Quantum computation and complexity theory

M.N. Vyalyi (Dorodnitsyn Computing Centre of RAS, Moscow)

Lectures at Zhejiang University, August 2015 (draft version)

Contents

1 Classical and quantum complexity classes

Computational complexity theory studies the properties of resource-bounded computation. The main goal is to distinguish between 'easy' and 'hard' problems. Of course, one need formal definitions before studying the question by mathematical tools.

The most important computational resources are time and space. But there are other, non-standard, types of resources:

- Interaction
  - Oracle: knows the answer and trusted
  - Prover (provers): knows the answer but do not trusted, should present a proof.
- Laws of nature as a special sort of a computation resource
  - Probabilistic processes
  - Quantum processes

The last option is the subject of consideration in these lectures.

We briefly recall the basic definitions and facts of computational complexity theory.

1.1 Deterministic computation, computational problems and complexity classes

There are many models of computation. The basic model is Turing machines. For Turing machine time and space of computation are easily defined. Turing machine operates in steps and duration of each step is constant. So the number of steps is proportional to running time. Data in Turing machine are stored in cells of equal size. So computation of TM requires a space proportional to the number of cells occupied by data.

Remark 1. From physical point of view a possibility of storage discrete symbols are based on laws of quantum mechanics. In this sense, all computations are quantum!

In computational theory we are interested in complexity of a *computation problem*. The standard way to define a problem is to use *languages*.

Let A^* be a set of strings over an alphabet A. A subset $L \subseteq A^*$ is called a *language*. In the sequel, we assume $A = \{0, 1\}$ without loss of generality.

There is a natural computation problem related to a language L— *decision problem for the language* L: given a string x, to check $x \in L$.

In study probabilistic and quantum computational models more important are *promise problems*. Formally, a promise problem is two nonintersecting languages L_0 and L_1, $L_0 \cap L_1 = \varnothing$. The related computation problem is to check $x \in L_1$ provided $x \in L_0 \cup L_1$.

Informally, we take some properties of inputs for granted and we are interesting only in those instances of a problem that satisfy these properties.

Note that a Turing machine computes a function. For decision problems the function is the indicator function of language:

$$\chi_L(x) = \begin{cases} 1, & \text{if } x \in L, \\ 0, & \text{otherwise.} \end{cases}$$

For promise problems we require that function f computed by TM satisfies two conditions

$$f(x) = 1, \quad \text{if } x \in L_1,$$
$$f(x) = 0, \quad \text{if } x \in L_0.$$

The values of f on the rest of inputs are arbitrary.

From now on we identify TMs and deterministic algorithms. We say that an algorithm *verifies* $x \in L$ if it computes the indicator function of L.

A *complexity class* is a collection of computational problems that can be solved using specified resources.

We will use the worst-case complexity. It is the most standard way to define running time of an algorithm solving decision problem for a language. Let $T_M(x)$ be the number of steps that is made by TM M on the input x. The worst-case complexity of the algorithm represented by M is

$$t_M(n) = \max_{|x|=n} T_M(x).$$

We are interested in asymptotic behavior of $t_M(n)$. The very important case is polynomial growth $\text{poly}(n)$. By definition,

$$t(n) \in \text{poly}(n) \quad \text{if } \exists c_1, c_2 \in \mathbb{Z}_+ : t(n) \leq c_1 n^{c_2}.$$

The one of the most important complexity classes is the class P. It formalizes the notion of a problem that can be solved efficiently.

Definition 1. $L \in$ P if there is a deterministic algorithm that verifies $x \in L$ in polynomial time w.r.t. the input length.

A similar definition is made for promise problems.

The class of efficiently computed functions is called FP.

Definition 2. A function $f\colon A^* \to A^*$ is in FP if it can be computed in polynomial time.

Sometimes problems in P are called *tractable* and problems outside P are called *intractable*.

1.2 The class NP

The second important complexity class is NP. It captures many problems that are presumably intractable.

It is the simplest example of using interaction resource. There are two parties involved in computation: Verifier and Prover.

Prover is able to compute any function.

Verifier is able to compute function from P only.

Verifier does not trust Prover. Thus, Prover should present a *proof* (a *certificate*) to convince Verifier.

An algorithm running on an input and a certificate should satisfy two requirements.

Definition 3. $L \in$ NP if there exists $V(x, y) \in$ P and the polynomial $p(\cdot)$ such that

(completeness) if $x \in L$ then $V(x, y) = 1$ for some y such that $|y| \leq p(|x|)$;

(soundness) if $x \notin L$ then $V(x, y) = 0$ for all y .

It is clear from definition that P $\subseteq$ NP. We give several examples of NP problems that presumably lie outside P.

Definition 4 (3-SAT)**.** Input: a description of 3-CNF C, i.e. a conjunctions of disjunctions of literals (a literal is either a Boolean variable or the negation of a variable).

Question: decide whether C is satisfiable, i.e. $C(x) = 1$ for some assignment x of variables.

Definition 5 (3-COLOR)**.** Input: a graph $G(V, E)$.

Question: decide whether the graph G has a proper coloring in 3 colors. A proper 3-coloring is a function $f\colon V \to \{0, 1, 2\}$ such that $(uv) \in E$ implies $f(u) \neq f(v)$.

Definition 6 (Partition)**.** Input: a sequence of positive integers $a_1, \ldots, a_n$.

Question: decide whether there exists a balanced partition, i.e. a subset $S \subseteq [n]$ such that

$$\sum_{j \in S} a_j = \sum_{j \notin S} a_j.$$

In the last problem we assume that integers are presented in binary. It is easy exercise to construct an algorithm for the partition problem running in time $\mathrm{poly}(\sum_j a_j)$.

Definition 7. Let $\mathcal{C}$ be a complexity class. Then the class co-$\mathcal{C}$ consists of languages $\bar{L}$, where $L \in \mathcal{C}$.

The following proposition is quite obvious.

Proposition 1. P = co-P.

Definition of NP is asymmetric: completeness and soundness conditions differ. Thus it is not clear for a language $L \in$ co-NP whether it lies in NP. The question NP $\stackrel{?}{=}$ co-NP is open. It is widely believed that NP $\neq$ co-NP and P $\subset$ NP $\cap$ co-NP.

There are problems from NP $\cap$ co-NP that considered as computationally hard. One of these problems is very important to the field of quantum computation.

Definition 8 (Prime factorization)**.** Input: an integer n.

Output: the prime factorization of n.

Prime factorization is not a decision problem. To convert it into a decision problem we use the standard way.

Let $f\colon \{0,1\}^* \to \{0,1\}^*$ be a function computing the prime factorization of an input.

Define the following language

$$L_f = \{\langle x, j\rangle \in \{0,1\}^* : j\text{th bit } f(x) \text{ equals } 1\}.$$

Proposition 2. $L_f \in$ NP $\cap$ co-NP.

Sketch of proof. The prime factorization is unique. So it can be used as a certificate for both languages L_f and $\bar{L}_f$. □

1.3 Probabilistic computation

A probabilistic algorithm has an access to a source of random bits. Random bits are independent and both values have the probability 1/2.

The result of computation is a random variable $\mathrm{Res}(x)$, where x is an input. Errors are possible. So the first question is: how to define a successful computation?

The idea is to bound an error probability. We say that an algorithm has the probability of error a if for all x

$$\mathbf{Pr}[\mathrm{Res}(x) \neq f(x)] < a \leq \frac{1}{2}.$$

It is appeared that the cases $a < 1/2$ ('with a cut point') and $a = 1/2$ ('without cut point'), where a is constant, differ drastically in complexity.

It seems that the difference is related to possibility of *amplification*.

If $a = 1/2 - \varepsilon$, $\varepsilon > 0$, then the error probability can be reduced efficiently.

The amplification procedure is very simple: repeat the computation t times and take the most frequent answer.

By direct computation one can check the following fact.

Proposition 3. *If* $\mathbf{Pr}[\mathrm{Res}(x) \neq f(x)] < \frac{1}{2} - \varepsilon$ *then the probability of error of the amplified algorithm is less than*

$$\left(2\sqrt{\frac{1}{4} - \varepsilon^2}\right)^t$$

Thus a cut point guarantees that the error probability exponentially decreases because $2\sqrt{\frac{1}{4} - \varepsilon^2} < 1$.

Without cut point the proposition gives no useful bound and there is no way to decrease error probability by an efficient procedure.

The above cases lead to definitions of two complexity classes. Note that for fixed values of random bits a probabilistic algorithm becomes deterministic.

Definition 9 (The class BPP)**.** $(L_0, L_1) \in$ BPP if there exist a polynomial $p(\cdot)$ and a deterministic algorithm $V(x, r)$ running in polynomial time such that

- (completeness) if $x \in L_1$ then $\mathbf{Pr}_{r \leftarrow \mathcal{U}_{p(|x|)}}[V(x, r) = 1] \geq 2/3$;
- (soundness) if $x \in L_0$ then $\mathbf{Pr}_{r \leftarrow \mathcal{U}_{p(|x|)}}[V(x, r) = 1] < 1/3$.

Here $\mathcal{U}_m$ is the uniform distribution on the strings of length m.

Proposition 4. P $\subseteq$ BPP.

Proposition 5. *There are* BPP *complete promise problems.*

Existence of BPP complete languages is a long-standing open problem.

For probabilistic computation without cut point we define the following complexity class.

Definition 10. $(L_0, L_1) \in$ PP if there exist a polynomial $p(\cdot)$ and a deterministic algorithm $V(x, r)$ running in polynomial time such that

- (completeness) if $x \in L_1$ then $\mathbf{Pr}_{r \leftarrow \mathcal{U}_{p(|x|)}}[V(x, r) = 1] > 1/2$;
- (soundness) if $x \in L_0$ then $\mathbf{Pr}_{r \leftarrow \mathcal{U}_{p(|x|)}}[V(x, r) = 1] < 1/2$.

The class PP contains BPP by definition. Also definition implies the following fact (completeness and soundness conditions are symmetric in the definition).

Proposition 6. PP = co-PP.

But PP is much wider. For example, PP $\supseteq$ NP as it shown below.

We relate PP with counting problems and gap functions.

To define gap functions we look at definitions of NP, BPP and PP from another point of view.

A *nondeterministic Turing machine* is able to make different transitions at a step. It chooses a transition arbitrary. Thus computation of an NTM is not a sequence of configurations, it is a rooted tree. A branch of the tree from the root to a leaf (halting state) is called a *computation path*.

Any NTM can be easily converted to an NTM that at each nondeterministic step makes a choice between two transitions.

An NTM is *running in polynomial time* if every computation path is polynomially bounded by the input size. In this case there are finitely many computation paths (typically, exponentially many).

We consider NTMs with accepting and rejecting halting states. So a computation path is either accepting or rejecting. We denote the number of accepting computation paths by $\mathrm{acc}_M(x)$, where M is a machine and x is an input.

Definition 11 (#P-functions and gap functions)**.** A function $f\colon \{0,1\}^* \to \mathbb{Z}_+$ is called a #P-*function* if there exist an NTM M running in polynomial time such that

$$f(x) = \mathrm{acc}_M(x)$$

for all x.

A function $f\colon \{0,1\}^* \to \mathbb{Z}$ is called a *gap function* if $f(x) = g(x) - h(x)$ for #P-functions g, h.

The following fact is easily checked.

Proposition 7. $(L_0, L_1) \in \mathrm{NP}$ *iff there exists an NTM* M *running in polynomial time such that* $x \in L_1 \Leftrightarrow \mathrm{acc}_M(x) > 0$ *and* $x \in L_0 \Leftrightarrow \mathrm{acc}_M(x) = 0$.

To define BPP in terms of NTMs one need to count probabilities of accepting computation paths instead of the cardinality. Here we skip details.

As for the class PP, we have a remarkable result.

Theorem 8 (Fenner, Fortnow, Kurtz, 1991)**.** $(L_0, L_1) \in \mathrm{PP}$ *iff there exists a gap function* f *such that* $x \in L_1 \Leftrightarrow f(x) > 0$ *and* $x \in L_0 \Leftrightarrow f(x) \le 0$.

Corollary 9. $\mathrm{PP} \supseteq \mathrm{NP}$.

The proof of Theorem 8 is based on arithmetic closure properties of gap functions.

Lemma 10. *If* f, g *are gap functions then* $f \pm g$ *and* $f \cdot g$ *are also gap functions.*

Proof. It suffices to prove that #P-functions are closed under addition and multiplication.

Let $f(x) = \mathrm{acc}_{M_1}(x)$ and $g(x) = \mathrm{acc}_{M_2}(x)$.

The machine $M_1 \vee M_2$ in the starting state chooses $i \in \{0,1\}$ and then run the machine M_i. So $\mathrm{acc}_{M_1 \vee M_2}(x) = \mathrm{acc}_{M_1}(x) + \mathrm{acc}_{M_2}(x)$.

The machine $M_1 \wedge M_2$ runs M_1 and, if it accepts, run M_2 on the same input. So $\mathrm{acc}_{M_1 \wedge M_2}(x) = \mathrm{acc}_{M_1}(x) \cdot \mathrm{acc}_{M_2}(x)$. □

Proof of Theorem 8. Note that $f(x) = 2^{p(|x|)}$ is a #P-function for a polynomial $p(n)$. For $(L_0, L_1) \in \mathrm{PP}$ it is easy to construct an NTM M such that

$$\Pr_{r \leftarrow \mathcal{U}_{p(|x|)}}[V(x,r) = 1] = \frac{\mathrm{acc}_M(x)}{2^{p(|x|)}}.$$

Therefore $x \in L_1 \Leftrightarrow f(x) > 0$ and $x \in L_0 \Leftrightarrow f(x) < 0$ for a gap function $f(x) = 2\,\mathrm{acc}_M(x) - 2^{p(|x|)}$.

To prove the opposite we need more sophisticated construction.

At first, note that for #P-functions f, g

$$u(x) - v(x) > 0 \Leftrightarrow 2u(x) - (2v(x)+1) > 0 \text{ and } u(x) - v(x) \le 0 \Leftrightarrow 2u(x) - (2v(x)+1) < 0.$$

Thus w.l.o.g. we assume that $x \in L_1 \Leftrightarrow a(x) > 0$ and $x \in L_0 \Leftrightarrow a(x) < 0$ for a gap function $a(x) = f(x) - g(x)$.

Let $f(x) = \mathrm{acc}_{M_0}(x)$, $g(x) = \mathrm{acc}_{M_1}(x)$ are #P-functions and $L(n)$ is a polynomial such that $L(n) > t_{M_0}(n)$, $L(n) > t_{M_1}(n)$.

We assume that M_i choose between two transitions at each nondeterministic step. Thus a computation paths on an input x are indexed by binary strings p shorter than $L(|x|)$.

To fit the definition of PP we should construct a polynomial $p(n)$ and a binary-valued function $V(x, r) \in \mathrm{P}$, where $|r| = p(|x|)$.

Set $p(n) = L(n) + 1$. To define $V(x, r)$ we divide a string r in three blocks: the first bit a, a path p and a suffix q such that p is an index of computation path of a machine M_a running on the input x. The function $V(x, apq)$ should satisfy the following conditions:

$$
\begin{aligned}
&|\{q : V(x, 0pq) = 1\}| - |\{q : V(x, 0pq) = 0\}| = 2 && \text{if } p \text{ is an accepting path for } M_0,\\
&|\{q : V(x, 0pq) = 1\}| - |\{q : V(x, 0pq) = 0\}| = 0 && \text{if } p \text{ is a rejecting path for } M_0,\\
&|\{q : V(x, 1pq) = 1\}| - |\{q : V(x, 0pq) = 0\}| = -2 && \text{if } p \text{ is an accepting path for } M_1,\\
&|\{q : V(x, 1pq) = 1\}| - |\{q : V(x, 0pq) = 0\}| = 0 && \text{if } p \text{ is a rejecting path for } M_1.
\end{aligned}
$$

It is easy to construct $V(x, r) \in \mathrm{P}$ satisfying these conditions. The second and the fourth conditions are satisfied by choosing

$$V(x, apq) = \bigoplus_{j=1}^{|q|} q_j,$$

the first— by

$$V(x, 0pq) = \begin{cases} 1 + \bigoplus\limits_{j=1}^{|q|} q_j \oplus \prod\limits_{j=1}^{|q|} q_j & \text{if } |q| \text{ is odd,} \\ \bigoplus\limits_{j=1}^{|q|} q_j \oplus \prod\limits_{j=1}^{|q|} q_j & \text{if } |q| \text{ is even,} \end{cases}$$

and the third— by

$$V(x, 0pq) = \begin{cases} \bigoplus\limits_{j=1}^{|q|} q_j \oplus \prod\limits_{j=1}^{|q|} q_j & \text{if } |q| \text{ is odd,} \\ 1 + \bigoplus\limits_{j=1}^{|q|} q_j \oplus \prod\limits_{j=1}^{|q|} q_j & \text{if } |q| \text{ is even.} \end{cases}$$

(Recall that the length of q is at least 1 by construction.)

Direct calculation shows that

$$\Pr_{r \leftarrow \mathcal{U}_{p(|x|)}}[V(x, r) = 1] = \frac{1}{2} + \frac{f(x) - g(x)}{2^{L(|x|)}}.$$

Thus $(L_0, L_1) \in \mathrm{PP}$. □

1.4 Quantum computation

From complexity point of view quantum computation is similar to probabilistic computation. But quantum processes provide more intricated distributions.

There are many models of quantum computation. We restrict our attention on the standard one.

Let us describe this model at high level. There is a classical control device that operates with a *'large'* closed quantum system. It performs the following actions:

- prepare the fixed initial state of the quantum system;

- perform a *quantum evolution* of the system;
- perform a *measurement* of the resulting state, the measurement outcome is a random variable depending on the resulting state;
- read an answer from the measurement outcome.

Note that for probabilistic computation we use the uniform distributions only. For quantum computation such a choice is impossible. Quantum evolution may lead to distributions that can not be generated one from another by classical tools.

1.4.1 Quantum states, evolution and measurements

Let us recall the basic facts about quantum states and operations with them.

A state space of a closed quantum system is a complex Hilbert space $\mathbb{C}^n$. States are represented by vectors of unit length. But a global shift (multiplying by $e^{i\varphi}$) does not change a quantum state physically. So the actual state space is a complex projective space $\mathrm{P}\mathbb{C}^{n-1}$.

A measurement apparatus fixes a basis in the state space. It is called the *computational basis*.

For linear algebra we use Dirac's notation. The vectors in are $|1\rangle$, ..., $|n\rangle$. Usually, quantum computation starts from a basis vector in the computational basis.

For a state

$$|\psi\rangle = \sum_i \alpha_i |i\rangle, \qquad \sum_i |\alpha_i|^2 = 1, \tag{1}$$

the coordinates α_i in the computational basis are called *amplitudes*.

Evolution of a closed quantum system is defined by a unitary operator U.

Measurement of a state (1) produces a random basis vector $|i\rangle$ and the measurement outcome is its index i. Probability of i equals $|\alpha_i|^2$. So, the measurement realizes non-unitary evolution and breaks the closeness of the system.

1.4.2 Local operators and quantum circuits

What unitary operators can be evolution operators? To answer the question we need to introduce tensor structure on the state space and local operator.

In quantum mechanics the state space of a union of two subsystems $\boldsymbol{A}$ and $\boldsymbol{B}$ is the tensor product $\mathcal{A} \otimes \mathcal{B}$, where $\mathcal{A}$, $\mathcal{B}$ are the state spaces of subsystems $\boldsymbol{A}$ and $\boldsymbol{B}$ respectively.

In the standard form of quantum computation quantum systems are composed of elementary quantum systems: *qubits*. A qubit is 2-dimensional quantum system. It means that the measurement of qubit gives to possible outcomes denoted by 0 and 1.

Thus the state space of a qubit is $\mathbb{C}^2 = \mathbb{C}(|0\rangle, |1\rangle)$. The state space of a system composed of n qubits is $(\mathbb{C}^2)^{\otimes n}$. It has a dimension 2^n (that equals the number of binary strings of length n).

An elementary step of computation affect on a few qubits (or bits in the classical settings). The corresponding unitary operators are called *local operators*. A local operator acts on a fixed number of qubits:

$$U[S] \otimes I[\bar{S}],$$

where the operator U acts on the set of qubits S, $|S| = O(1)$, and the identity operator acts on remaining qubits. Matrix elements of a local operator are

$$U[S] \otimes I[\bar{S}] = \sum_{x,y \in \{0,1\}^n : x[\bar{S}] = y[\bar{S}]} u_{x[S],y[S]} |x\rangle\langle y|,$$

where $x[S]$ is a sequence of bits corresponding to the set S in the string x.

A *basis* is a set $\mathcal{B}$ of unitary local operators— *gates*. A quantum computing device can apply gates in arbitrary way. We always assume that a basis is closed under inverse: if $U \in \mathcal{B}$ then $U^\dagger \in \mathcal{B}$.

Definition 12. A *quantum circuit* over the basis $\mathcal{B}$ is a sequence

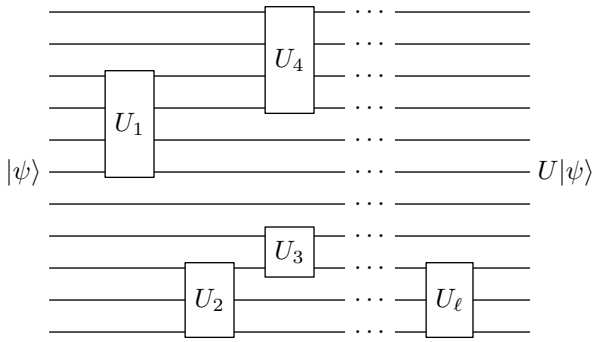
$$U_1[S_1], U_2[S_2], \ldots, U_\ell[S_\ell], \quad U_k \in \mathcal{B},\ S_k \subseteq \{1, \ldots, n\}.$$

The *size* of the circuit is L.

A circuit $U_1[S_1], U_2[S_2], \ldots, U_\ell[S_\ell]$ *realizes* the operator

$$U_\ell[S_\ell]U_{\ell-1}[S_{\ell-1}]\cdots U_2[S_2]U_1[S_1].$$

Graphically circuits are represented as shown on the picture



For a finite basis, quantum circuits have finite classical descriptions. We are also using infinite bases assuming approximations that will be discussed later.

Application of a circuit U to the initial vector $|\psi\rangle$ gives the state

$$U|0^n\rangle = \sum_x \alpha_x |x\rangle$$

According to a general rule, the distribution on measurement outcomes is

$$\mathop{\mathbf{Pr}}_{U|0^n\rangle}[x] = |\alpha_x|^2 = |\langle x|U|0^n\rangle|^2.$$

We see from this formula that a scalar operator (a global phase shift) does not affect on the distribution. Measurement results for vectors $|\psi\rangle$ and $e^{i\varphi}|\psi\rangle$ are the same.

Another observation is about use of additional qubits (*ancillas*). Suppose we realize an operator U such that

$$U(|0^n\rangle \otimes |0^m\rangle) = (W|\psi\rangle) \otimes |0^m\rangle.$$

Then it generates the same distribution on the first n bits as the operator W:

$$\mathop{\mathbf{Pr}}_{U|0^{n+m}\rangle}[x0^m] = |\langle x| \otimes \langle 0^m|U|0^n\rangle \otimes |0^m\rangle|^2 = |\langle x|W|0^n\rangle|^2.$$

1.4.3 Quantum algorithms

A quantum algorithm is described by a (classical) function $A\colon x \mapsto \langle U_x\rangle$. The function computes a description of a quantum circuit in the basis $\mathcal{B}$ that acts on m qubits.

The basis $\mathcal{B}$ can be infinite. So the description of gates in the circuit is approximate. Each matrix element of a gate in the computational basis is represented with precision ε. We will assume that matrix elements has a form

$$\frac{a}{q} + i\frac{b}{q}, \quad a, b \in \mathbb{Z},\ q \in \mathbb{Z}_+,$$

and $\varepsilon = 1/q$.

Gates of the basis are considered as elementary operations with the quantum system. Thus the running time of the algorithm can be measured by the running time of the classical algorithm computing the function $A\colon x \mapsto \langle U_x \rangle$. (A description of a circuit is not shorter than the size of the circuit.)

Definition 13. Let $\mathcal{B}$ be a basis. A problem (L_0, L_1) belongs to the class $\mathrm{BQP}_{\mathcal{B}}$ if there exists a polynomial time algorithm $A\colon x \mapsto \langle U_x \rangle$ such that

- matrix elements of the gates in the circuit U_x are represented with precision $(nL_x)^{-1}$, where L_x is the size of the circuit U_x and $n = |x|$;
- $x \in L_1$ iff $\mathbf{Pr}_{U_x|0^m\rangle}[b_1 = 1] > 2/3$;
- $x \in L_0$ iff $\mathbf{Pr}_{U_x|0^m\rangle}[b_1 = 1] < 1/3$.

Here b_1 is the answer qubit.

The class $\mathrm{BQP}_{\mathcal{B}}$ depends heavily on a choice of a basis. If $\mathcal{B}$ contains the identity operator only then no non-trivial problem lies in the class BQP_I. Another extreme case is the basis $\mathcal{U}$ of all unitary operators. Note that the definition remains reasonable in this case: there are $\Omega(2^d)$ matrix elements of an operator acting on d qubits. So the operators in the circuit U_x are at most $O(\log n)$-local (n is the input size). But $O(1)$-local operators are enough.

Let $\mathcal{B}_r$ be the basis consisting of all r-local operators.

Theorem 11. $\mathrm{BQP}_{\mathcal{U}} = \mathrm{BQP}_{\mathcal{B}_3} = \mathrm{BQP}_{\mathcal{B}_2}$.

Due to the theorem we use a shortcut BQP for $\mathrm{BQP}_{\mathcal{B}_2}$. We discuss this theorem later.

Now we turn to another important question: the role of precision in the definition of BQP.

1.4.4 Approximate representation of gates

Small perturbations of operators lead to small perturbations of distributions generated by measurements. To make this statement formal we need distance measures on the unitary group as well as on the set of probabilistic distributions.

For probabilistic distributions the *total variation distance* is the most appropriate:

$$\rho(p,q) = \max_E \left| \mathop{\mathbf{Pr}}_p[E] - \mathop{\mathbf{Pr}}_q[E] \right|,$$

where the maximum is taken over all possible events. It is well-known that the total variation distance is proportional to ℓ_1-distance.

Proposition 12. $\rho(p,q) = \frac{1}{2}\sum_j \left| \mathbf{Pr}_p[j] - \mathbf{Pr}_q[j] \right|$, *where j are elementary events.*

Distances between operators we measure in the *operator norm*

$$\|X\| = \sup_{|\psi\rangle \neq 0} \frac{\|X|\psi\rangle\|}{\||\psi\rangle\|}.$$

We will use the following properties of the operator norm:

$$\begin{gathered} \|X+Y\| \le \|X\| + \|Y\|, \\ \|XY\| \le \|X\|\,\|Y\|, \\ \|X^\dagger\| = \|X\|, \\ \|X \otimes Y\| = \|X\|\,\|Y\|. \end{gathered}$$

Lemma 13. *If $\|U - U'\| \le \varepsilon$ then the variation distance between distributions generated by measurements of the states $U|0\rangle$ and $U'|0\rangle$ is at most ε.*

Proof. Let

$$U|0\rangle = |\psi\rangle = \sum_x \psi_x |x\rangle, \qquad U'|0\rangle = |\xi\rangle = \sum_x \xi_x |x\rangle.$$

Using Cauchy–Schwartz and other elementary inequalities we have

$$2\rho(p,q) = \sum_x \left||\psi_x|^2 - |\xi_x|^2\right| = \sum_x \left(|\psi_x| - |\xi_x|\right)\left(|\psi_x| + |\xi_x|\right)$$

$$\le \sum_x |\psi_x - \xi_x| \left(|\psi_x| + |\xi_x|\right) \le \left(\sum_x |\psi_x - \xi_x|^2\right)^{1/2} \left(\sum_x (|\psi_x| + |\xi_x|)^2\right)^{1/2}$$

$$= \|\psi - \xi\| \cdot \left(\sum_x (|\psi_x|^2 + |\xi_x|^2 + 2|\psi_x| \cdot |\xi_x|)\right)^{1/2} \le 2\|\psi - \xi\| \le 2\varepsilon.$$

In the last line we use Cauchy–Schwartz again

$$\sum_x |\psi_x| \cdot |\xi_x| \le \|\psi\| \cdot \|\xi\| = 1.$$

□

Let A be a d-dimensional operator approximating a unitary operator U with precision ε. Then moduli of matrix elements of $A - U$ are less than ε and $\|A - U\| < \sqrt{d}\varepsilon$. By the triangle inequality $\|A\| < 1 + \sqrt{d}\varepsilon$.

