Baltic J. Modern Computing, Vol. 1 (2013), No. 1-2, 101-129

Quantum Query Algorithms

Alina VASILIEVA

Faculty of Computing, University of Latvia
Raina bulv. 29, LV-1459, Riga, Latvia

Alina.Vasiljeva@lu.lv

Abstract. Quantum computing is a method of computation based on the laws of quantum mechanics. This subfield of computer science aims to employ quantum mechanical effects for the efficient performance of computational tasks. In this article, we review the work by the author in a field of quantum algorithms development. In the first part of the article, exact and bounded-error quantum query algorithms for computing Boolean functions are presented. In the second part, a query model is applied for computing multivalued functions. The third part is devoted to nondeterministic query algorithms.

Keywords: quantum computing, query algorithms, algorithm complexity, algorithm design.

1. Introduction

Quantum computing is a subfield of computer science based on the laws of quantum mechanics. It applies the quantum mechanical effects for more efficient solution of computational problems than in a classical way. This area of science unites disciplines such as physics, mathematics and computer science. In the scope of computer science and mathematics the theoretical aspects, the potential and the limitations of quantum computers are studied. At the same time, physicists are working on developing practical implementations of quantum computing devices. This branch of science is very topical because it is a proven fact that quantum computing can solve certain problems faster than classical computing (Shor, 1997), (Grover,1996). Although quantum computers are not yet available to everyone, many scientists all over the world are working to make them universally available in the future. The physical implementation is very complex; however, several quantum computer prototypes have been developed and are used to solve computational problems (DiCarlo et al., 2009), (Politi et al., 2009), (Johnson et al., 2011). Nevertheless, these are still only demonstration models of quantum processors consisting of a small number of qubits. Theoretical results in the field of quantum information processing are already successfully implemented in such areas as quantum cryptography (Hiskett et al., 2006), (Dixon et al., 2008) and quantum teleportation (Jin et al., 2010), (Lee et al.,2011). Several companies assert they are prepared for marketing commercial quantum computing systems (IDQ, D-Wave). In spite of the progress achieved, there is still a long way to go until quantum computers are capable of solving real computational tasks. To be able to build complex and efficient quantum computer systems in the future, it is necessary to develop and improve theoretical foundations of quantum computing today.

1.1. Research Object

Algorithm complexity theory is a sub-field of computer science investigating the complexity of computational problems. One of the main tasks in complexity theory is designing efficient algorithms. The main object of the research is the query model (Buhrman and de Wolf, 2002). In this model, the definition of the function *f*(*X*) is known, but input $X=(x_1,x_2,...,x_N)$ is hidden in a black box. Input values can be accessed only by querying the black box about x_i values. In the process of computation, the query algorithm asks questions about variable values, receives answers from the black box, performs the computation, and finally produces the function value output. The goal is to develop a query algorithm that would compute the value of a certain function correctly for an arbitrary input. The complexity of a query algorithm is measured by the number of questions it asks based on worst-case input.

There are examples of efficient and impressive quantum algorithms already developed. The most famous are Shor's integer factoring algorithm (Shor, 1997), Grover's search algorithm (Grover, 1996) and algorithm for *XOR* function (Buhrman and de Wolf, 2002). Other examples of lower and upper bound estimations of quantum query algorithm complexity can be found in (Nielsen and Chuang, 2000), (de Wolf, 2001), (Ambainis and de Wolf, 2001), (Ambainis, 2004), (Ambainis, 2006), (Kravcevs, 2008), (Ščeguļnaja-Dubrovska, 2010).

1.2. Objectives of the Research

The overall goal of the research is to develop new, fast and efficient quantum algorithms for solving specific computational problems, as well as to improve the general construction techniques for algorithms. It is important to work out an approach for designing efficient quantum algorithms for arbitrary functions. A collection of existing methods (including, for instance, a method for evaluation of NAND formulas (Ambainis et al., 2010)) is not sufficiently large and the computation of arbitrary function is a complex task. The research is aimed at extending the collection of quantum algorithm constructing methods by introducing new methods.

There exist different types of query algorithms: deterministic, probabilistic, and nondeterministic, each exhibiting a specific behavior and defining different conditions for an algorithm to produce the correct result. In the quantum query model the following counterparts are studied: exact, bounded-error and nondeterministic quantum query algorithms. In this respect the aim of the study is to analyze relations between different complexity measures, to compare the classical and the quantum complexity of specific computational problems and finally to find examples with large separation between the optimal classical complexity and the quantum complexity of the same problem.

1.3. Summary of Results

The results consist of three parts:

- Quantum query algorithms for Boolean functions.
- Quantum query algorithms for multivalued functions.
- Nondeterministic query algorithms.

Summary of the first part of results:

- Several exact quantum query algorithm construction methods are presented, including a set of basic efficient algorithms for finite functions. The algorithm transformation and concatenation methods are helpful for enlarging the set of efficient exact algorithms. The methods can be used for generating examples of N versus $2N$ gaps between quantum and classical query complexity of a function.
- An exact quantum algorithm for verification of repetition codes is developed. The algorithm complexity is N while classically $2N$ queries are required.
- A method for constructing a bounded-error quantum query algorithm for conjunction $f = f_1 \wedge f_2$ using exact quantum query algorithms for sub-functions f_1 and f_2 is developed. The correct probability of a correct answer is $p = 4/5$ and the complexity is equal to the largest complexity of sub-algorithms: $\max(Q_E(f_1), Q_E(f_2))$.

In the second part of the article, quantum query algorithms for computing multivalued functions are examined:

- Different types of algorithms for computing multivalued functions are introduced.
- Examples are demonstrated where the quantum query algorithm complexity is lower than in the classical case.

In our opinion, this section contains the most important results. The main result is the example developed where the quantum query complexity of the function is N, while classically $3N$ queries are required to compute the same function. Function is not based on *XOR* operation and there is no error probability for an algorithm.

In the third part, nondeterministic query algorithms are examined. First, the results of designing algorithms according to the traditional nondeterministic query model are presented. A new type of algorithm is introduced: the dual nondeterministic quantum query algorithm. The properties of such algorithms are investigated. Examples of efficient dual nondeterministic quantum query algorithms are demonstrated. Second, a new alternative model for nondeterministic computation is proposed. The elaborated model is demonstrated through the example of computing a specific Boolean function, for which the gap between deterministic and nondeterministic query complexity is demonstrated to be 7^N versus $O(3^N)$.

Most part of results has been already published and we provide references to publications at the beginning of each section. More details and full proofs are available in referenced papers and (Vasilieva, 2012).

2. Theoretical Background

2.1. Classical Decision Trees

The classical version of the query model is known as *decision trees.* The definition of the Boolean function $f(X)$ is known, but the black box contains the input $X = (x_1, x_2, ..., x_N)$ and can be accessed by querying x_i values. The algorithm must allow determination of the correct value of a function for an arbitrary input. The complexity of the algorithm is measured by the number of queries on the worst-case input. For more details, see the survey on decision tree complexity (Buhrman and de Wolf, 2002).

A *deterministic decision tree* is a tree with internal nodes labeled with variables x_i, arrows labeled with possible variable values and leafs labeled with function values. The deterministic decision tree always follows the same computational path for each input and produces correct result with probability $p = 1$. The deterministic complexity of a function f is denoted by $D(f)$.

A *probabilistic decision tree* may contain internal nodes with a probabilistic branching, i.e., multiple arrows exiting from the above node, each one labeled with a probability for algorithm to follow. The total probability to obtain a result r after application of an algorithm on certain input X equals the sum of probabilities for each leaf labeled with r to be reached.

A *nondeterministic decision tree* differs from the deterministic one by an additional possibility that there can be more than one arrow labeled with the same value exiting each tree vertex. The nondeterministic decision tree computes Boolean function $f(X)$, if for an arbitrary input X it is true that:

- if $f(X)=1$, then a path exists from the root to the leaf with result 1;
- if $f(X)=0$, then a path exists from the root to the leaf with result 0;
- there is no path from the tree root to the leaf with incorrect function value.

2.2. Quantum Computing

The basics of quantum computing are available in the following books and papers: (Feynman, 1982), (Deutsch, 1985), (Cleve et al., 1998), (Gruska, 1999), (Childs et al., 2003), (Kaye et al., 2007).

An n-dimensional quantum pure state is a unit vector in an n-dimensional Hilbert space. Let $|0\rangle, |1\rangle, ..., |n-1\rangle$ be an orthonormal basis for $\mathbb{C}^n$. Then, any state can be expressed as $|\psi\rangle = \sum_{i=0}^{n-1} \alpha_i |i\rangle$ for some $\alpha_i \in \mathbb{C}^n$. The norm of $|\psi\rangle$ is 1: $\sum_{i=0}^{n-1} |\alpha_i|^2 = 1$. States $|0\rangle, ..., |n-1\rangle$ are called *basis states*. Any state of the form $\sum_{i=0}^{n-1} \alpha_i |i\rangle$ is called a *superposition* of $|0\rangle, ..., |n-1\rangle$. The coefficient α_i is called an *amplitude* of $|i\rangle$.

The state of a quantum system can be changed by applying the *unitary transformation*. The unitary transformation U is a linear transformation on $\mathbb{C}^n$ that maps vector of unit norm to another or the same vector of unit norm. Formally, unitary transformation is represented by a unitary matrix.

There are various types of the quantum measurement; the quantum query model uses the simplest one – the full measurement in the computation basis. Performing this measurement on a state $|\psi\rangle = \alpha_0 |0\rangle + ... + \alpha_{n-1} |n-1\rangle$ produces the outcome $|i\rangle$ with probability $|\alpha_i|^2$. The measurement changes the state of the system to $|i\rangle$ and destroys the original state $|\psi\rangle$.

2.3. Quantum Query Model

The quantum query model is also known as the quantum black box model. This model is the quantum counterpart of decision trees and is intended for computing Boolean functions. For a detailed description, see (Buhrman and de Wolf, 2002), (Ambainis, 2004), (Cleve et al., 1998), (Kaye et al., 2007), (de Wolf, 2001).

