УДК 004.415.2, 004.588

MODELS OF QUANTUM ALGORITHMS (DECISION MAKING ALGORITHMS): BASIC PROGRAMMING TECHNIQUES

Barchatova Irina[1], Ulyanov Sergey[2], Yamafuji Kazuo[3]

[1]PhD Student;
Dubna International University of Nature, Society and Man,
Institute of system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: i.a.barhatova@gmail.com.

[2]Doctor of Science in Physics and Mathematics, professor;
Dubna International University of Nature, Society and Man,
Institute of system analysis and management;
141980, Dubna, Moscow reg., Universitetskaya str., 19;
e-mail: ulyanovsv@mail.ru.

[3]PhD, professor;
Dept. of Mechanical and Control Eng., University of Electro-Communications;
1-5-1 Chofu, Chofugaoka ,182 Tokyo, Japan;
e-mail: yamafuji@yama.mce.uec.ac.jp.

Classification of quantum algorithms models is introduced. Simple models of quantum algorithms as decision making algorithms (Deutsch and Deutsch-Jozsa) are described. Basic programming techniques are discussed.

Keywords: quantum algorithms, classifications of quantum algorithms models, basic programming techniques.

МОДЕЛИ КВАНТОВЫХ АЛГОРИТМОВ (АЛГОРИТМЫ ПРИНЯТИЯ РЕШЕНИЯ): БАЗОВЫЕ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Бархатова Ирина Александровна[1], Ульянов Сергей Викторович[2], Ямафуджи Кацуо[3]

[1]Аспирант;
ГБОУ ВО «Международный Университет природы, общества и человека «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: i.a.barhatova@gmail.com.

[2]Доктор физико-математических наук, профессор;
ГБОУ ВО «Международный Университет природы, общества и человека «Дубна»,
Институт системного анализа и управления;
141980, Московская обл., г. Дубна, ул. Университетская, 19;
e-mail: ulyanovsv@mail.ru.

[3]Доктор наук, профессор;
Факультет механики и технической кибернетики (интеллектуальные системы),
Университет передачи информации;
1-5-1, Япония, Токио, Chofu, Chofugaoka ,182;
e-mail: yamafuji@yama.mce.uec.ac.jp.

В статье приведена классификация моделей квантовых алгоритмов. Описаны простые модели квантовых алгоритмов типа принятия решения (такие как алгоритмы Дойча и Джоза-Дойча), а также базовые технологии программирования для реализации данных алгоритмов на классическом компьютере.

Ключевые слова: квантовые алгоритмы, классификации квантовых алгоритмов, основные технологии программирования.

Classification and general structure of QA gates

One of the most important issues in quantum computing is the design of QA's. There are known very few of them. Apparently, we are lacking the basic principles underlying the quantum version of algorithm problem solving. One attempt to understand the basic principles of QA design may proceed with the comparison with the known strategies of designing classical algorithms in computational science.

Basic principles of QA design

Many of the most popular models of quantum computation are direct quantum generalizations of well known classical constructs. This includes quantum Turing machine, gate arrays and walks. These models use unitary evolution as the basic mechanism of information processing and only at the end do we make measurements, converting quantum information into classical information in order to read out classical answer. In the more familiar gate array model computational steps are unitary operations, developing a large entangled state prior to some final measurements for the output. Just two ideas from quantum computing (and some algorithmic ingenuity) are considered. The first of two ideas is amplitude amplification. The second idea is that any classical (either deterministic or probabilistic) computation can be simulated on a quantum computer. More precisely, (i) in the circuit a classical model, a classical circuit with N gates can be simulated by a quantum circuit with $O(N)$ gates; (ii) in the query model (when only the number of queries is counted), a classical computation with queries can be simulated by a quantum computation with N queries.

Thus, this greatly simplifies description of quantum algorithms [1-10]. Instead of describing a quantum algorithm, we can describe a classical algorithm that succeeds with some small probability ε. Then, we can transform the classical algorithm to a quantum algorithm and apply the amplitude amplification to the quantum algorithm. The result is a quantum algorithm with the running time or the number of queries that is times the one for the classical algorithm with which we started. A similar reasoning can be applied, if instead of a purely classical algorithm, we started with a classical algorithm that involves quantum subroutines. Such algorithms can also be transformed into quantum algorithms with the same complexity [11-35].

Another approach in quantum computing consists in the formalism of the measurement based quantum computation. In this case we start with a given fixed entangled state of many q-bits and perform computation by applying a sequence of measurements to designated q-bits in designated bases. The choice of basis for later measurement may depend on earlier measurement outcomes and the final result of the computation is determined from the classical data of all the measurement outcomes. In contrast to unitary evolution, measurements are irreversibly destructive, involving much loss of potential information about a quantum state's identity.

Thus it is interesting, and at first sight surprising, that we can perform universal quantum computation using only measurements as computation steps. Two principle schemes of measurement based computation are teleportation quantum computation and so-called cluster model of one-way quantum computer. From another standpoint, the appeal of hidden-variable theories is that they provide one possible solution to the measurement problem. For example, even if an observer were placed in coherent superposition, that observer would still have a sequence of definite experiences, and the probability of any such sequence could be calculated.

For this case, hidden-variable theory is simply a way to convert a unitary matrix that maps one quantum state to another into a stochastic matrix that maps the initial probability distribution to the final one in some fixed basis. A hidden-variable theory can be based on networks flows: if we examine the entire history of a hidden variable, then we could efficiently solve problems that are believed to be intractable even for quantum computers. By sampling histories, one could, for example, search an unordered database of N items for a single «marked item» using only $O\left(N^{\frac{1}{3}}\right)$ database queries.

By comparison, Grover's quantum search algorithm (QSA) requires $\theta\left(N^{\frac{1}{2}}\right)$ queries, while classical algorithms require $\theta(N)$ queries.

The results are surprising i.e., giving a hidden variable, the distribution over its possible values at any single time is governed by standard quantum mechanics and is therefore efficiently samplable on a quantum computer. So if examining the variable's history confers any extra computation power, then it can only be because of correlations between the variable's values at different times.

Remark. Quantum computation explores the possibilities of applying quantum mechanics to computer science. If built, quantum computers would provide speed-ups over conventional computers for a variety of problems. The two most famous results in this area are Shor's quantum algorithms for factoring and finding discrete logarithms and Grover's quantum search algorithm show that quantum computers can solve certain computation problems significantly faster than any classical computers. Shor's and Grover's algorithms have been followed by a lot of other results. Each of these algorithms has been generalized and applied to several other problems. New algorithms and new algorithmic paradigms (such as adiabatic computing which is the quantum counterpart of simulated annealing) have been discovered. Several aspects adiabatic quantum-computational model can be explored and use a way that directly maps any arbitrary circuit in the standard quantum-computing model to an adiabatic algorithm of the same depth. Many quantum algorithms are developed for the so-called oracle model in which the input is given as an oracle so that the only knowledge we can gain about the input is asking queries to the oracle. We use the query complexity as our measure of complexity.

The query complexity of an algorithm A computing a function F is the number of queries used by A. The query complexity of F is the minimum query complexity of any algorithm computing F. We are interested in proving lower bounds of the query complexity of specific functions and consider methods of computing such lower bounds. The two most successful methods for proving lower bounds on quantum computations are the following: the adversary method and the polynomial method. An alternative measure of complexity would be to use the time (temporal) complexity which counts the number of basic operations used by an algorithm.

The temporal complexity is always at least as large as the query complexity since each query takes one unit step, and thus a lower bound on the query complexity is also a lower bound on the temporal complexity. For most existing QAs the temporal complexity is within polylogarithmic factors of the query complexity.

The barrier to better understanding of the quantum query model is the lack of simple mathematical representations of quantum computations. While classical query complexity (both deterministic and randomized) has a natural intuitive description in terms of decision trees there is no such easy description of quantum query complexity. The main difference between the classical and quantum case is that classical computations branch into non-interacting subcomputations (as represented by the tree) while in quantum computations there is no obvious analog of branching because of the possibility of destructive interference between sub-computations. The bounded-error model is both relevant to understanding powerful explicit non-query quantum algorithms (such as Shor's factoring algorithm) and is theoretically important as the quantum analogue of the classical decision tree model.

We are interested in studying classical and quantum complexities because an oracle sometimes gives a separation between them. For example, there was shown one problem where we need exponentially many queries in the bounded error classical case but only a single query is needed in the quantum case. Another occasion to study a query complexity is when a temporal complexity is hard. In such case the number of queries we make gives a lower bound for the temporal complexity. In fact, currently there is no lower bound method for quantum temporal complexity that gives super-linear bounding and we get lower bounds heuristic on quantum temporal complexity by studying quantum query complexity.