In the description of a quantum circuit all gates are approximated. We need to upperbound the distance between the product of the gates and the product of approximations.

Proposition 14. *Let U_j be unitary operators and $\|A_j - U_j\| < \delta$. Then*

$$\|U_L U_{L-1} \dots U_1 - A_L A_{L-1} \dots A_1\| < (1+\delta)^L - 1.$$

Proof. By induction. We have

$$\begin{aligned} &\|U_L U_{L-1} \dots U_1 - A_L A_{L-1} \dots A_1\| \\ =&\|(U_L - A_L)U_{L-1} \dots U_1 + A_L(U_{L-1} \dots U_1 - A_{L-1} \dots A_1)\| \\ \le&\delta + (1+\delta)\Big((1+\delta)^{L-1} - 1\Big) = (1+\delta)^L - 1. \end{aligned}$$

□

From Lemma 13 and the last proposition we conclude that if $\delta < (nL)^{-1}$ then the distributions on the measurement outcomes for $U_L U_{L-1} \dots U_1 |0^m\rangle$ and $A_L A_{L-1} \dots A_1 |0^m\rangle$ are $O(d^{1/2} n^{-1})$ close w.r.t. the total variation distance.

We have noted above that $d = O(\log n)$. Therefore the completeness and soundness parameters 2/3 and 1/3 in the definition of $\mathrm{BQP}_{\mathcal{B}}$ are perturbed due to approximation errors by $O(n^{-1/2})$, which tends to 0 for $n \to \infty$.

1.5 Complexity bounds for BQP

1.5.1 Lower bounds

The inclusion $\mathrm{P} \subseteq \mathrm{BQP}$ can be proved easily from the definition.

Theorem 15 (Bernstein, Vazirani, 1997)**.** $\mathrm{BPP} \subseteq \mathrm{BQP}$.

To include BPP to BQP we need more arguments. Let $(L_0, L_1) \in \mathrm{BPP}$. To place the problem in BQP we should simulate the function $V(x, r)$ by a quantum circuit and draw r from the uniform distribution.

We briefly recall how to simulate classical computation by quantum circuits. At first, we need to represent the output $A(x)$ of an algorithm A by the value of classical Boolean circuit. It can be done *uniformly*. i.e. there is a polynomial time algorithm that takes an input x and produces a description of a Boolean circuit C such that $C(x) = A(x)$. W.l.o.g. we assume that the circuits is over the complete basis $\{\neg, \wedge, \oplus\}$.

The second step is to convert a Boolean circuit to a quantum one. Classical operations do not change the computational basis. So they are represented by permutation operators.

Here we face a difficulty: quantum circuits are reversible by definition while the Boolean operation $\wedge$ is not.

To simulate irreversible gates we use ancillas. Let define a permutation operator U_f for a Boolean function f by a rule

$$U_f \colon |x, b\rangle \mapsto |x, b \oplus f(x)\rangle.$$

Then it is a routine to convert a Boolean circuit $f_1, f_2, \ldots, f_L$ to a quantum circuit consisting of operators U_{f_j}. For each Boolean gate we use an ancilla bit to keep a result of operation. All ancillas are initially zero.

This 'straightforward' conversion do not suffice our purposes. If the Boolean circuit computes a function F then the converted circuit realizes an operator

$$\widetilde{U}_F \colon |x, 0^L\rangle \mapsto |x, G(x), F(x)\rangle.$$

To realize U_F with ancillas we need to place zeroes instead of a garbage $G(x)$. It can be done by the 'uncomputing' trick. Note that $U_f^2 = I$ by definition. So applying all gates in the circuit except the last one in the reverse order we get the circuit of size $2L - 1$ that realizes the operator U_F with ancillas.

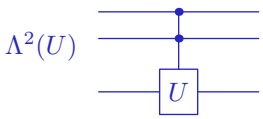
It is worth to mention that the simulation of classical computation uses the gates $U_\neg = X$, $U_\wedge = \Lambda^2(X)$ and $U_\oplus = \Lambda(X)[1, 3]\Lambda(X)[2, 3]$. Here we introduce notation for commonly used gates. X is one of the Pauli operators:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

$\Lambda^k(U)$ denotes a k-qubit controlled-U operator. By definition

$$\Lambda^k(U)|x_1, \ldots, x_k\rangle \otimes |\xi\rangle = \begin{cases} |x_1, \ldots, x_k\rangle \otimes |\xi\rangle & \text{if } x_1 \cdots x_k = 0, \\ |x_1, \ldots, x_k\rangle \otimes U|\xi\rangle & \text{if } x_1 \cdots x_k = 1. \end{cases}$$

Graphically, we represent an operator $\Lambda^k(U)$ as shown in the figure.



Proof of Theorem 15. Let (L_0, L_1) be a BPP problem. By definition,

$$x \in L_1 \quad \Leftrightarrow \quad \mathop{\mathbf{Pr}}_{r \leftarrow \mathcal{U}_{p(|x|)}} [V(x, r) = 1] \geq 2/3;$$

$$x \in L_0 \quad \Leftrightarrow \quad \mathop{\mathbf{Pr}}_{r \leftarrow \mathcal{U}_{p(|x|)}} [V(x, r) = 1] < 1/3$$

for a function $V(x, r) \in \mathrm{P}$.

We have shown that $V(x, r)$ can be computed by a reversible circuit $\widetilde{V}_x$ (depends on r). It remains to make the uniform distribution on strings r of the length $p(|x|)$.

That is simple. Recall that the *Hadamard gate* H is 1-qubit gate

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

To generate a uniform distribution on the measurement outcomes apply $H^{\otimes p(|x|)}$ to the initial state $|0\rangle^{p(|x|)}$. The resulting state is

$$|\psi\rangle = \bigotimes_{j=1}^{p(|x|)} \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big) = \frac{1}{2^{p(|x|)/2}} \sum_{r\in\{0,1\}^{p(|x|)}} |r\rangle.$$

Measuring $|\psi\rangle$ in the computational basis produces the uniform distribution on r.

Thus for the circuit $U_x = \widetilde{V}_x H^{\otimes p(|x|)}$ we have

$$U_x|0^m, 0^{p(|x|)}\rangle = \frac{1}{2^{p(|x|)/2}} \sum_r |V(x,r), 0^{m-1}, r\rangle$$

Probability of 1 in the first qubit coincides with probability of 1 in the probabilistic algorithm. □

Another corollary from simulating classical computation by reversible circuits is the amplification property for BQP. Similar to BPP, error probabilities can be amplified. The amplification procedure is the same: apply U_x to t copies of the initial state take the most frequent answer. The arguments used for BPP amplification remains valid for BQP. Two remarks should be noted.

At first, computing majority can be done by a reversible circuit as it explained above.

At second, the distribution of the first qubits after measuring the state $|\psi\rangle^{\otimes t}$ is the product of distributions for $|\psi\rangle$.

Lemma 16 (BQP amplification)**.** *For each $r = \mathrm{poly}(n)$ a problem (L_0, L_1) is in $\mathrm{BQP}_{\mathcal{B}}$ iff there exists a polynomial time algorithm $A\colon x \mapsto \langle U_x\rangle$ such that*

- *matrix elements of the gates in the circuit U_x are represented with precision $(nL_x)^{-1}$, where L_x is the size of the circuit U_x and $n = |x|$;*
- *$x \in L_1$ iff $\mathbf{Pr}_{U_x|0^m\rangle}[b_1 = 1] > 1 - 2^{-r(n)}$;*
- *$x \in L_0$ iff $\mathbf{Pr}_{U_x|0^m\rangle}[b_1 = 1] < 2^{-r(n)}$.*

1.5.2 Upper bound

Theorem 17 (Adleman, DeMarrais, Huang, 1997)**.** $\mathrm{BQP} \subseteq \mathrm{PP}$.

Let $(L_0, L_1) \in \mathrm{BQP}$ and $U = U_L \cdots U_1$ be a quantum circuit for an input x. For brevity we suppress in the sequel dependence of U and other objects on x. The circuit U is applied to the state $|0^m\rangle$. The state $U|0^m\rangle$ is measured, The probability to observe 1 in the first bit can be expressed as a matrix element of the product of $2L + 1$ operators:

$$\mathop{\mathbf{Pr}}_{U_x|0^m\rangle}[b_1 = 1] = \langle 0^m|U_1^\dagger \cdots U_L^\dagger \Pi_1^{(1)} U_L \cdots U_1|0^m\rangle = \langle 0^m|P|0^m\rangle,$$

where $\Pi_1^{(1)}$ is the projection operator on the subspace $b_1 = 1$:

$$\Pi_1^{(1)}|b_1, \ldots, b_m\rangle = \begin{cases} |b_1, \ldots, b_m\rangle & \text{if } b_1 = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Matrix elements of the factors in this product are complex numbers in the form

$$\frac{u}{nL} + i\frac{v}{nL}, \quad u, v \in \mathbb{Z}.$$

The denominator of the matrix element is $(nL)^{2L+1}$. Real and imaginary parts of numerators of a matrix element of each factor are computed in polynomial time by the construction of the circuit. So $\mathbf{Pr}_{U_x|0^m\rangle}[b_1 = 1] > 1/2$ iff $2S - (nL)^{2L+1} > 0$, where $S = (nL)^{2L+1}\langle 0^m|P|0^m\rangle$ is the matrix element of a product of matrices with integer real and imaginary parts, which are computed in polynomial time.

So (L_0, L_1) is in PP due to the following lemma. (Note that $(nL)^{2L+1}$ is a gap function due to the same lemma with $m = 0$).

Lemma 18. *Let $P = P_1 \cdots P_N$ be the product of $(2^m \times 2^m)$ matrices with integer real and imaginary parts and there exists a polynomial time algorithm to compute*

$$A(j, b, c) = \mathrm{Re}\langle b|P_j|c\rangle, \quad B(j, b, c) = \mathrm{Im}\langle b|P_j|c\rangle.$$

Then $\mathrm{Re}\langle 0^m|P|0^m\rangle$ and $\mathrm{Im}\langle 0^m|P|0^m\rangle$ are gap functions.

Recall that P depends on an input x as well as $\mathrm{Re}\langle 0^m|P|0^m\rangle$ and $\mathrm{Im}\langle 0^m|P|0^m\rangle$.

Proof. We prove that $\mathrm{Re}\langle 0^m|P|0^m\rangle$ is a gap function. For $\mathrm{Im}\langle 0^m|P|0^m\rangle$ the proof is similar.

By definition,

$$\langle 0^m|P|0^m\rangle = \sum_{0^m = b_0, \ldots, b_N = 0^m} \prod_{j=1}^{N} \big(A(j, b_{j-1}, b_j) + iB(j, b_{j-1}, b_j)\big).$$

The first step is to expand the products and separate terms of equal sign.

For a sequence $b = (b_0, \ldots, b_N)$ we define a sign $\pi(b, J)$ of a subset $J \subseteq [N] = \{1, \ldots, N\}$ such that $|[N] \setminus J|$ is even. It is equal $+1$ iff the total number of j such that either $j \in J$ and $A(j, b_{j-1}, b_j) > 0$, or $j \notin J$ and $B(j, b_{j-1}, b_j) > 0$ is even. Otherwise $\pi(b, J) = -1$.

Using this notation we write

$$\begin{aligned}
&\mathrm{Re}\langle 0^m|P|0^m\rangle \\
&\quad = \sum_b \sum_{J:\pi(b,J)=+1} \prod_{j\in J} |A(j, b_{j-1}, b_j)| \prod_{j\notin J} |B(j, b_{j-1}, b_j)| \\
&\quad - \sum_b \sum_{J:\pi(b,J)=-1} \prod_{j\in J} |A(j, b_{j-1}, b_j)| \prod_{j\notin J} |B(j, b_{j-1}, b_j)| \\
&\quad = \sum_b \sum_{\alpha\in\{0,1\}^N} \prod_{j=1}^{N} C(j, \alpha, b) - \sum_b \sum_{\alpha\in\{0,1\}^N} \prod_{j=1}^{N} D(j, \alpha, b),
\end{aligned}$$

where C and D are computed in polynomial time.

Claim. For any $C(j, y) \in \mathrm{P}$, $j \in [N]$, $y \in \{0, 1\}^M$, the function

$$\sum_y \prod_j C(j, y)$$

is a #P-function.

Proof of the claim. W.l.o.g. we assume that $N = 2^k$. $C(j, y) \in \mathrm{P}$ therefore $C(j, y) \le 2^{p(k+M)}$ for a polynomial $p(n)$.

An NTM M makes $M + kp(k + M)$ binary choices. Each computation path is represented by a pair $(y, b(j))$, where $j \in [N]$, $y \in \{0, 1\}^M$, and the $b(j)$ is an integer written in binary by use of $p(k + M)$ bits.

The path $(y, b(j))$ is accepting iff $b(j) < C(j, y)$ for each j. It is clear that this condition can be checked in polynomial time.

Thus for any y the number of accepting paths $(y, b(j))$ is

$$\prod_j C(j, y)$$

and the total number of accepting paths is

$$\sum_{y} \prod_{j} C(j, y)$$

The claim is proved and the lemma follows. □

2 BQP-complete problems

As for many other complexity classes, the class BQP can be represented by a single problem that is the hardest in the class. A *reduction* is a way to compare complexity of problems.

2.1 Reductions

Let us define *polynomial reductions* (Karp reductions) of promise problems.

Definition 14. A promise problem (L_0, L_1) is reduced to a promise problem (K_0, K_1) by a function f computed in polynomial time if

$$x \in L_1 \;\Leftrightarrow\; f(x) \in K_1, \quad x \in L_0 \;\Leftrightarrow\; f(x) \in K_0.$$

Notation $(L_0, L_1) \leqslant_p (K_0, K_1)$.

If $(L_0, L_1) \leqslant_p (K_0, K_1)$ by a function f then any decision algorithm for (K_0, K_1) can be converted to a decision algorithm for (L_0, L_1):

1. On an input x compute $f(x)$.
2. Run a decision algorithm for (K_0, K_1).
3. Output the answer.

Correctness of this algorithm follows from the definition of reduction. Extra time required to compute $f(x)$ is polynomial. So, a reduction shows that (K_0, K_1) is harder than (L_0, L_1).

Proposition 19 (transitivity of reductions)**.** $(L_0, L_1) \leqslant_p (K_0, K_1)$ *and* $(K_0, K_1) \leqslant_p (M_0, M_1)$ *imply* $(L_0, L_1) \leqslant_p (M_0, M_1)$.

The proposition follows from a simple observation that a composition of functions of polynomial growth is a function of polynomial growth.

Reductions give a way to indicate the hardest problems in a complexity class $\mathcal{C}$.

Definition 15. A problem $P \in \mathcal{C}$ is called $\mathcal{C}$ complete if $L \leqslant_p P$ for any problem $L \in \mathcal{C}$.

Thus for complexity classes containing P, any complete problem represents the whole class: any algorithm to solve this particular problem can be transformed to an algorithm solving any problem in the class (combine the algorithm with a reduction to the complete problem).

Sometimes it is more convenient to define a class by presentation of a complete problem for a class.

The class NP contains complete problems as it was shown by Cook and Levin. Now the list of NP complete problems is immense. Having a complete problem, one can enlarge the list of complete problems just reducing this particular problem to a new candidate to the list.

We have presented several examples of NP complete problem in the previous lecture.

Theorem 20 (Cook (1971), Karp (1972), Lovász (1973), Stockmeyer (1973))**.** *The problems* 3-SAT*,* Partition *and* 3-COLOR *are* NP *complete.*

The most intriguing question in complexity theory is $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$. If $\mathrm{P} \subset \mathrm{NP}$ then there exists NP problems outside P that are not NP complete.

Prime factorization is a candidate in the set of problems that are not NP complete but lie outside P.

For the class BQP many complete promise problems are also known. The question about complete languages is open. It is considered as a very hard question.

Note that complete problems change the goal of designing efficient quantum algorithms. Instead of thinking about sophisticated quantum procedures solving a problem, one can thinks about a reduction of the problem in question to a known BQP complete problem.

For this purpose it is desirable to have many examples of complete problems. It appears that BQP complete problems are ton so numerous as NP complete problems.

Below we present the examples.

2.2 Quantum circuits and BQP-complete problems

An examples of BQP-complete problems are easily derived from definition of the quantum algorithm. Recall that a quantum algorithm is an application of a quantum circuit to the initial state $|0^n\rangle$ following by the measurement of the answer qubit.

Definition 16. Evaluation of a qubit measurement.
Input: a description $\langle U\rangle$ of a quantum circuit.
Promise: either $\mathbf{Pr}_{U|0^n\rangle}[b_1 = 1] > 2/3$ or $\mathbf{Pr}_{U|0^n\rangle}[b_1 = 1] < 1/3$.
Question: decide whether $\mathbf{Pr}_{U|0^n\rangle}[b_1 = 1] > 1/2$.

Proposition 21. *The problem* Evaluation of a qubit measurement *is* BQP*-complete.*

Proof. Let $(L_0, L_1) \in$ BQP. A reduction is just a function $x \mapsto \langle U_x\rangle$ from the definition of the class BQP. □

Definition 17. Evaluation of a matrix element.
Input: a description $\langle U\rangle$ of a quantum circuit.
Promise: either $\langle 0^n|U|0^n\rangle > 1/3$ or $\langle 0^n|U|0^n\rangle < -1/3$
Question: decide whether $\langle 0^n|U|0^n\rangle < 0$.

Proposition 22. *The problem* Evaluation of a matrix element *is* BQP*-complete.*

Proof. Let prove that Evaluation of a qubit measurement is reduced to Evaluation of a matrix element. For an input $\langle U\rangle$ of the former problem we construct an instance $V = UZ[1]U^\dagger$ of the latter, where $Z[1]$ is the Pauli operator Z applied to the answer qubit. It is clear that this transformation can be done in polynomial time.

The correctness of the reduction follows from the equality

$$\mathop{\mathbf{Pr}}_{U|0^n\rangle}[x_1 = 1] = \frac{1}{2}\big(1 - \langle 0^n|UZ[1]U^\dagger|0^n\rangle\big).$$

(To verify the equation note that the right-hand side is $\frac{1}{2}\big(1 - \mathbf{E}_{U|0^n\rangle}[Z[1]]\big)$.)

So $\mathbf{Pr}_{U|0^n\rangle}[x_1 = 1] > 2/3$ iff $\langle 0^n|V|0^n\rangle < -1/3$ and $\mathbf{Pr}_{U|0^n\rangle}[x_1 = 1] < 1/3$ iff $\langle 0^n|V|0^n\rangle > 1/3$. □

Remark 2. Error probability of a quantum algorithm can be amplified to an exponentially small value. Using the amplification one can easily extend the previous result to the problem of locating the value $\langle 0^n|U|0^n\rangle$ of a matrix element promised that the matrix element does not belong to an interval (a, b), $0 < a < b < 1$.

To get more interesting examples we need to restrict a basis used.

A requirement to a basis used is a possibility to efficiently generate the whole class of measurement distributions of the states $U|0^n\rangle$, where $U \in \mathbf{U}((\mathbb{C}^2)^{\otimes n})$. There several sufficient conditions for that: circuits in the basis $\mathcal{B}$ realize

- all unitary operators modulo scalar operators, i.e. the group $\mathbf{U}((\mathbb{C}^2)^{\otimes n})/\mathbf{U}(1)$;
- all operators from $\mathbf{U}((\mathbb{C}^2)^{\otimes n})/\mathbf{U}(1)$ with ancillas;
- operators from a dense subgroup of $\mathbf{U}((\mathbb{C}^2)^{\otimes n})/\mathbf{U}(1)$ with ancillas (approximate realization).

For finite bases the only last option remains.

Actually, there is a more restrictive sufficient condition. Only orthogonal transformation can be realized under the same conditions. To replace unitary transformations by orthogonal ones one can use a simple trick: keep complex amplitudes in an additional real qubit.

More formally, a state $(\alpha + i\beta)|x\rangle$, where x is in the computational basis, will be encoded by the state $(\alpha|0\rangle + \beta|1\rangle) \otimes |x\rangle$ in the extended state space. We assume that the additional qubit placed at the first has the index 0.

Let $\mathcal{B}$ be an arbitrary unitary basis. We define a real-version of the basis as

$$\mathcal{B}_{\mathbb{R}} = \{U' : U' = \mathrm{Re}(U) - iY[0]\,\mathrm{Im}(U),\ U \in \mathcal{B}\}.$$

It can be verified by direct computation that U' applied to an encoded state $|\psi\rangle$ get the state encoding $U|\psi\rangle$.

2.3 Exact realization in infinite bases

We recall two theorems about realization of the unitary group with ancillas.

Theorem 23 (Deutsch, 1989)**.** *Any unitary operator is realized with ancillas in the basis $\mathcal{B}_3$ of all 3-local operators is complete.*

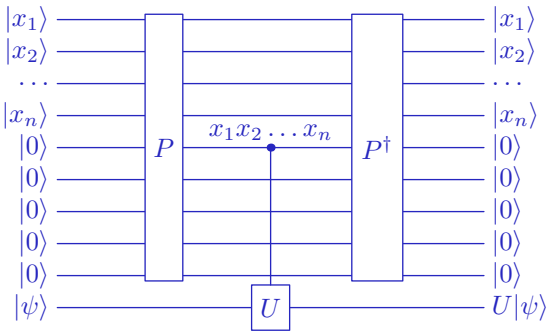
The proof is a combination of several facts that can be easily checked.

Lemma 24. *An arbitrary unitary operator U on the space $\mathbb{C}^M$ can be represented as a product of $M(M-1)/2$ matrices of the form*

$$\begin{pmatrix} 1 & 0 & \dots & & & \dots \\ \vdots & \ddots & 0 & & & \dots \\ 0 & \dots & 1 & 0 & & \dots \\ 0 & \dots & & \begin{pmatrix} a & b \\ c & d \end{pmatrix} & 0 & \dots \\ 0 & \dots & & & 1 & 0 \\ \dots & & & & & \ddots \\ 0 & \dots & & & & 1 \end{pmatrix}, \text{ where } \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathbf{U}(2).$$

We are interested in the case $M = 2^m$ corresponding to the state space of m qubits. In this case each operator in the form described in the lemma is conjugated to $\Lambda^{m-1}(U)$, where U acts on one qubit, by a permutation operator. This permutation operator can be efficiently realized by a reversible circuit C_u, $u \in \{0,1\}^m$, in the basis $\{X, \Lambda(X), \Lambda^2(X)\}$.

To realize $\Lambda^{m-1}(U)$ one need a reversible circuit P computing the product of Boolean variables. The circuit realizing $\Lambda^{m-1}(U)$ with ancillas is shown in the figure.



Thus the unitary group is generated (with ancilla) in the basis

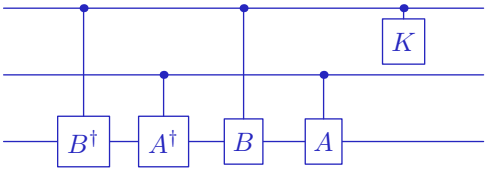
$$\{X, \Lambda(X), \Lambda^2(X), \Lambda(U)\}, \quad U \in \mathbf{U}(2)$$

of 3-local operators. Theorem 23 follows from this fact.

A stronger result was proved by Barenco, Bennett, Cleve, DiVincenzo, Margolus, Shor, Sleator, Smolin and Weinfurter.

Theorem 25 (Barenco etc., 1995)**.** *Any unitary operator up to a global phase is realized with ancillas in the basis containing $\Lambda(X)$ and all unitary operators acting on one qubit.*

Sketch of the proof. Toffoli gate $\Lambda^2(X)$ is realized by the circuit

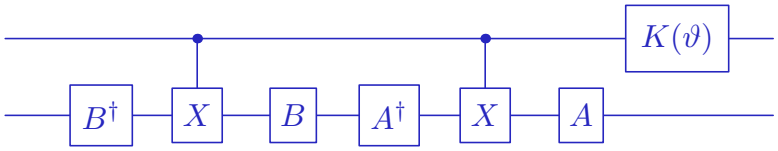


Here

$$A = \frac{1}{\sqrt{2}}\begin{pmatrix} -i & -1 \\ 1 & i \end{pmatrix}; \quad B = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}; \quad K = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$$

such that $A^2 = -I, \quad B^2 = -I, \quad ABA^\dagger B^\dagger = -iX$.

It remains to realize $\Lambda(U)$ by circuits in the basis $\{\Lambda(X), U\}$, $U \in \mathbf{U}(2)$. Note that $\mathbf{U}(2)/\mathbf{U}(1) \cong \mathbf{SO}(3)$ and each 3-dimensional rotation is a composition of two reflections. Thus



Here $K(\vartheta)$ is a controlled phase shift

$$K(\vartheta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\vartheta} \end{pmatrix}$$

□

A reduction from the unitary group to the orthogonal group increases the locality of operator by 1. But $\Lambda(X)$ has real elements. So, we have a corollary from Theorem 25.

Corollary 26. *The problems* Evaluation of a qubit measurement *and* Evaluation of a matrix element *for the circuits in the basis of all orthogonal operators acting on two qubits are* BQP *complete.*

2.4 Finite bases

In the case of finite bases it is only possible to obtain an approximate representation of operators as products of basis elements.

Let us recall formal definitions. The operator $U\colon (\mathbb{C}^2)^{\otimes n} \to (\mathbb{C}^2)^{\otimes n}$ *is approximated by the operator* $\tilde{U}\colon (\mathbb{C}^2)^{\otimes N} \to (\mathbb{C}^2)^{\otimes N}$ *with precision* δ *using ancillas* if, for arbitrary $|\xi\rangle$ in $(\mathbb{C}^2)^{\otimes n}$, the inequality

$$\left\|\tilde{U}\left(|\xi\rangle \otimes |0^{N-n}\rangle\right) - U|\xi\rangle \otimes |0^{N-n}\rangle\right\| \leq \delta \left\||\xi\rangle\right\|$$

holds.

Definition 18. A basis $\mathcal{B} = \{U_1, \ldots, U_r\}$ is *universal* if any operator $U \in \mathbf{U}(d)/\mathbf{U}(1)$ can be approximated with ancillas with arbitrary precision.

Definition 19. A basis $\mathcal{B} = \{U_1, \ldots, U_r\}$ is *complete* if it generates a dense subgroup of the unitary group $\mathbf{U}(d)$ modulo phase, or the orthogonal group $\mathbf{O}(d)$ in the real case. Ancillas are allowed.

A complete basis is a universal one.

The problems Evaluation of a qubit measurement and Evaluation of a matrix element remain BQP complete for any complete finite basis $\mathcal{B}$ for $\mathbf{O}(4)$.

To build reductions we should find approximations efficiently. For this purpose Kitaev-Solovay theorem is in use.

2.4.1 Kitaev–Solovay theorem

Theorem 27 (Kitaev, Solovay)**.** *Let $\mathcal{B}$ be a complete finite basis of operators in $\mathbf{SU}(d)$, where d is a constant. Then there exists an algorithm that takes a description of an operator $U \in \mathbf{SU}(d)$ and an accuracy bound ε and produces a description of a $\mathcal{B}$-circuit W such that $\|W - U\| < \varepsilon$. The running time of the algorithm is* $\mathrm{poly}(\log(1/\varepsilon))$.

Corollary 28. *For any complete basis the problems* Evaluation of a qubit measurement *and* Evaluation of a matrix element *are* BQP*-complete.*

We outline the proof of Kitaev–Solovay theorem skipping technical details.

Let G be a group generated by operators from the basis $\mathcal{B}$. Note that each element V in the group G has an efficient description: a sequence of indices of operators in $\mathcal{B}$ applied to get V. Hereinafter we assume this representation. For any $V \in G$ and $U \in \mathbf{SU}(d)$ the norm $\|U - V\|$ can be efficiently approximated with high precision (note that the dimension of operators $d = O(1)$).

The algorithm builds approximations with increasing precision in the following way. Let $\|U - V\| < \varepsilon$ for $V \in G$. Then $\|UV^{-1} - I\| < \varepsilon$. At the next step the algorithm finds $V' \in G$ such that $\|UV^{-1} - V'\| < \varepsilon/q$, where q is a constant. It implies that $\|U - V'V\| < \varepsilon/q$ too. If a step can be performed in polynomial time then repetitions give an exponentially closed approximation in polynomial time.