A quantum computation with T queries is a sequence of unitary transformations:

$$U_0\,,\,Q_0\,,\,U_1\,,\,Q_1\,,\,...\,,\,U_{T-1}\,,\,Q_{T-1}\,,\,U_T\,.$$

U_i's can be arbitrary unitary transformations not depending on input bits. Q_i's are unitary query transformations. Computation starts in the initial state $\left|\vec{0}\right\rangle$. Then transformations U_0, Q_0,..., Q_{T-1}, U_T are applied and the final state measured.

In algorithms created by the present research, the following definition of a query transformation is used - if the input is a state $\left|\psi\right\rangle = \sum_i \alpha_i \left|i\right\rangle$, then the output is:

$$\left|\gamma_i\right\rangle = \sum_{i=0}^{n-1} (-1)^{\varphi_i}\,\alpha_i \left|i\right\rangle, \text{ where } \varphi_i \in \{x_1,...,x_N,0,1\}.$$

In other words, for each query for each basis state $\left|i\right\rangle$ a variable assignment φ_i may be arbitrarily chosen. It is also allowed to skip the variable assignment for any particular basis state, i.e. to set $\varphi_i = 0$ for $\left|i\right\rangle$; or inverse amplitude value sign by setting $\varphi_i = 1$ for a particular $\left|i\right\rangle$. Depending on the value of the assigned variable, the sign of the amplitude of the quantum basis state either changes to the opposite or remains unchanged.

Formally, any transformation has to be defined by a unitary matrix:

$$Q = \begin{pmatrix} (-1)^{\varphi_0} & 0 & ... & 0 \\ 0 & (-1)^{\varphi_1} & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & (-1)^{\varphi_{n-1}} \end{pmatrix}$$

After all query transformations Q_i are applied (alternating with fixed intermediate unitary transformations U_i), the last remaining action is to extract the result value from the final quantum state. It is achieved by measuring this state and interpreting the quantum basis state observed after that. A value of a function is assigned to each basis state. The probability of obtaining the result r after applying an algorithm on input X equals the sum of squared moduli of all amplitudes, which correspond to outputs with value r.

Quantum query algorithms can be conveniently represented in diagrams, and this approach is used throughout the paper. Fig. 1 demonstrates a graphical representation of an algorithm in a general form.

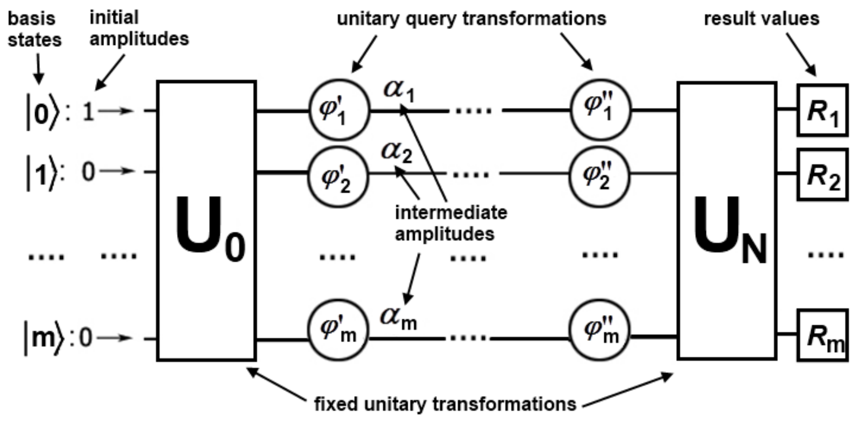

Fig. 1. Graphical representation of a quantum query algorithm

A quantum query algorithm *computes f exactly* if the output equals *f*(*X*) with probability $p = 1$, for all $X \in \{0,1\}^N$. Complexity is equal to the number of queries and is denoted by $Q_E(f)$ (Buhrman and de Wolf, 2002).

A quantum query algorithm *computes f with bounded-error* if the output equals *f*(*X*) with probability $p > 2/3$, for all $X \in \{0,1\}^N$. Complexity is equal to the number of queries and is denoted by $Q_p(f)$ (Buhrman and de Wolf, 2002).

3. Quantum Query Algorithms for Boolean Functions

In this section, the results of designing different types of quantum query algorithms are presented, simplifying the task of algorithm construction for an arbitrary function.

3.1. Exact Quantum Query Algorithms

This section is based on the papers (Dubrovska and Mischenko-Slatenkova, 2006), (Dubrovska, 2007), (Vasilieva, 2009).

Exact algorithms always produce a correct answer with probability *p* = 1. The error probability in algorithms of this kind is not allowed, the said limitation significantly complicating the design of the above algorithms. There are a significant number of efficient quantum algorithms with an error probability already developed. Applying those, the quantum algorithm speedup comparing to the best known classical algorithm is known to be quadratic (Grover, 1996) or even exponential (Shor, 1997). However, in certain types of computer software, we cannot allow even a small probability of error, for example, in spacecraft, aircraft, or medical software. For this reason, the development of exact algorithms is very important. By contrast with non-exact algorithms, the largest known complexity separation between the quantum exact and classical deterministic algorithm until recently was only *N* versus 2*N* for *XOR* function (Deutsch, 1985), (Cleve, 1998). Another category of exact quantum algorithms are algorithms for promise problems. In such problems, the domain of the function is restricted, i.e. input is promised to belong to a subset of all possible inputs. Examples of exact quantum query algorithms for promise problems can be found in (Deutsch and Jozsa, 1992), (Simon, 1994), (Freivalds and Iwama, 2009).

A long standing open question was whether it is possible to achieve a larger gap between quantum exact and classical deterministic query complexity of a total function with no error allowed. The conjecture about relation between complexity measures was the following:

$$Q_E(f) \geq \frac{D(f)}{2}.$$

Many authors have worked to either prove or refute this conjecture. This problem has been considered, for instance, in (Agadžanjans, 2010), (Lace, 2004), (Lace, 2008). Examples of a borderline gap of N versus $2N$ have been presented, but for a long time nobody has been able to improve this result. Just recently a first example was described in a paper (Ambainis, 2012) presenting an algorithm with the superlinear gap between classical deterministic and quantum exact algorithm complexity: N versus $O(N^{0.8675}...)$.

Our contribution consists of:

- new examples of N versus $2N$ complexity separation;
- techniques for enlarging a set of efficiently computable Boolean functions;
- methods for generating algorithms producing an infinite set of instances of N versus $2N$ complexity separations.

To simplify calculations in the process of algorithm development and debugging, i.e. to automate the verification process, we developed a simple program using *Wolfram Mathematica* software.

3.1.1. Basic Exact Quantum Query Algorithms

We start with exact quantum query algorithms for two problems of a finite input size. Both algorithms are of lower complexity than the best possible classical algorithms. Further these algorithms are used as a base for building advanced algorithms.

Function 1: $EQUALITY_3(X) = \neg(x_1 \oplus x_2) \wedge \neg(x_2 \oplus x_3)$.

Classical deterministic complexity: $D(EQUALITY_3) = 3$.

An exact quantum query algorithm with two questions is developed (Fig. 2).

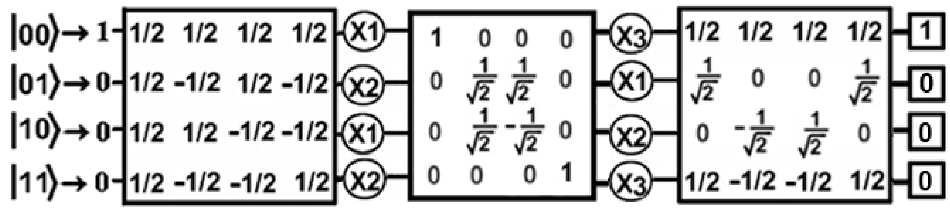

Fig. 2. Exact quantum query algorithm with two queries for *EQUALITY*$_3$

Function 2: $PAIR\_EQUALITY_4(x_1, x_2, x_3, x_4) = \neg(x_1 \oplus x_2) \wedge \neg(x_3 \oplus x_4)$.

Classical deterministic complexity: $D(PAIR\_EQUALITY_4) = 4$.

An exact quantum query algorithm with two questions is developed (Fig. 3).

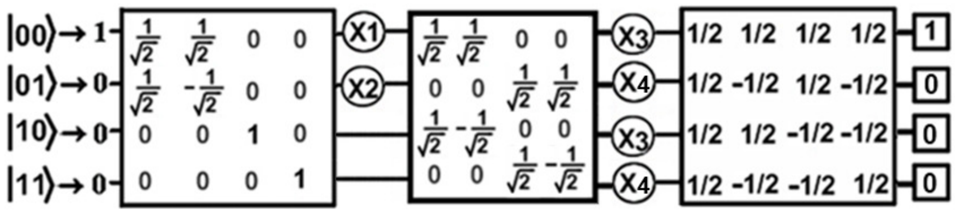

Fig. 3. Exact quantum query algorithm with two queries for *PAIR_EQUALITY*$_4$

3.1.2. Algorithm Transformation Methods

Quantum query algorithm transformation methods are introduced useful for enlarging a set of exactly computable Boolean functions. Each method receives an exact quantum query algorithm on input, processes it according to the rules producing as the result a slightly different exact algorithm computing another function.

Methods are the following:

- Output value assignment inversion.
- Output value assignment permutation.
- Permutation of query variables.

Applying transformation methods to two basic algorithms produced two sets of Boolean functions.

Set *QFunc3* – eight three-argument Boolean functions, where for each function there is an exact quantum query algorithm, which computes it with two queries.

Set *QFunc4* – twenty four four-argument Boolean functions, where for each function there is an exact quantum query algorithm, which computes it with two queries.

Table 1. Results of applying transformation methods to *EQUALITY*$_3$ algorithm

X	*EQUALITY*	Output value assignment permutation			Output value assignment inversion			
	(1,0,0,0)	(0,1,0,0)	(0,0,1,0)	(0,0,0,1)	(0,1,1,1)	(1,0,1,1)	(1,1,0,1)	(1,1,1,0)
000	1	0	0	0	0	1	1	1
001	0	0	0	1	1	1	1	0
010	0	0	1	0	1	1	0	1
011	0	1	0	0	1	0	1	1
100	0	1	0	0	1	0	1	1
101	0	0	1	0	1	1	0	1
110	0	0	0	1	1	1	1	0
111	1	0	0	0	0	1	1	1
D(f)	3	3	3	3	3	3	3	2
$Q_E(f)$	**2**	**2**	**2**	**2**	**2**	**2**	**2**	**2**

Transformation methods can be applied to every new exact quantum query algorithm, thus constructing a larger set of efficiently computable Boolean functions. Moreover, exact algorithms obtained this way further can be used as building blocks for more complex algorithms.