One of the powers of quantum computation comes from the fact that we can query in superposition. It means that if we are given a set of n elements from 1 to n we can query an oracle in parallel once to obtain a superposition of $f(1)$ through $f(n)$. However, we can in a sense only learn one of the $f(i)$'s from such a query. The real power of quantum computation comes from interference. I.e., the information in the state

e.g., $f(i)$'s can be combined by means of unitary quantum gates in non-trivial way and we can extract a global property of the input.

Implementation of a QA is based on a quantum algorithmic gates (QAG). In the language of classical computing a quantum computer is programmed by QAG designing. In this section we report relatively few such gates because the basic principles underlying the quantum version of programming are in their infancy and QAs have been programmed by ad-hoc techniques. This is suggested by the relationships between fundamentals of classical and quantum computations. In connection with it we need to distinguish between fundamentals for designing algorithms. The fact that we can understand is that the fundamentals of quantum computation do not mean that we know keys to set-up QA although it can be of great help.

Classification and general structure of QA's

One way to understand the basic principles of QA design is by comparison with the known strategies of designing classical algorithms in computational science.

1. On the classification of QA's: design strategies of classical algorithms and quantum computation

Let us come to the point of analyzing the classical strategies of algorithm design from the point of view of quantum computation. We shall consider the classification which is introduced by Levitin (1999) who has done a reformulation which includes and categorizes in a nice fashion other classifications schemes in a nice fashion: there are four classical general design techniques which we shall describe briefly by its definition and with a simple example to illustrate them.

This is suggested by the relationships between fundamentals of classical and quantum computation. In connection with it, it is useful to distinguish between fundamentals for designing algorithms. It is instructive to consider the classical strategies of algorithm design from the quantum computation point of view.

There are four classical general design techniques shown in Table 1.

Table 1.Generic types of algorithms

N	**Algorithm Type**	**Algorithm Meaning**
1	*Brute Force Algorithm*	It amounts to solving a problem by directly applying its crude formulation. *Example.* $a^n = a \cdot a \cdots a$, n times.
2	*Divide-and-Conquer Algorithms*	The original problem is partitioned into a number of smaller problems, usually of the same kind. These in turn are solved and their solutions are combined to get a solution of the bigger problem. This strategy is usually applied recursively in order to obtain a greater profit. *Example.* $a^n = a^{\lfloor n/2 \rfloor} \cdot a^{\lfloor n/2 \rfloor} \cdot a^{n-2\lfloor n/2 \rfloor}$.
3	*Decrease-and-Conquer Algorithms*	The original problem is reduced to a smaller one, which is usually solved by recursion and the solution obtained is applied to find a solution of the original problem. *Example.* 1) $a^n = a^{n-1} \cdot a$ (decrease-by-one variety); 2) $a^n = \left(a^{\lfloor n/2 \rfloor}\right)^2$ if n even, $a^n = \left(a^{\lfloor n/2 \rfloor}\right)^2 \cdot a$ if n odd (decrease-by-half variety)
4	*Transform-and-Conquer Algorithms*	The original problem is transformed into another equivalent problem, which is more amenable to solution with simpler techniques. *Example.* a^n is computed by exploiting the binary representation of n.

These techniques are described below using, as an example, the problem of computing, $a^n \bmod p$ which is of great importance in public-key encryption algorithms.

Table 2 lists these classical strategies with some-known and less trivial examples of representative algorithms.

There are important algorithms built upon a mixture of these basic techniques; for example, the Fast Fourier Transform (*FFT*) employs both Divide-and-Conquer and Transform-and-Conquer techniques.

Table 3 presents the quantum version of Table 2 by classifying the most useful of the known QA's.

Table 2. Classification of classical algorithms

Classical Technique	**Algorithm Example**
Brute Force	Searching the Largest
Divide-and-Conquer	Quick sort
Decrease-and-Conquer	Euclid's Algorithm
Transform-and-Conquer	Gaussian Elimination

Table 3. Classification of QAs

Quantum Technique	**Algorithm Example**
Brute Force	Grover's Algorithm Deutsch-Jozsa' Algorithm Simon's Algorithm
Divide-and-Conquer	∅
Decrease-and-Conquer	∅
Transform-and-Conquer	Shor's Algorithm

Remark. Firstly, Grover's QSA algorithms have placed (the Object is missing) in the category of *Brute Force* algorithms. The strategy is similar to its classical counterpart which is of *Brute Force* type. The difference lies in the fact that the quantum operation is realized through a unitary operator which implements the reversible quantum computation. Although the *Brute Force* technique gives usually low efficient algorithms it is very important for several reasons. One is that there are important cases like searching problem where the *Brute Force* method outperforms more sophisticated strategies like *Divide-and-Conquer*. We find Grover's algorithm as a realization of the *Brute Force* technique at the quantum level and for general purpose at the same time.

Remark. Secondly, Shor's algorithm have included in the category of *Transform-and-Conquer* algorithms. Shor solves the factorization problem by reducing it to the problem of finding the period of a certain function in number theory, which in turn is solved with the aid of the fundamentals of quantum computing. Having realized this, we point out that the classical version of *Transform-and-Conquer* algorithms is very rare. This may explain why Shor's algorithm, although power than Grover's, it has a more reduced range of applications.

Remark. Thirdly, the most notorious aspect of Table 2 is the absence of QA's based on the *Divide-and-Conquer* technique, which is by far the most general and used strategy in classical computation. This may partly account for the list of QA's being so short. Moreover, if we resort to the basic features of quantum computation we may explain somehow why this entry is empty in Table 1.4. We know that a quantum register supports the superposition of many states at the same time. This implies that the q-bits of the quantum registers are strongly correlated (*entangled*) and their joint state is not separable into a product of states of

smaller sub-registers. Thus quantum parallelism and entanglement renders unnaturally any try to implement the strategy of *Divide-and-Conquer* in a quantum register at least in a straightforward and naive fashion. A quantum computer is a quantum system time evolution of which can be thought of as a computation. Quantum-mechanical systems can be used (in principle) to implement a wide variety of computational algorithms with enhanced efficiency. The principle of superposition in quantum mechanics, according to which a system can be in a linearly superposed state of more than one eigenstate, is the key to this increased efficiency.

It is useful to consider also a physical role in quantum computation process of two non-classical (quantum) phenomena: *quantum interference* and *quantum entanglement*. Entanglement allows to encode data into non-trivial multi-particle superpositions of some pre-selected basis states. Quantum interference, which is a dynamical process, allows to evolve initial states (inputs) into some prescribed way. Unlike single-particle interference multi-particle quantum interference does not have any classical analogue and can be viewed as an inherently quantum process. Quantum computing uses the quantum interference of different computational paths to enhance correct outcomes and suppress erroneous outcomes of computations. A common pattern underpinning QA's can be identified when quantum computation is viewed as multi-particle interference.

From the quantum mechanics standpoint, QAs typically fall into two categories: (i) s*uperposition-driven* algorithms; and (ii) *entanglement-driven* algorithms.

In the superposition-driven algorithm the QA tends to amplify the amplitudes of a set of «good» states to a certain extent, e.g., states that are specified by a predicate given in the form of a quantum circuit shrink the amplitude of the «bad» states. The Grover QSA for searching an unordered list is an example of such an algorithm.

The entanglement-driven algorithm can be described as follows. Assume a function $f : X \to Y$ from a (finite) domain X to a co-domain Y. This function does not have to be injective; however, for a quantum computing to be able to perform f with respect to a suitable encoded X and Y, the function f has to be embedded into a unitary matrix U_f. Compute simultaneously the image of all inputs $x \in X$ using $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$ for all $x \in X$ by preparing an equal superposition $\sum_{x \in X} |x\rangle$ in the X register and the ground state $|0\rangle$ in the Y register first, and then applying the quantum circuit U_f to obtain $\sum_{x \in X} |x\rangle|f(x)\rangle$. This entangled state can then be written as: $\sum_{y \in \mathrm{Im}(f)} \left(\sum_{x: f(x)=y} |x\rangle \right) |y\rangle$, corresponding to a separation of the pre-images of f. Measurement of the second register yields one of these pre-images. The algorithm of Shor, described below (see, in details Chapter 4) is an example of such an algorithm. In this case, the function f is given by $f(x) = a^x \bmod N$, where N is the number to be factored and a is a random element in Z_N.

Global properties of functions that can be determined by such a QA's are said to be Computable by Quantum Parallelism (QPC). At least in the case of two-valued functions, the QPC properties that can be determined by means of a single computation are an exponentially small fraction of all the possible global properties.

2. Quantum algorithms and their classification

QAs consist of a fixed number of computational steps, represented by quantum gates which are comparable to single, elementary instructions in classical computer programming. More complex programming instructions and constructs known from classical programming languages like while-loops which lead to a variable number of steps, are not implementable in the (non-uniform) quantum circuit model. Hybrid algorithms remedy they are made of a classical program, with quantum subprograms, built by elementary quantum operations. Most of the quantum algorithms (relevant in practice) are hybrid. However, this chapter focuses mainly on quantum subprograms.