Note that on each step we should find an approximation in a small neighborhood of the identity operator.

For a more detailed description we will use geometric tools. The group $\mathbf{SU}(d)$ is a real manifold of dimension $d^2 - 1$. It is equipped with a distance $d(U, V) = \|U - V\|$. The distance in invariant under the group multiplication: $d(UW, VW) = d(U, V)$. By B_r we denote a metric ball of identity, i.e. the set $\{U \in \mathbf{SU}(d) : \|U - I\| \leq r\}$. It is easy to see that $\operatorname{diam} \mathbf{SU}(d) = 2$. So $B_2 = \mathbf{SU}(d)$.

Recall definitions of nets in metric spaces. An *ε-net for $R \subseteq S$* is a set $\Gamma \subseteq S$ whose ε-neighborhood contains R, i.e., for any $x \in R$ there is a $y \in \Gamma$ such that $d(x, y) \leq \delta$. The ε-net Γ is called *α-sparse* $(0 < \alpha < 2)$ if it has no points outside

the δ-neighborhood of R and the distance between any two distinct points of Γ is greater than $\alpha\varepsilon$.

An (r,ε)-*net* in $\mathbf{SU}(d)$ is an ε-net for B_r. The ratio $q = r/\varepsilon$ is called *quality* of the net. It follows from volume consideration that the size of an α-sparse $(r, r/q)$-net is $\mathrm{poly}(q/\alpha)$.

The size of an (r,ε)-net can be arbitrary large. In what follows we assume that after any net construction redundant points are removed. If Γ is a (r,ε)-net then by removing of redundant points one can efficiently construct an α-sparse $(r, \varepsilon/(1-\alpha))$-net $\Gamma' \subseteq \Gamma$.

The net Γ' is just a maximal subset in the net Γ such that the distance between any pair of its elements is greater than $\alpha\varepsilon/(1-\alpha)$. Then each element of Γ is $(\alpha\varepsilon/(1-\alpha))$-close to some element of Γ'. It implies that Γ' is a $(\alpha\varepsilon/(1-\alpha)+\varepsilon)$-net.

To make better nets we use multiplications of nets and commutators of nets. For sets $A, B \subseteq \mathbf{SU}(M)$ these operations are

$$AB = \big\{UV : U \in A,\, V \in B\big\}, \qquad [A,B] = \big\{UVU^{-1}V^{-1} : U \in A,\, V \in B\big\}.$$

The following lemma describes a behavior of the commutator operation on the neighborhoods of the identity operator.

Lemma 29. $[B_r, B_s] \subseteq B_{2rs}$; $\quad B_{rs/4} \subseteq [B_r, B_s] B_{3rs(r+s)}$.

The 'shrinking' operation on nets Γ_1, Γ_2 is defined as an $(1/6)$-sparse subnet of $[\Gamma_1, \Gamma_2]$. It follows from the previous lemma that the shrinking of nets of B_r and B_s lies in B_{2rs}. So the shrinking makes a net of a much smaller neighborhood of the identity. But its quality degrades as the following lemma states.

Lemma 30. *Let Γ_1 be an $(r_1,\, r_1/400)$-net, and Γ_2 an $(r_2,\, r_2/400)$-net. Then shrinking these nets is a $(r_1 r_2/4,\, r_1 r_2/80)$-net provided $r_1 + r_2 < 10^{-6}$.*

To restore the quality we use the operation of 'telescoping'. The telescoping of nets Γ_1, Γ_2 is $\Gamma_1\Gamma_2$.

Lemma 31. *Let Γ_1 be an (r_1, δ_1)-net, and Γ_2 an (r_2, δ_2)-net, where $\delta_1 \le r_2$. Then $\Gamma_1\Gamma_2$ is an (r_1, δ_2)-net.*

The algorithm uses two families Γ_j, $\widetilde{\Gamma}_j$ of nets defined by the following inductive rule for sufficiently large j:

- $\widetilde{\Gamma}_j$ is obtained by the shrinking operation applied to the nets $\Gamma_{\lceil j/2 \rceil}$ and $\Gamma_{\lfloor j/2 \rfloor}$;
- Γ_j is obtained by the telescoping operation applied to the nets $\widetilde{\Gamma}_{j-1}$ and $\widetilde{\Gamma}_j$.

We require that for all j the net Γ_j is a $(20^{-j}/5, 20^{-j}/2000)$-net and the net $\widetilde{\Gamma}_j$ is a $(20^{-j}/100, 20^{-j}/2000)$-net.

The above lemmata guarantee that these bounds hold after shrinking and telescoping operations provided

$$\frac{2}{5} \cdot 20^{-j/2+1/2} < 10^{-6},$$

which holds for $j \ge 12$.

As for the nets Γ_j, $\widetilde{\Gamma}_j$ for $j < 12$, they can be extracted from an ε-net Γ_s of the whole group $\mathbf{SU}(d)$ for $\varepsilon = (2000 \cdot 20^{11})^{-1}$.

The net Γ_s has the size $O(1)$ and it is 'hardwired' in the algorithm.

To approximate the operator U with the precision δ, the algorithm builds a sequence of approximations $W_1, W_2, \ldots$ such that $\|U - W_k\| \le \delta_k = (2000 \cdot 20^{11})^{-k}$ until $\delta_k < \delta$. It takes $O(\log(1/\delta))$ steps and the same number of the nets Γ_j, $\widetilde{\Gamma}_j$ is used.

Note that for chosen parameters the sizes of Γ_j, $\widetilde{\Gamma}_j$ are $O(1)$: they are very tight nets but of very small neighborhoods of identity.

Having an δ_k-approximation W_k, the algorithm searches for an δ_{k+1}-approximation of UW_k^{-1}. The latter operator belongs to B_{δ_k} and the approximation can be found in the net $\widetilde{\Gamma}_k$. The search takes a polynomial time.

2.4.2 Generation of dense subgroups

To prove completeness of particular bases we also need a simple geometric lemma.

Lemma 32 (Kitaev)**.** *Let $\mathcal{M}$ be a Hilbert space of finite dimension $M \geq 3$. Consider the subgroup $H \subset \mathbf{U}(\mathcal{M})$, the stabilizer of the 1-dimensional subspace generated by some unit vector $|\xi\rangle \in \mathcal{M}$. Let V be an arbitrary unitary operator not fixing the subspace $\mathbb{C}(|\xi\rangle)$. Prove that the set of operators $H \cup V^{-1}HV$ generates the whole group $\mathbf{U}(\mathcal{M})$.*

The result holds also for the orthogonal group $\mathbf{O}(\mathcal{M})$.

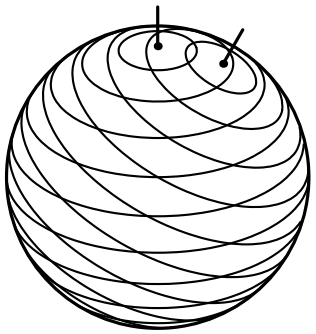
Corollary 33. *Let $\mathcal{M}_1$ and $\mathcal{M}_2$ be two different subspaces of co-dimension 1 (hyperplanes). Then $\mathbf{U}(\mathcal{M}_1) \cup \mathbf{U}(\mathcal{M}_2)$ generates $\mathbf{U}(\mathcal{M})$.*

The result holds also for the orthogonal group.

Proof of Lemma 32. It suffices to show that the group G generated by $H \cup H'$ acts transitively on the set of unit vectors. Indeed, suppose that for each unit vector $|\psi\rangle$ there exists $U_\psi \in G$ such that $U_\psi|\xi\rangle = |\psi\rangle$. Then

$$\mathbf{U}(\mathcal{M}) = \bigcup_{|\psi\rangle \in \mathcal{M}} U_\psi H.$$

The transitivity of the action of the group G is illustrated by the picture:



More formally, note that the orbits $H|\psi\rangle$, $H'|\psi\rangle$ of a unit vector $|\psi\rangle$ are the sets

$$Q(\vartheta) = \big\{|\eta\rangle : |\langle\eta|\xi\rangle| = \cos\vartheta\big\},$$
$$Q'(\vartheta') = \big\{|\eta\rangle : |\langle\eta|\xi'\rangle| = \cos\vartheta'\big\},$$

where ϑ, ϑ' are the angles between $|\psi\rangle$ and $|\xi\rangle$, $|\xi'\rangle$, respectively:

$$\cos\vartheta = |\langle\psi|\xi\rangle|, \quad \cos\vartheta' = |\langle\psi|\xi'\rangle|, \quad 0 \leq \vartheta, \vartheta' \leq \pi/2.$$

We denote by α the angle between the vectors $|\xi\rangle$ and $|\xi'\rangle$:

$$\cos\alpha = |\langle\xi|\xi'\rangle|, \quad 0 \leq \alpha \leq \pi/2.$$

For $\dim\mathcal{M} \geq 3$ the angle between the vector $|\xi\rangle$ and elements of $Q'(\vartheta')$ varies from $|\alpha - \vartheta'|$ to $\min\{\alpha + \vartheta', \pi/2\}$. Similarly, the angle between $|\xi'\rangle$ and elements

of $Q(\vartheta)$ varies from $|\alpha-\vartheta|$ to $\min\{\alpha+\vartheta,\pi/2\}$. Therefore

$$HQ'(\vartheta') = \bigcup_{|\alpha-\vartheta'|\le\vartheta\le\min\{\alpha+\vartheta',\pi/2\}} Q(\vartheta),$$
$$H'Q(\vartheta) = \bigcup_{|\alpha-\vartheta|\le\vartheta'\le\min\{\alpha+\vartheta,\pi/2\}} Q'(\vartheta').$$

It follows that

$$H'|\xi\rangle = Q'(\alpha),$$
$$HH'|\xi\rangle = \bigcup_{0\le\vartheta\le\min\{2\alpha,\pi/2\}} Q(\vartheta),$$
$$H'HH'|\xi\rangle = \bigcup_{0\le\vartheta'\le\min\{3\alpha,\pi/2\}} Q'(\vartheta'),$$

and so forth. Hence, acting on the vector $|\xi\rangle$ alternately with elements from H' and H sufficiently many times, we can obtain any unit vector $|\psi\rangle$. □

2.5 Complete finite bases: examples

We present in this section two remarkable results obtained by Shi.

Theorem 34 (Shi, 2002)**.** *Let S be any single-qubit real gate such that S^2 is basis-changing. Then $\{\Lambda(X), S\}$ is complete.*

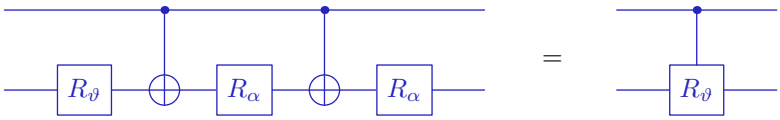
Let start from consideration of 2-dimensional orthogonal case. Any operator in $\mathbf{O}(2)$ is either a rotation R_ϑ by an angle ϑ or the composition of a rotation and a reflection. The reflection can be fixed. Moreover, for $U \in \mathbf{O}(2)$ the square U^2 is always rotation.

If ϑ/π is rational then $\langle R_\vartheta\rangle$ is finite. Otherwise $\langle R_\vartheta\rangle \approx \mathbf{SO}(2)$. Hereinafter we use notation $\langle\mathcal{B}\rangle \approx G$ to express the fact that $\mathcal{B}$ generates a dense subgroup in the group G.

So, if ϑ/π is irrational then any rotation can be approximated by powers of S. In this case Theorem 34 follows from the theorem by Barenco etc.

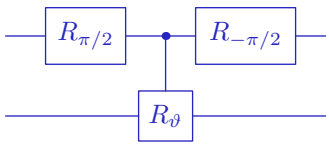
Lemma 35. *The basis $\{\Lambda(X), R_\alpha\}$, $0 \le \alpha < 2\pi$ is complete.*

Proof. Show that controlled rotation can be realized in this basis. It is easy to check that



if $\vartheta + 2\alpha = 2\pi$.

Circuits in the form



generate a dense subgroup in $\mathbf{O}(L_1)$, where $L_1 = \mathbb{R}(|00\rangle, |01\rangle)$. (We enumerate qubits from top to below.)

Similar circuits generate a dense subgroup in $\mathbf{O}(L_2)$, where $L_2 = \mathbb{R}(|00\rangle, |10\rangle)$. (Interchange qubits.)

Thus, by Lemma 32 we get

$$\langle \Lambda(X), R_\alpha \rangle \approx \mathbf{O}(L), \qquad \text{where } L = \mathbb{R}(|00\rangle, |10\rangle, |11\rangle)).$$

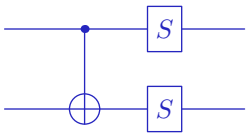
The group $\mathbf{O}(L)$ is the stabilizer of $\mathbb{R}(|11\rangle)$ in the whole state space of two qubits. The operator $\Lambda(X)[1,2]$ does not fix $\mathbb{R}(|11\rangle)$. Applying Lemma 32 once more we come to the result. □

For the case of rational ϑ, we need a number-theoretic lemma by Włodarski.

Lemma 36 (Włodarski, 1969)**.** *If α is not an integer multiple of $\pi/4$, and $\cos\beta = \cos^2\alpha$, then either α or β is an irrational multiple of π.*

Proof of Theorem 34 for the case of finite order gate. Let S be a rotation by an angle ϑ (the other case is treated similarly). We assume that ϑ/π is rational.

Let find eigenvalues of the operator W realized by the circuit



By direct calculation we see that in the computational basis the operator is represented by the matrix

$$W = \begin{pmatrix} \cos^2\vartheta & -\cos\vartheta\sin\vartheta & -\cos\vartheta\sin\vartheta & \sin^2\vartheta \\ \cos\vartheta\sin\vartheta & \cos^2\vartheta & -\sin^2\vartheta & -\cos\vartheta\sin\vartheta \\ \cos\vartheta\sin\vartheta & -\sin^2\vartheta & \cos^2\vartheta & -\cos\vartheta\sin\vartheta \\ \sin^2\vartheta & \cos\vartheta\sin\vartheta & \cos\vartheta\sin\vartheta & \cos^2\vartheta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} =$$

$$\begin{pmatrix} \cos^2\vartheta & -\cos\vartheta\sin\vartheta & \sin^2\vartheta & -\cos\vartheta\sin\vartheta \\ \cos\vartheta\sin\vartheta & \cos^2\vartheta & -\cos\vartheta\sin\vartheta & -\sin^2\vartheta \\ \cos\vartheta\sin\vartheta & -\sin^2\vartheta & -\cos\vartheta\sin\vartheta & \cos^2\vartheta \\ \sin^2\vartheta & \cos\vartheta\sin\vartheta & \cos^2\vartheta & \cos\vartheta\sin\vartheta \end{pmatrix}$$

In the basis

$$|e_1\rangle = \begin{pmatrix} -1 \\ 1 \\ 1 \\ 1 \end{pmatrix}; \quad |e_2\rangle = \begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \end{pmatrix}; \quad |e_3\rangle = \begin{pmatrix} 1 \\ 1 \\ -1 \\ 1 \end{pmatrix}; \quad |e_4\rangle = \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \end{pmatrix}$$

the matrix of the operator is

$$\begin{pmatrix} c & 0 & s & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -s & 0 & c & 0 \end{pmatrix}, \qquad \text{where } c = \cos(2\vartheta),\ s = \sin(2\vartheta).$$

Thus, $|e_2\rangle$ is an eigenstate with eigenvalue 1. The characteristic polynomial on the orthogonal complement to $|e_2\rangle$ is computed directly:

$$\begin{vmatrix} c-\lambda & s & 0 \\ 0 & -\lambda & 1 \\ -s & c & -\lambda \end{vmatrix} = -\lambda^3 + c\lambda^2 + c\lambda - 1 = -(\lambda+1)(\lambda^2 - (1+c)\lambda + 1)$$

The eigenvector $|\xi_1\rangle$ corresponding to the eigenvalue -1 is easily computed:

$$|\xi_1\rangle = s|e_1\rangle - (c+1)|e_3\rangle + (c-1)|e_4\rangle.$$

In the orthogonal complement $L_1^\perp$ to the subspace $L_1 = \mathbb{R}(|e_2\rangle, |\xi_1\rangle)$ the operator W acts as a rotation by an angle α such that

$$\cos\alpha = \frac{1+\cos(2\vartheta)}{2} = \cos^2\vartheta.$$

We assume that ϑ/π is rational but ϑ is not integer multiple of $\pi/4$. Włodarski's lemma implies that α/π is irrational.

The rest of the proof is repeated applications of Lemma 32. The powers of W^2 generate a dense subgroup in the stabilizer of $|\xi_1\rangle$ in the subspace $L_2 = \mathbb{R}(|e_1\rangle, |e_3\rangle, |e_4\rangle)$.

Note that the operator $\Lambda(X)[1,2]$ preserves $|e_2\rangle$ but does not preserve the subspace $\mathbb{R}(|\xi_1\rangle)$. (The operator transposes $|e_2\rangle$ and $|e_4\rangle$ and fixes $|e_1\rangle$ and $|e_2\rangle$.)

By Lemma 32 we conclude that the group $G = \langle \Lambda(X), S\rangle$ generates a dense subgroup in $\mathbf{O}(L_2)$.

The operator $\Lambda(X)[2,1]$ does not preserve the subspace $\mathbb{R}(|e_2\rangle)$. Applying Lemma 32 once more we get the result. □

The second theorem of Shi claims that Toffoli gate $\Lambda^2(X)$ and the Hadamard gate H form a complete basis.

Theorem 37 (Shi, 2002)**.** $\{\Lambda^2(X), H\}$ *is complete.*

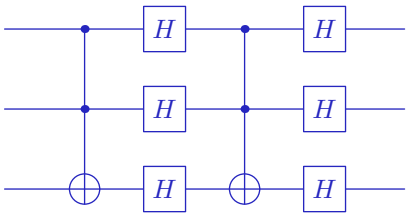
The theorem is proved by similar arguments as Theorem 34. For the sake of completeness we include the proof.

Proof. We denote the group generated by $\Lambda^2(X)$ and H by G.

The first step is to construct an operator of rotation by an angle incommensurable with π. The operator

$$U = (H^{\otimes 3}\Lambda^2(X)[1,2,3])^2 = (H^{\otimes 3}\Lambda^2[1,2,3](X)H^{\otimes 3})\Lambda^2(X)[1,2,3]$$

fits the purpose. It is represented by a circuit



The operator $\Lambda^2[1,2,3]$ is a reflection about the hyperplane orthogonal to the vector

$$|\xi_7\rangle = \frac{1}{\sqrt{2}}(|110\rangle - |111\rangle).$$

The operator $H^{\otimes 3}\Lambda^2[1,2,3](X)H^{\otimes 3}$ is conjugated to $\Lambda^2[1,2,3]$. So it is also a reflection about the hyperplane orthogonal to the vector

$$|\xi_8\rangle = H^{\otimes 3}|\xi_7\rangle = \frac{1}{2}(|001\rangle - |011\rangle - |101\rangle + |111\rangle).$$

Thus U is a rotation by an angle α such that

$$\cos\frac{\alpha}{2} = \langle\xi_7|\xi_8\rangle = -\frac{1}{2\sqrt{2}}\,.$$

It implies that $\cos\alpha = -3/4$. The non-unit eigenvalues $\lambda_\pm$ of U are the roots of equation

$$\lambda^2 - \frac{3}{2}\lambda + 1 = 0.$$

It implies that $\lambda_\pm$ are not algebraic integers and α/π is irrational.

Thus G contains a dense subgroup in $\mathbf{O}(L)$, where $L = \mathbb{R}(\xi_7, \xi_8)$. The subgroup acts trivially on the orthogonal complement to L.

We are going to apply Lemma 32 repeatedly. For this purpose we choose a basis in $L^\perp$:

$$\begin{aligned} |\xi_1\rangle &= |000\rangle, & |\xi_4\rangle &= |001\rangle + |011\rangle, \\ |\xi_2\rangle &= |010\rangle, & |\xi_5\rangle &= |101\rangle + |110\rangle + |111\rangle, \\ |\xi_3\rangle &= |100\rangle, & |\xi_6\rangle &= |011\rangle - |101\rangle \end{aligned}$$

and a set of operators

$$\begin{aligned} U_1 &= H[3], & U_4 &= \Lambda^2(X)[2,3,1], \\ U_2 &= U_1\Lambda^2(X)[2,3,1]U_1, & U_5 &= U_1\Lambda^2(X)[1,2,3]U_1, \\ U_3 &= U_1\Lambda^2(X)[1,3,2]U_1, & U_6 &= \Lambda^2(X)[1,3,2]. \end{aligned}$$

By direct calculations one can check that each U_i, $1 \le i \le 6$, preserves $\mathbb{R}(|\xi_j\rangle : 1 \le j < i)$ and does not preserve $\mathbb{R}(|\xi_i\rangle)$. Thus

$$\langle U, U_i, \ldots, U_6\rangle \approx \mathbf{O}(\mathbb{R}(|\xi_j\rangle : i \le j \le 8))$$

and the theorem follows. □

2.6 *k*-fold forrelations

Typically a BQP-complete problem is a question about exponentially large sums with a promise that prevents the problem to be complete in a wider class PP.

In this section we present an interesting example of BQP-complete problem. The problem is related to oracle separation results and might be useful in further investigations of power of the class BQP.

For Boolean functions $f_1, \ldots, f_k \colon \{0,1\}^n \to \{0,1\}^n$, a k-fold forrelation is the quantity

$$\Phi_{f_1,\ldots,f_k} = \frac{1}{2^{(k+1)n/2}} \sum_{x,\ldots,x_k \in \{0,1\}^n} (-1)^{f_1(x)}(-1)^{x_1 \cdot x_2}(-1)^{f_2(x)} \ldots (-1)^{x_{k-1}\cdot x_k}(-1)^{f_k(x)}.$$

Definition 20. k-FOLD FORRELATION.

Input: k Boolean circuits $C_1, \ldots, C_k$, which compute the Boolean functions $f_j \colon \{0,1\}^n \to \{0,1\}^n$.

Promise: either $\Phi_{f_1,\ldots,f_k} \le 1/100$ or $\Phi_{f_1,\ldots,f_k} \ge 3/5$.

Question: decide whether $\Phi_{f_1,\ldots,f_k} > 1/2$.

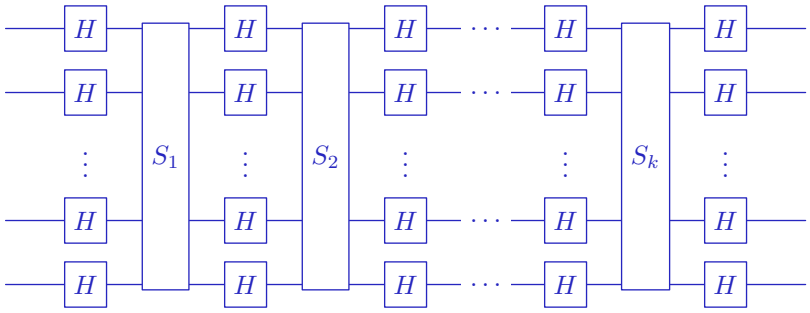
Theorem 38 (Aaronson, Ambainis, 2014)**.** k-FOLD FORRELATION *is* BQP*-complete.*

This problem has a typical form for BQP-complete problems. It is a question about the value of an exponentially large sum. In this problem all terms are ± 1 and the rule of computing of the sign is given by a sequence of Boolean circuits. It looks very close to the definition of the class PP. Nevertheless, the sum has a special form. In particular, its value is at most $2^{(k+1)n/2}$ (it will be clear from the proof below). Also, the promise of the problem can be amplified to make the bounds of $\Phi_{f_1,\ldots,f_k}$ exponentially close to ± 1. For general PP problems this amplification property does not known.

On the other hand, the problem seems to be computationally hard for classical models of computation. Moreover, one can introduce *restricted Hadamard depth* problems by restriction on the value k. Actually, the computational complexity of the problem with $k = 1$ is not known.

To prove the theorem we should to prove two facts: (i) k-FOLD FORRELATION lies in BQP and (ii) any BQP-complete problem is reduced to k-FOLD FORRELATION.

The first fact is easy to see. Indeed, $\Phi_{f_1,\dots,f_k}$ is a matrix element $\langle 0^n|U|0^n\rangle$ of an operator U realized by the circuit



Controlled sign operators S_j in this circuit are diagonal in the computational basis and change sign according to the value of f_j:

$$S_j|x\rangle = (-1)^{f(x)}|x\rangle.$$

In this way the k-FOLD FORRELATION problem is reduced to the EVALUATION OF A MATRIX ELEMENT problem. To complete a reduction we need to realize a controlled sign operator by a quantum circuit. Recall that the standard reversible operator corresponding to the function f_j, namely

$$U_j|x,b\rangle = |x, f_j(x)\oplus b\rangle,$$

is realized by a quantum circuit of polynomial size w.r.t. the size of the Boolean circuit representing the function. A controlled sign operator is realized using this circuit and an ancilla qubit:

$$S_j = X[n+1]H[n+1]U_jH[n+1]X[n+1]$$

because

$$\begin{aligned} X[n+1]H[n+1]U_jH[n+1]X[n+1]|x,0\rangle &= |x,0\rangle && \text{if } f(x)=0,\\ X[n+1]H[n+1]U_jH[n+1]X[n+1]|x,0\rangle &= -|x,0\rangle && \text{if } f(x)=1. \end{aligned}$$

Here $n+1$ is the index of the ancilla.

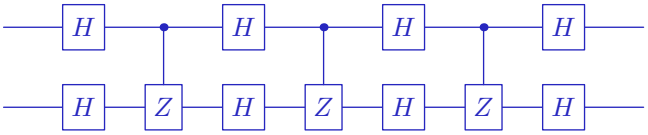
To prove BQP-hardness we apply the second theorem of Shi. The basis $\{\Lambda^2(X), H\}$ is complete. Note that

$$\Lambda^2(Z) = H[3]\Lambda^2(X)H[3]$$

so the basis $\{\Lambda^2(Z), H\}$ is also complete. So the EVALUATION OF A MATRIX ELEMENT problem restricted to the circuits in this basis is BQP-complete: any circuit can be approximated in this basis.

The operator $\Lambda^2(Z)$ is a controlled sign operator for the function $x_1x_2x_3$. It looks similar to the above circuit used to express the k-fold forrelation. The only difference is in applications of Hadamard gates. In a circuit in the basis $\{\Lambda^2(Z), H\}$ the Hadamards are applied in an arbitrary way and in the circuit expressing the k-fold forrelation Hadamards are applied to all qubits simultaneously. We call such circuits all-Hadamard circuits for brevity.

To overcome this difficulty we introduce a gadget simulating an application of Hadamards to a specific set of qubits by a circuit in which Hadamards are applied to all qubits. The gadget has the form It includes 4 Hadamards. Adding 4 Hadamards for the rest of qubits change nothing because $H^2 = I$. So the gadget can be realized by an all-Hadamard circuit.



By direct calculation we get

$$(H^{\otimes 2}\Lambda(Z)[1,2])^3 = \mathrm{SWAP},$$

where the operator SWAP transposes tensor factors:

$$\mathrm{SWAP}\,|a,b\rangle = |b,a\rangle.$$

Thus the gadget realizes the operator $\mathrm{SWAP}\circ H^{\otimes 2}$.

Swapping tensor factors is not essential. While converting a general $\{\Lambda(Z), H\}$ circuit to an all-Hadamard circuit we should only keep a permutation of tensor factors induced by applications of the gadget.

Thus, using the gadget, we are able to convert into all-Hadamard form any $\{\Lambda(Z), H\}$ circuit such that Hadamards are applied for pairs of qubits. For a general circuit we add an ancilla qubit and change an application of the Hadamard to a significant qubit by an application of H to this qubit and the ancilla. (Of course, we need to keep the track of the ancilla after gadget applications.)

The last small problem is about the parity of the number of Hadamards in the initial circuit. If this number is odd, then we apply an odd number of Hadamards to the ancilla qubit. It multiplies the value of the matrix element $\langle 0^N|U0^N\rangle$ by the factor $1/\sqrt{2}$. Recall that one can amplify the probabilities and the Evaluation of a matrix element remains BQP-complete if the promise is either $\langle 0^N|U|0^N\rangle > 1-\varepsilon$ or $\langle 0^N|U|0^N\rangle < -1+\varepsilon$ for any $\varepsilon > 0$. So the above conversion to all-Hadamard circuit should be applied to an amplified circuit. (Note that $3/5 < 1/\sqrt{2}$.)