3.1.3. Exact Quantum Query Algorithm for Verifying Repetition Code

A repetition code is a data transmission error detection scheme. (r, N) code repeats each N-bit block r times (Cover and Thomas, 1991). Verification procedure for the repetition code involves checking whether in each group of r the consecutive blocks of size N all blocks are equal.

(2,1) repetition code verification procedure is represented as the process of computing the following Boolean function ($N = 2k$):

$$VERIFY_N(X) = \begin{cases} 1, \textit{ if } \left(x_1 = x_2\right) \wedge \left(x_3 = x_4\right) \wedge \left(x_5 = x_6\right) \wedge \ldots \wedge \left(x_{2k\text{-}1} = x_{2k}\right) \\ 0 \text{ , } \textit{otherwise} \end{cases}$$

We propose an exact quantum query algorithm computing $VERIFY_N$ function using $N/2$ queries, while classically N queries are required (Fig. 4).

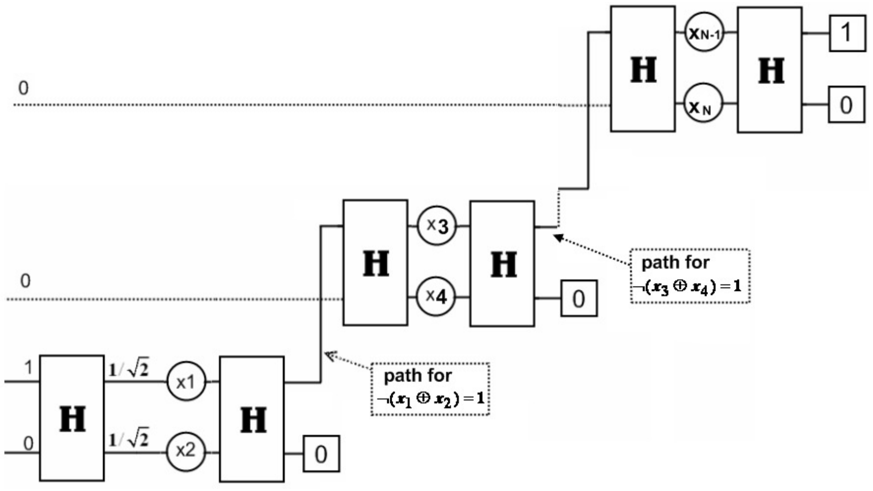
0
0
1
0
H
$1/\sqrt{2}$
$1/\sqrt{2}$
x1
x2
H
0
path for $\neg(x_1 \oplus x_2) = 1$
H
x3
x4
H
0
path for $\neg(x_3 \oplus x_4) = 1$
H
x_{N-1}
x_N
H
1
0

Fig. 4. Exact quantum algorithm for computing the Boolean function $VERIFY_N$

(r,1) repetition code verification procedure is represented as the process of computing the following Boolean function:

$$VERIFY^{r}_{r\cdot N}(X) = \begin{cases} 1, \textit{ if } EQUALITY_r\left(x_1,\ldots,x_r\right) \wedge \ldots \wedge EQUALITY_r\left(x_{(N-1)r+1},\ldots,x_{Nr}\right) \\ 0 \text{ , } \textit{otherwise} \end{cases}$$

Theorem 1. $D(VERIFY^{r}_{r\cdot N}) = rN$.

Proof idea. By function sensitivity on any accepting input.

Theorem 2. $Q_E(VERIFY^{r}_{r\cdot N}) = (r-1)N$.

Proof idea. We use the similar construction as in Fig. 4, but insert algorithms for $EQUALITY_r$ as sub-routines instead of algorithms for XOR_2.

3.1.4. Algorithm Concatenation Methods

Finally, we generalize described approaches for quantum query algorithm design and present methods for generating complex quantum algorithms from simple building blocks. Specifically, these methods can be used for generating examples of N versus $2N$ gaps between quantum and classical query complexity of the function.

Given an exact quantum query algorithms QA_1, QA_2 computing Boolean functions $f_1(X)$, $f_2(Y)$ with complexity $Q_E(QA_1) = m_1$, $Q_E(QA_2) = m_2$ it is possible to build a new exact quantum query algorithms with complexity $Q_E(QA_3) = m_1 + m_2$ for computing the following Boolean functions:

- conjunction of basic functions: $f_{\wedge}(XY) = f_1(X) \wedge f_2(Y)$;
- disjunction of basic functions: $f_{\vee}(XY) = f_1(X) \vee f_2(Y)$;
- *XOR* of basic functions: $f_{\oplus}(XY) = f_1(X) \oplus f_2(Y)$.

Fig. 5 demonstrates constructions for computing a conjunction and a disjunction of basic functions. In case of *XOR* operation, instances of the algorithm QA_2 have to be concatenated to algorithm's QA_1 outputs in a similar way. However, this time the second algorithm has to be concatenated to all outputs – both, the accepting and the rejecting. Additionally, for QA_2 instances concatenated to accepting outputs of QA_1, the result values assigned to the states have to be inversed.

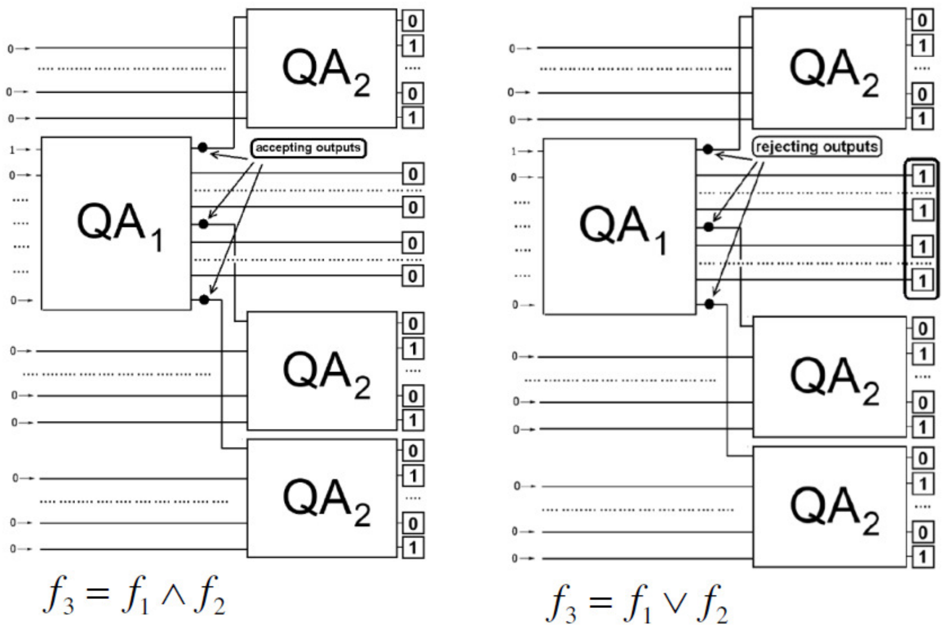

Fig. 5. Algorithm concatenation methods for computing conjunction and disjunction

3.2. Bounded-Error Quantum Query Algorithms

This section is based on the paper (Vasilieva and Mischenko-Slatenkova, 2010).

Every Boolean function can be presented as a logical formula in a conjunctive normal form (CNF). Formula is in CNF if it is a conjunction (*AND*s) of disjunctions (*OR*s) of variables or negated variables. Therefore, a fast and efficient algorithm for conjunction is necessary. Algorithms for computing this kind of function have been previously presented in (Dubrovska, 2007), (Kerenidis and de Wolf, 2004), (Lace, 2008).

We present a bounded-error quantum query algorithm for computing the conjunction $f(XY) = f_1(X) \wedge f_2(Y)$, which improves previous results.

Fig. 6 presents a quantum algorithm, which computes the conjunction of two bits $f(x_1, x_2) = x_1 \wedge x_2$ using one query with correct answer probability $p = 4/5$.

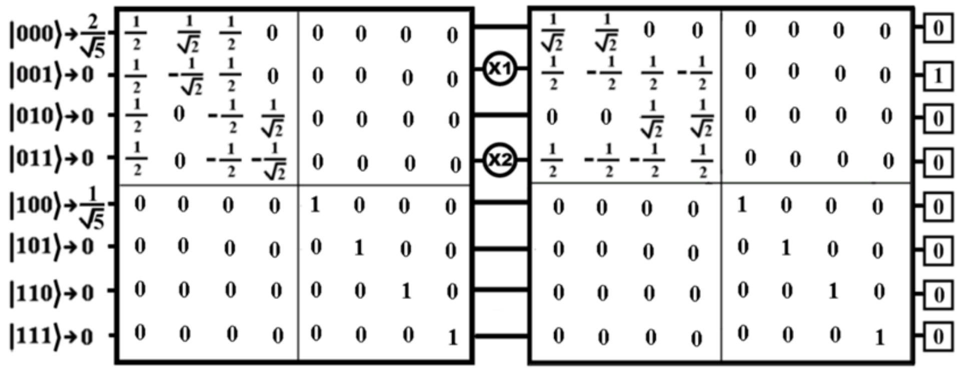

Fig. 6. Quantum algorithm computing the conjunction of two bits

Classical complexity lower bound is the following: the Boolean function $AND_2(x_1, x_2)$ can be computed by a randomized classical decision tree with one query with the maximum probability $p = 2/3$.

During the analysis of an algorithm a way to generalize it for computing the conjunction of sub-functions was discovered. The following algorithm construction method is formulated.

Given exact quantum query algorithms QA_1, QA_2 for computing Boolean functions $f_1(X_1)$, $f_2(X_2)$ with complexity $Q_E(QA_1) = m_1$, $Q_E(QA_2) = m_2$ it is possible to build a quantum query algorithm computing a function $f(XY) = f_1(X) \wedge f_2(Y)$ with probability $p = 4/5$ and complexity $Q_E(QA) = \max(m_1, m_2)$.

It should be noted that the method is applicable to basic exact algorithms satisfying specific properties.

The most significant advantage of this method is that the overall algorithm complexity does not exceed the greatest complexity of sub-algorithms. To compute a composite function, additional queries are not required. However, the cost for efficient computing is the error probability.