It is the aim of this chapter to provide a summary of QAs which give an advantage over known classical algorithms and which are based on the quantum circuit model of computation. Although there are not many different QAs or quantum subprograms respectively, there is no claim to completeness because no all improvements and implementations by means of using other gate sets are mentioned. Furthermore, one must

distinguish between the QAs and the problems they are applied to since some algorithms (subprograms) solve more than just one problem or even class of problems.

Finally, it cannot be ruled out that there are some more insignificant QAs which escaped through investigation as it stands at the mid-year of 2005. It should be noted that all these QAs were exclusively developed by hand.

Since there exist only a few QAs, classifying them may tend to be arbitrary. However, there are characteristics which can lead to partitions of QAs. One possible classification depends on the technique that the algorithm uses: The first class consists of algorithms which are based on determining a common property of all the output values. A representative of this class is Shor's algorithm, where the period of the function $f(x) = a^x \bmod N$ is calculated.

Remark. The period of a function $f : 0,1^n \to 0,1^m$ is the value $f(x) = a^x \bmod N$, with $f(x) = f(x + rm)$, where m is an integer, such that $x, x + rm \in 0,1,...,2^n - 1$.

The second class contains those algorithms which use amplitude amplification, like Grover's algorithm. The third class contains algorithms which derive benefit from both methods characterizing the previous two groups, such as the approximate counting algorithm.

In a slightly different way, Nielsen and Chuang divide quantum algorithms into three classes. The first class consists of those algorithms based on quantum versions of the Fourier transform. Examples are the Deutsch-Jozsa algorithm as well as Shor's algorithms for factoring and the discrete logarithm. Most of these problems are instances of the hidden subgroup problem in group theory. They second class of quantum algorithms are the quantum search algorithms. At last, the third class of algorithms includes *quantum simulations*, i.e. algorithms, which simulate a quantum physical system on a quantum computer.

A classifications based on the programming technique seems tempting. However, the preparation of an equally weighted superposition which can be also regarded as an application of the quantum Fourier transform (QFT), seems to be an imperative step in any QA and cannot serve as a distinguishing mark. Whether this can be interpreted as a Fourier transform depends on the problem. The best example is Grover's algorithm, where there are no reasons to read the superpositioning as a QFT. Deutsch's problem, however, can be also regarded as a group theoretic hidden subgroup problem. Therefore, the interpretation of the superposition step as a QFT might be appropriate.

Amplitude amplification means to increase the amplitudes of solution states while decreasing the amplitudes of non-solution states by using a generalization of Grover's iteration. This characterizes many quantum search algorithms. Hogg's algorithm for structured combinatorial search problems, such as highly constrained k-SAT, is an exception, since it is also a quantum search algorithm, but it does not use Grover's technique. Instead, its power seems to come from a claver exploitation of the hidden problem structure. The algorithm is described in more detail below. Another algorithm, which does not seem to be integrable into these classifications mentioned above is the route finding algorithm. It finds routes between specified start and end nodes in an undirected, weighted graph which contains no edges, connecting a node to itself, and no multiple edges between the same two nodes. It is mainly based on the idea of superpositional graphs and the realization of a special quantum *AND* operator, which is iteratively applied until an entangled state is generated, which, when measured, collapses to a state representing a path from start to end. A detailed description is omitted.

General structure of QA's

A QA calculates the qualitative properties of the function f. From a mathematical standpoint, a function f is the map of one logical state into another.

The problems solved by a QA can be stated as follows:

A function f is given: $\{0,1\}^n \to \{0,1\}^m$; find a certain property of the function f.

Or in the symbolic form as:

Input	*A function f: {0,1}^{n} →{0,1}^{m}*
Problem	*Find a certain property of f*

Fig. 1 is a block diagram showing a gate approach for simulation of a QA using classical computers. In Fig. 1, an input is provided to a QA and the QA produces an output. However, the QA can be transformed to produce a QAG such that an input vector (corresponding to the QA input) is provided to the QAG to produce an output vector (corresponding to the QA output).

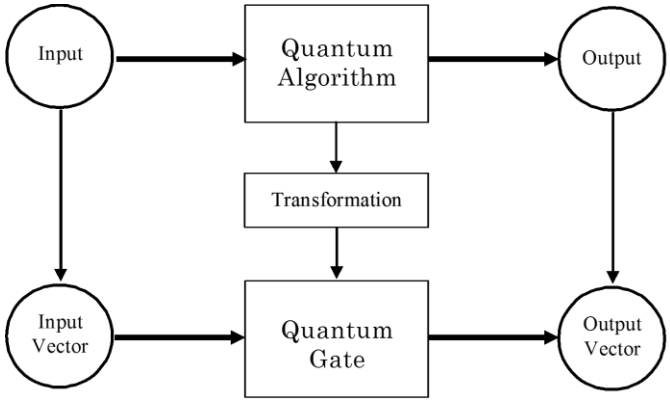

Figure 1. The gate approach for simulation of quantum algorithms using classical computers

Fig. 2 shows classification tree of QA's for quantum soft computing and control engineering applications. QA's are either decision-making or searching as described above. As shown, as example, in Fig. 2, Quantum Genetic Search Algorithms (QGSA) follows from Grover's and Shor's algorithms, and background for Robust KB design of Fuzzy Controllers follows from Deutch's, Deutch-Josa's, Grover's and/or Shor's algorithms.

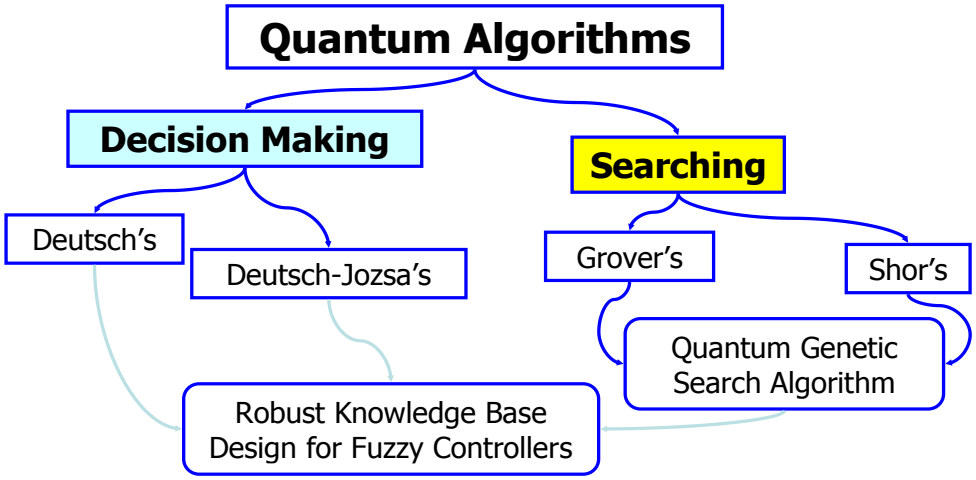

Figure 2. Classification of quantum algorithms

Fig. 3 is a block diagram showing the design process of the QAG. Let us briefly consider the design process of QAG.

In Fig. 3 an input block of the QA is a function f that maps binary strings into binary strings. This function f is represented as a map table, defined for every string its image. The function is first encoded into a unitary matrix operator U_F depending on the properties of f. In a certain sense, this operator calculates f when its input and output strings are encoded into canonical basis vectors of a complex Hilbert Space. The operator U_F maps the vector code of every string into the vector code of its image by f. The quantum block operates on basis vectors in a complex Hilbert space. The vectors operated on by the quantum block are provided to a decoder, which decodes the vectors to produce an answer.

Once generated, the matrix operator U_F is embedded into a quantum gate G. The quantum gate G is a unitary matrix whose structure depends on the form of matrix U_F and on the problem to be solved. The quantum gate is a unitary operator built from the dot composition of other more specific operators. The specific operators are described as tensor products of smaller matrices.