3 Efficient quantum algortihms

A direct way to demonstrate a power of quantum computation is to suggest a problem that is hard for classical computation but admits efficient quantum algorithms to solve it.

On this way only partial success was achieved during two decades of research.

Today we discuss some known efficient quantum algorithms and their perspectives from complexity theory view.

3.1 Factoring and discrete log

Efficient quantum algorithms for factoring and discrete log were found by P. Shor in 1994. The impact of these result on the whole area of quantum computing were immense.

Definition 21. Integer factoring.

Input: a positive integer N represented in binary.

Output: the prime factorization of N: binary representations of integers p_1, α_1, p_2, α_2, $\ldots$, p_s, α_s such that

$$N = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_s^{\alpha_s}.$$

To make an output unique we assume that $p_1 < p_2 < \cdots < p_s$.

Definition 22. Discrete log.

Input: positive integers N, x, g.

Promise: x lies in the subgroup of $(\mathbb{Z}/N\mathbb{Z})^*$ generated by g.

Output: the discrete log of x with respect to g, i.e. an integer r such that

$$x = g^r \mod N.$$

Both problems are widely used in cryptography. The popular cryptographic systems such as RSA public-key cryptosystem rely on an assumption that these problems are computationally hard. A successful use of RSA is a sort of pragmatical argument in the favor of this assumption.

In the picture of complexity classes the problems lie rather low. To speak of their complexity the decision versions of the problems should be stated. As it mentioned, it can be done in the straightforward way. One can define languages

$$\text{FAC} = \{\langle N, i\rangle : i\text{th bit of the prime decomposition of } N \text{ is } 1\},$$
$$\text{DiLog} = \{\langle N, g, x, i\rangle : i\text{th bit of the } \log_g x \text{ is } 1\}.$$

Proposition 39. FAC $\in$ NP $\cap$ co-NP, DiLog $\in$ NP $\cap$ co-NP.

Proof. Inclusions FAC $\in$ NP, DiLog $\in$ NP are obvious. For the factoring problem one can choose a certificate in the form of the prime decomposition. For the discrete log problem an appropriate certificate is a binary representation of discrete log.

But the same certificates can be used to show inclusions $\overline{\text{FAC}} \in$ NP, $\overline{\text{DiLog}} \in$ NP. It follows directly from definition of languages. Knowing the prime factorization, one can efficiently verify its correctness and compute ith bit of it. The same is true for DiLog. □

The techniques used in quantum algorithms for integer factoring and discrete log were studied intensively. For many related problems efficient quantum algorithms were also found. But none of them is suitable to prove that BQP contains some reasonable classical complexity class.

In the next sections we outline two major quantum procedures used in these algorithms.

3.2 Hidden subgroup problem and Fourier sampling

Let G be a group. Group elements are represented in some prescribed way by binary strings. There is an oracle that computes a function $F\colon G \to \{0,1\}^N$ with the following property:

$$F(x) = F(y) \Leftrightarrow x - y \in H, \tag{2}$$

where $H \subseteq G$ is a subgroup (a hidden subgroup). If (2) holds we say that the oracle *hides* the subgroup H.

The goal is to find H (a set of generators of H).

A rather general form of the hidden subgroup problem is specified by actions of groups. Let $G : X \to X$ be an action of the group G and $x_0 \in X$ be a point of space X. Then the oracle

$$F(g) = gx_0$$

hides the stabilizer of x_0, i.e. a subgroup $\operatorname{Stab} x_0 = \{g \in G : gx_0 = x_0\}$. The stabilizer problem is to find $\operatorname{Stab} x_0$ for an action of a group G given.

Integer factoring and discrete log problems are reduced to the stabilizer problem.

For factorization, there is a well-known probabilistic reduction of factorization to the period finding problem.

Definition 23. Period finding.

Input: positive integers N, x such that $(N, x) = 1$.

Output: the order of x in the multiplicative group $(\mathbb{Z}/N\mathbb{Z})^*$, i.e. the smallest positive integer r such that

$$x^r = 1 \mod N.$$

The period finding to the hidden subgroup problem is straightforward: $G = \mathbb{Z}$; $X = \mathbb{Z}/N\mathbb{Z}$, $gy = y^g \mod N$. It is clear that $\operatorname{Stab} x$ is generated by the period of x modulo N.

For discrete log one should use an action of $\mathbb{Z}^2$ on the set $\mathbb{Z}/N\mathbb{Z}$ by the rule

$$(a, b)\colon y \to g^a y^b \mod N.$$

For this action $\operatorname{Stab} x$ is generated by $(-r, 1)$ and $(\varphi(N), 0)$, where φ is the Euler's totient function.

It is worth to present an example of the stabilizer problem for an non-Abelian group. Let Γ be a graph on n vertices. The symmetric group S_n acts on n-vertex graphs by permutations of vertices. The stabilizer of a graph is just the automorphism group $\operatorname{Aut}\Gamma$.

It is well-known that Graph Isomorphism problem (GI) is reduced to the finding of the automorphism group of a graph.

Thus, HSP for non-Abelian groups can be used to solve GI. It is another example of 'practically hard' problem.

The simplest form of the hidden subgroup problem is so-called Simon's problem. In this case $G = (\mathbb{Z}/2\mathbb{Z})^n$. We present an efficient quantum algorithm for the Simon's problem and then discuss a possibility to generalize it to other groups.

The group $(\mathbb{Z}/2\mathbb{Z})^n$ has a natural representation by binary strings with the group operation is represented by the bitwise XOR. We may regard G as the n-dimensional vector space over the field $\mathbb{F}_2$. Any subgroup of G is a subspace, so it can be represented by a basis.

In what follows we need also a dual subgroup H^*. It is the group

$$H^* = \{h \in G \text{ such that } h \cdot z = 0 \text{ for all } z \in H\},$$

where $h \cdot z = \sum_j h_j z_j$ is the natural inner product. The dual subgroup can be considered as the orthogonal complement of H with respect to the inner product introduced above. In another way it is the group of characters of the group G/H.

It is clear from the standard linear algebra that H can be efficiently restored from the generators of H^* and vice versa.

Lemma 40. *There is a oracle quantum algorithm for the hidden subgroup of $(\mathbb{Z}/2\mathbb{Z})^n$ running in polynomial time.*

Proof. The algorithm operates with two registers: the n-qubit query register and the m-qubit answer register. The algorithm repeats several times the following three steps.

The first step is to apply Hadamard $H^{\otimes n}$ to the first register of the initial state $|0^n\rangle \otimes |0^m\rangle$. Resulting state

$$|\psi_1\rangle = \frac{1}{N^{1/2}} \sum_{x\in\{0,1\}^n} |x\rangle \otimes |0^m\rangle,$$

where $N = 2^n$.

The second step is to apply the oracle operator to produce a state

$$|\psi_2\rangle = \frac{1}{N^{1/2}} \sum_{x\in\{0,1\}^n} |x\rangle \otimes |f(x)\rangle.$$

The third step is to apply Hadamard $H^{\otimes n}$ to the first register once more. Then measure the resulting state

$$|\psi_3\rangle = \frac{1}{N} \sum_{x,y\in\{0,1\}^n} (-1)^{x\cdot y} |y\rangle \otimes |f(x)\rangle.$$

Fortunately, the outcome of the measurement is uniformly distributed on the subgroup H^*. Let's verify this claim by calculating the probability of the outcome y:

$$\mathop{\mathbf{Pr}}_{|\psi_3\rangle}[y] = \frac{1}{N^2} \sum_a \left(\sum_{x:f(x)=a} (-1)^{x\cdot y} \right)^2 = \frac{1}{N^2} \sum_a \sum_{\substack{x',x'':\\ f(x')=f(x'')=a}} (-1)^{x'\cdot y + x''\cdot y}$$

The last sum can be rewritten using the definition of the hidden subgroup oracle and changing the order of summation:

$$\mathop{\mathbf{Pr}}_{|\psi_3\rangle}[y] = \frac{1}{N^2} \sum_{x'-x''\in H} (-1)^{(x'-x'')\cdot y} = \begin{cases} \dfrac{H}{N}, & \text{if } y \in H^*, \\ 0, & \text{otherwise.} \end{cases}$$

Thus after repeating the above steps k times the algorithm produces k samples from H^* and the samples are independent and uniformly distributed. For $k = \Omega(n)$ the samples generate H^* with high probability. We prove this claim in separate Proposition 41.

So with high probability the algorithm finds H^* and uses it to recover the hidden subgroup H. □

Proposition 41. *Let $h_1, \ldots, h_k$ be independent uniformly distributed random elements of the group $G = (\mathbb{Z}/2\mathbb{Z})^m$. Then they generate the entire group G with probability $\geq 1 - 2^{m-k}$.*

Proof. We denote the probability to generate G by $p(k)$. If $h_1, \ldots, h_k$ do not generate the whole group G then they are contained in some maximal proper subgroup $G' \subset G$. For each G' the probability of such an event does not exceed 2^{-k}, because $|G'| \leq |G|/2$. Therefore $p(k) \geq 1 - K(G) \cdot 2^{-k}$, where $K(G)$ is the number of maximal proper subgroups of the group G. The subgroups are subspaces of $\mathbb{F}_2$-vector space G. Thus the maximal proper subgroups are dual to the minimal nonzero ones

(use an inner product). Each minimal nonzero subgroup is generated by a single nonzero element, so that $K(G) < |G| - 1$. □

Now we return to the general case of the hidden subgroup problem. To generalize the algorithm for the Simon's problem we use the *Fourier transform*. Over the group $\mathbb{Z}/N\mathbb{Z}$ it is defined as

$$|x\rangle \mapsto |\hat{x}\rangle = \frac{1}{\sqrt{N}} \sum_{y \in \mathbb{Z}/N\mathbb{Z}} e^{2\pi i x y/N} |y\rangle.$$

To define the Fourier transform over an arbitrary finite group one need irreducible representations of the group. Let $\hat{G}$ be the complete set of irreducible representations of the group G. Then the Fourier transform is defined by

$$|x\rangle \mapsto |\hat{x}\rangle = \frac{1}{\sqrt{G}} \sum_{\rho \in \hat{G}} \sum_{j,k=1}^{\dim \rho} \sqrt{\dim \rho}\, \rho(x)_{jk} |\rho, j, k\rangle.$$

For an Abelian group all irreducible representations are 1-dimensional and the coefficients of the Fourier transform are just characters. For the group $(\mathbb{Z}/2\mathbb{Z})^n$ the Fourier transform is represented by $H^{\otimes n}$.

Let's discuss a possibility to generalize the Simon's algorithm to arbitrary hidden subgroup problem.

In general terms main steps in the Simon's algorithm are: (i) to produce a *coset state*

$$\sqrt{\frac{H}{G}} \sum_{x \in G:H} |xH\rangle \otimes |F(x)\rangle, \quad \text{where } |xH\rangle = \frac{1}{\sqrt{H}} \sum_{h \in H} |xh\rangle,$$

by application of an oracle; (ii) application of the Fourier transform over the group G; (iii) measure the resulting state and use samples to restore H.

The first step can be done for arbitrary group. To perform the second step one need an efficient algorithm for the Fourier transform over the group. Such algorithms do exist for Abelian groups and for some non-Abelian groups. Say, there is an efficient realization of the Fourier transform over the symmetric group S_n.

The third step can be efficiently realized for any Abelian group. For non-Abelian groups it fails in general case. Say, the hidden subgroup of S_n can not be restored from the Fourier sampling outlined above.

3.3 Phase estimation

The second idea used in efficient quantum algorithms for the problems discussed above is phase estimation.

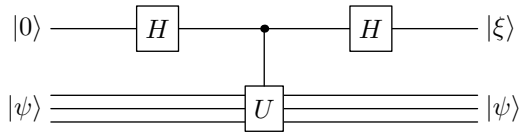
Let U be a unitary operator and $|\psi\rangle$ be an eigenstate of U

$$U|\psi\rangle = \lambda|\psi\rangle,$$

where $\lambda = \exp(2\pi i\varphi)$ is the corresponding eigenvalue.

The problem is to estimate the value of the phase φ.

We assume that U is realized by a quantum circuit of the size S. Then consider the following circuit:



An action on the control qubit is easily calculated:

$$|\xi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1 & 0\\ 0 & \lambda\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}|0\rangle = \frac{1}{2}\begin{pmatrix}1+\lambda & 1-\lambda\\ 1-\lambda & 1+\lambda\end{pmatrix}|0\rangle = \frac{1}{2}\begin{pmatrix}1+\lambda\\ 1-\lambda\end{pmatrix}.$$

So, the measurement of the control bit gives 0 with probability

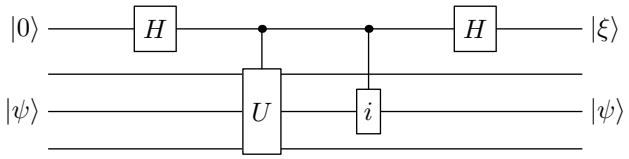
$$\mathbf{Pr}(|\xi\rangle, 0) = \left|\frac{1+\lambda}{2}\right|^2 = \frac{1}{4}\left((1+\cos(2\pi\varphi))^2 + \sin(2\pi\varphi)^2\right) = \frac{1+\cos(2\pi\varphi)}{2}.$$

It implies that application of the above circuit to several control bits and measurement these bits produces a Bernoulli distribution with probability of 0 equals to $p = \mathbf{Pr}(|\xi\rangle, 0)$.

The fraction ν of 0s in the measurement results is close to p. By Chernoff bound, we see that after s trials

$$\mathbf{Pr}\left[|\nu - p| > \delta\right] < 2e^{-2\delta^2 s}.$$

We have designed a circuit to estimate $\cos(2\pi\varphi)$. To estimate the sine a similar circuit can be applied:



After calculations we get

$$\mathbf{Pr}(|\xi\rangle, 0) = \left|\frac{1+i\lambda}{2}\right|^2 = \frac{1-\sin(2\pi\varphi)}{2}$$

for this circuit.

We conclude the analysis by the proposition

Proposition 42. *The phase φ can be estimated with additive precision δ and error probability ε by a quantum circuit of the size $O(S\delta^{-2}\log(1/\varepsilon))$.*

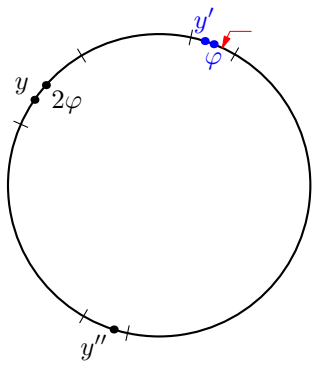
Interesting applications such as a period finding quantum algorithm and so on require the better dependence on the precision δ. To estimate a phase with exponential precision the powers of the operator U should be used. Note that the state $|\psi\rangle$ is an eigenstate for any power U^k and the corresponding eigenvalue is λ^k.

Assume that powers of U^k can be realized by quantum circuits of the size $\mathrm{poly}(S, \log k)$. Note that the assumption is strong: simple repetitions of the circuit for U give the circuits of size $\mathrm{poly}(S, k)$.

The previous proposition implies that with a very small error probability all phases $2^k\varphi$ for U^{2^k}, $1 \le k \le n$, can be estimated with some constant precision $\delta < \pi/8$ by use of polynomial size quantum circuit. From this information φ can be restored with additive precision $\pi/2^{n+3}$. For this purpose the following proposition should be repeatedly applied.

Proposition 43. *If $|y - 2\varphi| < \delta < \pi$ then either $|y' - \varphi| < \delta/2$ or $|y'' - \varphi| < \delta/2$, where y', y'' are the solutions of the equation $2x \equiv y \pmod{2\pi}$.*

To choose between two cases an $< \pi/8$ approximation of φ is enough, see a picture



How the above cosine and sine circuits work on an arbitrary state? Expand a state in use in the eigenstate basis:

$$|\psi\rangle = \sum_{k=1}^{N} c_k |\psi_k\rangle, \quad U|\psi_k\rangle = \exp(2\pi i \varphi_k)|\psi_k\rangle.$$

The eigenstates are orthogonal, so for the cosine circuit we have

$$\mathbf{Pr}(|\psi\rangle, 0) = \sum_k |c_k|^2 \frac{1+\cos(2\pi\varphi_k)}{2} = \sum_k |c_k|^2 p_k$$

and a similar equation holds for the sine circuit.

It is important that repetitions of the cosine and the sine circuits do not mix different eigenvalues. Thus with probability $|c_k|^2$ the measurement results are drawn from the Bernoulli distribution with the parameter p_k.

Thus, that to apply the efficient phase estimation procedure for a problem we need an operator U satisfying two requirements

- there are polynomial size circuits for powers U^k;
- an information to solve the problem can be extracted from phases of most eigenvalues of U.

A rather general case is an efficient action of $\mathbb{Z}^n$. An action $\mathbb{Z}^n \colon X \to X$ is efficient if $a \cdot x$ can be computed in $\mathrm{poly}(\mathrm{size}(a), \mathrm{size}(x))$ time.

Theorem 44 (Kitaev, 1995)**.** *There is a polynomial quantum algorithm to find* $\operatorname{Stab} x$ *for an efficient action of* $\mathbb{Z}^n$.

It is easy to see that the actions used in reductions of integer factorization and discrete log to the stabilizer problem are efficient. Later some other applications to number-theoretical and algebraic problems were found. Similarly to the hidden subgroup problem, none of them is complete for an interesting complexity class.

3.4 Quantum adiabatic computation

Another idea for efficient use of quantum resource was proposed by Farhi, Goldstone, Gutmann and Sipser in 2000. It is based on *quantum adiabatic theorem*.

The theorem deals with the time-dependent Hamiltonians and time evolution of their eigenstates. Assuming that evolution is slow enough the the theorem claims that the evolution of the ground state of the initial Hamiltonian is ended in the ground state of the final Hamiltonian.

In the exposition of quantum adiabatic theorem we follow the paper of Ambainis and Regev.

Let $H(t)$, $0 \le t \le 1$, be a time-dependent Hamiltonian, $|\Psi(t)\rangle$ be an eigenstate of the Hamiltonian and $\gamma(t)$ be the corresponding eigenvalue. The evolution of the eigenvalue is defined by Schroedinger equation

$$i\frac{d}{dt}|\Psi(t)\rangle = H(t)|\Psi(t)\rangle.$$

Assume that for any moment of time there is a spectral gap λ around $\gamma(t)$. It means that $|\gamma(t) - \gamma'(t)| > \lambda$ for any other eigenvalue $\gamma'(t)$ of $H(t)$.

To make an evolution slow we choose a (large) parameter T and change the Hamiltonian by $H(t/T)$.

Theorem 45 (quantum adiabatic theorem by Ambainis and Regev)**.** *If*

$$T \ge \frac{10^5}{\delta^2} \cdot \max\left(\frac{\|H'\|^3}{\lambda^4}, \frac{\|H'\| \cdot \|H''\|}{\lambda^3}\right)$$

then the final state of the adiabatic evolution started from the state $|\Psi(0)\rangle$ is δ-close to the $|\Psi(1)\rangle$.

Here $\|F\|$ is the maximum of the operator norm of an operator F on the segment $[0; 1]$.

Typical use of quantum adiabatic evolution is to solve combinatorial optimization problems. Let H_0 be a Hamiltonian with the ground state that can be easily prepared (a 'known' ground state). Let H_1 be a Hamiltonian corresponding to some combinatorial optimization problem (see examples below in Subsection 3.5.1). Time evolution is defined by a Hamiltonian

$$H(t) = (1-t)H_0 + tH_1.$$

To simulate time evolution by a quantum circuit one can approximate the adiabatic Hamiltonian $H(t/T)$ by a piece-wise constant Hamiltonian

$$\tilde{H}(t) = \left(1 - \frac{j}{T}\right) H_0 + \frac{j}{T} H_1, \qquad \text{for } j \le t < j+1.$$

For this Hamiltonian time evolution operator $U_T \colon |\Psi(0)\rangle \mapsto |\Psi(T)\rangle$ is expressed in the form

$$U_T = e^{-i\tilde{H}(T-1)} e^{-i\tilde{H}(T-2)} \dots e^{-i\tilde{H}(0)}.$$

It can be shown that if Hamiltonians H_0, H_1 are local then this operator approximates time evolution rather well. Locality of Hamiltonian means that the Hamiltonian is a sum of local terms, i.e. Hamiltonians acting on $O(1)$ qubits.

Suppose that the conditions of the quantum adiabatic theorem are satisfied. Then we get a solution of a combinatorial problem by measuring the resulting state of quantum adiabatic evolution after time T assuming that the starting state is a known ground state.

Relation between quantum adiabatic evolution and the standard model of quantum circuits have been studied. In 2005 the picture became clear. Both models are equivalent. Evolution time T is polynomially related to the size of the corresponding quantum circuit.

Thus the question is about the bounds on T for NP-complete problems. Unfortunately, all known bounds for the spectral gap in this case are exponentially small. It means that evolution is too long.

Nevertheless, the idea of quantum adiabatic algorithm appears to be very attractive for many researches. Numerical experiments were used to demonstrate that QAA performs better than classical algorithms.

From complexity theory view the experiments say nothing definitive about the power of the method. It is worth to mention that in complexity theory we deal with asymptotics and the experiments with QAA are low-scale by obvious reasons.

From practical view there is no need in a theorem if you have a solution of your problem. Thus people start designing quantum devices realizing QAA in hope that they will be competitive with respect to classical algorithms.

Even this— practical— goal is very hard to reach. Decades of research in the area of combinatorial optimization result in a lot of useful heuristic algorithms. In many cases these algorithms operate in a wide range of parameters much better than one can expect assuming P $\neq$ NP. So, the place of quantum adiabatic computation in this highly competitive area is not clear.

3.5 Quantum approximate optimization algorithm

We present in this section an algorithm proposed by Farhi, Goldstone and Gutmann in 2014. The algorithm generates approximate solutions of combinatorial optimization problems.

Recall that quantum computation differs from classical probabilistic one by distributions that can be produced. So the design of efficient quantum algorithms is in fact the search for non-trivial distributions that can be produced by quantum circuits.

QAOA implements this idea in a very straightforward way. It takes a uniform distribution on the set of solutions and transforms it to increase probabilities of better solutions.

We will apply the algorithm to Boolean CSP problems.

3.5.1 CSP and MAX-CSP

Boolean Constraint Satisfaction Problem (CSP) is defined by a set of variables $x_1, x_2, \ldots, x_n$ and a collection of *constraints*. Each constraint is specified by a subset S of variables and a Boolean function $c_S\colon \{0,1\}^r \to \{0,1\}$. Here $r = |S|$ is the size of the set S.

Possible solutions of a particular instance of CSP are assignments of values to variables in the instance. An assignment $x_1 = \alpha_1$, $x_2 = \alpha_2$, $\ldots$, $x_n = \alpha_n$ satisfies a constraint c_S if $c_s(\alpha_{j_1}, \ldots, \alpha_{j_r}) = 1$, where $S = \{j_1, \ldots, j_r\}$.

In the CSP the question about existence of a consistent assignment, i.e. an assignment that satisfies all constraints. The CSP problem is in the class NP and covers many particular NP-complete problems.

Example 1. 3-SAT problem is the question about satisfiability of 3-CNF. CNF is a conjunction of clauses. A clause is a disjunction of literals (variables and negations of variables).

Disjunctions can be viewed as constraints. It is well-known that 3-SAT is NP-complete.

In the MAX-CSP the goal is to maximize the number of satisfied constraints. To speak of computational complexity of an optimization problem we assume the standard conversion to a decision form. Namely, the optimization problem $f(x) \to \max$ has a decision form 'is there x such that $f(x) > t$?'. Here the input of the problem is the pair (x, t).

Many particular cases of MAX-CSP are NP-complete while the corresponding CSP problems lie in the class P.

Example 2. MAX-CUT problem is to find a cut in a graph of maximal size.

Recall that a cut in a graph is specified by a partition of the vertex set into two parts: $V = V_0 \sqcup V_1$. The cut contains edges of the graph connecting different parts.

To convert MAX-CUT into MAX-CSP let's introduce Boolean variables x_v, $v \in V$, for the vertices of the graph. The value 0 of x_v indicates that $v \in V_0$ and

the value 1 indicates that $c \in V_1$. Constraints are specified by edges and a Boolean function $f(\alpha, \beta) = \alpha \oplus \beta$. The function takes value 1 iff $\alpha \neq \beta$. So the number of satisfied constraints is the size of the corresponding cut.

A graph admits a consistent assignment iff it is bipartite. So, this variant of a CSP problem is in P.

But the decision form of MAX-CUT is NP-complete.

Example 3. An instance of MAX3LIN problem is a system of Boolean linear equations. There are exactly 3 variables in each equation. The goal is to maximize the number of satisfied equations.

Again, CSP version of the problem lies in P. But MAX3LIN is NP-complete.

Even approximate the optimum in a MAX-CSP might appear to be computationally hard. Let's recall the results of this sort about MAX-CUT and MAX3LIN. We way that an algorithm for a maximization problem with the objective function $f(x)$ gives a c-approximation if it produces a solution x such that

$$c \max_y f(y) \leq f(x) \leq \max_y f(y).$$

Theorem 46 (Goemans, Williamson, 1995; Håstad, 2001)**.** *There exists a polynomial time algorithm that gives α_{GW}-approximation ration for MAX-CUT, where*

$$\alpha_{\mathrm{GW}} = \min_{0<\theta<\pi} \frac{\theta/\pi}{(1-\cos\theta)/2} \approx 0.87856.$$

Assuming P $\neq$ NP *no polynomial time algorithm gives* 16/17*-approximation ratio for MAX-CUT.*

Theorem 47 (Håstad, 1997)**.** *Assuming* P $\neq$ NP *for any $\varepsilon > 0$ no polynomial time algorithm gives $(1/2 + \varepsilon)$-approximation ratio for MAX3LIN.*

3.5.2 The general framework of QAOA

An instance of MAX-CSP is specified by variables and constraints. Let n be the number of variables and m be the number of constraints.

The algorithm operates with n qubits.

Let introduce the *objective function operator*. It is a Hermitian operator $C = \sum_{j=1}^{m} C_j$, where $C_j|x\rangle = |x\rangle$ if the constraint j is satisfied on the assignment x and $C_j|x\rangle = 0$ otherwise. The operator is diagonal in the computational basis and $C|x\rangle = f(x)|x\rangle$, where $f(x)$ is the objective function.

The algorithm also uses the operator $B = \sum_k X_j$, where $X_{[j]}$ is the Pauli operator X applied to the jth qubit.

At first, the algorithm prepares the state

$$|s\rangle = \frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} |x\rangle.$$

Note that measuring $|s\rangle$ produces the uniform distribution on the set of assignments. The algorithms tries to improve the distribution acting by operators $e^{-i\gamma C}$ and $e^{-i\beta B}$.

After p rounds the resulting state is

$$|\gamma, \beta\rangle = \prod_{\ell=1}^{p} e^{-i\beta_\ell B} e^{-i\gamma_\ell C} |s\rangle.$$

Then the algorithm measures the state $|\gamma, \beta\rangle$ and output the result.

Parameters p, β_ℓ, γ_ℓ should be chosen to specify the exact form of the algorithm.

Of course, the best choice of the angles β_ℓ, γ_ℓ is to maximize the expectation of the operator C on the resulting state $|\gamma, \beta\rangle$. For a fixed p we denote by M_p the best result achieved by a choice of the angles:

$$M_p = \max_{\gamma,\beta} \langle \gamma, \beta | C | \gamma, \beta \rangle.$$

If $p = O(1)$ then the best values of the angles β_ℓ, γ_ℓ can be found by brute force search on an appropriate grid of possible values.

Realization of the algorithm by quantum circuits is rather simple if each constraint includes $O(1)$ variables. In this case the operator C is local. Note that the above examples MAX-CUT and MAX3LIN satisfy this requirement.

3.5.3 Relation to adiabatic computation

As it mentioned the idea behind the algorithm is to increase probabilities of better assignments. The idea stems from a relation of QAOA to quantum adiabatic computation.

Note that $|s\rangle$ is the eigenstate of B with the maximal eigenvalue n. Indeed, the eigenvalues of X are ± 1 and the eigenstate for the eigenvalue 1 is $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. It is easy to see that

$$|s\rangle = \bigotimes_{k=1}^{n} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle).$$

So $B|s\rangle = n|s\rangle$.