Proposed quantum query algorithm for computing conjunction is more efficient than the best possible classical deterministic analogue and ensures better correct answer probability than the best possible classical probabilistic algorithm.

Further action line for algorithm improvement is the following:

- to extend the number of clauses of computable conjunction to *N*;

- to increase the correct answer probability.

4. Quantum Query Algorithms for Multivalued Functions

This chapter is based on the papers (Vasilieva, 2010), (Vasilieva, 2011).

The query model is mostly used to compute Boolean functions. However it is possible to apply the query model to functions with larger domain and wider range as well. In this section, we consider even a more uncommon case: computing of *multivalued functions (multifunctions).* The study of the query complexity of multifunctions has been inspired by the book on communication complexity (Kushilevitz and Nisan, 1997).

Multifunction is a left-total binary relation associating values from the domain set with one or more values from a range set. Function is simply a special case of multifunction, where each value from a domain set is mapped to no more than one value from a range set.

We consider the following kind of multifunctions:

$$M(X):\{0,1\}^N \to \mathbb{N}\text{, where } X=(x_1,x_2,...,x_N)\text{, } x_i\in\{0,1\}.$$

The major motivation for studying query complexity of multivalued function is a potential possibility to achieve a larger gap between quantum and classical query complexity than *N* versus 2*N*.

We analyze the possibility of computing multivalued functions in the query model and propose different types of query algorithms for this task. Three examples are demonstrated where the quantum query algorithm complexity is lower than the classical query algorithm complexity.

4.1. Computing Multifunctions in a Query Model

Computation of usual functions in a query model has been studied in detail: for each input, the algorithm has to output correct function value with a certain probability. However, it is not obvious how to extend a query model to compute multivalued functions. We propose three different ways of description on how the query algorithm computes a multifunction and define three types of query algorithms based on these options.

Definition 1. *The query algorithm computes multifunction M(X) in a* ***definite manner****, if for each X it outputs one certain correct value from the result set with probability p = 1. The classical query complexity is denoted by $C_D(M)$. The quantum query complexity is denoted by $Q_D(M)$.*

The type of the classical decision tree computing a multifunction in a definite manner is the deterministic decision tree. In the quantum version, the corresponding algorithm type is the exact quantum query algorithm.

Definition 2. *The query algorithm computes multifunction M(X) in a* ***randomly distributed manner****, if for each X it outputs arbitrary values from a result set with arbitrary probabilities (for each value such probability has to be positive) never delivering an incorrect value. The classical query complexity is denoted by $C_{RD}(M)$. The quantum query complexity is denoted by $Q_{RD}(M)$.*

The above definition is more natural and takes account of the essence of multifunction as a mathematical object. In a classical query model, probabilistic decision trees should be used to produce the described behavior. Quantum query algorithms seem to be better suited for computing multifunctions in a distributed manner because of the superposition principle.

Definition 3. *The query algorithm computes multifunction M(X) in a* ***uniformly distributed manner****, if for each X it outputs each value from a result set with equal probability never delivering an incorrect value. The classical query complexity is denoted by* $C_{UD}(M)$*. The quantum query complexity is denoted by* $Q_{UD}(M)$*.*

This definition poses a serious constraint to designing of a query algorithm. However, in our opinion this definition is the most reasonable in the sense of computing a multifunction and algorithms of that type have the most practical applications.

4.2. Example 1 of Computing a Multifunction

First, a three-variable multifunction $M_1 : \{0,1\}^3 \to \{1,2,3,4\}$ is examined, that is defined in Table 2.

The classical uniformly distributed query algorithm complexity lower bound is the following: $C_{UD}(M_1) = 2$.

The quantum query algorithm is demonstrated in Fig. 7 that computes the multifunction using one query: $Q_{UD}(M_1) = 1$.

Table 2. Definition of the multifunction M_1

X	$M_1(X)$	X	$M_1(X)$
000	{ 1 , 3 }	100	{ 2 , 4 }
001	{ 1 , 4 }	101	{ 2 , 3 }
010	{ 2 , 3 }	110	{ 1 , 4 }
011	{ 2 , 4 }	111	{ 1 , 3 }

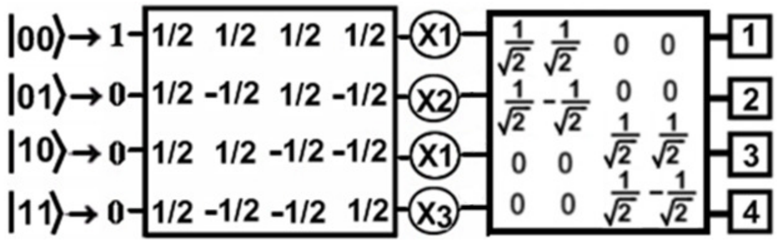

Fig. 7. Uniformly distributed quantum algorithm for multifunction M_1

Subsequently, multifunction is generalized to the case of N variables:

$M_1^{GEN1} : \{0,1\}^N \to \{1,2,...,2(N-1)\}$, where

$\forall\ 1 < i \le N$: if $(x_1 \oplus x_i = 0)$, then $M_1^{GEN1}(X) = 2(i-1)-1$

otherwise $M_1^{GEN1}(X) = 2(i-1)$

Classically two queries are required to compute the multifunction in a uniformly distributed manner: $C_{UD}(M_1^{GEN1}) = 2$.

The quantum query algorithm is demonstrated in Fig. 8 that computes the multifunction using one query: $Q_{UD}(M_1^{GEN1}) = 1$.

The second generalization of multifunction is the following:

$$M_1^{GEN2} : \{0,1\}^{N^2} \to \{1,2,...,2(N-1)\}\text{, where}$$

$\forall\ 1 < i \le N$: if $((x_1 \oplus x_2 \oplus ... \oplus x_N) \oplus (x_{(i-1)N+1} \oplus x_{(i-1)N+2} \oplus ... \oplus x_{(i-1)N+N}) = 0)$,

then $M_1^{GEN2}(X) = 2(i-1)-1$

otherwise $M_1^{GEN2}(X) = 2(i-1)$

The classical algorithm complexity lower bound is: $C_{UD}(M_1^{GEN2}) = 2N$.

The quantum query algorithm is demonstrated in Fig. 9 that computes the multifunction using N queries: $Q_{UD}(M_1^{GEN2}) \le N$.

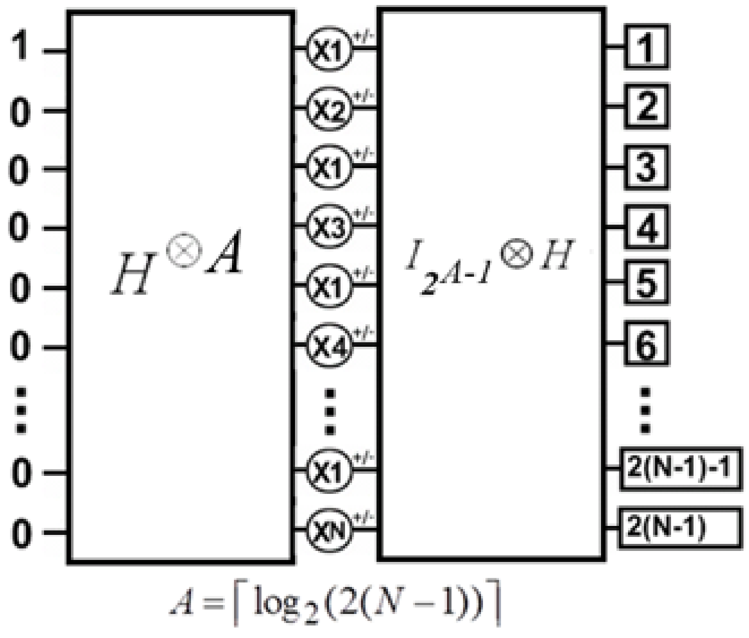

Fig. 8. Uniformly distributed quantum algorithm for multifunction M_1^{GEN1}

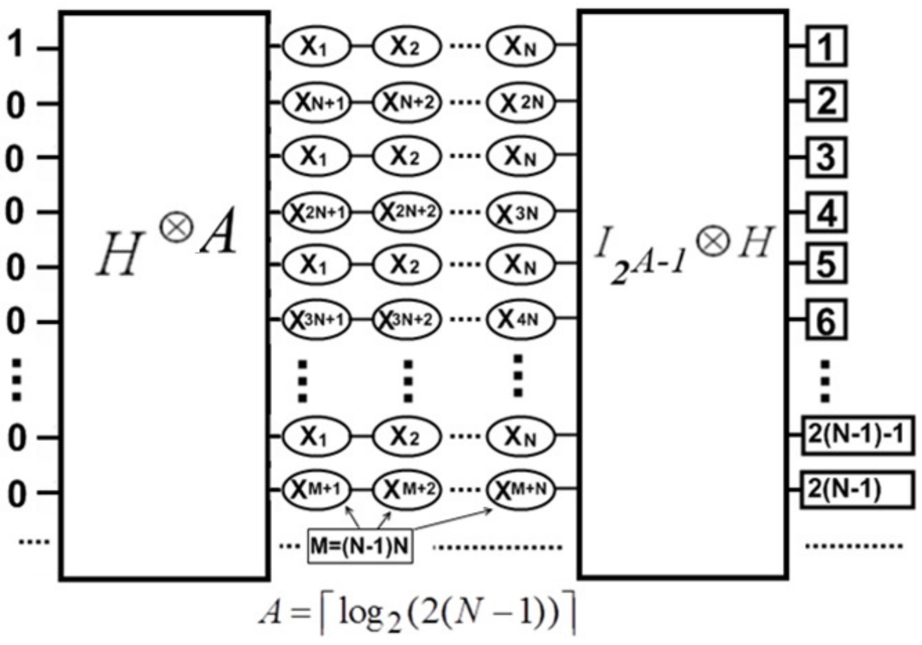
1
0
0
0
0
0
0
0
$H^{\otimes A}$
X_1 X_2 X_N
X_{N+1} X_{N+2} X_{2N}
X_{2N+1} X_{2N+2} X_{3N}
X_{3N+1} X_{3N+2} X_{4N}
X_{M+1} X_{M+2} X_{M+N}
M=(N-1)N
$I_{2^{A-1}} \otimes H$
1
2
3
4
5
6
2(N-1)-1
2(N-1)
$A = \lceil \log_2(2(N-1)) \rceil$

Fig. 9. Uniformly distributed quantum algorithm for multifunction M_1^{GEN2}

4.3. Example 2 of Computing a Multifunction

First, a four-variable multifunction $M_2 : \{0,1\}^4 \to \{1,2,3,4\}$ is examined, that is defined in Table 3.