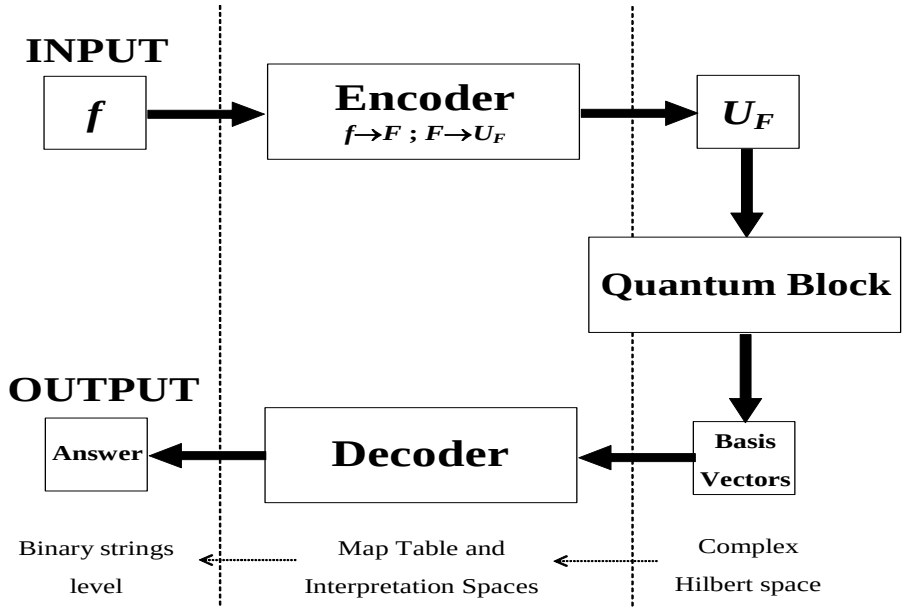

Figure 3. Schematic diagram of QA

Fig. 4 shows the general structure of a quantum circuit for a QAG.

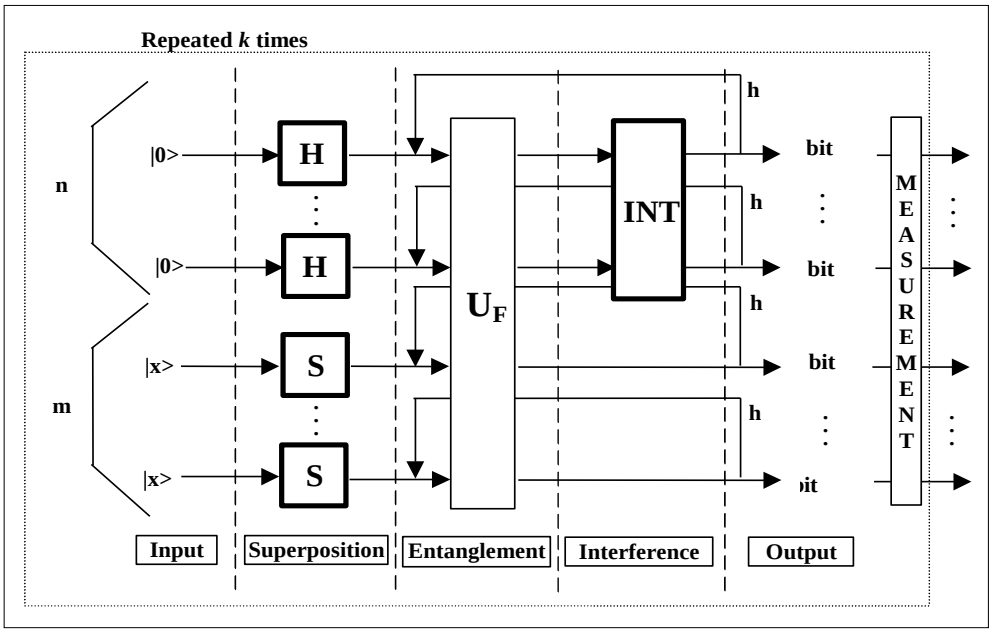

Figure 4. A quantum circuit

The quantum circuit is a high-level description of how these smaller matrices are composed using tensor and dot products in order to generate the final quantum gate as shown in Fig. 4.

Thus, the mathematical background of this approach is based on mappings between the quantum block operations in the complex Hilbert space. The encoder and decoder operate in a map table and interpretation space, and input/output occurs on a binary string level. The Clifford and Pauli groups are the background for universal QAG design for simulation of a QA's on classical computers.

As shown in Fig. 4, the general structure of the QAG is based on three quantum operators (superposition, entanglement, and interference) and measurement. The QAG acts on an initial canonical basis vector to generate a complex linear combination (called a superposition) of basis vectors as an output. This superposi-

tion contains the full information to answer the initial problem. After the superposition has been created, measurement takes place in order to extract the answer information. In quantum mechanics, a measurement is a non-deterministic operation that produces as output only one of the basis vectors in the entering superposition. The probability of every basis vector of being the output of measurement depends on its complex coefficient (probability amplitude) in the entering complex linear combination.

Therefore, the segmental action of the quantum gate and of measurement makes up a quantum block (see Fig. 3). The quantum block is repeated *k* times in order to produce a collection of *k* basis vectors. Since measurement is a non-deterministic operation, these basis vectors will not necessarily be identical, and each basis vector encodes a peace of the information needed to solve the problem. The last part of the algorithm involves interpretation of the collected basis vectors in order to get the final answer for the initial problem with some probability.

Fig. 5 shows the general structure of QA that includes almost all of described peculiarities.

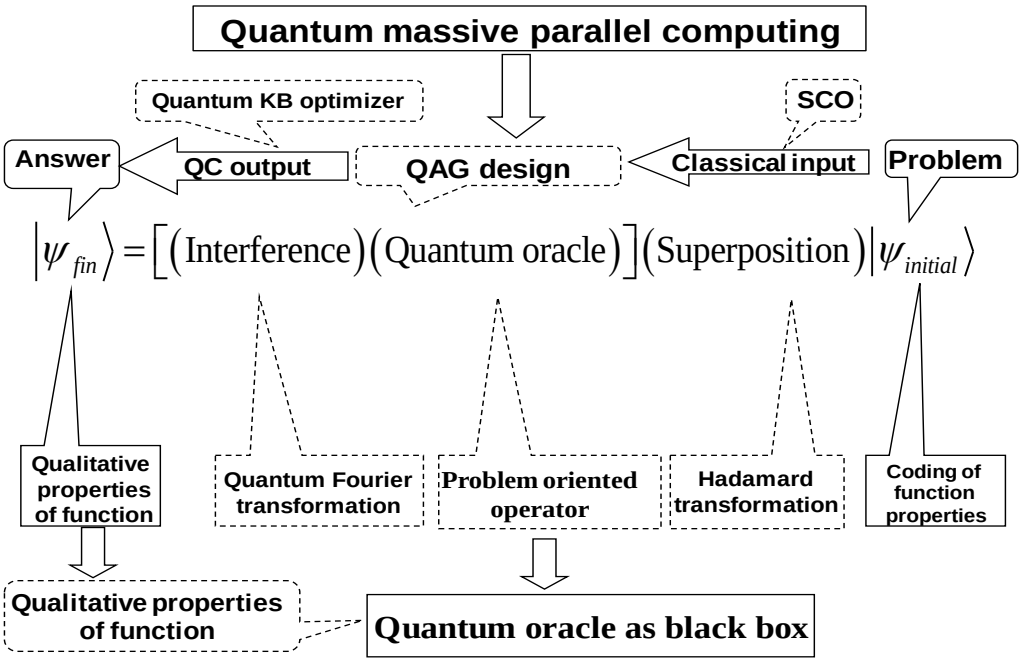

Figure 5. General structure of QA

Let us discuss briefly any mathematical backgrounds and its physical peculiarities for quantum computing based on QAG.

Basic programming techniques and simple quantum algorithms

It is assumed that certain computational problems can be solved on a quantum computer with a lower complexity than possible on classical computers since there are problems which are efficiently solvable by quantum algorithms but not classically up to now. In doing so, quantum algorithms take advantage of basic computational techniques, namely *superpositioning*, *quantum parallelism* and *quantum interference*. Additionally, *entanglement* seems to be a source of computational power which quantum algorithm can benefit from.

In this item fundamental programming techniques are discussed on the basic of the simple quantum algorithm for Deutsch's problem or its generalization, the Deutsch-Jozsa problem. Also, a quantum algorithm for quantum teleportation is explained, demonstrating the power of quantum entanglement. Both problems, quantum teleportation and Deutsch-Jozsa, were already used as test problems for quantum circuit evolutions.

Decision-making QA models: Deutsch's, Deutsch-Jozsa's, Simon's and Berstein-Vazirani benchmarks QA

Classical computer science relies on the concept of Turing machines as a unifying model of universal computation. According to the modern Church-Turing Thesis, this concept is interpreted in the form that every physical reasonable model of computation can be *efficiently* simulated on a probabilistic Turing machine. Recently this understanding, which was taken for granted for a long time, has required a severe reorientation because of the emergence of new computers that do not rely on classical physics but, rather, use effects predicted by quantum mechanics.

Remark. It has been realized that, by using the principle of quantum mechanics, there are problems for which a putative quantum computer could outperform any classical computer. QA's are benefit from the application of the superposition principle to the internal states of the quantum computer, which are considered to be states in a (finite-dimensional) Hilbert space. As a result, these algorithms lad to a new theory of computation and might be of central importance to physics and computer science. Striking examples of QA's are Shor's factoring algorithm, Grover's QSA and QA's for quantum error-correcting codes.