Starting in $|s\rangle$ we run the quantum adiabatic algorithm with the time-dependent Hamiltonian $H(t/T) = (1 - t/T)B + (t/T)C$. Hopefully the evolution will finish near the state with the maximum eigenvalue of C.

Using Trotter formula

$$e^{t(A+B)} = \lim_{q\to\infty} \left(e^{tA/q} e^{tB/q}\right)^q,$$

one can approximate the final state of the adiabatic algorithm by application of long chain of operators $e^{-i\beta_\ell B} e^{-i\gamma_\ell C}$.

From these observations it is natural to conjecture the convergence

$$\lim_{p\to\infty} M_p = \max_x C(x).$$

3.5.4 The case of fixed p

The convergence result shows that a variant of QAOA is just an adiabatic computation.

But the case of $p = O(1)$ is interesting in itself. In this case convergence to the optimum does not hold. Actually we disturb slightly the uniform distribution and it would give a rather small increase in expectation value.

But regarding the inapproximability results states above this small advantage might be very important. QAOA was suggested less than an year ago. So we know a little about it up to the moment.

A partial success was achieved in the analysis of the bounded occurrence case. In this case each variable participates in $O(1)$ constraints. For the MAX-CUT problem it means that a graph has bounded degree. For the MAX3LIN problem it means that in addition to the requirement that every equation contains a few variables (namely, 3) we put the requirement that every variable occurs in a few equations.

Farhi, Goldstone and Gutmann have analyzed the bounded occurrence case of MAX3LIN. For this case it is also known that the approximation is computationally hard. Let every variable occurs in at most D equations.

Theorem 48 (Trevisan, 2001). *Assume that* P $\neq$ NP. *Then there is a constant* K *such that no polynomial time algorithm gives* $(1/2 + K/D^{1/2})$*-approximation ratio for MAX3LIN with* D*-bounded occurrences.*

When Farhi etc. published their work the best known approximation ratio for this problem was $1/2 + K/D$. They proved that QAOA provides the better ratio.

Theorem 49 (Farhi, Goldstone and Gutmann, 2014). *For sufficiently large* D *and a constant* K *QAOA with* $p = 1$ *ensures* $(1/2 + C/(D+1)^{3/4})$*-approximation for the problem MAX3LIN with* D*-bounded occurrences.*

Actually they proved that a fraction $1/2 + K/(D+1)^{3/4}$ of equations can be satisfied in any linear system with 3 variables in each equation and each variable occurs in at most $D+1$ equations. It is easy to see that usual randomized argument gives the fraction $1/2$.

This work was the first example of formal result in hardness approximation area achieved by use of quantum algorithms.

Recently this result was beaten by the work of 10 authors: Barak, Moitra, O'Donnel, Raghavendra, Regev, Steurer, Trevisan, Vijayaraghavan, Witmer and Wright.

Theorem 50 (the optimal bound). *For any odd* k *in any MAX-kLIN system with* D*-bounded occurrences the fraction* $1/2 + \Omega(1/D^{1/2})$ *of equations can be satisfied and they can be found by an efficient classical algorithm.*

Below we outline the proof of Theorem 49.

Equation $x_a + x_b + x_c = b$ contributes the term

$$\frac{1}{2}(1 + (-1)^b Z_a Z_b Z_c)$$

into the objective function operator. Here Z is the Pauli operator.

Computing the expectation of the objective operator

$$\langle \gamma, \beta | C | \gamma, \beta \rangle$$

one can extract the constant $1/2$. The corresponding term is just the half of equations in the system. In the sequel we use the objective function operator in the form

$$C = \sum_{a,b,c} d_{abc} Z_a Z_b Z_c.$$

Let $\beta = \pi/4$, $\gamma = -K/D^{3/4}$, where K is a constant. Then the expected value of C per equation is at least $\Omega(1/D^{3/4})$ and the theorem follows.

We analyze local terms in the operator C separately. For an equation $x_1 + x_2 + x_3 = b$ the expectation of the corresponding term is

$$(-1)^{d_{123}} \langle s | e^{i\gamma C} e^{i\beta B} Z_1 Z_2 Z_3 e^{-i\beta B} e^{-i\gamma C} | s \rangle.$$

In the operator B all terms except $X_1 + X_2 + X_3$ commute with $Z_1 Z_2 Z_3$ and canceled. From $\beta = \pi/4$ we get

$$e^{i\beta X} Z e^{-i\beta X} = Y.$$

So we rewrite the expectation in the form

$$\pm \langle s | e^{i\gamma C} Y_1 Y_2 Y_3 e^{-i\gamma C} | s \rangle.$$

The operator C contains the term $\pm\frac{1}{2} Z_1 Z_2 Z_3$. Let $\widetilde{C}$ be the sum of all terms in C that contain at least one of the operators Z_1, Z_2, Z_3. All other terms commute with $Y_1 Y_2 Y_3$ and cancelled.

Computing the conjugation of $Y_1Y_2Y_3$ by $e^{\pm i\gamma\frac{1}{2}Z_1Z_2Z_3}$, we get

$$\pm\frac{1}{2}\langle s|e^{i\gamma\widetilde{C}}(\cos(2\gamma)Y_1Y_2Y_3 \mp \sin(2\gamma)X_1X_2X_3)e^{-i\gamma\widetilde{C}}|s\rangle.$$

There are two terms in the last expression. The second term with $X_1X_2X_3$ has the coefficient $-\frac{1}{2}\sin\gamma$. For the chosen value of γ the coefficient is positive. The value

$$\langle s|e^{i\gamma\widetilde{C}}X_1X_2X_3e^{-i\gamma\widetilde{C}}|s\rangle$$

is lowerbounded by $1-\frac{3}{2}D\gamma^2$.

On the other hand the first term is

$$O(\cos\gamma|\gamma|^3D^{3/2})$$

and the result follows after elementary calculations.

How to get the bounds on these terms?

The X and Y operators are off diagonal so we have

$$\langle s|e^{i\gamma\widetilde{C}}X_1X_2X_3e^{-i\gamma\widetilde{C}}|s\rangle =$$
$$\sum_{z_1,z_2,z_3}\sum_{z_1',z_2',z_3'}\langle s|e^{i\gamma\widetilde{C}}|z_1,z_2,z_3\rangle\langle z_1,z_2,z_3|X_1X_2X_3|z_1',z_2',z_3'\rangle\langle z_1',z_2',z_3'|e^{-i\gamma\widetilde{C}}|s\rangle =$$
$$\sum_{z_1,z_2,z_3}\langle s|e^{i\gamma\widetilde{C}}|z_1,z_2,z_3\rangle\langle\bar{z}_1,\bar{z}_2,\bar{z}_3|e^{-i\gamma\widetilde{C}}|s\rangle. \quad (3)$$

Let expand the operator $\widetilde{C}$ in the form

$$\widetilde{C} = Z_1C_1 + Z_2C_2 + Z_3C_3 + Z_1Z_2C_{12} + Z_1Z_3C_{13} + Z_2Z_3C_{23},$$

where C_j is the sum of terms $\pm\frac{1}{2}Z_aZ_b$ coming from equations on the variables j, a, b, $a,b\notin\{1,2,3\}$, $j\in\{1,2,3\}$, and C_{jk} is the sum of terms $\pm\frac{1}{2}Z_a$ coming from equations on the variables j, k, b, $a\notin\{1,2,3\}$, $j,k\in\{1,2,3\}$.

Now we simplify the sum (3). Recall that

$$|s\rangle = \bigotimes_j\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle).$$

Tensor factors corresponding the variables that do not occur in equations counted in $\widetilde{C}$ contribute the factor 1 to each term of the sum (3). Now expand the tensor product in factors 1, 2, 3. We get

$$\frac{1}{8}\sum_{z_1,z_2,z_3}\langle\tilde{s}|\langle z_1,z_2,z_3|e^{i\gamma\widetilde{C}}|z_1,z_2,z_3\rangle\langle\bar{z}_1,\bar{z}_2,\bar{z}_3|e^{-i\gamma\widetilde{C}}|\bar{z}_1,\bar{z}_2,\bar{z}_3\rangle|\tilde{s}\rangle, \quad (4)$$

where

$$|\tilde{s}\rangle = \bigotimes_{j\in J}\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$$

is taken on all variables j occuring in C_j, C_{jk}.

The terms like $Z_1Z_2C_{12}$ in the expansion of $\widetilde{C}$ cancelled in (4) because

$$\sum_{z_1,z_2,z_3}\langle z_1,z_2,z_3|e^{i\gamma Z_1Z_2C_{12}}|z_1,z_2,z_3\rangle\langle\bar{z}_1,\bar{z}_2,\bar{z}_3|e^{-i\gamma Z_1Z_2C_{12}}|\bar{z}_1,\bar{z}_2,\bar{z}_3\rangle| = I.$$

The terms Z_jC_j give $e^{2i\gamma z_jC_j}$ in the sum over z_1, z_2, z_3. So we rewrite (3) as

$$\frac{1}{8}\sum_{z_1,z_2,z_3}\langle\tilde{s}|e^{2i\gamma(z_1C_1+z_2C_2+z_3C_3)}|\tilde{s}\rangle$$

Note that the sum of $e^{2i\gamma(z_1C_1+z_2C_2+z_3C_3)}$ is expressed as the sum of cosines of $C_1\pm C_2\pm C_3$. Thus the estimation of the term $X_1X_2X_3$ is in fact an estimation

of cosines sum. The value of γ is small, so to obtain the lower bound we use the inequality

$$\cos x \geq 1 - \frac{1}{2}x^2.$$

As for $Y_1Y_2Y_3$ terms, it can be estimated in a similar way. But now a phase term $\pm i$ is appeared in the expansion (3) and we obtain a sum of sines. Using the fact that γ is small the expectations of sines can be upperbounded.

The reader can find a more detailed exposition in the original preprint `arXiv:1412.6062`.

3.5.5 Some conclusions

After the 10-author paper we have returned to the previous situation. There are reasonable quantum algorithms but they do not affect computational complexity theory.

But it is not an end of the game. The power of QAOA is not exhausted. We indicate one especially interesting direction for research. In the original version of QAOA the starting state corresponds to the uniform distribution on the set of assignments. Of course, this distribution is affected mainly by bad assignments.

There is a known technique to find more appropriate classical distributions. Take for example the MAX-CUT problem. Goemans–Williamson algorithm uses a distribution generated by solving an SDP problem. To each variable x_j the SDP solution assigns a unit vector u_j in n-dimensional Euclidean space. A random hyperplane h defines an assignment to Boolean variables according to the signs $h(u_j)$.

Goemans and Williamson have proved that the expectation of the objective function on this distribution is at least α_{GW}-optimal. So this distribution is much more appropriate to start the further optimization.

Moreover, the famous Unique Games Conjecture by Khot claims that the constant α_{GW} cannot be enlarged unless P = NP. The conjecture is about classical algorithms and it's status is unclear now. So an attempt to suggest a quantum algorithm to overcome Goemans–Williamson algorithm looks both reasonable and ambitious.

As the first step in this direction one should think about appropriate operator B for the Goemans–Willamson distribution. In other words we are interesting in a Hamiltonian with the ground state generating the Goemans–Williamson distribution. Of course, a simpler Hamiltonian makes the analysis simpler. So the question: does exist a local Hamiltonian with the ground state generating the Goemans–Williamson distribution?

4 Relativization

Let's reproduce the diagram of complexity classes[1]

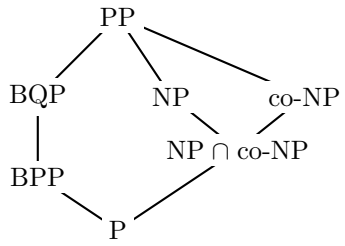


Figure 1: The complexity classes diagram revisited

Main question in the area: are the inclusions shown on the diagram proper?

The answer is far beyond modern techniques. But there are simplified versions of the main question. And we are able to answer these questions.

Generally, there are two ways to make a reasonable argument in favor of a proper inclusion claim:

- to suggest a natural candidate problem that lies in a broader class but plausibly do not lie in the narrower class;
- to make a weaker claim about inclusions and prove it.

These ways are complementary in some sense. The former makes the proper inclusion question more specified and gives a direction either to prove or to refute it. But it is a sort of informal argument and there is no idea how far from the solution we are.

The latter approach change the question. So, it is not clear what relation is between the initial question and the weakened one. On the other hand we obtain a formally correct statement about complexity classes.

Now we discuss the second approach. An informal idea behind it is to replace a 'real world' complexity class $\mathcal{C}$ by a 'relativized world A' complexity class $\mathcal{C}^A$.

So we change a definition of a complexity class and expect that the question about relativized classes has some relation to 'real world' complexity classes.

The question about proper inclusions in relativized worlds is much more tractable. But in most cases the answer is indeterminate. Choosing a relativized world one can make an inclusion between 'natural' complexity classes either proper or improper.

This way we find a first barrier to prove hardness results in computational complexity theory: if we are using techniques that works in all relativized worlds then we can not prove a hardness result.

But this negative impact on the theory is accompanied by a (much weaker) positive impact. As it becomes clear later we establish on this way a relation between complexity of computational problems and weak model of computation (query complexity).

4.1 Oracles and relativized complexity classes

To relativize complexity classes we use *oracles*. In a broad sense an oracle is a 'magic' device computing a function. Taking a computational model with restricted

[1] Actually it is a small fragment of the whole diagram of all known complexity classes, see Complexity Zoo.

resources we add to the model an access to the oracle. On this way we get a relativized model and the relativized complexity class is defined naturally.

To be more definitive let's take a Turing machine as a basic computational model. Oracle version of the Turing machine has an additional tape— *query tape*. The machine writes a string to the query tape and is able to make a query x at some moment. The string x is just the content of the query tape. The oracle gives the answer $\mathcal{O}(x)$. We assume that the answer overwrites the query on the oracle tape. The main requirement to the oracle is: $\mathcal{O}(x)$ is a function. The answer on the query is made in one step of computation.

The standard definition of an oracle is the following: $\mathcal{O}^A$ computes a characteristic function of a language A:

$$\mathcal{O}(x) = \begin{cases} 1, & \text{if } x \in A, \\ 0, & \text{otherwise.} \end{cases}$$

The above definition of oracle Turing machine gives an easy way to define relativized version of usual complexity classes such as P, NP, co-NP, BPP. Say, P^A contains those languages that can be recognized by an A-oracle Turing machine running in polynomial time.

Example 4. Let A be an NP-complete language. Then $A \in \mathrm{P}^A$ and $\bar{A} \in \mathrm{P}^A$.

A more general way to define oracle complexity classes is to use oracles that form a complexity class themself. Say, P^{NP} contains those languages that can be recognized by a A-oracle Turing machine running in polynomial time and $A \in \mathrm{NP}$. The previous example can be restated in these notation as

Proposition 51. $\text{co-NP} \in \mathrm{P}^{\mathrm{NP}}$.

Definition 24. An inclusion $\mathcal{C}_1 \subseteq \mathcal{C}_2$ is *relativized* if $\mathcal{C}_1^A \subseteq \mathcal{C}_2^A$ holds for any oracle A.

Example 5. The inclusion $\mathrm{P} \subseteq \mathrm{NP}$ is relativized. To see it we should specify definition of a relativized class NP^A.

The straightforward way is to change the requirement to the verifier to be in P by the requirement to the verifier to be in P^A.

Then the claim becomes trivial.

Now we are ready to make the main statement about complexity classes inclusions in relativized worlds.

Theorem 52. *(1) All inclusions on the Figure 1 are relativized.*
(2) Opposite inclusions are not relativized.

Later we prove partially this statement. Our main interest in the quantum class BQP. But it is very useful to start from oracle separations of the classical classes.

4.2 Relativized P vs NP question

Theorem 53. *There exists an oracle A such that $\mathrm{P}^A = \mathrm{NP}^A$.*
There exists an oracle B such that $\mathrm{P}^B \subset \mathrm{NP}^B$ (the inclusion is proper).

The theorem was proved by Baker, Gill, and Solovay in 1975. We present the proof that was suggested by Bennett and Gill in 1981.

The first claim in the theorem is easier. To prove it we need to introduce a new complexity class PSPACE. It is a space complexity class. We put a restriction on the space used by an algorithm. For Turing machines the quantitative measure of space is clear: it is the number of tape cells visited by a machine during the computation.

Definition 25. $L \in$ PSPACE if there exists a Turing machine M recognizing L and using poly(n) space.

Recall a well-known fact from complexity theory.

Proposition 54. *There are* PSPACE*-complete languages under polynomial time reductions.*

Now we take a PSPACE-complete language as an oracle A.

Claim 55. $\mathrm{P}^A = \mathrm{PSPACE}$.

Proof. In one direction it is just a reformulation of a polynomial time reduction. Let L be a PSPACE language. Take a function f that reduces L to A. By definition

$$x \in L \Leftrightarrow f(x) \in A.$$

So A-oracle algorithm running in polynomial time and recognizing the language L computes $f(x)$, queries the oracle $f(x)$ and returns the oracle answer. So PSPACE $\subseteq$ P^A.

In the other direction, we replace oracle queries by subroutines computing the oracle answer in polynomial space. □

Claim 56. $\mathrm{NP}^A = \mathrm{PSPACE}$.

Proof. Let $L \in \mathrm{NP}^A$. By definition it means that there exist $V \in \mathrm{P}^A$ and a polynomial $q(\cdot)$ such that

$$x \in L \Rightarrow \exists y : |y| = p(|x|) \wedge V(x, y),$$
$$x \notin L \Rightarrow \forall y \; |y| = p(|x|) \Rightarrow V(x, y).$$

PSPACE algorithm recognizing L tries all possible certificates y of the length $p(|x|)$. For each certificate y the algorithm computes $V(x, y)$ in polynomial space. If it finds a certificate such that $V(x, y)$ is true then it answers in the positive. Otherwise it answers in the negative. It is clear that the whole process uses polynomial space (and exponential time but now we have no time restrictions). Thus $\mathrm{NP}^A \subseteq$ PSPACE.

The opposite direction follows immediately from the previous claim:

$$\mathrm{PSPACE} = \mathrm{P}^A \subseteq \mathrm{NP}^A$$

the latter inclusion is relativized. □

These two claims show us that $\mathrm{P}^A = \mathrm{NP}^A$ for a PSPACE-complete oracle A. Actually one can say more about PSPACE-complete oracles. Using such an oracle collapses the whole diagram of complexity classes.

Claim 57. PP $\subseteq$ PSPACE *and this inclusion is relativized.*

Proof. Let $L \in$ PP. By definition there exists a probabilistic algorithm M recognizing L with error probability $< 1/2$ and running in polynomial time. It implies that the algorithm uses a polynomially many random bits.

PSPACE algorithm recognizing L tries all possible values of random bits used by M and compute the probability of accepting answer. It is clear that polynomial space is enough for the process. Also the process remains valid relative to any oracle. □

Claim 58. $\mathrm{PSPACE}^{\mathrm{PSPACE}} = \mathrm{PSPACE}$.

Proof. Let A be a PSPACE-complete oracle. It is sufficient to show that $\mathrm{PSPACE}^A \subseteq \mathrm{PSPACE}$.

Let's replace oracle queries in the A-oracle algorithm by subroutines computing the query answers. Computing each answer takes a polynomial space and the space can be reused for subsequent queries. □

Thus we have the equalities of complexity classes relative to a PSPACE oracle A:

$$\mathrm{P}^A = \mathrm{NP}^A = \text{co-}\mathrm{NP}^A = \mathrm{BPP}^A = \mathrm{BQP}^A = \mathrm{PP}^A = \mathrm{PSPACE}^A.$$

It should be noted that the most of people believe that this relativized world differs drastically from the 'natural' world (the empty oracle). They conjecture that all these classes are different!

The next step is to prove a harder part of Theorem 53. We should present an oracle B such that $P^B \subset \mathrm{NP}^B$. It can be done in several ways. One way is to use diagonalization technique and define the oracle in question step-by-step to fool all polynomial time oracle algorithms.

But we prefer an another way and use *random oracles*. This technique will be useful in oracle separations of the quantum class BQP.

Let's explain an intuition behind the proof. We are going to take an oracle B drawn from a suitable probabilistic distributions on the set of oracles. The goal is to ensure that any B-oracle polynomial time algorithm can not simulate B-oracle nondeterministic polynomial time algorithm with high probability.

Also we change oracles to be used. Namely, we will use in the constructions *length-preserving oracles*.

Definition 26. A function $B\colon \{0,1\}^* \to \{0,1\}^*$ is a length-preserving oracle if $|B(x)| = |x|$ for all $x \in \{0,1\}^*$.

Again we assume that the B-oracle gives the answer to a query in one unit of time.

Using length-preserving oracles does not change the relativized worlds in the part concerning the complexity classes into consideration. It is quite clear that a usual Boolean function oracle can be simulated a length-preserving one: just ignore bits of answer except the first bit. Also there is a simple way to transform a length-preserving oracle to a Boolean function one.

For this purpose let's define a language B' for a length-preserving oracle B as follows:

$$x \in B' \Leftrightarrow x = \pi(y,i) \wedge 1 \leq i \leq |y| \wedge B(y)_i = 1.$$

Here π is an efficiently computed pairing function, i.e. a bijection $\pi\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ such that π and π^{-1} are computed in polynomial time. It is easy to construct such a function.

For any reasonable complexity class $\mathcal{C}$ containing P we have

$$\mathcal{C}^B = \mathcal{C}^{B'}$$

because it is easy to reconstruct in polynomial time answers of the oracle B from the answers of the oracle B' and vice versa.

At this moment we are ready to present a language that appears to be hard for a random length-preserving oracle B (the choice of distribution will be made later).

Definition 27. Let $B\colon \{0,1\}^* \to \{0,1\}^*$ be a length-preserving oracle.

By definition $L_B = \{x \mid \exists y\ B(y) = x\}$.

Thus the language L_B is just the image of the whole set of binary strings under the map B.

Claim 59. *$L_B \in \mathrm{NP}^B$ for any length-preserving oracle B.*

Proof. Let $V(x,y) = 1$ iff $B(y) = x$. Then $V(x,y) \in \mathrm{P}^B$ by obvious reasons and the claim follows. □

We choose a probabilistic distribution on the set of length-preserving oracle such that the distribution is uniform if conditioned to the set of strings of length n for any n. We will call such a distribution *uniform*.

To finish the proof of Theorem 53 we prove that $L_B \notin \mathrm{P}^B$ relative to an oracle B drawn from a uniform distribution with probability 1.

The proof consists of several steps. The most important step is to prove the following claim.

Claim 60 (main step)**.** *Let M be a polynomial time oracle machine running in time $T(n)$ on the strings of length n. If $T(n) < 2^{n-2}$ then*

$$\mathop{\mathbf{Pr}}_B[M^B \textit{ fails on the input } 1^n] = \mathop{\mathbf{Pr}}_B[M^B(1^n) \neq \chi_{L_B}(1^n)] \geq \frac{1}{4}\,.$$

Next step is much easier.

Claim 61. *If $\mathbf{Pr}_B[M^B$ fails on the input $1^n] < 1/4$ for infinitely many n then $\mathbf{Pr}_B[L(M^B) \neq L_B] = 1$.*

Proof. Note that events '$L(M^B) \neq L_B$ when restricted to the inputs of the length n' are independent due to the choice of the distribution. □

The last step is to notice that there are countably many polynomial time oracle machines and the union of countable many events of probability 0 has the probability 0. If $T(n) = \mathrm{poly}(n)$ then $T(n) < 2^{n-2}$ for infinitely many n. So $\mathbf{Pr}_B[L_B \in \mathrm{P}^B] = 0$ q.e.d.

The last two steps will be the same in the later proofs of oracle separation results. Now we return to the main step.

Proof of the Claim 60. We fix the oracle values on the of length $\neq n$ arbitrary and denote the set of oracles having the fixed values by $\mathcal{F}$. It is sufficient to prove the claim conditioned $B \in \mathcal{F}$.

Let $\mathcal{B}$ be the set of 'bad' oracles $B \in \mathcal{F}$ such that the string 1^n does not have B-inverse:

$$\forall y\ B(y) \neq 1^n.$$

In particular, $1^n \notin L_B$ for $B \in \mathcal{B}$.

Let $\mathcal{G}$ be the set of 'good' oracles $B \in \mathcal{F}$ such that the string 1^n has the unique B-inverse y_0:

$$B(y_0) = x \quad \wedge \quad \forall y \neq y_0\ B(y) \neq 1^n.$$

In particular, $1^n \in L_B$ for $B \in \mathcal{G}$.

Now we establish a relation between bad and good oracles. For $y \in \{0,1\}^n$ and an oracle $B \in \mathcal{B}$ the oracle $B_y \in \mathcal{G}$ is defined as follows:

$$B_y(x) = \begin{cases} 1^n, & \text{if } x = y; \\ B(x), & \text{otherwise.} \end{cases}$$

It is easy to observe that if a machine M^B runs in time $T(n) \leq 2^{n-2}$ then it makes at most 2^{n-2} oracle queries.

So for $\geq 3/4$ values of y the machine running on the input 1^n does not query y. In this case M^B and M^{B_y} give the same answer. But this answer is wrong either for one machine or for another due to the choice of good and bad oracles.

Figure 2 is useful to follow the remaining calculations.

Let N_1 be the number of oracles $B \in \mathcal{B}$ such that M^B fails on the input 1^n.

Let N_2 be the number of oracles $B \in \mathcal{G}$ such that M^B fails on the input 1^n.

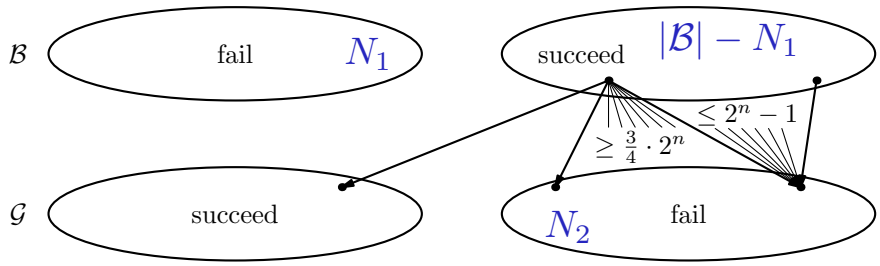


Figure 2: relation between bad and good oracles

Let N be the total number of oracles $B \in \mathcal{F}$ such that M^B fails on the input 1^n. By elementary combinatorial and calculus arguments we get

$$N_2 \geq (|\mathcal{B}| - N_1) \cdot \frac{3}{4} \cdot \frac{2^n}{2^n - 1},$$

$$N \geq N_1 + N_2 \geq N_1 + (|\mathcal{B}| - N_1) \cdot \frac{3}{4} \cdot \frac{2^n}{2^n - 1} \geq \frac{3}{4}|\mathcal{B}|,$$

$$\mathop{\mathbf{Pr}}_{B \in \mathcal{F}}[\text{fail}] \geq \frac{3}{4} \mathop{\mathbf{Pr}}_{B \in \mathcal{F}}[\mathcal{B}] = \frac{3}{4} \cdot \left(\frac{2^n - 1}{2^n}\right)^{2^n} > \frac{3}{4} \cdot \frac{1}{3} = \frac{1}{4}$$

and the claim follows. □

4.3 Quantum oracles

Let A be an oracle. We define a quantum oracle query as an application of the unitary operator

$$\mathcal{O}^A \colon |x, b\rangle \mapsto |x, A(x) \oplus b\rangle$$

Oracle quantum circuits are the quantum circuits in some universal basis with the oracle operator added.

So an oracle quantum circuit realizes the operator

$$U_T \mathcal{O}^A U_{T-1} \mathcal{O}^A U_{T-2} \ldots \mathcal{O}^A U_1,$$

where U_j are products of gates taken from some universal basis.

In definition of quantum oracle algorithms we prefer to 'hardwire' input bits into circuits.

Definition 28. Quantum oracle algorithm for a decision problem is specified by an oracle A and a (classical) TM machine M such that:

- on the input x the machine M produces a description of an oracle quantum circuit realizing a unitary operator U_x acting on m qubits;
- the measurement of the vector $U_x|0^m\rangle$ generates the probability distribution p_x on the strings of length m;
- the algorithm *accepts* x if the probability of $y_1 = 1$ is greater than $2/3$ for y drawn from the distribution p_x;
- the algorithm *rejects* x if the probability of $y_1 = 0$ is greater than $2/3$ for y drawn from the distribution p_x.

Definition of the class BQP^A follows naturally.