Table 3. Definition of the multifunction M_2

X	$M_2(X)$	X	$M_2(X)$
0000	{1}	1000	{1,2,3,4}
0001	{1,2,3,4}	1001	{4}
0010	{1,2,3,4}	1010	{2}
0011	{3}	1011	{1,2,3,4}
0100	{1,2,3,4}	1100	{3}
0101	{2}	1101	{1,2,3,4}
0110	{4}	1110	{1,2,3,4}
0111	{1,2,3,4}	1111	{1}

The classical uniformly distributed query algorithm complexity lower bound is: $C_{UD}(M_2) \geq 3$.

The quantum query algorithm is demonstrated in Fig. 10 that computes the multifunction using one query: $Q_{UD}(M_2) = 1$.

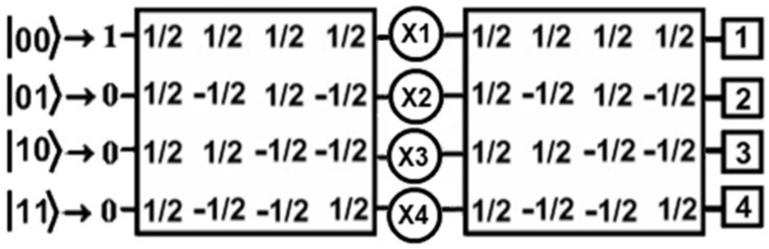

Fig. 10. Uniformly distributed quantum algorithm for multifunction M_2

Subsequently, multifunction is generalized to the case of N variables:

$$M_2^{GEN1} : \{0,1\}^{4N} \to \{1,2,3,4\}.$$

Imagine that $4N$ variables are arranged on four vertical lines (v-lines) in such a way that:

$$\forall i \in \{0,...N-1\}, \forall k \in \{1,2,3,4\} : x_{4i+k} \text{ belongs to } v\text{-line number } k.$$

For example, $x_1, x_5, x_9, x_{13},...$ are placed on the 1st v-line, $x_2, x_6, x_{10}, x_{14},...$ - on the 2nd, and so on (see Fig. 11 for illustration).

The result set for each input X of the multifunction is defined as follows:

1. $M_2^{GEN1}(X) = \{1\}$, if all four v-lines of X contain either odd or even number of "1".
2. $M_2^{GEN1}(X) = \{2\}$, if 1st and 3rd v-lines of X have odd number of "1" and 2nd and 4th have even number of "1", or vice versa: 1st an 3rd – even and 2nd and 4th - odd.
3. $M_2^{GEN1}(X) = \{3\}$, if 1st and 2nd v-lines of X have odd number of "1" and 3rd and 4th have even number of "1", or vice versa: 1st and 2nd – even and 3rd and 4th - odd.
4. $M_2^{GEN1}(X) = \{4\}$, if 1st and 4th v-lines of X have odd number of "1" and 2nd and 3rd have even number of "1", or vice versa: 1st and 4th - even and 2nd and 3rd - odd.
5. In all other cases, $M_2^{GEN1}(X) = \{1,2,3,4\}$.

The classical algorithm complexity lower bound is: $C_{UD}(M_2^{GEN1}) \geq 3N$.

The quantum query algorithm is demonstrated in Fig. 11 that computes multifunction in a uniformly distributed manner using N queries: $Q_{UD}(M_2^{GEN1}) \leq N$.

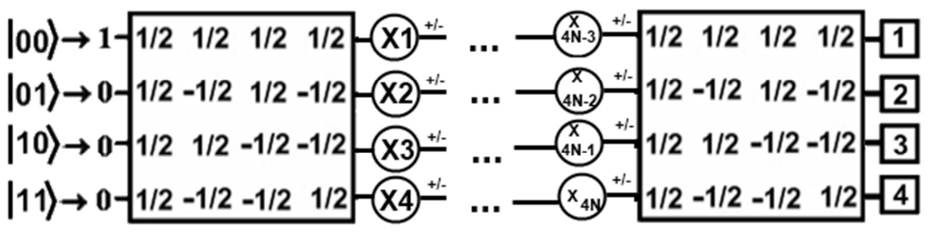

Fig. 11. Uniformly distributed quantum algorithm for multifunction M_2^{GEN1}

Finally, another one generalization on multifunction is created and as a result a different multifunction $M_2^{GEN2}:\{0,1\}^N \to \{1,2,...,N\}$ is obtained.

This time randomly distributed query algorithm complexity is examined.

The classical algorithm complexity lower bound is: $C_{RD}(M_2^{GEN2}) \geq \frac{N}{2}+1$.

The quantum query algorithm is demonstrated in Fig. 12 that computes multifunction in randomly distributed manner using one query: $Q_{RD}(M_2^{GEN2})=1$.

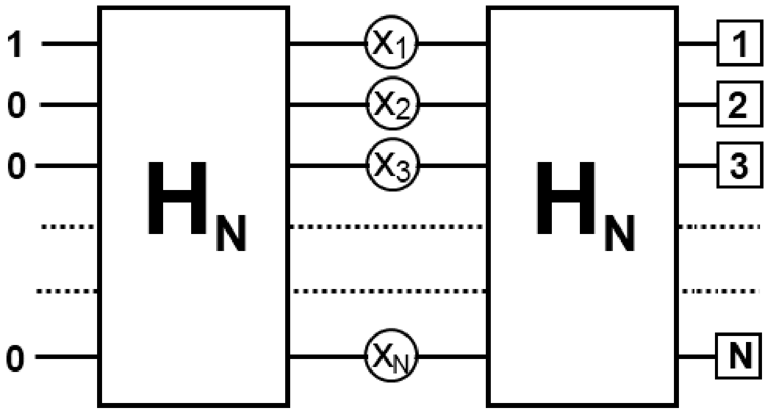

Fig. 12. Randomly distributed quantum algorithm for multifunction M_2^{GEN2}

4.4. Example 3 of Computing a Multifunction

The last example in our opinion is the most interesting. First, a multifunction $M_3:\{0,1\}^8 \to \{1,2,3,4\}$ is examined. Table 4 describes the rules, where exactly one rule is true for each input *X*. In the last column of the corresponding table row, the multifunction result set for examined input *X* is specified.

Classically six queries are required to compute the multifunction in a uniformly distributed manner: $C_{UD}(M_3)=6$.

The quantum query algorithm is developed (Fig. 13) that computes multifunction using two queries: $Q_{UD}(M_3) \leq 2$.

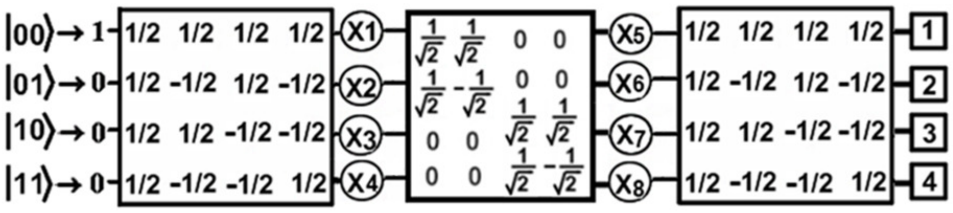

Fig. 13. Uniformly distributed quantum algorithm for multifunction M_3

Table 4. Definition of the multifunction M_3

Rules for input $X = (x_1, ..., x_8)$	Result set $M_3(X)$
$(x_1 = x_2 = x_3 = x_4) \& (x_5 = x_7)$	{1,2}
$(x_1 = x_2 = x_3 = x_4) \& (x_5 \neq x_7)$	{3,4}
$(x_1 = x_2 \& x_3 = x_4 \& x_1 \neq x_3) \& (x_5 = x_7)$	{3,4}
$(x_1 = x_2 \& x_3 = x_4 \& x_1 \neq x_3) \& (x_5 \neq x_7)$	{1,2}
$(x_1 \neq x_2 \& x_3 = x_4 \& x_1 = x_3) \& (x_6 = x_7)$	{1,4}
$(x_1 \neq x_2 \& x_3 = x_4 \& x_1 = x_3) \& (x_6 \neq x_7)$	{2,3}
$(x_1 \neq x_2 \& x_3 = x_4 \& x_1 \neq x_3) \& (x_6 = x_7)$	{2,3}
$(x_1 \neq x_2 \& x_3 = x_4 \& x_1 \neq x_3) \& (x_6 \neq x_7)$	{1,4}
$(x_1 = x_2 \& x_3 \neq x_4 \& x_1 = x_3) \& (x_5 = x_8)$	{1,4}
$(x_1 = x_2 \& x_3 \neq x_4 \& x_1 = x_3) \& (x_5 \neq x_8)$	{2,3}
$(x_1 = x_2 \& x_3 \neq x_4 \& x_1 \neq x_3) \& (x_5 = x_8)$	{2,3}
$(x_1 = x_2 \& x_3 \neq x_4 \& x_1 \neq x_3) \& (x_5 \neq x_8)$	{1,4}
$(x_1 \neq x_2 \& x_3 \neq x_4 \& x_1 = x_3) \& (x_6 = x_8)$	{1,2}
$(x_1 \neq x_2 \& x_3 \neq x_4 \& x_1 = x_3) \& (x_6 \neq x_8)$	{3,4}
$(x_1 \neq x_2 \& x_3 \neq x_4 \& x_1 \neq x_3) \& (x_6 = x_8)$	{3.4}
$(x_1 \neq x_2 \& x_3 \neq x_4 \& x_1 \neq x_3) \& (x_6 \neq x_8)$	{1,2}

Subsequently, two generalizations of quantum algorithm are developed.

The first generalization is based on quantum parallelism and computes 8N-variable multifunction. Quantum algorithm uses two queries to compute the multifunction (Fig. 14). Classically, six queries are required.