The goal of the QA is to learn some property of the function f . The QA's presented in this item exploit the fundamental principles of quantum mechanics: *interference, superposition* and *entanglement* that quantum physics offer (see Fig. 5).

We have explored these principles in various QA's (see below).

Main quantum operation in quantum computing. Quantum computing can manipulate quantum information by means of unitary transformations. In particular, they can work with superpositions.

Superposition operator. For instance, a single-q-bit Walsh-Hadamard operation H transforms a q-bit from $|0\rangle$ to $|+\rangle$ and from $|1\rangle$ to $|-\rangle$. When H is applied to a superposition such as $|+\rangle$, it follows by the linearity of quantum mechanics that the resulting state is

$$\frac{1}{2}\left\{\left(|0\rangle+|1\rangle\right)+\left(|0\rangle-|1\rangle\right)\right\}=0\,.$$

This illustrates the phenomenon of *destructive interference*, by which component $|1\rangle$ of the state is erased. Consider now an n-q-bit quantum register initialized to $|0^n\rangle$. Applying a Walsh-Hadamard transform to each of these q-bits yields an equal superposition of all n-bit classical states: $|0^n\rangle \xrightarrow{H} \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle$.

Unitary operation U_f: Quantum oracle. Consider now a function $f:\{0,1\}^n \to \{0,1\}$, that maps n-bit strings to a single bit. On a quantum computer, because unitary transformations are *reversible*, it is natural to implement it as a unitary transformation U_f that maps $|x\rangle|b\rangle$ to $|x\rangle|b\oplus f(x)\rangle$, where x is an n-bit string, b is a single bit, and «$\oplus$» denotes the Exclusive – OR (XOR). Schematically, $|x\rangle|b\rangle \xrightarrow{U_f} |x\rangle|b\oplus f(x)\rangle$.

Remark. Much of current interest in quantum computation was spurred by Peter Shor's momentous discovery that quantum computers can in principle factor large numbers and extract discrete logarithms in polynomial time and thus break much of contemporary cryptography such as the RSA cryptosystem and the Diffie-Hellman public-key exchange. However, this does not provide advantage of quantum computation because nobody knows for sure that these problems genuinely hard for classical computers. On the other hand, it has been demonstrated that quantum computers can solve some problems exponentially faster than any classical computer provided the input is given as an *oracle*, and even if we allow bounded errors. In this model, some function $f:\{0,1\}^n \to \{0,1\}$ is given as a black-box, which means that the only way to obtain knowledge about f is to query the black-box on chosen inputs. In the corresponding quantum oracle model, a function f is provided by a black-box that applies unitary transformation U_f to any chosen quantum state, as described by: $|x\rangle|b\rangle \xrightarrow{U_f} |x\rangle|b\oplus f(x)\rangle$.

Action effects of quantum computing operations. The linearity of quantum mechanics gives rise to two important phenomena.

1. *Quantum parallelism*: We can compute f on arbitrary many classical inputs by a single application of U_f to a suitable superposition: $\sum_x \alpha_x |x\rangle|b\rangle \xrightarrow{U_f} \sum_x \alpha_x |x\rangle|f(x)\oplus b\rangle$. When this is done, the additional output q-bit may become entangled with the input register;
2. *Phase kick-back*: The outcome of f can be recorded in the *phase* of the input register rather than being XOR-ed to the additional output q-bit:

$$|x\rangle|-\rangle \xrightarrow{U_f} (-1)^{f(x)}|x\rangle|-\rangle; \qquad \sum_x \alpha_x |x\rangle|-\rangle \xrightarrow{U_f} \sum_x \alpha_x (-1)^{f(x)}|x\rangle|-\rangle .$$

The fundamental questions in quantum computing are following:

(1). Where does the surprising computational advantage provided by quantum mechanics come from?

(2). What is the non-local property of quantum mechanics that leads to such an advantage?

(3). Do superposition and interference provide the quantum advantage?

Probably the most often heard answer is that the power of quantum computing comes from the *use of entanglement*, and indeed there are very strong arguments in favor of this believe.

For our purposes it will suffice to model the quantum system as a «*block box*» and focus our attention on two discrete observables out of a complete set, which we shall call the input and output register. Following the standard notation from quantum computing, we shall indicate the computation of a function $f: A \to B$ as $|x\rangle|0\rangle \to |x\rangle|f(x)\rangle$, the first is ket describing the state of the input register and the second is the state of the output register. Kets are labeled according to the elements of A and B they represent.

As mentioned above one of the most powerful features of quantum computation is quantum parallelism. The superposition principle of quantum mechanics allows us to prepare the computing in a coherent superposition of a set $I \subseteq A$ of input states. After a single run, all of the corresponding outputs $f(x)$ appear in the final state, according to the temporal evolution

$$\sum_{x\in I} |x\rangle \otimes |0\rangle \to \sum_{x\in I} |x\rangle \otimes |f(x)\rangle .$$

Remark. Unfortunately, this is no «pay one, take N ». In fact, the result is an entangled state of the input and output registers, and there is no single measurement allowing us to extract from it all the computed values of $f(x)$. However, it may well be possible to distil from this final state some *global property* of the function, thus exploiting quantum parallelism. One of the most famous examples presented by D. Deutsch, who showed that a single quantum computation may suffice to state whether a two-valued function of a two-valued variable is constant or not. D. Deutsch and R. Jozsa generalized this result showing that the problem of classifying a given function $f: \{0,\ldots,2N-1\} \to \{0,1\}$ as «*not constant*» or «*not balanced*» can be solved in polynomial time by means of a quantum computing (the time required by a classical solution is exponential). Also D.R. Simon showed that the problem of determining if a function is invariant under a so-called XOR mask, while it is classically intractable, admits an efficient quantum solution. All of the algorithms cited above (apart from the last, for which Simon also considered a fully probabilistic generalization) are characterized by a variable running time and zero error probability. They consist of a non-classical computation like $\sum_{x\in I} |x\rangle \otimes |0\rangle \to \sum_{x\in I} |x\rangle \otimes |f(x)\rangle$ followed by a measurement of the final state of the computing, as a result of which either the correct answer is obtained or the relevant information is destroyed and an explicitly inconclusive result is returned. In the latter case one has to go through the whole procedure again, so that only an average running time for the QA can be estimated.

Remark. Global properties of functions that can be determined by such an algorithm are said to be *Computable by Quantum Parallelism* (QPC). This definition was put forward by Jozsa who also demon-

strates that, at least in the case of two-valued functions, the QPC properties that can be determined by means of a single computation are an exponentially small fraction of all the possible global properties.

The Deutsch's quantum decision-making algorithm

The Deutsch's QA is the simplest QA, nevertheless it is important because it shows us the computing power of a quantum computation.

Qualitative computational model of Deutsch's Problem. Let us consider a Boolean function f that maps $\{0,1\} \to \{0,1\}$. There are exactly four functions of this type:

$$\textit{Two constant functions}: \begin{array}{l} f_1(0) = f_1(1) = 0, \\ f_2(0) = f_2(1) = 1; \end{array} \text{ and,}$$

$$\textit{two balanced functions}: \begin{array}{ll} f_3(0) = 0, & f_3(1) = 1, \\ f_4(0) = 1, & f_4(1) = 0. \end{array}$$

Problem: Is it possible, by computing f only once, to find out whether the binary numbers $f(0)$ and $f(1)$ are the same or different?

Remark. Classical intuition tells us that we have to evaluate both $f(0)$ and $f(1)$, i.e., to compute f twice, to give a conclusive answer to the previous question. This is so when one resort to quantum mechanics and to its most peculiar features, such as *entanglement* and *superposition.*

In this, one would like to determine whether a function $f:\{0,1\} \to \{0,1\}$ is constant or balanced, i.e., whether $f(0) = f(1)$ or $f(0) \neq f(1)$ is using a quantum computing technology.

The four possible outcomes of f are

$f(0) = f(1) \quad = 0$	$(constant)$
$f(0) = f(1) \quad = 1$	$(constant)$
$f(0) = 0, f(1) = 1$	$(balanced)$
$f(0) = 1, f(1) = 0$	$(balanced)$

Ordinarily, one has to determine *both* $f(0)$ and $f(1)$ to infer the nature of the function, since the knowledge of one does not shed light on the value of the other. However, it was shown that by applying a certain sequence of unitary operators (gates) on a given initial *quantum-mechanical* state, and then making just *one* measurement on the basis final state, the nature of the function f can be determined.

Mathematical model and computational steps in Deutsch's QA. Deutsch's problem, as abovementioned, in its simplest form concerns the analysis of single binary functions $f(x): B \to B$, where $B = \{0,1\}$ is the set of possible values for a single bit.