Definition 29. A language L is in the class BQP^A if there exists a quantum oracle algorithm M^A running in polynomial time such that:

- if $x \in L$ then M^A accepts x;
- if $x \notin L$ then M^A rejects x.

There are oracles such that the class BQP^A is not contained in classical A-oracle complexity classes. These oracles indicate a 'strength' of quantum computing. Contrary, there are oracles such that BQP^A does not contain NP^A. These oracles indicate a 'weakness' of quantum computing. We consider both cases.

4.4 BQP has a limited power in a relativized world

Theorem 62 (Bennett, Bernstein, Brassard, Vazirani, 1997)**.** $\mathrm{NP}^A \setminus \mathrm{BQP}^A \neq \varnothing$ *for a random length-preserving oracle A with probability* 1*.*

The class BQP is closed under taking the complement of a language. So this oracle separation result holds also for the relativized classes co-NP^A. Thus we obtain a corollary.

Corollary 63. $(\mathrm{NP}^A \cap \text{co-}\mathrm{NP}^A) \setminus \mathrm{BQP}^A \neq \varnothing$ *for a random length-preserving oracle A with probability* 1*.*

To prove Theorem 62 we are going to repeat the arguments from the proof of Theorem 53. The general framework remains the same. In particular, we will use the same language $L_B = \{x \mid \exists y\ B(y) = x\}$ and will prove that the language is hard for oracle BQP algorithms. Also we will use the same classes of oracles $\mathcal{F}$, $\mathcal{B}$ and $\mathcal{G}$ in the proof.

But now we should develop a different approach to the main step in the proof (Claim 60).

In the previous argument it was clear that the behavior of a B-oracle algorithm and a B_y-oracle algorithm is the same if the algorithms do not ask query y. A quantum algorithm can query in superpositions. So a 'closeness' of the B-oracle algorithm and the perturbed B_y-oracle algorithm is plausible but needs extra argument to be verified. Time bounds for these arguments differs from the classical ones.

More definitely, for each oracle $B \in \mathcal{B}$ we are going to find many strings y satisfying the property: if M^B succeeds on 1^n then M^{B_y} fails on 1^n. For this purpose we will use the fact that unitary operators $\mathcal{O}^B$ and $\mathcal{O}^{B_y}$ are close and generate close distributions.

To state formally the claim about the closeness we introduce some notation. Let

$$U = U_T \mathcal{O} U_{T-1} \mathcal{O} U_{T-2} \dots \mathcal{O} U_1$$

be the oracle circuit produced by the machine M on the input 1^n.

Let's define vectors

$$|\psi_j\rangle = U_j \mathcal{O}^B U_{j-1} \mathcal{O}^B U_{j-2} \dots \mathcal{O}^B U_1 |0^m\rangle,$$
$$|\psi_j^y\rangle = U_j \mathcal{O}^{B_y} U_{j-1} \mathcal{O}^{B_y} U_{j-2} \dots \mathcal{O}^{B_y} U_1 |0^m\rangle.$$

Lemma 64. *For any $\varepsilon > 0$ there exists a set $S \subseteq \{0,1\}^n$ such that (i) $|S| < 4T^2/\varepsilon^2$ and (ii) $|\psi_T - \psi_T^y| \leq \varepsilon$ for $y \notin S$.*

Proof. Let $|E_j\rangle = \mathcal{O}^{B_y}|\psi_j\rangle - \mathcal{O}^B|\psi_j\rangle$.

Assume w.l.o.g. that the queries are applied for the same set of qubits and these qubits are the last qubits in the list. Then expansion of the vector $|\psi_j\rangle$ by query values has the form:

$$|\psi_j\rangle = \sum_{v \in \{0,1\}^n} \alpha_v^{(j)} |u_v\rangle \otimes |v\rangle$$

(here $|u_v| = 1$ but $|u_v\rangle$ is not necessarily a basis vector).

Application of the oracle query operator change the vector as follows

$$\mathcal{O}^B|\psi_j\rangle = \sum_{v\neq y} \alpha_v^{(j)}|u'_v\rangle \otimes |v\rangle + \alpha_y^{(j)}|u''_v\rangle \otimes |y\rangle,$$

$$\mathcal{O}^{B_y}|\psi_j\rangle = \sum_{v\neq y} \alpha_v^{(j)}|u'_v\rangle \otimes |v\rangle + \alpha_y^{(j)}|u'''_v\rangle \otimes |y\rangle.$$

So $|E_j\rangle = \alpha_y^{(j)}|u'''_v\rangle \otimes |y\rangle - \alpha_y^{(j)}|u''_v\rangle \otimes |y\rangle$ and $|E_j|^2 \leq 4|\alpha_y^{(j)}|^2$.

Let S be the set of y such that

$$\sum_0^{T-1} |\alpha_y^{(j)}|^2 > \frac{\varepsilon^2}{4T}\ .$$

Operators are unitary. So

$$\sum_{j=0}^{T} \sum_y |\alpha_y^{(j)}|^2 = T$$

and $|S| \leq 4T^2/\varepsilon^2$.

Now relate $|\psi_T\rangle$ and $|\psi_T^y\rangle$:

$$|\psi_T\rangle = U_T\mathcal{O}^B|\psi_{T-1}\rangle = U_T(\mathcal{O}^{B_y}|\psi_{T-1}\rangle - |E_{T-1}\rangle) = \cdots =$$

$$U_T\mathcal{O}^{B_y}U_{T-2}\mathcal{O}^{B_y}|\psi_{T-2}\rangle - \tilde{U}_{T-1}|E_{T-1}\rangle = \cdots = |\psi_T^y\rangle - \sum_{j=0}^{T-1} \tilde{U}_j|E_j\rangle,$$

where $\tilde{U}_j$ are unitary.

Take $y \notin S$. Then

$$\|\psi_T - \psi_T^y\|^2 = \Big\| \sum_{j=0}^{T-1} \tilde{U}_j|E_j\rangle \Big\|^2 \leq T \sum_{j=0}^{T-1} \big\|\tilde{U}_j|E_j\rangle\big\|^2.$$

Operators $\tilde{U}_j$ are unitary and $\|E_j\|^2 \leq 4|\alpha_y^{(j)}|^2$. Thus

$$\|\psi_T - \psi_T^y\|^2 \leq 4T \cdot \frac{\varepsilon^2}{4T} = \varepsilon^2$$

and the lemma follows. □

Lemma 64 says that vectors $U^B|0^m\rangle$ and $U^{B_y}|0^m\rangle$ are close. Recall that it implies that the measurement distributions on these states are also closed.

Proposition 65. *If $\|\psi - \xi\| < \varepsilon$ then $\rho(p,q) < \varepsilon$.*

Now we are ready to state and prove a quantum version of the main step.

Claim 66. *Let M be a polynomial time oracle machine running in time $T(n)$ on the strings of length n. If $T(n) < \frac{1}{24}2^{n/2}$ then*

$$\mathop{\mathbf{Pr}}_B[M^B \textit{ fails on the input } 1^n] = \mathop{\mathbf{Pr}}_B[M^B(1^n) \neq \chi_{L_B}(1^n)] \geq \frac{1}{4}\ .$$

Theorem 62 follows after this claim immediately.

Proof. Let $\varepsilon = \frac{1}{6}$ and $T(n) < \frac{1}{24}2^{n/2}$. For at least $2^n - \frac{144}{24^2}2^n = \frac{3}{4}\cdot 2^n$ values of y accepting probabilities for oracles $B \in \mathcal{B}$ and $B_y \in \mathcal{G}$ are 1/6-close.

For these y if M^B succeeds on the input 1^n then M^{B_y} fails on the input 1^n.

To complete the proof we repeat computations from the proof of Theorem 53. □

Remark 3. Time bound $O(2^{n/2})$ in the proof of the quantum claim is considerably less than time bound $O(2^n)$ in the proof of the classical claim. Actually the bounds are tight.

The well-known Grover search algorithm can be applied to recognize language L_B in time $O(\mathrm{poly}(n)2^{n/2})$.

The proof of Theorem 62 can be restated as a proof of optimality of the Grover search algorithm.

4.5 BQP is powerful in a relativized world

In this section we prove that there exists an oracle A such that

- $\mathrm{BQP}^A \setminus \mathrm{NP}^A \neq \varnothing$;
- $\mathrm{BQP}^A \supset \mathrm{BPP}^A$.

Again we will use random oracle arguments.

To unify arguments we introduce the class $\mathrm{BPP}_{\mathrm{path}}$ to cover both BPP and NP simultaneously.

To define the class $\mathrm{BPP}_{\mathrm{path}}$ we add an ability of 'postselection' to probabilistic computation. Recall that usual probabilistic computation can be described by a deterministic algorithm $A(x,r)$ taking two arguments. The latter argument is just random bits used in computation. So, the accepting probability is

$$\frac{1}{2^{|r|}} \sum_r A(x,r).$$

All strings r are counted with the same weight in this formula.

Now suppose that a computation device is equipped by a *selector* $S(x,r)$. It is also a deterministic algorithm. If $S(x,r) = 1$ then we say that r is selected. Computation with postselection has a 'magic' ability to take into account selected strings only while counting probabilities.

Definition 30. $L \in \mathrm{BPP}_{\mathrm{path}}$ if there exist a polynomial $p(\cdot)$ and deterministic algorithms $A(x,r)$ (acceptor), $S(x,r)$ (selector) running in polynomial time such that

- if $x \in L$ then $\mathbf{Pr}_{r \leftarrow \mathcal{U}_{p(|x|)}}[A(x,r) = 1 \mid S(x,r) = 1] \geq 2/3$;
- if $x \notin L$ then $\mathbf{Pr}_{r \leftarrow \mathcal{U}_{p(|x|)}}[A(x,r) = 1 \mid S(x,r) = 1] < 1/3$;
- $\mathbf{Pr}_{r \leftarrow \mathcal{U}_{p(|x|)}}[S(x,r) = 1] > 0$.

Here $\mathcal{U}_m$ is the uniform distribution on the strings of length m.

The following proposition is straightforward.

Proposition 67. $\mathrm{BPP}_{\mathrm{path}} \supseteq \mathrm{BPP}$, $\mathrm{BPP}_{\mathrm{path}} \supseteq \mathrm{NP}$ *and these inclusions are relativized.*

Proof. The trivial postselector ($S(x,r) = 1$ for all r) gives the first inclusion.

Let's prove the second inclusion. Let L be a language in NP and $V(x,y)$ be the corresponding verifier. It means that

$$x \in L \Rightarrow \exists y\, V(x,y) = 1, \qquad x \notin L \Rightarrow \forall y\, V(x,y) = 0.$$

Define a postselector by the rules $S(x,y,b) = V(x,y)$ for all $(y,b) \neq (0^m, 0)$; $S(x, 0^m, 0) = 1$. The latter rule guarantees that $\mathbf{Pr}_{(y,b) \leftarrow \mathcal{U}_{m+1}}[S(x,y,b) = 1] > 0$.

For $x \notin L$ we have $\mathbf{Pr}_{(y,b) \leftarrow \mathcal{U}_{m+1}}[V(x,y) = 1 \mid S(x,y,b) = 1] = 0$.

For $x \in L$ we denote by A the number of y such that $V(x,y) = 1$. Then

$$\Pr_{(y,b)\leftarrow\mathcal{U}_{m+1}}[V(x,y) = 1 \mid S(x,y,b) = 1] \geq \frac{2A}{2A+1} \geq \frac{2}{3}.$$

Thus $L \in \mathrm{BPP}_{\mathrm{path}}$. □

Theorem 68 (Aaronson, 2009)**.** *There exists an oracle A such that* $\mathrm{BQP}^A \backslash \mathrm{BPP}^A_{\mathrm{path}} \neq \varnothing$.

To prove Theorem 68 we need an oracle problem that can be efficiently solved by a quantum algorithm but is $\mathrm{BPP}_{\mathrm{path}}$-hard relative to a random oracle. We follow the construction from Aaronson's work on forrelated distributions.

4.5.1 Forrelated distributions

It will be convenient to consider oracles that compute a pair of Boolean functions (actually it is a Boolean function with additional argument) and to present Boolean functions in the form $f\colon \{0,1\}^n \to \{1,-1\}$ (i.e. replace 0 by 1 and 1 by -1 in the range of a function).

For any function $f\colon \{0,1\}^n \to \mathbb{R}$ we define the Fourier transform $\hat{f}\colon \{0,1\}^n \to \mathbb{R}$ as follows

$$\hat{f}(y) = \frac{1}{\sqrt{N}} \sum_{x\in\{0,1\}^n} (-1)^{x\cdot y} f(x),$$

where $N = 2^n$ and $x \cdot y$ is the inner product.

To prove Theorem 68 we apply the random oracle method. A distribution used in the proof is somewhat tricky and uses forrelated distributions on the pairs of Boolean functions.

Forrelated distributions are related to the 1-FOLD FORRELATION problem.

A *forrelated pair* (f,g) of Boolean functions is obtained in the following way. Take a random function $v\colon : \{0,1\}^n \to \mathbb{R}$ by drawing each value independently from the Gaussian distribution with mean 0 and variance 1. Set

$$f(x) = \begin{cases} 1, & \text{if } v(x) \geq 0, \\ -1, & \text{otherwise,} \end{cases} \qquad g(y) = \begin{cases} 1, & \text{if } \hat{v}(y) \geq 0, \\ -1, & \text{otherwise.} \end{cases}$$

We denote the forrelated distribution by $\mathcal{F}$.

It is clear from the construction that $f(x)$ is uniformly distributed on the set of Boolean functions of n variables. The Fourier transform is the orthogonal and the Gaussian distribution is symmetric under orthogonal transforms. So $g(y)$ is also uniformly distributed on the set of Boolean functions of n variables. But the pair (f,g) is not uniformly distributed on the set of pair of Boolean functions: the functions f and g are globally correlated ('forrelated').

It is crucial for the proof that these global correlations almost do not affect the values of forrelated pair in small sets of points. To make formal statement, let's define a notion of ε-almost k-wise independence (of a distribution).

A distribution $\mathcal{D}$ on N-bit strings is ε-almost k-wise independent if

$$1 - \varepsilon \leq \frac{\mathbf{Pr}_{X\leftarrow\mathcal{D}}[X_S = Y]}{\mathbf{Pr}_{X\leftarrow\mathcal{U}}[X_S = Y]} \leq 1 + \varepsilon$$

for any k-element subset $S \subset \{1,\dots,N\}$ and $Y \in \{0,1\}^k$. By X_S we denote the restriction of the string X to the positions marked by the set S: if $S = \{j_1 < j_2 < \cdots < j_k\}$ then $(X_S)_i = X_{j_i}$. By $\mathcal{U}$ we denote the uniform distribution on the pairs of Boolean functions.

Theorem 69 (Aaronson, 2009)**.** *If $k \leq \sqrt[4]{N}$ then $\mathcal{F}$ is $O(k^2/\sqrt{N})$-almost k-wise independent.*

To fool $\mathrm{BPP}_{\mathrm{path}}$ verifiers we will use the following distribution on the oracles. Take a random infinite binary string ω drawn from the uniform distribution. If $\omega_n = 1$ then an oracle F is a pair (f, g) of Boolean functions on n variables drawn from the uniform distribution $\mathcal{U}_n$. If $\omega_n = 0$ the pair of functions is drawn from the forrelated distribution $\mathcal{F}_n$.

Let L be a language $\{0^n \mid \omega_n = 1\}$. Our goal is to prove that $L \in \mathrm{BQP}^F$ with probability $c > 0$ and $L \notin \mathrm{BPP}^F_{\mathrm{path}}$ with probability 1. It immediately implies that there exists an oracle F such that $\mathrm{BQP}^F \setminus \mathrm{BPP}^F_{\mathrm{path}} \neq \varnothing$.

4.5.2 Quantum algorithm to detect a forrelation

Global correlations in a forrelated pair can be detected by a quantum algorithm having an oracle access to the values of the functions.

The algorithm repeats the following procedure several times. Take an n-qubit query register and 1-qubit answer register in the basis state $|0^n\rangle \otimes |0\rangle$. Apply Hadamard operators to all query qubits to obtain the uniform superposition of all possible queries

$$|0^n\rangle \otimes |0\rangle \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes |0\rangle.$$

Then apply oracle to query the values of f. It gives the state

$$\frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} |x\rangle \otimes |\frac{1 - f(x)}{2}\rangle.$$

Now apply Pauli operator Z to the answer qubit and uncompute the query (just ask the query once more). The current state is

$$\frac{1}{\sqrt{N}} \sum_{x \in \{0,1\}^n} f(x)|x\rangle \otimes |0\rangle.$$

Again apply Hadamards to the query qubits to obtain the state

$$\frac{1}{N} \sum_{x,y \in \{0,1\}^n} f(x)(-1)^{x \cdot y}|y\rangle \otimes |0\rangle.$$

Now query the values of g, apply Z and uncompute the query to produce the state

$$\frac{1}{N} \sum_{x,y \in \{0,1\}^n} f(x)(-1)^{x \cdot y} g(y)|y\rangle \otimes |0\rangle.$$

The next step is to apply Hadamards again. The state after this step is

$$\frac{1}{N^{3/2}} \sum_{x,y,z \in \{0,1\}^n} f(x)(-1)^{x \cdot y} g(y)(-1)^{y \cdot z}|z\rangle \otimes |0\rangle.$$

Now measure query bits in computational basis. The probability to observe 0^n is

$$p(f, g) = \frac{1}{N^3} \left(\sum_{x,y} f(x)(-1)^{x \cdot y} g(y) \right)^2.$$

It appears that the value $p(f, g)$ differs significantly for the uniform and the forrelated distributions. For the uniform distribution it can be easily computed:

$$\mathop{\mathbf{E}}_{\mathcal{U}}[p(f, g)] = \frac{1}{N}$$

and from Markov's inequality we get

$$\mathop{\mathbf{Pr}}_{\mathcal{U}}[p(f, g) \geq 0.01] \leq \frac{100}{N}.$$

So, the expectation of $p(f,g)$ is very small for the uniform distribution. Contrary, the expectation for the forrelated distribution is large.

Lemma 70. $\mathbf{Pr}_{\mathcal{F}}[p(f,g) < 0.07] \leq 2^{-\Omega(N)}$.

Proposition 71. *If an oracle F is drawn from the oracle distribution described above then* $\mathbf{Pr}[L \in \mathrm{BQP}^F] > 0.99$.

Proof. A quantum algorithm recognizing the language L on inputs 0^n, where $n > n_0$, repeats the above procedure 20 times and answers in the negative (reports the forrelated distributions) if 0^n is observed at least once. Here n_0 is a constant, which is choosed later. Answers for the inputs of lengths $\leq n_0$ are hardwired into the algorithm.

Let E be an intersection of events: (I) '$p(f_n, g_n) < 0.01$ for all $n > n_0$ such that $\omega_n = 1$' and (II) '$p(f_n, g_n) > 0.07$ for all $n > n_0$ such that $\omega_n = 0$'.

Conditioned on the event E, the probability of the positive answer in the case of $\omega_n = 0$ (i.e. the answer is wrong) is at most $0.93^{20} < 1/5$. On the same condition, the probability of the negative answer in the case of $\omega_n = 1$ (i.e. the answer is wrong) is at most $0.01 \cdot 20 = 1/5$. So the error of the algorithm in any case is at most $1/5$. It implies $L \in \mathrm{BQP}^F$.

The complement of the event E is a union of events '$\mathbf{Pr}_{\mathcal{U}_n}[p(f,g) \geq 0.01]$' and '$\mathbf{Pr}_{\mathcal{F}}[p(f,g) < 0.07]$ ' for $n > n_0$. Probabilities of these events were bounded earlier. We get from the union bound

$$\mathbf{Pr}[\neg E] \leq \sum_{n>n_0} \frac{100}{2^n} + \sum_{n>n_0} \gamma^{-2^n}$$

for a positive constant γ. Thus one can choose n_0 such that the probability of E is greater than 0.99. □

4.5.3 Hardness part of Theorem 68

To prove hardness of L relative to a random oracle drawn from the above distribution, we rely on ε-almost k-wise independency of the forrelated distributions.

Lemma 72. *Let M be a* $\mathrm{BPP}_{\mathrm{path}}$ *machine running in polynomial time and reading strings drawn from a probability distribution $\mathcal{D}$ that is $\varepsilon(n)$-almost* $\mathrm{poly}(n)$*-wise independent, where $1/\varepsilon(n)$ grows faster than any polynomial. Then*

$$\frac{1-\varepsilon(n)}{1+\varepsilon(n)} \leq \frac{\mathbf{Pr}_{\mathcal{D}}[M \text{ accepts}]}{\mathbf{Pr}_{\mathcal{U}}[M \text{ accepts}]} \leq \frac{1+\varepsilon(n)}{1-\varepsilon(n)}.$$

Proof. The machine M reads a polynomial number of bits and uses random strings r. For each string of random bits r the collection of read bits is the unique. The machine either accepts or rejects on the string r as well either selects or does no select. Due to almost independency the fraction $a_{\mathcal{D}}$ of accepting r for the distribution $\mathcal{D}$ and $a_{\mathcal{U}}$ for the uniform distribution $\mathcal{U}$ are close:

$$1 - \varepsilon(n) \leq \frac{a_{\mathcal{D}}}{a_{\mathcal{U}}} \leq 1 + \varepsilon(n).$$

The same inequalities hold for the fractions $s_{\mathcal{D}}$, $s_{\mathcal{U}}$ of selected r:

$$1 - \varepsilon(n) \leq \frac{s_{\mathcal{D}}}{s_{\mathcal{U}}} \leq 1 + \varepsilon(n).$$

To complete the proof note that accepting probabilities equal $a_{\mathcal{D}}/s_{\mathcal{D}}$ and $a_{\mathcal{U}}/s_{\mathcal{U}}$ respectively. □

From the lemma we conclude that for any $\mathrm{BPP}_{\mathrm{path}}$ machine M and sufficiently large n the probability over the oracle distribution that M fails on the input 0^n is at least $1/2 - o(1) > 1/3$.

Thus $\mathbf{Pr}[L(M) = L] = 0$ and with probability 1 we have $L \notin \mathrm{BPP}^F_{\mathrm{path}}$.

5 Quantum ineractive proofs

5.1 Interactive proof systems

Here we consider computational models that include interaction between parties involved in computation. There are many different models of this type.

We restrict our attention to *interactive proof systems*. There are two parties in this case. *Prover* has unlimited computation power. *Verifier* is computationally restricted. Prover and Verifier communicate by sending *messages*.

Let (L_0, L_1) be a promise decision problem. Prover and Verifier have a common input string x satisfying the promise: $x \in L_0 \cup L_1$.

The goal of Prover is to convince Verifier that $x \in L_1$, i.e. the answer in the instance of the problem is 'yes'. The goal of Verifier is to output the correct answer. This difference in the goals lead to asymmetric requirements to the communication protocol.

The *completeness* condition means that in the case $x \in L_1$ Prover can convince Verifier to accept the input. It means that Prover can choose his messages to guarantee acceptance.

The *soundness* condition means that in the case $x \in L_0$ Prover can not to convince Verifier. It means that for any Prover messages Verifier rejects the input.

Formal definitions differ for different types of Verifier and details of protocols (the number of rounds in communication).

5.1.1 Deterministic Verifier

Let Verifier be a deterministic algorithm running in polynomial time. In this case Prover can simulate a behavior of Verifier and send to Verifier a description of the whole protocol of communication. Verifier is able to check the correctness of the description. So the interaction can be compressed in one round— one message from Prover to Verifier. We come to definition of the class NP: the Prover message is a certificate in the definition of NP.

5.1.2 Probabilistic Verifier

Now we give to Verifier more power. He has a source of random bits. We assume that the source is *public*, i.e. Prover can see the values of random bits. In this case communication becomes nontrivial: Prover does not know Verifier messages in advance if they are chosen randomly. But Prover can simulate all Verifier computations with random bits. Thus Verifier messages V_1, V_2, ..., V_{m-1} except the last one can be just random strings. The last message V_m is a function of the input x, all previous messages and some random bits. The values of V_m are either 1 'accept' or 0 'reject'. All Prover messages P_k are functions of the input x and the previous messages.

To specify an *interactive proof system* (V, P) one should specify the number of rounds $m(x) = \text{poly}(|x|)$, the size of messages $q(x) = \text{poly}(|x|)$, the acceptance function $V_m(x, P_1, V_1, \ldots, P_m) \in \text{P}$ and the Prover strategy— collection of functions $P_k(x, P_1, V_1, \ldots, P_{k-1}, V_{k-1})$.

Definition 31. A problem (L_0, L_1) has an interactive proof system (V, P) with the completeness parameter $2/3$ and the soundness parameter $1/3$ if

- (completeness) if $x \in L_1$ then $\mathbf{Pr}[V_m = 1] > 2/3$, where V_k are drawn independently from the uniform distribution on $\{0,1\}^{q(x)}$.
- (soundness) if $x \in L_0$ then $\mathbf{Pr}[V_m = 1] < 1/3$ for any interactive proof system (V, P').

Definition 32. The class IP consists of promise problems that have interactive proof systems.

It is easy to see that probabilities can be amplified by repetitions of the protocol. So, the class IP does not changed if completeness and soundness parameters are more tight, say, for any $r = \text{poly}(n)$, , where n is the input size, the completeness parameter may be set $1 - 2^{-r}$ and the soundness parameter may be set 2^{-r}.

For the constant number of communication rounds similar definitions can be made. We get the class MA for one round of communication, the class AM for two rounds and so on. Notations in this case indicate that public coins is used. In this case Verifier is called *King Arthur* and Prover is called *Merlin*.

Theorem 73 (Babai, Moran, 1988)**.** *Let* AM$[k]$ *be a class of problems that have a public-coin IP with* k *messages. Then* AM$[k]$ = AM$[2]$ = AM *for all* $k \geq 3$.

Most of people believe in the conjecture that AM $\subset$ IP.

For private coins (Prover does not see the values of random bits) the definitions should be changed and the corresponding complexity classes are denoted by IP$[r]$. For the polynomial number of rounds there is no difference between public and private coins.

One of the most important results in complexity theory is a characterization of IP via space bounds on computation.

Let PSPACE be the class of problems that can be solved by an algorithm using a polynomially bounded space.

Theorem 74 (Shamir, 1992)**.** IP = PSPACE.

We do not prove the theorem here. Some remarks are worth to be mentioned.

The proof of inclusion IP $\subseteq$ PSPACE is based on an another representation of the class PSPACE by families of Boolean circuits of polynomial depth. We discuss them below.

The proof of inclusion IP $\supseteq$ PSPACE is based on *arithmetization technique*. The Boolean functions are represented by polynomials over a finite field. It gives to Verifier a possibility to reveal Prover cheating.

At last, it follows from the proof that completeness parameter can be set 1 (one-side error).

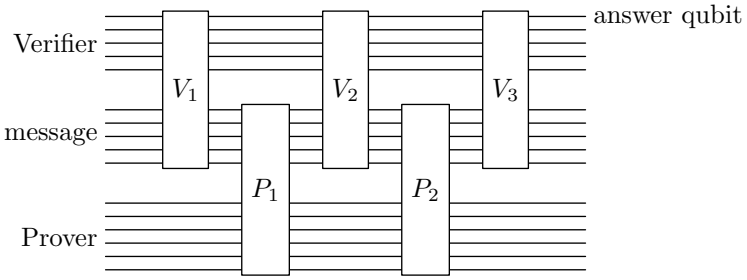
5.1.3 Quantum Verifier and quantum Prover

Now we assume that Prover and Verifier are able to operate with quantum data.

In description of quantum protocols we assume that the input x is 'hardwired' in the construction of the interactive system and suppress x in notation.

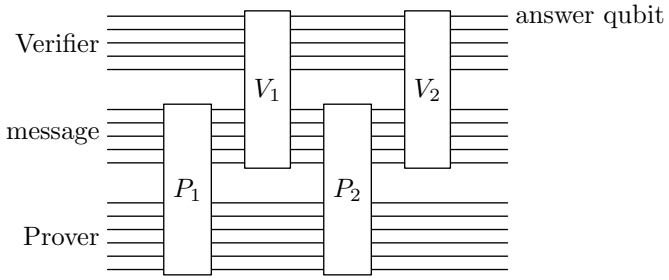
There are three quantum registers: $\mathcal{P}$ is private data of Prover containing q_P qubits, $\mathcal{V}$ is private data of Verifier containing q_V qubits and $\mathcal{M}$ is a message register containing q_M qubits.

A communication is a sequence of unitary operators in the form shown on the picture for the case of 4 messages.