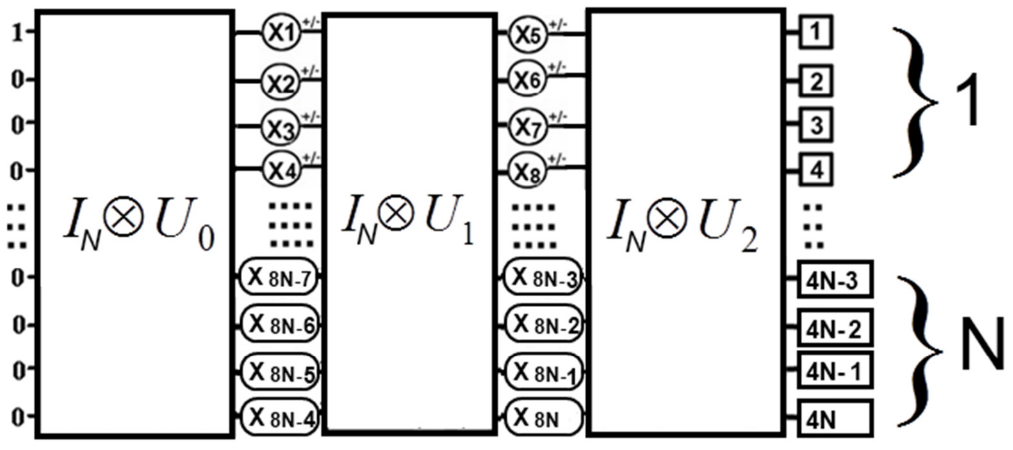

Fig. 14. Uniformly distributed quantum algorithm for multifunction M_3^{GEN1}

The idea of the second generalization is to substitute every x_i by $x_i^1 \oplus x_i^2 \oplus ... \oplus x_i^{N_i}$:

$$M_3^{GEN2}(x_1^1,...,x_1^{N_1},x_2^1,...,x_2^{N_2},......,x_8^1,...,x_8^{N_8}) = M_3(x_1^1 \oplus ... \oplus x_1^{N_1},......,x_8^1 \oplus ... \oplus x_8^{N_8}).$$

The classical algorithm complexity lower bound is: $C_{UD}(M_3^{GEN2}) = 6N$.

The quantum query algorithm is developed (Fig. 15) that computes multifunction using $2N$ queries: $Q_{UD}(M_3^{GEN2}) \leq 2N$.

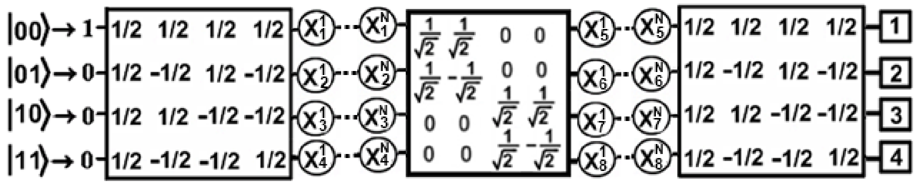

Fig. 15. Uniformly distributed quantum algorithm for multifunction M_3^{GEN2}

5. Nondeterministic Query Algorithms

A nondeterministic finite automaton, as introduced in (Rabin and Scott, 1959), is a machine with many choices in its movements. On every stage it may choose one of several further internal states. The nondeterministic machine accepts a tape if there is at least one winning combination of choices of states leading to a designated final state. This is a traditional point of view on nondeterminism. In the first part of the section, traditional nondeterministic quantum query model is examined.

In (Floyd, 1967), nondeterministic algorithms are considered conceptual devices for simplifying the design of backtracking algorithms. The above study supports a view that algorithms are nondeterministic not in the sense of being random, but in the sense of having a *free will*. In the second part of the section, the above mentioned free will is investigated and alternative definition of nondeterministic query algorithm is proposed.

In (Hopcroft and Ullman, 1969), the detailed definitions of nondeterministic finite automata, pushdown automata, Turing machine, and related results in complexity theory are provided.

5.1. Traditional Nondeterministic Quantum Query Model

This section is based on the paper (Dubrovska, 2007).

Nondeterministic quantum query algorithms (NQQA) were examined in (de Wolf, 2003). For instance, it was proved that it is possible to compute a function

$$f(X)=1 \Leftrightarrow |X| \neq 1$$

using one query for all N, though it is proved that the best classical nondeterministic algorithm requires all N questions.

Definition 4. (de Wolf, 2003) *A **nondeterministic** quantum query algorithm for f is defined to be a quantum algorithm that outputs 1 with positive probability if* $f(X)=1$ *and that always outputs 0 if* $f(X)=0$.

$NQ_1(f)$ denotes the query complexity of a nondeterministic quantum algorithm for *f*.

5.1.1. Dual Nondeterministic Quantum Query Algorithms

We introduce the concept of a *dual nondeterministic quantum query algorithm* and study the relations between complexity of exact, nondeterministic and dual nondeterministic quantum query algorithms.

Definition 5. *A* ***dual nondeterministic*** *quantum query algorithm for f is defined to be a quantum algorithm that outputs 0 with positive probability if* $f(X)=0$ *and that always outputs 1 if* $f(X)=1$.

$NQ_0(f)$ denotes the query complexity of a dual nondeterministic quantum algorithm for *f*.

5.1.2. Properties of NQQA

We present several theorems related to nondeterministic query complexity. The most important results are related to computing the following composite functions:

$$MULTI\_AND_m(x_1,...,x_{mn}) = \underbrace{f_n(x_1,...,x_n) \wedge f_n(x_{n+1},...,x_{2n}) \wedge ... \wedge f_n(x_{(m-1)n+1},...,x_{mn})}_{m}$$

$$MULTI\_OR_m(x_1,...,x_{mn}) = \underbrace{f_n(x_1,...,x_n) \vee f_n(x_{n+1},...,x_{2n}) \vee ... \vee f_n(x_{(m-1)n+1},...,x_{mn})}_{m}$$

Theorem 3. *Let Q1 be an exact quantum query algorithm that computes a Boolean function f with k queries. Consequently, a dual nondeterministic quantum query algorithm Q2 exists, computing a function $MULTI\_AND_m(f)$ with the same k queries for all m.*

Theorem 4. *For an arbitrary Boolean function f,*

$$NQ_0(MULTI\_AND_m(f)) \leq Q_E(f).$$

Theorem 5. *Let Q1 be an exact quantum query algorithm that computes Boolean function f with k queries. Consequently, a nondeterministic quantum query algorithm Q2 exists computing the function $MULTI\_OR_m(f)$ with the same k queries for all m.*

Theorem 6. *For an arbitrary Boolean function f,*

$$NQ_1(MULTI\_OR_m(f)) \leq Q_E(f).$$

Theorem 7. *Let f_i be an arbitrary Boolean function. Let us examine a function $F = f_1 \wedge f_2 \wedge ... \wedge f_n$. A dual nondeterministic quantum query algorithm Q exists computing F with $max(Q_E(f_1), Q_E(f_2), ..., Q_E(f_n))$ queries.*

Theorem 8. *Let f_i be an arbitrary Boolean function. Let us examine a function $F = f_1 \vee f_2 \vee ... \vee f_n$. A nondeterministic quantum query algorithm Q exists computing F with $max(Q_E(f_1), Q_E(f_2), ..., Q_E(f_n))$ queries.*

5.1.3. Application of NQQA Properties

In this section, we present several examples of dual nondeterministic quantum query algorithms that are better than the best possible classical counterparts.

In the first example, the following Boolean function is considered:

$$H_7(x_1,x_2,x_3,x_4,x_5,x_6,x_7)=(\neg(x_1\oplus x_2)\wedge(x_1\oplus x_3))\wedge((x_4\oplus x_5)\wedge(x_6\oplus x_7)).$$

Classical deterministic complexity is $D(H_7)=7$.

A dual nondeterministic quantum algorithm is demonstrated in Fig. 16 computing H_7 using two queries.

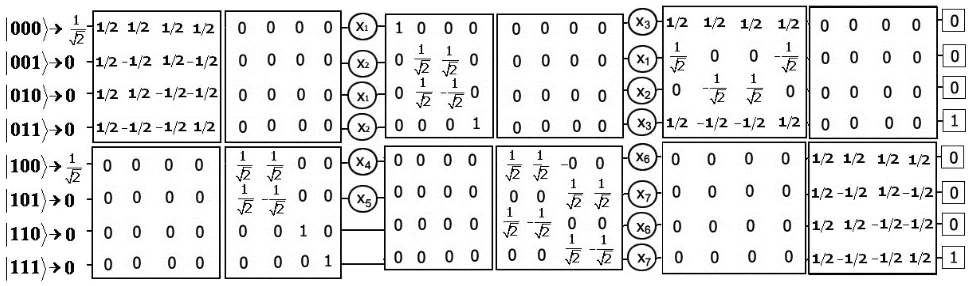

Fig. 16. Dual nondeterministic quantum query algorithm for H_7

In the second example, the following Boolean function is considered:

$$Control_N(x_1,x_2,...,x_k,x_{k+1},...,x_{2k-1})=1 \quad\Leftrightarrow\quad \begin{cases} x_{k+1}=x_1\oplus x_2 \\ x_{k+2}=x_1\oplus x_2\oplus x_3 \\ \dots\dots\dots\dots\dots\dots \\ x_{2k-2}=x_1\oplus x_2\oplus\dots\oplus x_{k-1} \\ x_{2k-1}=x_1\oplus x_2\oplus\dots\oplus x_{k-1}\oplus x_k \end{cases}$$

Classical deterministic complexity is $D(Control_N)=N$.

A dual nondeterministic quantum algorithm is demonstrated in Fig. 17 computing $Control_N$ using two queries.

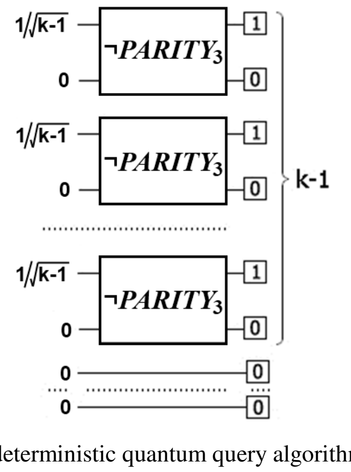

Fig. 17. Dual nondeterministic quantum query algorithm for $Control_N$

In the third example, the following Boolean function is considered:

$$PAIR\_EQUALITY_N(X) = \neg(x_1 \oplus x_2) \wedge \neg(x_3 \oplus x_4) \wedge \ldots \wedge \neg(x_{2k-1} \oplus x_{2k}),$$

where $N = 2k$.