Clearly there are exactly four such functions, which may be described by their *truth tables*, as shown in Table 4.

According to qualitative model description of Deutsch's problem, these four functions can be divided into groups: the two «*constant*» functions, for which $f(x)$ is independent of x (f_{00} and f_{11}), and the two «*balanced*» functions, for which $f(x)$ is zero for one value of x and unity for the other (f_{01} and f_{10}).

Table 4. The four possible binary functions mapping one bit to another

x	$f_{00}(x)$	$f_{01}(x)$	$f_{10}(x)$	$f_{11}(x)$
0	0	0	1	1
1	0	1	0	1

Some unknown function is given f (known to be one of these four functions), it is possible to determine which of the four functions it is by applying f to two known inputs: 0 and 1. This procedure also provides enough information to determine whether the function is constant or balanced.

However, knowing whether the function is constant or balanced corresponds to only one bit of information, and so it might be possible to answer this question using only one evaluation of the function f. Equivalently, it might be possible to determine the value of $f(0) \oplus f(1)$ using only one evaluation of f. (The symbol $\oplus$ indicates addition mod2, and for two one bit numbers, a and b, $a \oplus b$ equals 0 if a and b are the same, and 1 if they are different.)

In fact this can be achieved as long as the calculation is performed using a quantum computing rather than a classical one.

Quantum operator (entanglement) for quantitative description of function property. Quantum computing of necessity use reversible logic, and so it is not possible to implement the binary function f directly. It is, however, possible to design a propagator U_f, which captures f within a reversible transformation by using a system with two input q-bits and two output q-bits as follows: $|x\rangle|y\rangle \xrightarrow{U_f} |x\rangle|y \oplus f(x)\rangle$. The two input bits are preserved (x is preserved directly, while y is preserved by combining it with $f(x)$, the desired result), and so U_f corresponds to a reversible transformation.

Remark. Note that for any one bit number a, $0 \oplus a = a$, and so values of $f(x)$ can be determined by setting the second input bit to 0.

Using this propagator and appropriate input states it is possible to evaluate *both* $f(0)$ and $f(1)$ using as following: $|0\rangle|0\rangle \xrightarrow{U_f} |0\rangle|f(0)\rangle$ *and* $|1\rangle|0\rangle \xrightarrow{U_f} |1\rangle|f(1)\rangle$.

Quantum circuit for Deutsch's QA. The approach outlined above, in which the state of a quantum computing is described explicitly, swiftly becomes unwieldy, and it is useful to use more compact notations. One particularly simple approach is to use quantum circuits, which may be drawn by analogy with classical electronic circuits. In this approach lines are used to represent «wires» down which q-bits «flow», while boxes represent quantum gates, which perform appropriate unitary transformations. For example, the analysis of f can be summarized using the circuit shown in Fig. 6.

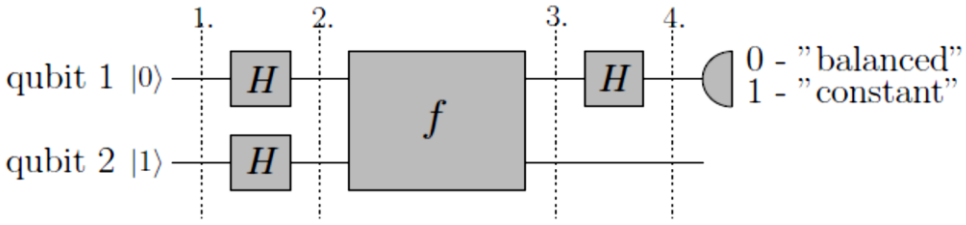

Figure 6. Deutsch's QA circuit

Quantum computation parallelism in Deutsch QA. According to the Deutsch's algorithm (Fig. 6), we should simply perform the following operations:

– we take two q-bits and prepare them in the product state $|0\rangle|1\rangle$;

– we apply (step 1 in Fig. 6) to each q-bit the unitary Hadamard transformation H, that produces the superposition states as following:

– $|0\rangle \to \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, $|1\rangle \to \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

– ;

– we evaluate (step 2 in Fig. 6) the unknown function on the first gubit and then store the result in the second q-bit in the following way: $|x\rangle|y\rangle \xrightarrow{U_f} |x\rangle|y \oplus f(x)\rangle$;

– we apply (step 3 in Fig. 6) again the transformation H to each q-bit; and

– we perform (step 4 in Fig. 6) a measurement on the first q-bit.

– We are using the following physical interpretation of measurement results: If the first q-bit is in state $|0\rangle$, the function is constant; if it is in the state $|1\rangle$, the function is balanced.

Describing the algorithm in detail, the application of the Hadamard transforms H to each q-bit of the initial state realizes $|0\rangle|1\rangle \to \frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle)$. Then the evaluation of the unknown function produces

$$\frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) \to \frac{1}{2}\{|0\rangle[|f(0)\rangle - |1 \oplus f(0)\rangle] + |1\rangle[|f(1)\rangle - |1 \oplus f(1)\rangle]\}.$$

At this stage, depending on which function we consider, we have four possible outcomes. Finally, by applying once again the Hadamard transform H to each q-bit, we obtain

$\frac{1}{2}\{\|0\rangle[\|0\rangle - \|1\rangle] + \|1\rangle[\|0\rangle - \|1\rangle]\}$	$\to$	$\|0\rangle\|1\rangle$	*for* f_1
$\frac{1}{2}\{\|0\rangle[\|1\rangle - \|0\rangle] + \|1\rangle[\|1\rangle - \|0\rangle]\}$	$\to$	$-\|0\rangle\|1\rangle$	*for* f_2
$\frac{1}{2}\{\|0\rangle[\|0\rangle - \|1\rangle] + \|1\rangle[\|1\rangle - \|0\rangle]\}$	$\to$	$\|1\rangle\|1\rangle$	*for* f_3
$\frac{1}{2}\{\|0\rangle[\|1\rangle - \|0\rangle] + \|1\rangle[\|0\rangle - \|1\rangle]\}$	$\to$	$-\|1\rangle\|1\rangle$	*for* f_4

We emphasize, that, regardless the specific Boolean function implemented, the second q-bit always ends up in state $|1\rangle$. However, even though it does not convey any information on the parity properties of the function, it may turn our to be useful to check eventual errors taking place in a real experiment.

We shall briefly recall abovementioned example before confronting the problem of its generalization.

Example. Suppose the calculation begins with the second q-bit in the superposition $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Then

$\|x\rangle\left(\frac{\|0\rangle - \|1\rangle}{\sqrt{2}}\right)$	$\xrightarrow{U_f}$	$\|x\rangle\left(\frac{\|0 \oplus f(x)\rangle - \|1 \oplus f(x)\rangle}{\sqrt{2}}\right)$
	=	$\begin{cases} \|x\rangle\left(\frac{\|0\rangle - \boxed{\|1\rangle}}{\sqrt{2}}\right) & \text{if } f(x) = 0, \\ \|x\rangle\left(\frac{\boxed{\|1\rangle} - \|0\rangle}{\sqrt{2}}\right) & \text{if } f(x) = 1 \end{cases}$
	=	$(-1)^{f(x)}\|x\rangle\left(\frac{\|0\rangle - \|1\rangle}{\sqrt{2}}\right)$

Remark. We have used the fact that $0 \oplus a = a$, as before, while $1 \oplus a = 1$ if $a = 0$ and 0 if $a = 1$. The value of $f(x)$ is now encoded in the overall phase of the result, with the q-bits left otherwise unchanged. While this is not particularly useful, suppose the calculation begins with the first q-bit also in a superposition of states, namely $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Then

$\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)\frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle)$	$\xrightarrow{U_f}$	$\frac{1}{\sqrt{2}}\left((-1)^{f(0)}\|0\rangle + (-1)^{f(1)}\|1\rangle\right)\frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle)$
	$=$	$\frac{(-1)^{f(0)}}{2}\left(\|0\rangle + (-1)^{f(0)\oplus f(1)}\|1\rangle\right)(\|0\rangle - \|1\rangle)$

with the first q-bit ending up in the superposition $\frac{1}{\sqrt{2}}(|0\rangle \pm |1\rangle)$, with desired answer ($f(0) \oplus f(1)$) encoded as the *relative* phase of the two states contributing to the superposition. This relative phase can be measured, and so the value of $f(0) \oplus f(1)$ (that is, whether $f(x)$ is constant or balanced) has been determined using only one application of the propagator U_f, that is only one evaluation of the function $f(x)$.

This approach can be easily implemented using also a quantum circuit. This circuit starts off from eigenstates, uses Hadamard transformations to convert these into superpositions, applies the propagator U_f to these superpositions, and finally use another pair of Hadamard transforms to convert the superpositions back into eigenstates which encode the desired result. So far this is simply using a quantum computing to simulate a classical computing implementing classical algorithms. With a quantum computing, however, it is not necessary to start with the system in some eigenstate; instead it is possible to begin with a superposition of states.