Note that Verifier acts 3 times. It is naturally to give him ability to process the last Prover's message.

If the number of messages is odd, we assume that the first message is sent by Prover. Say, 3-messages communication has the following form



After all communication rounds the answer qubit is measured. The measurement outcome is the answer of Verifier.

Quantum interactive proof system (QIP) (V, P) is specified by the number of rounds $m(x) = \text{poly}(|x|)$, the sizes $q_P(x) = \text{poly}(|x|)$, $q_V(x) = \text{poly}(|x|)$, $q_M(x) = \text{poly}(|x|)$ of quantum registers $\mathcal{P}$, $\mathcal{V}$, $\mathcal{M}$ respectively, the descriptions of quantum circuits $V_k \in \mathbf{U}(\mathcal{V} \otimes \mathcal{M})$ in a universal finite basis and the Prover strategy— collection of unitary operators $P_k \in \mathbf{U}(\mathcal{P} \otimes \mathcal{M})$. (Recall that V_k and P_k is also depend on x.)

Let $\text{acc}_{V,P}(x)$ be a probability of the outcome 1 in the measurement of the answer qubit after communication of Prover P and Verifier V starting from the all-zero initial state $|0^{q_P+q_V+q_M}\rangle$.

Definition 33. A problem (L_0, L_1) has a quantum interactive proof system (V, P) with completeness parameter a and soundness parameter b if

- $\text{acc}_{V,P}(x) \geq a$ for $x \in L_1$;
- $\text{acc}_{V,P'}(x) \leq b$ for $x \in L_0$ and any Prover P'.

Remark 4. In the definition of QIP we demand that the message register is the same for all rounds of communication. In a more general communication protocol qubits are divided between parties arbitrary. Prover and Verifier apply in turn unitary transformation to the qubits in their possession and send some of them to the partner.

This generalized protocol can be easily transformed into QIP form. The message register in the QIP simulating the generalized protocol is large enough to contain all messages in the generalized protocol. By swapping qubits each party can place the qubits to be sent in the message register.

Later we will use generalized protocols assuming this conversion.

Definition 34. The class $\text{QIP}(m, a, b)$ consists of problems that have m-message quantum interactive proof system (V, P) with completeness parameter a and soundness parameter b.

If $m = \text{poly}(|x|)$ we use notation $\text{QIP}(a, b)$. By definition, $\text{QIP} = \text{QIP}(2/3, 1/3)$.

In what follows completeness and soundness parameters sometimes will be functions of the input size n.

Theorem 75 (Jain, Ji, Upahyay, Watrous, 2009)**.** $\text{QIP}(a(n), b(n)) = \text{PSPACE}$ *for any pair* a, b *such that* $a(n) - b(n) = \Omega(\text{poly}(n)^{-1})$.

The theorem shows that quantum and classical interactive proof systems define the same complexity class. It is an another illustration of general effect: the more power in a model the less difference between quantum and classic models.

5.2 Mixed states

Analyzing quantum communication, it is convenient to use more general definitions of a quantum state and quantum measurement.

A notion of *mixed quantum state* arises naturally from the following question. Prover sends to Verifier only a part of its qubits. In what state these qubits are? More exactly, let $|\psi\rangle$ be a state in the space $\mathcal{P} \otimes \mathcal{M}$. How to define a state of the register $\mathcal{M}$ only?

If $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$ is a product state, the answer is clear. But there are non-product states. To define the state of the second register for these states we need a more general formalism.

In this section we recall the definitions used in this case.

Let $\mathbb{L}(\mathcal{X})$ be a space of all linear operators $\mathcal{X} \to \mathcal{X}$. This space is equipped by the inner product $\langle X, Y\rangle = \mathrm{Tr}(X^\dagger Y)$.

A Hermitian operator A is called *positive semidefinite* if all eigenvalues of A are nonnegative. The set of positive semidefinite operator is denoted by $\mathrm{Pos}(\mathcal{X})$.

An operator $\rho \in \mathrm{Pos}(\mathcal{X})$ is a *density operator* if $\mathrm{Tr}(\rho) = 1$. The set of density operators is denoted by $\mathbb{D}(\mathcal{X})$.

Density operators represent *mixed quantum states*. Pure states $|\psi\rangle$ are represented in the density operator formalism by operators of rank 1: $|\psi\rangle\langle\psi|$. Expanding a density operator in the basis of eigenstates we see that any quantum state is a convex combination (or probability distribution) of pure states:

$$\rho = \sum_k p_k |\psi_k\rangle\langle\psi_k|, \qquad p_k \geq 0, \quad \sum_k p_k = 1.$$

Such a representation is not unique.

A *POVM measurement* with outcomes Γ is a set $\{P_a : a \in \Gamma\}$ of positive semidefinite operators P_a such that

$$\sum_{a\in\Gamma} P_a = I_{\mathcal{X}}.$$

If a state ρ is measured then the probability of an outcome a is $\langle P_a, \rho\rangle$.

This notion generalize the previous definition of the measurement in the orthonormal basis. In this case operators are $P_k = |k\rangle\langle k|$. For a pure state $\rho = |\psi\rangle\langle\psi|$ we get the same probability of an outcome k:

$$\mathop{\mathbf{Pr}}_{\rho}[k] = \langle |k\rangle\langle k|, |\psi\rangle\langle\psi|\rangle = \mathrm{Tr}(|k\rangle\langle k||\psi\rangle\langle\psi|) = |\langle k, \psi\rangle|^2.$$

Now define a state of a subsystem of a quantum system. The reasonable requirement is to get the same probabilities of outcomes for measurements involving the qubits of the subsystem only.

Let $\rho \in \mathbb{D}(\mathcal{P} \otimes \mathcal{M})$ and $\{I_{\mathcal{P}} \otimes P_a : a \in \Gamma\}$ is a measurement of qubits in the register $\mathcal{M}$. Then

$$\mathop{\mathbf{Pr}}_{\rho}[a] = \langle I_{\mathcal{P}} \otimes P_a, \rho\rangle.$$

Expand ρ in the sum of tensor products of operators:

$$\rho = \sum_j \sigma_j \otimes \zeta_j.$$

Then

$$\mathop{\mathbf{Pr}}_{\rho}[a] = \sum_j \mathrm{Tr}((I_{\mathcal{P}} \otimes P_a)(\sigma_j \otimes \zeta_j)) = \sum_j \mathrm{Tr}(\sigma_j)\langle P_a, \zeta_j\rangle = \langle P_a, \mathrm{Tr}_{\mathcal{P}}(\rho)\rangle.$$

The standard arguments of tensor algebra show that a density operator

$$\mathrm{Tr}_{\mathcal{P}}(\rho) = \sum_j \mathrm{Tr}(\sigma_j)\zeta_j$$

does not depend on a choice of expansion in the sum of tensor products. It is called a *partial trace* of ρ.

By definition, if two registers $\mathcal{P} \otimes \mathcal{M}$ are in a state ρ then a state of the register $\mathcal{M}$ is $\mathrm{Tr}_{\mathcal{P}}(\rho)$.

Any mixed state can be considered as a part of large system in a pure state. The latter is called a *purification*. For $\rho \in \mathbb{D}(\mathcal{M})$, a purification can be found in the system of two registers $\mathcal{M} \otimes \mathcal{M}^*$. Expand ρ in the eigenstate basis

$$\rho = \sum_j p_j |\xi_j\rangle\langle\xi_j|$$

and set

$$|\psi\rangle = \sum_j \sqrt{p_j}|\xi_j\rangle \otimes |\eta_j\rangle, \quad \text{where } |\eta_j\rangle = \langle\xi_j| \in \mathcal{M}^*.$$

Direct calculation of partial trace shows that

$$\mathrm{Tr}_{\mathcal{M}^*} |\psi\rangle\langle\psi| = \rho.$$

5.3 The class QMA

One-message quantum interactive proof systems correspond to the classical complexity class MA. So we use additional notation $\mathrm{QIP}(1, a, b) = \mathrm{QMA}(a, b)$.

The general definition can be adapted for one-message case in the following way.

Definition 35. A promise problem (L_0, L_1) is in the class $\mathrm{QMA}(a, b)$ if there exists a classical algorithm $x \mapsto \langle U_x \rangle$ running in polynomial time such that $\langle U_x \rangle$ is a description of a quantum circuit acting on two registers $\mathcal{V}$ and $\mathcal{M}$ and

- (completeness) if $x \in L_1$ then $\mathbf{Pr}_{U_x(0^q \otimes \rho)}[v_1 = 1] \geq a$ for some state $\rho \in \mathbb{D}(\mathcal{M})$;
- (soundness) if $x \in L_0$ then $\mathbf{Pr}_{U_x(0^q \otimes \rho)}[v_1 = 1] \leq b$ for all states $\rho \in \mathbb{D}(\mathcal{M})$.

The class $\mathrm{QMA} = \mathrm{QMA}(2/3, 1/3)$ is a natural generalization of the classes NP (deterministic case) and MA (probabilistic case). So it is worth to discuss the properties of QMA.

The first observation is about pure and mixed Prover messages. In general definition a message is a mixed state because Prover can send only a part of his qubits. But in the case of QMA it makes no difference. Sending a purification, Prover guarantees the same accept probabilities. Moreover, note that probability is a linear functional on a density operator. Any mixed state is a convex combination of pure states. So the maximum of accept probability is attained at a pure state.

5.3.1 Amplification of probabilities

Lemma 76. *If* $a - b = \Omega(\mathrm{poly}(n)^{-1})$ *then* $\mathrm{QMA}(a, b) = \mathrm{QMA}(1 - 2^{-r}, 2^{-r})$ *for any* $r = \mathrm{poly}(n)$.

Proof. The idea of the amplification is the same: repeat protocol many times (say, k times) and output the positive answer iff the fraction of outcomes 1 is greater than $(a + b)/2$. Denote the measurement outcomes in the copies of the protocol by $z_1, \ldots, z_k$.

The probability of obtaining outcomes $z_1, \ldots, z_k$ by applying k copies of U to the message $\rho \in \mathcal{M}^{\otimes k}$ is

$$\mathop{\mathbf{Pr}}_{\rho}[z_1, \ldots, z_k] = \mathrm{Tr}\big(X^{(z_1)} \otimes \cdots \otimes X^{(z_k)} \rho\big),$$

where

$$X^{(a)} = \mathrm{Tr}_{\mathcal{V}}\Big(U^\dagger \Pi_1^{(a)} U\big(I_{\mathcal{M}} \otimes |0^q\rangle\langle 0^q|\big)\Big).$$

Here $\Pi_1^{(a)}$ is the projection onto the subspace of states having a in the first qubit.

Note that $X^{(0)} + X^{(1)} = I$. Thus the operators $X^{(0)}$ and $X^{(1)}$ are diagonalized in the same basis. We denote basis vectors in this basis by $|d\rangle$. Then

$$\mathop{\mathbf{Pr}}_{|d\rangle}[z] = \langle d|X^{(z)}|d\rangle \quad \text{and} \quad \mathop{\mathbf{Pr}}_{|d\rangle}[0] + \mathop{\mathbf{Pr}}_{|d\rangle}[1] = 1.$$

Maximal probability $\max_\rho \mathbf{Pr}_\rho[1]$ for $\rho \in \mathbb{D}(\mathcal{M})$ is attained on a basis vector $|d\rangle$ because for any density operator $\mathrm{Tr}(X^{(1)}\rho)$ is a convex combination of matrix elements $\langle d|X^{(1)}|d\rangle$.

Now we rewrite the probability of outcomes $z_1, \ldots, z_k$ as a convex combination

$$\mathop{\mathbf{Pr}}_{\rho}[z_1, \ldots, z_k] = \sum_{d_1, d_2, \ldots, d_k} p_{d_1,\ldots,d_k} \prod_{j=1}^{k} \mathop{\mathbf{Pr}}_{|d_j\rangle}[z_j], \qquad \sum_{d_1, d_2, \ldots, d_k} p_{d_1,\ldots,d_k} = 1$$

where $p_{d_1,\ldots,d_k}$ are diagonal elements of ρ in the basis $|d_1, \ldots, d_k\rangle$.

So the maximal accept probability is attained on a basis vector $|d_1, \ldots, d_k\rangle$. For a basis vector the probability is a product

$$\mathop{\mathbf{Pr}}_{|d_1,\ldots,d_k\rangle}[z_1, \ldots, z_k] = \prod_j \mathop{\mathbf{Pr}}_{|d_j\rangle}[z_j].$$

Suppose that

$$p = \max_{\rho \in \mathbb{D}(\mathcal{M})} \mathop{\mathbf{Pr}}_{\rho}[1] = \mathop{\mathbf{Pr}}_{|d_j\rangle}[1] \geq a.$$

Then Prover sends the state $|d_j\rangle^{\otimes k}$. The amplified algorithm outputs 1 if

$$\sum_j z_j \geq k\frac{a+b}{2} = \ell.$$

So the accept probability of the amplified algorithm is

$$\sum_{j \geq \ell} \binom{k}{j} p^j (1-p)^{k-j}.$$

By Chernoff bound it is greater than

$$1 - \exp(-2(p - (a+b)/2)^2 k) \geq 1 - \exp(-k(a-b)^2/2).$$

Suppose now that $p = \max_\rho \mathbf{Pr}_\rho[1] \leq b$ (the negative case). We should upper-bound a sum in the form

$$A_\ell(\vec{p}) = \sum_{|J| \geq \ell} \prod_{j \in J} p_j \prod_{j \notin J} (1 - p_j), \quad \text{where } p_j = \mathop{\mathbf{Pr}}_{|d_j\rangle}[1] \leq b.$$

Take an index j and expand $A_\ell(\vec{p})$ on a variable p_j:

$$A_\ell(\vec{p}) = p_j A_{\ell-1}(\vec{p}_{\setminus j}) + (1 - p_j) A_\ell(\vec{p}_{\setminus j}),$$

where $\vec{p}_{\setminus j}$ is formed by the rest of variables. It is easy to see that $A_{\ell-1}(\vec{p}) \geq A_\ell(\vec{p})$ for any vector $\vec{p} = (p_j)$. So $A_\ell(\vec{p})$ grows when p_j grows. Thus the probability to accept is upperbounded by

$$\sum_{j \geq \ell} \binom{k}{j} p^j (1-p)^{k-j}.$$

By Chernoff bound it is smaller than

$$\exp(-2((a+b)/2 - b)^2 k) \leq \exp(-k(a-b)^2/2).$$

It follows from the bounds derived that if $a - b = \Omega(\mathrm{poly}(n)^{-1})$ then error probabilities can be made $O(2^{-r})$ by $k = \mathrm{poly}(n)$ repetitions for any $r = \mathrm{poly}(n)$. □

5.3.2 Bounds on computational complexity of QMA

It follows from definition that QMA = QMA$(2/3, 1/3) \supseteq$ MA. No better lower bounds on the complexity of QMA is known.

As for upper bounds of complexity the basic result is the following.

Theorem 77 (Kitaev, 2002)**.** QMA $\subseteq$ PP.

Proof. Let $(L_0, L_1) \in$ QMA. Suppose that Verifier applies the circuit U in a QMA proof system for an input x.

If Prover provides a message ρ the probability to accept is expressed as

$$\mathop{\mathbf{Pr}}_{\rho}[v_1 = 1] = \mathrm{Tr}(X\rho), \quad \text{where } X = \mathrm{Tr}_{\mathcal{V}}(U^{\dagger}\Pi_1 U(I_{\mathcal{M}} \otimes |0^q\rangle\langle 0^q|)).$$

It is maximized if $\rho = |\psi\rangle\langle\psi|$, where $|\psi\rangle$ is the eigenvector of X corresponding to the maximal eigenvalue $\lambda_{\max}$. Thus

$$\max_{\rho} \mathop{\mathbf{Pr}}_{\rho}[v_1 = 1] = \lambda_{\max}.$$

To estimate the maximal eigenvalue one can estimate trace of X^d due to obvious inequalities

$$\lambda_{\max}^d \leq \mathrm{Tr}(X^d) \leq 2^m \lambda_{\max}^d.$$

Take $d > m + 2$. Then

$$\lambda_{\max} \geq \frac{2}{3} \quad \Rightarrow \quad \mathrm{Tr}(X^d) \geq \left(\frac{2}{3}\right)^d,$$

$$\lambda_{\max} \leq \frac{1}{3} \quad \Rightarrow \quad \mathrm{Tr}(X^d) \leq 2^m \left(\frac{1}{3}\right)^d < \frac{1}{4} \cdot \left(\frac{2}{3}\right)^d.$$

So to distinguish between acceptance and rejection one should estimate $\mathrm{Tr}(X^{m+3})$ with multiplicative accuracy $1/2$.

W.l.o.g. we assume that the circuit U is a circuit in a complete basis $\{\Lambda^2(X), H\}$ of Toffoli gate and Hadamard gate. In this case matrix elements (and the trace) of X^d are fractions in the form

$$\frac{a}{2^{dh/2}}, \quad \text{where } a \in \mathbb{Z} \text{ and}$$

h is the number of Hadamards in the circuit.

The numerator of $\mathrm{Tr}(X^{m+3})$ is a gap function (difference of $\#P$-functions). The inequality

$$\frac{a}{2^{dh/2}} > \frac{1}{2} \cdot \left(\frac{2}{3}\right)^d$$

is equivalent to

$$4a^2 \cdot 9^d - 2^{d(2+h)} > 0.$$

Left-hand side of this inequality is also a gap function f due to basic arithmetic properties of gap functions.

From completeness and soundness conditions for the QMA protocol we have two properties of the gap function $f(x)$ (recall that f depends on an input string x):

$$x \in L_1 \quad \Rightarrow \quad f(x) > 0,$$

$$x \in L_0 \quad \Rightarrow \quad f(x) < 0,$$

which mean that $(L_0, L_1) \in$ PP. □

Remark 5. There is an undirect argument against the equality QMA = PP. It implies that PP $\supseteq$ PH, where PH is the class called *polynomial hierarchy*. This inclusion contradicts relativized results and most of people do not believe in it.

We skip the definitions and the exact claim. Note only that the amplification of error probabilities plays a major role in the proof. For a general PP problem the amplification of this type is unknown.

5.3.3 A QMA complete problem

QMA is a quantum analog of NP. To make the analogy more clear, we present an example of QMA-complete problem that generalize the basic NP complete problem—satisfiability of 3-CNF. The problem is about eigenvalues of local Hamiltonians.

A Hermitian operator $H\colon (\mathbb{C}^2)^{\otimes n} \to (\mathbb{C}^2)^{\otimes n}$ is called a *k-local Hamiltonian* if it can be expressed in the form

$$H = \sum_j H_j[S_j],$$

where each term $H_j[S_j]$ is a Hermitian operator acting on a set of qubits S_j and $|S_j| \le k$.

We normalize local terms and assume that $0 \le H_j \le 1$, i.e. both H_j and $I - H_j$ are positive semidefinite.

Definition 36 (k-local Hamiltonian problem)**.** Input: a description of a k-local Hamiltonian and two numbers a, b. Here k is a fixed constant, $0 \le a < b$ and $b - a = \Omega(n^{-\alpha})$ ($\alpha > 0$ is a constant).

Promise: either H has an eigenvalue not exceeding a or all eigenvalues of H are greater than b.

Question: decide whether H has an eigenvalue not exceeding a.

Formally, the k-local Hamiltonian problem is a promise problem (L_0, L_1), where $x \in L_1$ iff $x = \langle H, a, b\rangle$ with parameters satisfying the above properties and H has an eigenvalue not exceeding a; and $x \in L_0$ iff $x = \langle H, a, b\rangle$ with parameters satisfying the above properties and all eigenvalues of H are greater than b.

Proposition 78. *k-local Hamiltonian problem is in* QMA.

Proof. Let the input Hamiltonian has a form

$$H = \sum_{j=1}^{r} H_j[S_j], \quad H\colon (\mathbb{C}^2)^{\otimes m} \to (\mathbb{C}^2)^{\otimes m}.$$

We are going to construct a QMA protocol with the following property: Prover sends a state $|\eta\rangle \in (\mathbb{C}^2)^{\otimes m}$ and Verifier applies a quantum circuit W such that the measurement of the 'answer' qubit gives the outcome 1 with probability $p = 1 - r^{-1}\langle\eta|H|\eta\rangle$.

If $|\eta\rangle$ is an eigenvector corresponding to an eigenvalue $\lambda \le a$, then the probability of the outcome 1 is

$$p = 1 - r^{-1}\langle\eta|H|\eta\rangle = 1 - r^{-1}\lambda \ge 1 - r^{-1}a,$$

and if every eigenvalue of H exceeds b, then

$$p = 1 - r^{-1}\langle\eta|H|\eta\rangle \le 1 - r^{-1}b.$$

Thus k-local Hamiltonian problem is in $\mathrm{QMA}(1 - r^{-1}a, 1 - r^{-1}b)$ and

$$(1 - r^{-1}a) - (1 - r^{-1}b) = \Omega(\mathrm{poly}(n)^{-1}).$$

From the amplification lemma we conclude that k-local Hamiltonian problem is in QMA.

We build the circuit W from circuits for local terms. Let $H_j = \sum_s \lambda_s |\psi_s\rangle\langle\psi_s|$. This operator acts on a constant number of qubits. Therefore we can realize the operator

$$W_j \colon |\psi_s, 0\rangle \mapsto |\psi_s\rangle \otimes \Big(\sqrt{\lambda_s}|0\rangle + \sqrt{1-\lambda_s}|1\rangle\Big)$$

by a circuit of constant size. It acts on the set of qubits $S_j \cup \{\text{“answer”}\}$, where “answer” denotes the qubit that will contain the measurement outcome.

Let $|\eta\rangle = \sum_s y_s|\psi_s\rangle$ be the expansion of $|\eta\rangle$ in the orthogonal system of eigenvectors of H_j. We have, by definition of the probability,

$$\begin{aligned}
\mathop{\mathbf{Pr}}_{W_j(|\eta\rangle\otimes|0\rangle)}[1] &= \langle\eta,0|\, W_j^\dagger \big(I \otimes \underbrace{|1\rangle\langle 1|}_{\text{answer}}\big) W_j\, |\eta,0\rangle \\
&= \left(\sum_s y_s^* \langle\psi_s,0|\right) W_j^\dagger \big(I \otimes \underbrace{|1\rangle\langle 1|}_{\text{answer}}\big) W_j \left(\sum_t y_t\, |\psi_t,0\rangle\right) \\
&= \sum_{s,t} \sqrt{1-\lambda_s}\; y_s^* \sqrt{1-\lambda_t}\; y_t\; \langle\psi_s|\psi_t\rangle \;=\; \sum_s (1-\lambda_s) y_s^* y_s \\
&= 1 - \langle\eta|H_j|\eta\rangle.
\end{aligned}$$

A general circuit W uses additional register of dimension r. It applies an operator

$$\sum_j |j\rangle\langle j| \otimes W_j$$

to the state $\Big(\frac{1}{\sqrt{r}} \sum_j |j\rangle\Big) \otimes |\eta, 0\rangle$. By linearity the probability of getting the outcome 1 is

$$\mathbf{Pr}[1] = \sum_j \frac{1}{r}\big(1 - \langle\eta|H_j|\eta\rangle\big) = 1 - r^{-1}\langle\eta|H|\eta\rangle.$$

We skip the details of construction of the circuit W. They are routine. □

Theorem 79 (Kempe, Kitaev, Regev, 2004)**.** 2-LOCAL HAMILTONIAN PROBLEM *is* QMA *complete.*

The proof is rather technical. We present the basic idea for a weaker result about QMA-completeness of 5-local Hamiltonians.

The idea goes back to Feynman: replacing a unitary evolution by a time independent Hamiltonian.

Suppose that Verifier applies a circuit $U = U_L \cdots U_1$ of size L in a QMA protocol. We will assume that U acts on $m + q$ qubits, the first m of which initially contain Prover's message $|\psi\rangle$, the rest being initialized by 0. The gates U_j act on pairs of qubits.

At first, we construct a $O(\log n)$-local Hamiltonian associated with the circuit. It acts on the space

$$\mathcal{L} = (\mathbb{C}^2)^{\otimes(m+q)} \otimes \mathbb{C}^{L+1}.$$

The second factor is a counter (clock register).

The Hamiltonian consists of three terms

$$H = H_{\text{in}} + H_{\text{prop}} + H_{\text{out}}.$$

The term H_{in} corresponds to the condition that, at step 0, all the qubits but m are in state $|0\rangle$. Specifically,

$$H_{\text{in}} = \left(\sum_{s=m+1}^{m+q} \Pi_s^{(1)}\right) \otimes |0\rangle\langle 0|,$$

where $\Pi_s^{(\alpha)}$ is the projection onto the subspace of vectors for which the s-th qubit equals α. The second factor in this formula acts on the space of the counter. (Informally speaking, the term $\Pi_s^{(1)} \otimes |0\rangle\langle 0|$ "collects a penalty" by adding 1 to the cost function whenever the s-th qubit is in state $|1\rangle$ while the counter being in state $|0\rangle$.)

The term H_{out} corresponds to the final state and equals

$$H_{\text{out}} = \Pi_1^{(0)} \otimes |L\rangle\langle L|.$$

Here we assume that the answer qubit has index 1. (That is, at step L the first qubit should be in state $|1\rangle$, or a penalty will be imposed.)

The last term H_{prop} describes the propagation of a quantum state through the circuit. It consists of L terms, each of which corresponds to the transition from $j-1$ to j:

$$H_{\text{prop}} = \sum_{j=1}^{L} H_j,$$
$$H_j = -\frac{1}{2} U_j \otimes |j\rangle\langle j-1| - \frac{1}{2} U_j^\dagger \otimes |j-1\rangle\langle j| + \frac{1}{2} I \otimes \big(|j\rangle\langle j| + |j-1\rangle\langle j-1|\big).$$

Each term H_j acts on two qubits of the space $(\mathbb{C}^2)^{\otimes(m+q)}$, as well as on the clock register.

The Hamiltonian looks rather mysterious. To reveal its nature the change of basis given by the operator

$$W = \sum_{j=0}^{L} U_j \cdots U_1 \otimes |j\rangle\langle j|$$

is useful. Under such a change, the Hamiltonian is transformed into its conjugate, $\widetilde{H} = W^\dagger H W$.

Let's calculate the conjugations of the terms of H. On the term H_{in} the conjugation has no effect:

$$\widetilde{H}_{\text{in}} = W^\dagger H_{\text{in}} W = H_{\text{in}}.$$

The action on the term H_{out} is:

$$\widetilde{H}_{\text{out}} = W^\dagger H_{\text{out}} W = \Big(U^\dagger \Pi_1^{(0)} U\Big) \otimes |L\rangle\langle L|.$$

Each operator H_j in the propagation term is the sum of three terms. Let us write the action of the conjugation on the first of them:

$$\begin{aligned}
&W^\dagger \big(U_j \otimes |j\rangle\langle j-1|\big) W \\
&\quad = \sum_{p,t} \big(U_p \cdots U_1 \otimes |p\rangle\langle p|\big)^\dagger \big(U_j \otimes |j\rangle\langle j-1|\big) \big(U_t \cdots U_1 \otimes |t\rangle\langle t|\big) \\
&\quad = \big((U_j \cdots U_1)^\dagger U_j (U_{j-1} \cdots U_1)\big) \otimes \Big(\big(|j\rangle\langle j|\big)^\dagger |j\rangle\langle j-1| \big(|j-1\rangle\langle j-1|\big)\Big) \\
&\quad = I \otimes |j\rangle\langle j-1|.
\end{aligned}$$

Conjugation of the two other terms proceeds analogously, so that we obtain

$$\begin{aligned}
\widetilde{H}_j &= W^\dagger H_j W \\
&= I \otimes \frac{1}{2}\Big(|j-1\rangle\langle j-1| - |j-1\rangle\langle j| - |j\rangle\langle j-1| + |j\rangle\langle j|\Big) = I \otimes E_j, \\
\widetilde{H}_{\text{prop}} &= W^\dagger H_{\text{prop}} W = I \otimes E,
\end{aligned}$$

where

$$E = \sum_{j=1}^{L} E_j = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} & & & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} & & \\ & -\frac{1}{2} & 1 & -\frac{1}{2} & \\ & & -\frac{1}{2} & & \ddots \\ 0 & & & \ddots & \ddots \\ & & & \ddots & \end{pmatrix}.$$

We have described a reduction of a QMA problem to local Hamiltonian problem. The correctness of the reduction stems from two facts.