Classical deterministic complexity is $D(PAIR\_EQUALITY_N) = N$.

A dual nondeterministic quantum algorithm is demonstrated in Fig. 18 computing *PAIR_EQUALITY*$_N$ using one query.

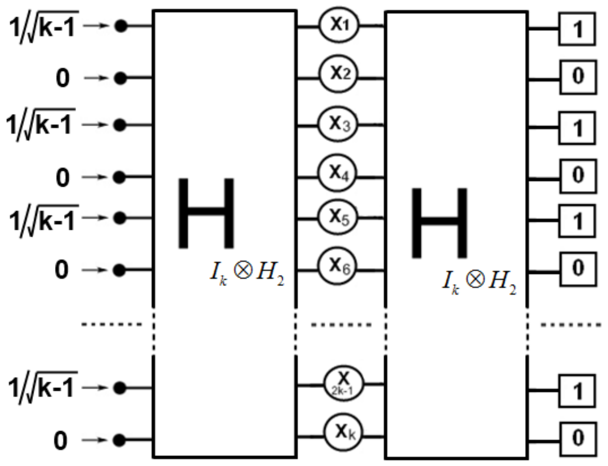

Fig. 18. Dual nondeterministic quantum query algorithm for *PAIR_EQUALITY*$_N$

5.1.4. Open Problems

The future direction of this research study is to prove stronger relations to other types of query algorithms, for example, to exact quantum algorithms of the same function, classical nondeterministic query algorithms and classical deterministic algorithms. It would also be useful to discover efficient quantum nondeterministic algorithms for specific functions, revealing large gaps between complexities of different kinds of algorithms.

5.2. Alternative Nondeterministic Query Model

This section is based on the paper (Vasilieva and Freivalds, 2011).

In (Floyd, 1967), a point of view is presented that algorithms are nondeterministic not in the sense of being random, but in the sense of having a *free will.*

We investigate the nature of the above-mentioned nondeterministic free will. We propose a way to measure the amount of nondeterminism in an algorithm. In the traditional nondeterministic query model the power of nondeterminism comes at no cost. The idea is that for the benefit of nondeterminism, the algorithm must "pay" with additional queries. We introduce an alternative definition of the nondeterministic query model, which incorporates the behavior described above. The definition of nondeterministic quantum query algorithms examined in the previous section seems a little counter intuitive. This is an additional motivation to introduce a new approach for nondeterminism in query algorithms.

5.2.1. Definition of the Alternative Nondeterministic Query Model

The idea of the model can be informally described as follows. Supposedly the task is to compute some arbitrary Boolean function $F(X)$ in an alternative nondeterministic query model. In such case, the first step is to define a nondeterministic helper function $G(X,Y)$. This function has to satisfy definite conditions. The second step is to design a deterministic query algorithm for the function $G(X,Y)$. Finally, the nondeterministic query complexity of the function $F(X)$ is equal to the complexity of the deterministic query algorithm for a nondeterministic helper function $G(X,Y)$.

Subsequently, we provide formal definitions for the computational model informally described above.

Definition 6. *The* ***nondeterministic helper function G(X,Y)*** *for the Boolean function F(X) is a partial Boolean function, which satisfies the following conditions:*

$\forall x1, ..., xn, \exists y1, ..., yk$, *so that* $G(x1, ..., xn, y1, ..., yk) = F(x1, ..., xn)$;

$\forall x1, ...,xn, \neg\exists y1, ..., yk$, *so that* $G(x1, ..., xn, y1, ..., yk) \neq F(x1, ..., xn)$.

Definition 7. *The* ***nondeterministic query complexity of the function F(X) with the fixed helper function G(X,Y)*** *is denoted by NDG(F) and is equal to the deterministic complexity of the G(X,Y): NDG(F) = D(G).*

An additional restriction on the deterministic query algorithm for the helper function $G(X,Y)$ is that after computing this function deterministically it should be possible to re-calculate or verify the value of $F(X)$ independently, using variable values extracted from the black box during the calculation of $G(X,Y)$.

Definition 8. *The* ***nondeterministic query complexity*** *of the function F(X) is denoted by ND(F) and is equal to the minimal nondeterministic query complexity of the function F(X) over all possible fixed helper functions G(X,Y):*

$$ND(F) = \min_{G(X,Y)} NDG(F).$$

5.2.2. Computing the Fano Plane Function

The Fano plane is a two-dimensional finite projective plane with the least number of points and lines (Weisstein et.al., 1974). Each vertex of the Fano plane is labeled by a variable number x_i (Fig. 19).

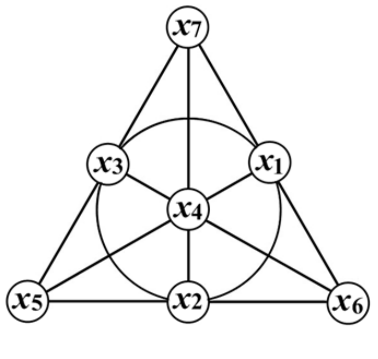

Fig. 19. Fano plane with vertices labeled by function *FANO*(*X*) variables

Definition 9. *A line of the Fano plane with Boolean values assigned to vertices is called* ***constant*** *if all vertices in a line have the same Boolean value assigned.*

There are two important properties of the Fano plane with Boolean values assigned to vertices:

- For any assignment of variable values there always is a constant line.
- For any assignment of variable values there cannot be two constant lines assigned with the different Boolean value at the same time.

These two properties allow defining of a Boolean function based on the Fano plane.

Definition 10. *Boolean function* ***FANO(x_1, ..., x_7)*** *is defined as follows. For an arbitrary input X=(x_1, ..., x_7) find a constant line in the Fano plane. Value of the FANO(X) function equals Boolean value assigned to vertices in that constant line.*

An example of *FANO*(*X*) function value assignment is illustrated in Fig. 20.

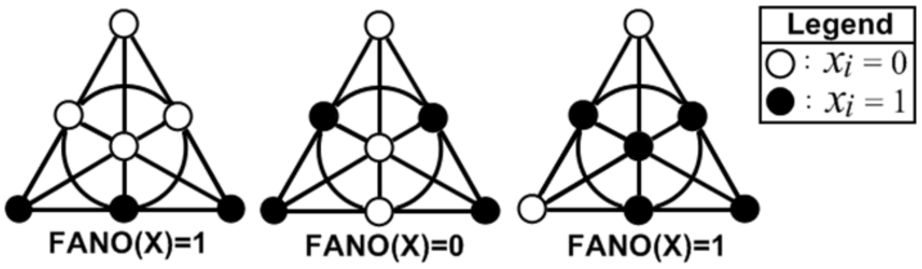

Fig. 20. Illustration of *FANO*(*X*) Boolean function value assignment

The classical deterministic query complexity of the Fano function is equal to number of variables: *D(FANO)*=7.

For computing the Fano plane function in an alternative nondeterministic query model a nondeterministic helper function $G_{FANO}(x_1,...,x_7,y_1,y_2,y_3)$ with three helper variables exists. To compute this helper function deterministically six queries are sufficient, thus, $ND_{GFANO}(FANO)$ = 6.

An important property of the Fano plane function is that definition can be applied recursively.

Definition 11. *Recursive Boolean function* $FANO^i$ *is defined as follows:*

$$FANO^1(X^1) = FANO(x_1, x_2, ..., x_7);$$

$$FANO^i(X^i) = FANO^1(FANO^{i-1}(X^{i-1}{}_1), FANO^{i-1}(X^{i-1}{}_2), ..., FANO^{i-1}(X^{i-1}{}_7)),$$

$$\text{where } X^i = X_1^{i-1} X_2^{i-1} ... X_7^{i-1}.$$

The classical deterministic query complexity of the recursive Fano function is equal to the number of variables: $D(FANO^N) = 7^N$.

A nondeterministic helper function exists ensuring the following nondeterministic complexity: $ND_G(FANO^N) = O(3^N)$.

5.2.3. Open Problems

The future work is to develop and improve the nondeterministic query model introduced in this paper. The scope of further investigation is very wide, from designing algorithms for certain problems in this model to performing a detailed complexity analysis and comparison to other computational models. Consideration of the Boolean function based on projective finite geometries, similar to the Fano plane function, seems to be a promising direction for searching of interesting examples. The most important further step is to define a quantum counterpart of the alternative nondeterministic query model and to investigate its properties.

6. Conclusion

In this article we presented an overview of results related to investigation of the problems of quantum query algorithm design and complexity. Different types of quantum query algorithms have been examined, such as exact, bounded-error and nondeterministic.

In the first part, quantum query algorithms for computing Boolean functions have been presented. Regarding exact quantum algorithms (for which the probability of a correct result is invariably $p = 1$), there was a long-standing open question and challenge: to establish whether a larger separation than N versus $2N$ can be achieved between the classical deterministic and the quantum exact query complexity for a total function?[1] Even the design of examples with an N versus $2N$ complexity gap is known to be a complex task. The majority of such examples are directly based on the involvement of an *XOR* operation in the definition of a computable function. In the course of the present research study, efforts were made to improve and develop universal techniques for designing exact quantum query algorithms. To simplify the process of generation and verification of quantum algorithms, a *Wolfram Mathematica* software program was developed. Using this tool, two examples of Boolean functions were generated with a small number of variables, for which the complexity of the quantum exact query algorithm is lower than the complexity of the classical deterministic query algorithm. Subsequently, quantum algorithm transformation methods were proposed, providing for a significant enlargement of the set of efficiently computable Boolean functions. Transformation methods were successfully applied to quantum algorithms for two basic Boolean functions. The approach demonstrated may be applied to already familiar and new exact quantum query algorithms. Additionally, an exact quantum query algorithm for the problem of repetition code verification was presented. In this example, a gap of N versus $2N$ between quantum and classical complexity is achieved.

Regarding bounded-error quantum query algorithms, an algorithm for computing conjunctions, i.e., the Boolean function $AND(x_1,\ldots,x_N)$, was presented. This is a basic and widely applicable function, which requires an efficient algorithm. The quantum algorithm presented computes the conjunction of two bits through a single query with the correct answer probability $p = 4/5$. The approach was extended by formulating a general method for computing the conjunction of two Boolean functions with the same probability and the number of queries equal to $\max(Q_E(f_1), Q_E(f_2))$.