Qualitative analysis of Deutsch's QA. Suppose we are given a function $f : \{0,1\} \to \{0,1\}$ and we are interested to know whether f is constant or not. Of course there are only four such functions (i.e., four instance of the problem), namely

$$\begin{cases} f_1(0) = 0 \\ f_1(1) = 0 \end{cases} \quad \begin{cases} f_2(0) = 1 \\ f_2(1) = 1 \end{cases} \quad \begin{cases} f_3(0) = 0 \\ f_3(1) = 1 \end{cases} \quad \begin{cases} f_4(0) = 1 \\ f_1(1) = 0 \end{cases}.$$

If all we can use a classical computing, there is only one-way to do the job: we must compute <u>*both*</u> $f(0)$ and $f(1)$, and compare them to check if they are equal. On the contrary, since in this simple case the property « f *is constant*» is QPC, a quantum computing gives us a fair chance of finding the solution at the cost of the single computation as follows:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle \to \frac{1}{\sqrt{2}}\{|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle\}.$$

After the computation, the calculator halts with its input and output registers in one of four possible states, corresponding to the four possible functions:

$\|f_1\rangle$	$=$	$\frac{1}{\sqrt{2}}[\|0\rangle\|0\rangle + \|1\rangle\|0\rangle]$	$\|f_2\rangle$	$=$	$\frac{1}{\sqrt{2}}[\|0\rangle\|1\rangle + \|1\rangle\|1\rangle]$
$\|f_3\rangle$	$=$	$\frac{1}{\sqrt{2}}[\|0\rangle\|0\rangle + \|1\rangle\|1\rangle]$	$\|f_4\rangle$	$=$	$\frac{1}{\sqrt{2}}[\|0\rangle\|1\rangle + \|1\rangle\|0\rangle]$

Since the above states are linearly dependent, they cannot be distinguished with certainty. This means that no measurement can establish which function was actually computed, or, which is the same, it's impossible to extract from the final state *both* the values of $f(0)$ and $f(1)$. However, we need only discriminate $|f_1\rangle$ and $|f_2\rangle$, the final states yielded by the constant functions, from $|f_3\rangle$ and $|f_4\rangle$. This case actually can

be done by measuring on the final state of the two registers on observable with the following non-degenerate eigenstates:

$\lvert Same\rangle$	=	$\frac{1}{2}(\lvert 0\rangle+\lvert 1\rangle)(\lvert 0\rangle-\lvert 1\rangle)$	$\lvert Different\rangle$	=	$\frac{1}{2}(\lvert 0\rangle-\lvert 1\rangle)(\lvert 0\rangle-\lvert 1\rangle)$
$\lvert Fail\rangle$	=	$\frac{1}{2}(\lvert 0\rangle+\lvert 1\rangle)(\lvert 0\rangle+\lvert 1\rangle)$	$\lvert Error\rangle$	=	$\frac{1}{2}(\lvert 0\rangle-\lvert 1\rangle)(\lvert 0\rangle+\lvert 1\rangle)$

These four states can be thought of as «flags» indicating the result of the computation and have been named according to their meaning. This becomes clearer as soon as we rewrite the final states on the basis of the above eigenvectors:

$\lvert f_1\rangle$	=	$\frac{1}{\sqrt{2}}\left[\lvert Fail\rangle+\lvert Same\rangle\right]$	$\lvert f_2\rangle$	=	$\frac{1}{\sqrt{2}}\left[\lvert Fail\rangle+\lvert Different\rangle\right]$
$\lvert f_3\rangle$	=	$\frac{1}{\sqrt{2}}\left[\lvert Fail\rangle-\lvert Same\rangle\right]$	$\lvert f_4\rangle$	=	$\frac{1}{\sqrt{2}}\left[\lvert Fail\rangle-\lvert Different\rangle\right]$

It is now evident that

1	*Projection along the eigenvector* $\lvert Same\rangle$ *can only take place if the state of the computing is either* $\lvert f_1\rangle$ *or* $\lvert f_2\rangle$*, i.e., when* f *is constant*
2	*Likewise, projection along the eigenvector* $\lvert Different\rangle$ *can only occur if* f *is not a constant function*
3	*Regardless the final state* $\lvert f_i\rangle$ *of the two registers after the computation, the measurement can yield a* $\lvert Fail\rangle$ *state with probability* $\frac{1}{2}$
4	*State* $\lvert Error\rangle$ *is orthogonal to the four final states listed above, therefore, it should show up only as a consequence of noise-induced errors*

Remark. According to the definition of the QPC class, the QA can either give us the correct answer or no answer at all: as long as every thought works properly, we'll never get a wrong result. This comes in handy when we are asked to solve a decision-making problem in which simply waiting has a much higher utility than taking a wrong action. In this case we can discard the $\lvert Fail\rangle$ results and base our decisions upon the meaningful answers, that we know to be correct.

Role of computational basis in Deutsch's problem. We will write $B=\{0,1\}$ for the additive group of integers mod2 and denote by $\mathcal{B}$ the Hilbert space of one q-bit (i.e., a two-dimensional Hilbert space) equipped with a standard basis denoted by $\{\lvert 0\rangle,\lvert 1\rangle\}$. $\mathcal{B}^n$ will denote the Hilbert space of n q-bits. The dual basis of $\mathcal{B}$ denoted by $\{\lvert 0'\rangle,\lvert 1'\rangle\}$ is defined by $\lvert 0'\rangle=\frac{1}{\sqrt{2}}(\lvert 0\rangle+\lvert 1\rangle)$, $\lvert 1'\rangle=\frac{1}{\sqrt{2}}(\lvert 0\rangle-\lvert 1\rangle)$. Hadamard transformation H interchanges the standard and dual basis.

Remark. In terms of real geometry, the dual basis lies on the 45^0 lines between the orthogonal directions $\lvert 0\rangle$ and $\lvert 1\rangle$ and H is the transformation given by reflection in a line at angle $\frac{1}{8}\pi$ to the $\lvert 0\rangle$ direction. Thus the eigenvectors of H (parallel and perpendicular to the mirror line) are $\cos\frac{\pi}{8}\lvert 0\rangle\pm\sin\frac{\pi}{8}\lvert 1\rangle$ belonging to $\lambda=\pm 1$, respectively. We will see that H is also the Fourier transform on the group B. The elements of $\mathcal{B}^n$ are n bit strings.

If $x=(x_1,\ldots,x_n)$ and $y=(y_1,\ldots,y_n)$ are in $\mathcal{B}^n$, then we write $x\oplus y=(x_1\oplus y_1,\ldots,x_n\oplus y_n)\in\mathcal{B}^n$, $x\cdot y=(x_1y_1\oplus\ldots\oplus x_ny_n)\in\mathcal{B}$. The operations on the right-hand sides in last equations are being the addition

and the multiplication mod2 in $\mathcal{B}$. Note that $x \cdot y$ is the parity of the number of places where x and y both have a bit value of 1.

We are given a «black-box» or oracle that computes a function $f : \mathcal{B}^n \to \mathcal{B}$ and we are required to decide whether a certain «global» property (i.e., a joint property of all the function values) holds of f. For quantum computation, the black-box is given as a unitary transformation U_f on $n+1$ q-bits given in standard basis by

$$U_f : \underbrace{|x_1\rangle|x_2\rangle\ldots|x_n\rangle}_{input}|y\rangle \to \underbrace{|x_1\rangle|x_2\rangle\ldots|x_n\rangle}_{Register\ 1}\underbrace{|y \oplus f(x_1, x_{2,\ldots,} x_n)\rangle}_{Register\ 2}.$$

We will often abbreviate $|x_1\rangle|x_2\rangle\ldots|x_n\rangle$ as $|x\rangle$ for $x \in \mathcal{B}^n$. Thus if y is initially set to zero, the value of f may be read from the last q-bit.

For the problem, referred to as Deutsch's XOR problem, we have $n=1$ so that f is one of the four possible functions $f : \mathcal{B} \to \mathcal{B}$. We are to decide whether $f(0) \oplus f(1)$ is 0 or 1. Equivalently, we wish to decide whether is f a constant function or a balanced function is.

