, we define

$$\mathcal{A} = \{\mu \in \mathcal{V}_{\ell+1} : \Pr(\mu) > 1/2\} \quad \text{and} \quad \mathcal{R} = \{\mu \in \mathcal{V}_{\ell+1} : \Pr(\mu) \leq 1/2\};$$

and for bounded error programs, we define

$$\mathcal{A} = \{\mu \in \mathcal{V}_{\ell+1} : \Pr(\mu) \geq 1/2 + \epsilon\} \quad \text{and} \quad \mathcal{R} = \{\mu \in \mathcal{V}_{\ell+1} : \Pr(\mu) \leq 1/2 - \epsilon\}$$

We call $\mathcal{A}$ and $\mathcal{R}$ the *accepting* and *rejecting* sets respectively.

Recall that $\mathcal{V}_{\ell+1}$ includes the final states reachable by all possible paths, both consistent and inconsistent. Then:

Definition 4 ([1]) *We call a stochastic or a quantum branching program* syntactic *if its accepting and rejecting set of state vectors form a partition of the set of sink states, i.e., if* $\mathcal{V}_{\ell+1} = \mathcal{A} \cup \mathcal{R}$.

Note that without the syntactic restriction, it might happen that $\mathcal{V}_{\ell+1} \neq \mathcal{A} \cup \mathcal{R}$, and that some inconsistent final state vector $\mu \in \mathcal{V}_{\ell+1}$ has the property that $1/2 - \epsilon < \Pr(\mu) < 1/2 + \epsilon$.

Theorem 1 (Deterministic Simulation Theorem) *Let function f be bounded error δ ($\delta \in (0, 1/2)$) computed by syntactic QBP P. Then there exists deterministic BP P' that computes f and has the following complexity characteristics: $Length(P') = Length(P)$ and*

$$Width(P') \leq \left(1 + \frac{2}{1 - 2\delta}\right)^{2Width(P)}.$$

Similarly, we can deterministically simulate classical stochastic BP.

Theorem 2 *If a function is computed with bounded error δ ($\delta \in (0, 1/2)$) by a width-w syntactic stochastic branching program, then it is also computed by a deterministic branching program of the same length, and width*

$$w' \leq \left(1 + \frac{2}{1 - 2\delta}\right)^{w}.$$

The construction of corresponding deterministic branching program is based on the following properties:

1. Quantum states are unit vectors (a set of quantum states form bounded set for $||\cdot||_2$ norm).
2. Unitary transformations of quantum states preserves a distance.
3. Bounded-error acceptance criteria.

Bounded-error acceptance criteria together with the properties 1 and 2 forms topological structure on the set of quantum states which leads to a desired deterministic branching program.

Notice that Theorem 1 and Theorem 2 imply that constant width quantum and stochastic branching programs can be simulated by a constant width deterministic branching programs and hence — by an NC^1 circuits. For more information and results see [1].

2.2 Stochastic Classical Simulation of Quantum Branching Programs

Theorem 3 (Stochastic Simulation Theorem) *Let function f be unbounded error (bounded error) computed by QBP Q. Then there exists SBP P that unbounded error computes f of the same length $Length(P) = Length(Q)$ and width $Width(P) = 4Width^2(Q) + 3$.*

The proof of the Theorem uses another technique which is a classical stochastic simulation technique for bounded-error quantum branching programs.Informally our result is that a resulting stochastic classical branching program is of the same length and almost the same width, but we lost bounded-error acceptance.

Our construction of classical stochastic branching program is based on:

1. Replacing complex matrices with real ones with dimension doubled and tensor product construction as a bridge between $||\cdot||_1$ norm and $||\cdot||_2$ norm. This construction gives new matrices with quadratic increase in dimension.

2. A Turakainen-type construction [10] to replace arbitrary real matrices with stochastic ones with "good properties" of the original ones.

Now we define probabilistic and quantum complexity classes based on branching programs as follows.

Definition 5 *Let BPP-BP and PP-BP be the classes of functions computable with bounded error and unbounded error respectively by stochastic branching program of polynomial size;*

Let BQP-BP and $PrQP$-BP be the classes of functions computable with bounded error and unbounded error respectively by quantum branching program of polynomial size.

Theorem 4

$PrQP$-$BP \subseteq PP$-BP

BQP-$BP \subseteq PP$-BP

Notice that Sasaki in [9] proved that BQP-$BP \subseteq BPL/poly$ where $BPL/poly$ is a class of languages accepted by Logarithmic-space bounded error nonuniform probabilistic Turing machines.

References

[1] F. Ablayev, C. Moore, and C. Pollett, Quantum and Stochastic Branching Programs of Bounded Width, *Electronic Colloquium on Computational Complexity*, TR02-013, 2002, available at `http://www.eccc.uni-trier.de/eccc/`

[2] F. Ablayev, A. Gainutdinova, and M. Karpinski. On computational Power of quantum branching programs. *Proc. FCT 2001*, Lecture Notes in Computer Science 2138: 59–70, 2001.

[3] L. Adleman, J. Demarrais, M. Huang. Quantum computability, SIAM J. on Computing. 26(5), 1997, 1524–1540.

[4] D. Barrington. Bounded-width polynomial branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences* 38(1): 150–164, 1989.

[5] L. Fortnow. One complexity theorist's view of quantum computing. *Theoretical Computer Science*, 2003, 292(3), 597–610.

[6] C. Moore, J. Crutchfield. Quantum Automata and Quantum Grammars. *Theoretical Computer Science,* 2000, 237, 275–306.

[7] A. Paz. Introduction to Probabilistic Automata. Academic-Press, 1971.

[8] M. Sauerhoff and D. Sieling. Quantum branching programs and space-bounded nonuniform quantum complexity. *ph/0403164, March 2004.*

[9] Y. Sasaki. Nonunifrom Quantum Complexity: Bounded-error quantum branching programs and their computational power. *Poster of Tokyo Office Quantum Computation Seminar*, 2002.

[10] P. Turakainen. Generalized automata and stochastic languages, *Proc. of AMS 21*, 1969, 303–309.

[11] I. Wegener. *Branching Programs and Binary Decision Diagrams.* SIAM Monographs on Discrete Mathematics and Applications. 2000.

[12] J. Watrous. On quantum and classical space bounded processes with algebraic transition amplitudes. *Proc. of FOCS 1999*, 341-351.