[1] Recently an example of a superlinear gap has been presented in (Ambainis, 2012).

The second part is devoted to computing multivalued functions in a query model. We proposed three types of query algorithms for computing multifunctions: definite, randomly distributed and uniformly distributed query algorithms. Since quantum algorithms actively employ the power of parallelism and computing in superposition, they are naturally well suited for computing multifunctions in a distributed manner. We presented three examples of computing multifunctions in classical and quantum versions of the query model.

In the third part, we examined nondeterministic query algorithms. Regarding the traditional quantum query model, we introduced a new notion of a dual nondeterministic quantum query algorithm. This type of algorithms was studied, and the discovered properties were applied to design efficient algorithms. Investigating the traditional nondeterministic query model we realized that it is counter-intuitive in some sense and does not employ the full power of nondeterminism. As an alternative, a different definition of a nondeterministic query model was introduced. The model was demonstrated on the basis of an example of computing the Fano plane Boolean function obtaining a 7^N versus $O(3^N)$ gap between the deterministic and the nondeterministic query complexity.

The goals of the research are achieved on the whole; however, further improvements are possible by continuing the study in the following directions:

- *Exact quantum algorithms*. It is still important to develop new quantum algorithms that are more than two times faster than the best possible classical deterministic algorithm. The idea for further work is to apply the combinatorial approach for algorithm design and to enlarge the size of the quantum system.
- *Computing multifunctions in the query model*. In our opinion, this study direction is the most promising. To be able to build even more efficient algorithms, it is necessary to perform an exhaustive analysis of the properties of functions and algorithms discovered during the research study. A very important action line is development of convenient and efficient methods for proving the classical complexity lower bounds for multifunctions.
- *Nondeterministic query algorithm*s. The most important further step is to define a quantum counterpart of the alternative nondeterministic query model and to investigate its properties.

Acknowledgements

I wish to express sincere thanks to my supervisor professor Rūsiņš Freivalds. Special thanks to professor Andris Ambainis and associate professor Juris Smotrovs for helpful comments.

This work has been supported by the European Social Fund within the project «Support for Doctoral Studies at University of Latvia».

References

Agadžanjans, R. (2010). *Complexity of Quantum Query Algorithms.* Doctoral Thesis, University of Latvia.

Ambainis, A. (2004). Quantum query algorithms and lower bounds (survey article). *Proc. of FOTFS III, Trends on Logic*, *23*, 15-32.

Ambainis, A. (2006). Polynomial degree vs. quantum query complexity. *Journal of Computer and System Sciences 72*, 220–238.

Ambainis, A. (2012). Superlinear advantage for exact quantum algorithms. CoRR, abs/1211.0721.

Ambainis, A., de Wolf, R. (2001). Average-case quantum query complexity. *Journal of Physics A 34*, 6741–6754.

Ambainis, A., Childs, A., Reichardt, B., et al. (2010). Any AND-OR Formula of Size N Can Be Evaluated in Time $N^{1/2+o(1)}$ on a Quantum Computer. *SIAM J. Comput., 39*, 2513-2530.

Buhrman, H., de Wolf, R. (2002). Complexity Measures and Decision Tree Complexity: A Survey. *Theoretical Computer Science v. 288(1)*, 21–43.

Childs, A., Cleve, R., Deotto, E., et al. (2003). Exponential algorithmic speedup by quantum walk. *35th ACM Symposium on Theory of Computing*, 59–68.

Cleve, R., Ekert, A., Macchiavello, C., Mosca, M. (1998). Quantum algorithms revisited. *Proc. of the Royal Society of London*, *A 454*, 339–354.

Cover, T. M., Thomas, J. A. (1991). *Elements of Information Theory.* Wiley-Interscience.

de Wolf, R. (2001). *Quantum Computing and Communication Complexity.* University of Amsterdam.

de Wolf, R. (2003). Nondeterministic Quantum Query and Quantum Communication Complexities. *SIAM Journal on Computing* (32(3)), 681-699.

Deutsch, D. (1985). Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. of the Royal Society of London A 400.*

Deutsch, D., Jozsa, R. (1992). Rapid solutions of problems by quantum computation. *Proceedings of the Royal Society of London*, *A 439*, 553-558.

DiCarlo, L., Chow, J. M., Gambetta, J. M., et al. (2009). Demonstration of two-qubit algorithms with a superconducting quantum processor. *Nature 460,*(460), 240-244.

Dixon, A., Dynes, J., Shields, A. (2008). Gigahertz decoy quantum key distribution with 1 Mbit/s secure key rate. *Optics Express, 16*(23).

Dubrovska, A. (2007). *Application of quantum computing for efficient algorithm construction*. Master's Thesis, University of Latvia.

Dubrovska, A. (2007). Properties and Application of Nondeterministic Quantum Query Algorithms. *Proc. od SOFSEM 2007: Theory and Practice of Computer Science, Volume II;* MatFyz Press; ISBN 80-903298-9-6, 37-49.

Dubrovska, A. (2007). Quantum Query Algorithms for Certain Functions and General Algorithm Construction Techniques. *Proc. of SPIE Vol. 6573, Quantum Information and Computation V,* (SPIE, Bellingham, WA) Article 65730F.

Dubrovska, A., Mischenko-Slatenkova, T. (2006). Computing Boolean Functions: Exact Quantum Query Algorithms and Low Degree Polynomials. *SOFSEM 2006, Student Research Forum;* MatFyz Press; ISBN 80-903298-4-5, 91-100.

D-Wave Systems sells its first Quantum Computing System to Lockheed Martin Corporation. http://www.dwavesys.com/en/pressreleases.html#lm_2011

Feynman, R. (1982). Simulating Physics with Computers. *International Journal of Theoretical Physics, 21* (No. 6/7).

Floyd, R. W. (1967). Nondeterministic Algorithms. *Journal of the ACM (JACM), 14*(4), 636-644.

Freivalds, R., Iwama, K. (2009). Quantum Queries on Permutations with a Promise. *Implementation and Application of Automata, Lecture Notes in Computer Science, 5642/2009*, 208-216.

Grover, L. (1996). A fast quantum mechanical algorithm for database search. *Proc. of 28th STOC'96*, 212–219.

Gruska, J. (1999). *Quantum Computing*. McGraw-Hill.

Hiskett, P., Rosenberg, D., Peterson, C. (2006). Long-distance quantum key distribution in optical fibre. *New J. Phys., 8*(193).

Hopcroft, J. E., Ullman, J. (1969). *Formal Languages and their Relation to Automata.* Addison-Wesley, Reading, MA.

ID Quantique SA - Network Encryption, Random Number Generators, Photon Counting. http://www.idquantique.com/

Jin, X.-M., Ren, J.-G., Yang, B. (2010). Experimental free-space quantum teleportation. *Nature Photonics* (4), 376 - 381.

Johnson, M., Amin, M., Gildert, S., et al. (2011). Quantum annealing with manufactured spins. *Nature, 473*, 194–198.

Kaye, P., Laflamme, R., Mosca, M. (2007). *An Introduction to Quantum Computing.* Oxford.

Kerenidis, R., de Wolf, R. (2004). Exponential Lower Bound for 2-Query Locally Decodable Codes via a Quantum Argument. *Journal of Computer and System Sciences*, 395-420.

Kravcevs, V. (2008). *Quantum algorithm complexity.* Doctoral Thesis, University of Latvia.

Kushilevitz, E., Nisan, N. (1997). *Communication complexity.* Cambridge University Press.

Lace, L. (2004). Enlarging gap between quantum and deterministic query complexities. *Proc. of Baltic DB&IS*, *2*, 81-91.

Lace, L. (2008). *Quantum Query Algorithms.* Doctoral Thesis, University of Latvia.

Lee, N., Benichi, H., Takeno, Y., et al. (2011). Teleportation of Nonclassical Wave Packets of Light. *Science, 332*(6027), 330-333.

Lehmer, D. H., Lehmer, E. (1974). A New Factorization Technique Using Quadratic Forms. *Mathematics of Computation* (28, 126), 625-635.

Nielsen, M., Chuang, I. (2000). *Quantum Computation and Quantum Information.* Cambridge University Press.

Politi, A., Matthews, J., O'Brien, J. (2009). Shor’s Quantum Factoring Algorithm on a Photonic Chip. *Science, 325*(5945), 1221.

Rabin, M. O., Scott, D. (1959). Finite automata and their decision. *IBM. Journal of Research and Development* (3:2), 114-125.

Shor, P. W. (1997). Polynomial time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing* (26(5)), 1484-1509.

Simon, I. (1994). String matching algorithms and automata. *Lecture Notes in Computer Science, 814*, 386–395.

Ščeguļnaja-Dubrovska, O. (2010). *Models of Quantum Computation.* Doctoral Thesis, University of Latvia.

Vasilieva, A. (2009). Exact Quantum Query Algorithm for Error Detection Code Verification. *Proc. of the Fifth Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS),* ISBN 978-80-87342-04-6, 200-207.

Vasilieva, A. (2010). Quantum Query Algorithms for Relations. *Proc. of the MFCS & CSL 2010 Satellite Workshop Randomized and quantum computation*, ISBN 978-80-87342-08-4, 78-89.

Vasilieva, A. (2011). Uniformly Distributed Quantum Query Algorithms for Multifunctions. *Proc. of the Annual Doctoral Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS)*, 86-93.

Vasilieva, A., Freivalds, R. (2011). Nondeterministic Query Algorithms. *Journal of Universal Computer Science (J. UCS)*, 859-873.

Vasilieva, A., Mischenko-Slatenkova, T. (2010). Quantum Query Algorithms for Conjunctions. *Proc. of the 9th International Conference UC 2010*, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, vol. 6079/2010, ISBN: 978-3-642-13522-4, 140-151.

Weisstein, E. W. Fano Plane. From MathWorld - A Wolfram Web Resource, http://mathworld.wolfram.com/FanoPlane.html.

Wolfram Mathematica computational software. http://www.wolfram.com/mathematica/.

Received March 12, 2013, revised July 26, 2013, accepted August 13, 2013