Suppose that the circuit U gives the outcome 1 with probability $\geq 1-\varepsilon$ on some Prover's message $|\xi\rangle$. It implies that the probability of the outcome 0, i.e.

$$\mathbf{Pr}[0] = \big\langle \xi, 0 \big| U^\dagger \Pi_1^{(0)} U \big| \xi, 0 \big\rangle$$

less than ε.

Then there is a state $|\widetilde{\eta}\rangle$ such that

$$\langle \widetilde{\eta}| \, \widetilde{H} \, |\widetilde{\eta}\rangle \leq \frac{\varepsilon}{L+1},$$

so H has an eigenvalue with the same upper bound.

This fact is rather simple. Note that

$$E|\psi\rangle = 0, \quad \text{where } |\psi\rangle = \frac{1}{\sqrt{L+1}} \sum_{j=0}^{L} |j\rangle.$$

Direct calculations show that the state $|\widetilde{\eta}\rangle = |\xi, 0\rangle \otimes |\psi\rangle$ fits the above bound.

In the rejection case the circuit U gives the outcome 1 with probability $\leq \varepsilon$ on all Prover's messages $|\xi\rangle$. It appears that all eigenvalues of H are at least $c\,(1-\sqrt{\varepsilon})\,L^{-3}$ for some constant c.

Now we return to the structure of the clock register. In the above construction the clock register has no qubit structure. So, the Hamiltonian constructed in the reduction is actually $O(\log n)$ local.

To get a local Hamiltonian we embed the clock register in a larger space. We take L qubits, enumerated from 1 to L. The suitable embedding $\mathbb{C}^{L+1} \to (\mathbb{C}^2)^{\otimes L}$ is

$$|j\rangle \mapsto |\underbrace{1, \ldots, 1}_{j}, \underbrace{0, \ldots, 0}_{L-j}\rangle.$$

The operators on the space $\mathbb{C}^{L+1}$ used in the construction of the Hamiltonian H are replaced as follows:

$|0\rangle\langle 0|$ is replaced by $\Pi_1^{(0)}$, $\qquad$ $|0\rangle\langle 1|$ is replaced by $\big(|0\rangle\langle 1|\big)_1 \Pi_2^{(0)}$,

$|j\rangle\langle j|$ is replaced by $\Pi_j^{(1)} \Pi_{j+1}^{(0)}$, $\qquad$ $|j-1\rangle\langle j|$ is replaced by $\Pi_{j-1}^{(1)} \big(|0\rangle\langle 1|\big)_j \Pi_{j+1}^{(0)}$,

$|L\rangle\langle L|$ is replaced by $\Pi_L^{(1)}$, $\qquad$ $|L-1\rangle\langle L|$ is replaced by $\Pi_{L-1}^{(1)} \big(|0\rangle\langle 1|\big)_L$.

In this way we get 3-local operators in the clock register and the Hamiltonian itself is 5-local.

Thus, we have replaced the Hamiltonian H, acting on the space $\mathcal{L} = (\mathbb{C}^2)^{\otimes(m+q)} \otimes \mathbb{C}^{L+1}$, by a new Hamiltonian H_{ext}, defined on the larger space $\mathcal{L}_{\text{ext}} = (\mathbb{C}^2)^{\otimes(m+q)} \otimes (\mathbb{C}^2)^{\otimes L}$. The operator H_{ext} maps the subspace $\mathcal{L} \subseteq \mathcal{L}_{\text{ext}}$ into itself and acts on it just as H does.

But the embedding causes a new problem: what to do with the extra states in the extended clock register? The idea is to put additional penalties to avoid these

states. Namely, we add another term to the Hamiltonian H_{ext}:

$$H_{\text{stab}} = I_{(\mathbb{C}^2)^{\otimes(m+q)}} \otimes \sum_{j=1}^{L-1} \Pi_j^{(0)} \Pi_{j+1}^{(1)}.$$

The null subspace of the operator H_{stab} coincides with the old working space $\mathcal{L}$, so that the supplementary term does not change the upper bound for the minimum eigenvalue for the measurement outcome 1.

In the rejection case the required lower bound for the eigenvalues of the operator $H_{\text{ext}} + H_{\text{stab}}$ can be obtained by the following arguments. Both terms leave the subspace $\mathcal{L}$ invariant, so we can analyze the action of $H_{\text{ext}} + H_{\text{stab}}$ on $\mathcal{L}$ and on its orthogonal complement $\mathcal{L}^\perp$ independently. On $\mathcal{L}$ we have $H_{\text{ext}} \geq c(1-\sqrt{\varepsilon})L^{-3}$ and $H_{\text{stab}} = 0$, and on $\mathcal{L}^\perp$ we have $H_{\text{ext}} \geq 0$ and $H_{\text{stab}} \geq 1$. In both cases

$$H_{\text{ext}} + H_{\text{stab}} \geq c(1-\sqrt{\varepsilon})L^{-3}.$$

To prove QMA completeness of 2-local Hamiltonian problem similar ideas of adding penalties is combined with more sophisticated techniques of perturbation theory.

5.4 The class QIP

Now we return to general classes $\text{QIP}(m, a(n), b(n))$, where n is the input size and $a - b = \Omega(\text{poly}(n)^{-1})$. They have nice properties and the proof of QIP = PSPACE is based on these properties.

5.4.1 Perfect completeness for QIP

It appears that the completeness parameter $a(n)$ can be made 1 by two additional messages in the protocol. In other words, the error can be made one-sided: in the case $x \in L_1$ Verifier gives a correct answer with probability 1.

We make some assumptions on the protocols and circuits used by Verifier.

We assume that Prover can decrease the probability of acceptance by any desired quantity. For this purpose, one additional message qubit can be added. Verifier accepts iff this qubit is set to 0. It is clear that Prover can change this qubit in the last message to multiply the acceptance probability by any factor < 1.

Also we assume that Verifier uses circuits that can be realize exactly the operator

$$\begin{aligned} T_{a(n)}|0\rangle &= \sqrt{a(n)}|0\rangle \quad -\sqrt{1-a(n)}|1\rangle, \\ T_{a(n)}|1\rangle &= \sqrt{1-a(n)}|0\rangle \quad +\sqrt{a(n)}|1\rangle. \end{aligned}$$

Here $a(n)$ is the completeness parameter.

Theorem 80. *Let $a(n)$ be a sequence such that the operator $T_{a(n)}$ is realized exactly. Then* $\text{QIP}(m, a, b) \subseteq \text{QIP}(m + 2, 1, 1 - (a - b)^2)$.

Proof. Let $(L_0, L_1) \in \text{QIP}(m, a, b)$. As it mentioned, we assume that for each $x \in L_1$ the acceptance probability is exactly $a(|x|)$.

QIP with perfect completeness has the following form.

1. Let R be Verifier's qubits in the original protocol. In the perfect protocol Verifier uses two additional 1-qubit registers B and B'. They are initially zero.

2. After running the original protocol, the perfect Verifier increments both B and B' iff the original Verifier rejects.

3. Then R and B' are sent to Prover.

4. Prover makes a transformation of qubits in his possession and returns qubits to Verifier.

5. Verifier subtracts B from B' (applies CNOT).

6. Verifier performs $T_{a(n)}$ on B. Then Verifier measures B and accepts iff B contains 0.

Let $x \in L_1$. Then the state of the register R after running the original protocol can be made $|\psi\rangle = \alpha|\psi_1\rangle + \beta|\psi_0\rangle$, where $\alpha = \sqrt{a(n)}$, $\beta = \sqrt{1-\alpha^2}$ and $|\psi_1\rangle$ and $|\psi_0\rangle$ are normalized projections of $|\psi\rangle$ onto accepting and rejecting states respectively. The state of all Verifier qubits at this moment is $\alpha|00\rangle|\psi_1\rangle + |00\rangle\beta|\psi_0\rangle$.

The next actions change the state as follows:

$$\begin{aligned}
&\alpha|00\rangle|\psi_1\rangle + |11\rangle\beta|\psi_0\rangle && \text{after step 2,}\\
&\alpha|0\rangle U(|0\rangle|\psi_1\rangle) + \beta|1\rangle U(|1\rangle|\psi_0\rangle) && \text{after step 4,}\\
&\alpha|0\rangle|\varphi_1\rangle + \beta|1\rangle|\varphi_0\rangle && \text{after step 5, where}\\
&|\varphi_1\rangle = U(|1\rangle|\psi_0\rangle), \quad |\varphi_0\rangle = X_1 U(|1\rangle|\psi_0\rangle).
\end{aligned}$$

After application of $T_{a(n)}$ on B the probability to accept is

$$\|\alpha\sqrt{a(n)}|\varphi_1\rangle + \beta\sqrt{1-a(n)}|\varphi_0\rangle\|^2$$

Prover can choose U such that $U(|0\rangle|\psi_1\rangle) = |0\rangle|\varphi\rangle$ and $U(|1\rangle|\psi_0\rangle) = |1\rangle|\varphi\rangle$. It implies that $|\varphi_0\rangle = |\varphi_1\rangle$ and the probability to accept is

$$(\alpha\sqrt{a(n)} + \beta\sqrt{1-a(n)})^2 = (a(n) + 1 - a(n))^2 = 1.$$

If $x \in L_0$ the probability α^2 to accept in the initial QIP does not exceed $b(n)$. Repeating the previous arguments, we get the bound of the acceptance probability in the perfect QIP:

$$\begin{aligned}
(\alpha\sqrt{a(n)} + \beta\sqrt{1-a(n)})^2 &\le (\alpha^2 + 1 - a(n))(a(n) + 1 - \alpha^2)\\
&= 1 - (a(n) - \alpha^2)^2 \le 1 - (a(n) - b(n))^2.
\end{aligned}$$

□

The requirement of exact realization of the operator $T_{a(n)}$ looks very artificial. We are going to rid of it.

Note that in a complete finite basis any operator can be approximated with exponential precision efficiently. Thus for any $a(n)$ there exists $a'(n) < a(n)$ such that $a(n) - a'(n) = O(2^{-\operatorname{poly}(n)})$ and the operator $T_{a'(n)}$ is realized exactly in the basis.

Due to inclusion $\mathrm{QIP}(m, a, b) \subset \mathrm{QIP}(m, a'(n), b(n)$ the theorem implies

$$\mathrm{QIP}(m, a, b) \subseteq \mathrm{QIP}(m + 2, 1, 1 - (a' - b)^2), \quad \text{where } a(n) - a'(n) = O(2^{-\operatorname{poly}(n)}).$$

Corollary 81. *If* $a - b = \Omega(\operatorname{poly}(n)^{-1})$ *then* $\mathrm{QIP}(m, a, b) \subseteq \mathrm{QIP}(m, 1, 1 - \operatorname{poly}(n)^{-1})$.

5.4.2 Parallelization

Contrary to the classic case, the number of messages in quantum interactive proofs can be made a small constant, namely, 3.

Theorem 82 (Kitaev, Watrous, 2000)**.** *Let* $m = \operatorname{poly}(n)$. *Then* $\mathrm{QIP}(m, 1, 1 - \varepsilon) \subseteq \mathrm{QIP}(3, 1, 1 - \varepsilon^2/(4m^2))$.

The idea of 3-message protocol is rather simple. W.l.o.g. we assume that the original protocol includes an odd number of messages.

Prover sends the whole history of communication in the original protocol. Verifier checks the correctness in a random point of the history.

This idea does not work in classical case. In quantum case quantum entanglement can be used to prevent Prover from cheating.

We present a more formal description of 3-message protocol. Let x be an input. In the original m-message perfect protocol Verifier applies operators $V_1, \ldots, V_k \in \mathbf{U}(\mathcal{V} \otimes \mathcal{M})$. Prover in the 3-message protocols can simulate all steps of the original protocol. Then the following steps are performed:

1. Prover sends registers $\boldsymbol{V}_1, \boldsymbol{M}_1, \ldots, \boldsymbol{V}_k, \boldsymbol{M}_k$, which are states in $\mathcal{V} \otimes \mathcal{M}$ during communication in the original protocol.
2. Verifier rejects if $\boldsymbol{V}_1$ does not contain all zeroes. Otherwise he applies V_k to $\boldsymbol{V}_k, \boldsymbol{M}_k$ and rejects if $\boldsymbol{V}_k, \boldsymbol{M}_k$ does not contain an accepting state. Otherwise he uncompute V_k (i.e. applies $V_k^\dagger$).
3. Verifier prepares the state $|\varphi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ in additional registers $\boldsymbol{B}, \boldsymbol{B}'$. Then he draws $r \in \{1, \ldots, k-1\}$ from the uniform distribution and applies V_r to $\boldsymbol{V}_r, \boldsymbol{M}_r$.
4. Verifier makes a controlled-swap between $\boldsymbol{V}_r$ and $\boldsymbol{V}_{r+1}$ with the control bit $\boldsymbol{B}$. He sends $\boldsymbol{M}_r, \boldsymbol{M}_{r+1}, \boldsymbol{B}', r$ to Prover.
5. Prover returns $\boldsymbol{B}'$. Verifier performs CNOT on $\boldsymbol{B}, \boldsymbol{B}'$. Then he applies Hadamard to the register $\boldsymbol{B}$ and accepts if $\boldsymbol{B}$ contains 0.

Let us check the perfect completeness of this protocol. If $x \in L_1$ then probability of error in the original protocol is 0. Let $P_1, \ldots, P_k$ be an optimal sequence of Prover actions in the original protocol. In 3-messages protocol actions of Prover are controlled by the value of $\boldsymbol{B}'$. If it contains 0 then Prover does nothing. Otherwise Prover applies P_{r+1} to $\boldsymbol{M}_r, \boldsymbol{P}_r$ and swaps $\boldsymbol{M}_r, \boldsymbol{P}_r$ and $\boldsymbol{M}_{r+1}, \boldsymbol{P}_{r+1}$.

For any value of $\boldsymbol{B}'$ the registers $\boldsymbol{V}_r, \boldsymbol{M}_r, \boldsymbol{P}_r$ and $\boldsymbol{V}_{r+1}, \boldsymbol{M}_{r+1}, \boldsymbol{P}_{r+1}$ after actions of Verifier and Prover are in the same state.

Thus before step 5 the state of Verifier's registers is

$$|00\rangle \otimes \rho + |11\rangle \otimes \rho$$

The state before the final measurement is easily calculated

$$(|00\rangle + |11\rangle) \otimes \rho \xrightarrow{\Lambda(X)[1,2]} (|00\rangle + |10\rangle) \otimes \rho \xrightarrow{H} |00\rangle \otimes \rho.$$

Therefore Verifier accepts with probability 1.

The proof of soundness condition is much more technical and omitted here.

5.4.3 Amplification

For 3-messages QIP with perfect completeness the same idea of amplification probabilities does work. The protocol is performed many times and Verifier accepts if he accepts in each copy.

Theorem 83 (Kitaev, Watrous, 2000)**.** $\mathrm{QIP}(3, 1, \varepsilon) \subseteq \mathrm{QIP}(3, 1, \varepsilon^r)$ *for any* $r = \mathrm{poly}(n)$.

5.4.4 Simplification of 3-message protocols: single-coin Arthur

The class QMAM is the class of problems admitting 3-messages QIP such that Verifier message is a sequence of random bits. Recall that such a verifier is called Arthur. Single-coin Arthur sends one random bit.

It is quite surprising that this very restrictive form of the protocol has the same power as general 3-message protocols.

Theorem 84 (Marriot, Watrous, 2005)**.** QMAM$(1, 1/2 + 2^{-r}) =$ QIP *for any* $r = \text{poly}(n)$.

Note that the soundness parameter in a single-coin QMAM can not be amplified. Prover can guess the value of random bit and convince Arthur with probability at least 1/2.

Due to previous results it is sufficient to consider QIP with 3 messages with perfect completeness and soundness 2^{-2r}.

Let $(L_0, L_1) \in \text{QIP}(3, 1, 2^{-2r})$. Fix an input x. Verifier applies circuits V_1 and V_2 in QIP. Prover's optimal actions are P_1 and P_2.

We denote the number of qubits in the message register by m, the number of qubits in the prover register by l and the number of qubits in the verifier register by k.

Single-coin QMAM protocol simulating 3-message protocol is described using these notations as follows:

1. Merlin sends a k-qubit register $\boldsymbol{V}$ to Arthur.
2. Arthur flips a fair coin and send the resulting random bit (1 or 0) to Merlin.
3. Merlin sends a m-qubit register $\boldsymbol{M}$ to Arthur.
4. If the value of random bit is 1 then Arthur applies V_2 to the register $(\boldsymbol{V}, \boldsymbol{M})$ and accept iff the value of the first qubit is 1 (it resembles the acceptance in the original protocol). Otherwise Arthur applies $V_1^\dagger$ to $(\boldsymbol{V}, \boldsymbol{M})$ and accepts iff all qubits of $\boldsymbol{V}$ are zero.

If $x \in L_1$ then for some P_1 and P_2 Verifier in the original protocol accepts with probability 1. In QMAM Merlin should act in the following way:

1. Apply P_1 to registers $(\boldsymbol{M}, \boldsymbol{P})$, where $\boldsymbol{P}$ corresponds to the l-qubit prover register in the original protocol.
2. Apply V_1 to $(\boldsymbol{V}, \boldsymbol{M})$ and send $\boldsymbol{V}$.
3. If a random bit is 1 then apply P_2 to $(\boldsymbol{M}, \boldsymbol{P})$ and send $\boldsymbol{M}$. Otherwise just send $\boldsymbol{M}$ without any action.

This protocol has perfect completeness. If the random bit is 1 then registers are the same as in the original protocol and Arthur accepts with probability 1. If the random bit is 0 then $\boldsymbol{V}$ is restored to the initial state of all zeroes. Thus Arthur also accepts with probability 1.

To check soundness condition a more sophisticated analysis of the protocol is needed. We skip it here.

5.4.5 Ideas to prove QIP $\subseteq$ PSPACE

Merriot–Watrous theorem reduces the proof of inclusion QIP $\subseteq$ PSPACE to the proof of inclusion QMAM$(1, 1 + 2^{-r}) \subseteq$ PSPACE and it is sufficient to consider only single-coin protocols.

We represent Arthur actions in a single-coin QMAM in the following way. Arthur receives a register $\boldsymbol{V}$, generates a random bit a and sends it to Merlin, then he receives a second register $\boldsymbol{M}$. After that he measures the registers with respect to a binary-values measurement

$$\{P_a, I - P_a\} \subset \text{Pos}(\mathcal{V} \otimes \mathcal{M}).$$

(In the previous description Arthur applies a unitary transformation and then measures the first qubit. Now we give him more freedom.)

If a state of $(\boldsymbol{V}, \boldsymbol{M})$ is ρ and the random bit is a then probability to accept is $\langle P_a, \rho \rangle$.

Now we introduce a register $\boldsymbol{X}$ corresponding to the random bit and an operator

$$Q = \frac{1}{2}|0\rangle\langle 0| \otimes P_0 + \frac{1}{2}|1\rangle\langle 1| \otimes P_1.$$

If Merlin is able to make states ρ_a before Arthur's measurement then the probability to accept is

$$\frac{1}{2}\langle P_0, \rho_0 \rangle + \frac{1}{2}\langle P_1, \rho_1 \rangle = \langle Q, X \rangle, \quad \text{where } X = \frac{1}{2}|0\rangle\langle 0| \otimes \rho_0 + \frac{1}{2}|1\rangle\langle 1| \otimes \rho_1. \tag{5}$$

Merlin can not choose ρ_a arbitrary. He can not change the register $\boldsymbol{V}$ after receiving the random bit. Thus

$$\mathrm{Tr}_{\mathcal{M}}(\rho_0) = \sigma = \mathrm{Tr}_{\mathcal{M}}(\rho_1), \quad \sigma \in \mathbb{D}(\mathcal{V}), \tag{6}$$

and it is the only constraint on the Merlin's choice. He may hold a purification of the state σ. In this case he can set the state of $(\boldsymbol{V}, \boldsymbol{M})$ to any choice of ρ_0 and ρ_1 satisfying the above equality.

The above constraint implies that

$$\mathrm{Tr}_{\mathcal{M}}(X) = I \otimes \sigma.$$

In the other direction, for any $X \in \mathrm{Pos}(\mathcal{X} \otimes \mathcal{V} \otimes \mathcal{M})$ such that $\mathrm{Tr}_{\mathcal{M}}(X) = I \otimes \sigma$ let's define

$$\rho_a = (\langle a| \otimes I_{\mathcal{V}\otimes\mathcal{M}})X(|a\rangle \otimes I_{\mathcal{V}\otimes\mathcal{M}}).$$

Then Equations (5) and (6) hold.

Thus the maximum of acceptance probability is the maximum in the following SDP program:

$$\begin{aligned}
&\langle Q, X \rangle \to \max \\
&\mathrm{Tr}_{\mathcal{M}}(X) \leq I_{\mathcal{X}} \otimes \sigma, \\
&X \in \mathrm{Pos}(\mathcal{X} \otimes \mathcal{V} \otimes \mathcal{M}), \\
&\sigma \in \mathbb{D}(\mathcal{V}).
\end{aligned}$$

The program has an exponential size. So, the above arguments show that QIP $\subseteq$ EXPTIME. Jain, Ji, Upahyay and Watrous propose a PSPACE algorithm to solve this particular form of SDP problem.

The idea of their algorithm is to construct a sequence of operators that converges to the maximum. Operators in the sequence have exponential size. But they can be expressed in succinct way. It means that there are algorithms running in polynomial space that approximate matrix elements of the operators in the sequence.

More exactly, the matrix elements of operators in consideration can be computed by uniform families of Boolean circuits of polynomial depth. 'Uniform' means that gates in the circuits can be computed efficiently by their indices.

It reduces an evaluation of the maximum in the SDP problem to evaluation of a uniform circuit of polynomial depth. It can be done recursively using polynomial space.

6 The bibliography

The bibliography does not pretend on completeness. References below are directly related to the lectures.

Books on quantum computation

[1] A. Kitaev, A. Shen, and M. Vyalyi. Classical and Quantum Computation, volume 47 of Graduate Studies in Mathematics. American Mathematical Society, 2002.

[2] M. A. Nielsen and I. L. Chuang. Quantum Computation and Quantum Information. Cambridge University Press, 2000.

Books on complexity

[3] Complexity Zoo: `https://complexityzoo.uwaterloo.ca/Complexity_Zoo`

[4] Arora S., Barak B. Computational complexity: a modern approach. Cambridge, UK: Cambridge Univ. Press, 2009.

[5] Ch. M. Papadimitriou. Computational complexity. Addison-Wesley, 1994.

[6] Sipser M. Introduction to the theory of computation. 3rd edition. Independence, KY, USA: Course Technology, 2012.

Papers on quantum computation

[7] S. Aaronson. BQP and the Polynomial Hierarchy, in Proceedings of ACM STOC 2010, pages 141-150. arXiv:0910.4698, ECCC TR09-104.

[8] S. Aaronson and A. Ambainis. Forrelation: A Problem that Optimally Separates Quantum from Classical Computing arXiv:1411.5729, ECCC TR14-155.

[9] L. Adleman, J. DeMarrais, and M. Huang. Quantum computability, SIAM Journal on Computing 26:1524-1540, 1997.

[10] D. Aharonov, van Dam, Kempe, Landau, Lloyd. Adiabatic Quantum Computation is Equivalent to Standard Quantum Computation. SIAM Journal of Computing, Vol. 37, Issue 1, p. 166-194 (2007). Preprint: arXiv:quant-ph/0405098v2

[11] Andris Ambainis, Oded Regev. An Elementary Proof of the Quantum Adiabatic Theorem. arXiv:quant-ph/0411152v2

[12] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter, Elementary gates for quantum computation, Phys. Rev. Ser. **A52** (1995), pp. 3457–3467; e-print `quant-ph/9503016`.

[13] C. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. SIAM J. Comput., 26(5):1510–1523, 1997. quant-ph/9701001.

[14] E. Bernstein and U. Vazirani. Quantum complexity theory. SIAM J. Comput., 26(5):1411–1473, 1997.

[15] Andrew M. Childs, Wim van Dam. Quantum algorithms for algebraic problems. Reviews of Modern Physics 82, 1-52 (2010). Preprint: arXiv:0812.0380v1

[16] D. Deutsch, Quantum computational networks, Proc. Roy. Soc. Lond. A 425, 73 (1989).

[17] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Michael Sipser. Quantum Computation by Adiabatic Evolution. 2000. arXiv:quant-ph/0001106v1.

[18] Edward Farhi, Jeffrey Goldstone, Sam Gutmann. A Quantum Approximate Optimization Algorithm, 2014. arXiv:1411.4028.

[19] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann, A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem, 2014, arXiv:1412.6062.

[20] Julia Kempe, Alexei Kitaev, Oded Regev. The Complexity of the Local Hamiltonian Problem. SIAM Journal of Computing, Vol. 35(5), p. 1070-1097 (2006). Preprint: arXiv:quant-ph/0406180v2.

[21] J. Kempe and O. Regev. 3-local Hamiltonian is QMA-complete. Quantum Information & Computation, 3(3):258-264, 2003.

[22] R. Jain, Z. Ji, S. Upadhyay, and J. Watrous. QIP = PSPACE. Journal of the ACM 58(6): article 30, 2011. Prerint: arXiv:0907.4737v2.

[23] A.Yu.Kitaev. Quantum measurements and the Abelian Stabilizer Problem. 1995. arXiv:quant-ph/9511026v1.

[24] A. Kitaev and J. Watrous. Parallelization, amplification, and exponential time simulation of quantum interactive proof systems. Proceedings of the 32nd ACM Symposium on Theory of Computing, pages 608-617, 2000.

[25] C. Marriott and J. Watrous. Quantum Arthur-Merlin games. Computational Complexity, 14(2): 122-152, 2005.

[26] Y. Shi. Both Toffoli and controlled-NOT need little help to do universal quantum computation. Quantum Information and Computation, 3(1):84–92, 2002. quant-ph/0205115.

[27] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM J. Comput. **26** (1997), 1484–1509; e-print arXiv:quant-ph/9508027.

[28] J. Watrous. Succinct quantum proofs for properties of finite groups, Proceedings of IEEE FOCS'2000, pp. 537-546, 2000. arXiv:cs.CC/0009002.

Papers on complexity and classical algorithms

[29] L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes, Journal of Computer and Systems Sciences 36:254-276, 1988.

[30] T. Baker, J. Gill, and R. Solovay. Relativizations of the P $\neq$ NP question, SIAM Journal on Computing 4:431-442, 1975.

[31] Boaz Barak, Ankur Moitra, Ryan O'Donnell, Prasad Raghavendra, Oded Regev, David Steurer, Luca Trevisan, Aravindan Vijayaraghavan, David Witmer, John Wright. Beating the random assignment on constraint satisfaction problems of bounded degree. ECCC, 2015. TR15–082.

[32] C. H. Bennett and J. Gill. Relative to a random oracle $\mathrm{P}^A \neq \mathrm{NP}^A \neq \text{co-NP}^A$ with probability 1. SIAM J. Comput., 10(1):96–113, 1981.

[33] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. Journal of Computer and System Sciences, 48(1):116-148, 1994.

[34] Goemans M., Williamson D. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM. 1995. V. 42. P. 1115–1145.

[35] S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In Randomness and Computation, volume 5 of Advances in Computing Research. JAI Press, 1989.

[36] Y. Han, L. Hemaspaandra, and T. Thierauf. Threshold computation and cryptographic security, SIAM Journal on Computing 26(1):59-78, 1997.

[37] Håstad J. Some optimal inapproximability results. Journal of the ACM. 2001. V. 48. P. 798–869.

[38] S. Khot. On the power of unique 2-prover 1-round games. In Proc. 34th ACM Symposium on Theory of Computing, 2002.

[39] Lászó Lovász, Coverings and coloring of hypergraphs, Proceedings of the Fourth Southeastern Conference on Combinatorics, Graph Theory, and Computing, Utilitas Math., Winnipeg, Man., 1973, pp. 3–12.

[40] A. Shamir, IP=PSPACE, J. Assoc. Comput. Mach. **39** (1992), no. 4, 869–877.

[41] A. Shen, IP=PSPACE: simplified proof, J. Assoc. Comput. Mach. **39** (1992), no. 4, 878–880.

[42] Larry J. Stockmeyer. Planar 3-colorability is polynomial complete. ACM SIGACT News, vol. 5, no. 3, 1973.

[43] Luca Trevisan. Non-approximability results for optimization problems on bounded degree instances, 2001. Proc. 30th STOC, pp 453-461.