Remark. It is clear, as we know any classical computer requires evaluating f twice to decide this. According to Deutsch's original method, the problem may be solved on a quantum computer after running U_f only once, but QA succeeds only with probability $\frac{1}{2}$ (and we know when it has been successful). The method is simply to run U_f on the input superposition $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, yielding the state $\frac{1}{\sqrt{2}}(|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle)$. Writing this state in the dual basis we have the four possibilities given as the following:

by two constant functions

$$\frac{1}{\sqrt{2}}(|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle) = \begin{cases} \frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|0\rangle) = \frac{1}{\sqrt{2}}(|0'\rangle|0'\rangle + |0'\rangle|1'\rangle) \\ \frac{1}{\sqrt{2}}(|0\rangle|1\rangle + |1\rangle|1\rangle) = \frac{1}{\sqrt{2}}(|0'\rangle|0'\rangle - |0'\rangle|1'\rangle) \end{cases}$$

and

two balanced functions

$$\frac{1}{\sqrt{2}}(|0\rangle|f(0)\rangle + |1\rangle|f(1)\rangle) = \begin{cases} \frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|1\rangle) = \frac{1}{\sqrt{2}}(|0'\rangle|0'\rangle + |1'\rangle|1'\rangle) \\ \frac{1}{\sqrt{2}}(|0\rangle|1\rangle + |1\rangle|0\rangle) = \frac{1}{\sqrt{2}}(|0'\rangle|0'\rangle - |1'\rangle|1'\rangle) \end{cases}.$$

Physical interpretation of measurement results in dual basis. Now measure the second q-bit in the dual basis. If the result is $0'$ (which occurs with probability $\frac{1}{2}$ in every case), then we have lost all the information about the function f. If the result is $1'$, then measurement of the first q-bit will reliably distinguish between constant and balanced functions.

Role of interference quantum operator in Deutsch's QA. Quantum computing is used the quantum interference operator of different computational paths to enhance correct outcomes and suppress erroneous outcome of computations. In effect, they follow the same logical paradigm as (multi-particle) interferometers. We can show how QA's may be cast in this manner using as example the Deutsch's problem solution.

Let us consider the Boolean function f that maps $\{0,1\}$ to $\{0,1\}$. There are exactly four such functions: two constant functions [$\{f(0)=f(1)=0\}$ and $\{f(0)=f(1)=1\}$] and two balanced functions [$\{f(0)=0, f(1)=1\}$ and $\{f(0)=1, f(1)=0\}$]. It turns out that it is possible to construct a controlled function evaluation such that two possible eigenvalues are produced which may be used to determine whether the function is constant or balanced. This is done in the following way.

A controlled-U transformation can be used to produce a particular phase shift on the control q-bit corresponding to its eigenvalue on the auxiliary q-bit. If two eigenvalues of the controlled-U transformation lead to different orthogonal states in the control q-bit, a single measurement on this q-bit will suffice to distinguish the two cases.

Let us formally define the operation of «evaluating» f in term of the f *-controlled-NOT* operation on two bits: the first contains the input value and the second contains the output value. If the second bit is initialized to 0, the f *-controlled-NOT* operation maps $(x,0)$ to $(x, f(x))$. This is clearly just a formalization of the operation of computing f. In order to make the operation reversible, the mapping is defined for all initial settings of the two bits, taking (x,y) to $(x, y \oplus f(x))$. A single evaluation of the f *-controlled-NOT* operation on quantum superpositions suffices to classify f as constant or balanced. This is the real advantage of the quantum method over classical.

Classically if the f *-controlled-NOT* operation may be performed only once then it is impossible to distinguish between balanced and constant functions. Whatever the outcome is both possibilities (balanced and constant) remain for f. This corresponds to our classical intuition about the problem since it involves determining not particular values of $f(0)$ and $f(1)$, but a global property of f. Classically to determine this global property of f, we have to evaluate both $f(0)$ and $f(1)$, that as we known involves evaluating $f(0)$ and $f(1)$, twice. Deutsch's QA has the same mathematical structure as the Mach-Zehnder interferometer, with the two phases settings $\phi = 0, \pi$.

It is best represented as the quantum network, where the middle operation is the f *-controlled-NOT* operation that can be defined as the following:

$$|x\rangle|y\rangle \xrightarrow{f-c-N} |x\rangle|y \oplus f(x)\rangle .$$

The initial state of the q-bits in the network is $|0\rangle(|0\rangle - |1\rangle)$ (apart from a normalization factor that will be omitted in the following). After the first Hadamard transform, the state of the two q-bits has the form $(|0\rangle + |1\rangle)(|0\rangle - |1\rangle)$. To determine the effect of the f *-controlled-NOT* operation on this state first of all note that for each $x \in \{0,1\}$,

$$|x\rangle(|0\rangle - |1\rangle) \xrightarrow{f-c-N} |x\rangle(|0 \oplus f(x)\rangle - |1 \oplus f(x)\rangle) = (-1)^{f(x)}|x\rangle(|0\rangle - |1\rangle) .$$

Therefore, the state after the f *-controlled-NOT* operation is

$$\left((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\right)(|0\rangle - |1\rangle) .$$

It means that for each x, the $|x\rangle$ term acquires a phase factor of $(-1)^{f(x)}$, that corresponds to the eigenvalue of the state of the auxiliary q-bit under the action of the operator that sends $|y\rangle$ to $|y \oplus f(x)\rangle$. As we known this state can be also written as $(-1)^{f(0)}\left(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle\right)(|0\rangle - |1\rangle)$ that after applying the second Hadamard transform to the first q-bit, becomes

$$(-1)^{f(0)}|f(0) \oplus f(1)\rangle(|0\rangle - |1\rangle) .$$

Therefore, the first q-bit is finally in state $|0\rangle$ if the function f is constant and in state $|1\rangle$ if the function is balanced, and a measurement of this q-bit distinguishes these cases with certainty.

By a suitable choice of a time-dependent Hamiltonian, Deutsch's QA can be implemented by an adiabatic quantum computing.

Deutsch-Jozsa quantum problem

In the second algorithm (*Deutsch* and *Jozsa*, 1992), referred as Deutsch- Jozsa's algorithm, we are given n and a function $f: B^n \to B$. It is promised that f is either constant or balanced (where balanced means that f takes values of 0 and 1 an equal number of times, i.e., 2^{n-1} times each). The problem is to decide whether f is balanced or constant. The method, described in detail in *Deutsch* and *Jozsa* (1992), involves running U_f *twice* (and using Hadamard transform H $O(n)$ times) to construct the state

$$|f\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in B^n} (-1)^{f(x)} |x\rangle . \qquad (1)$$

Then $|f\rangle$ for any constant function is orthogonal to the corresponding state for any balanced function and thus we can solve the decision problem with certainty by a suitable measurement on the resulting state. The QA always runs in time $O(2^n)$, at least in some cases.

Remark. Deutsch's XOR problem is the $n=1$ case of the above decision problem. However, the above algorithm, running U_f twice, offers no advantage over the obvious classical algorithm for $n=1$. Another distinction between the above two algorithms is that the XOR problem is solved only with probability $\frac{1}{2}$, whereas the second algorithm is always successful. An interesting recent innovation (see in details below) fully unifies and considerably improves the above two algorithms: the XOR problem may be solved with certainty and the state in Eq. (1) may be constructed by running U_f only *once*. The improved XOR algorithm is then more precise than $n=1$ case of the improved Deutsch's algorithm. The basic idea was to set the output register to the state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ before applying U_f. Note that

$$U_f : |x\rangle(|0\rangle - |1\rangle) \to \begin{cases} |x\rangle(|0\rangle - |1\rangle), & \text{if } f(x)=0 \\ |x\rangle(|1\rangle - |0\rangle), & \text{if } f(x)=1 \end{cases}.$$

Thus $U_f : \frac{1}{\sqrt{2^n}} \sum_{x \in B^n} |x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \to \left(\frac{1}{\sqrt{2^n}} \sum_{x \in B^n} (-1)^{f(x)} |x\rangle \right) \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, giving the state $|f\rangle$ in the first n q-bits after only one application of U_f. The last q-bit plays a curiously passive role in the fact that its state is unchanged in the process. This is reminiscent of the similarly passive role of the second register in Shor's QA.

The explicit description of the measurement on $|f\rangle$ that distinguishes balanced from constant functions is significant for subsequent developments. Firstly, apply the operation H to each of the n q-bits of $|f\rangle$. Denoting the resulting n-q-bit operation by H_n, we have, for each $x \in B^n$, $H_n : |x\rangle \to \frac{1}{\sqrt{2^n}} \sum_{y \in B^n} (-1)^{x \cdot y} |y\rangle$.

Note that $H_n |0 \ldots 0\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in B^n} |y\rangle$ is the equal superposition of all the standard basis states and that up to an overall sign this coincides with $|f\rangle$ for f constant. Since $H_n H_n = I$ it follows that $H_n |f\rangle = |0 \ldots 0\rangle$ for f

to be constant. Thus if f is balanced then $H_n|f\rangle$ must be orthogonal to $|0\ldots0\rangle$, i.e., $|f\rangle$ lies in the span of $\{|x\rangle : x \neq 0\ldots0\}$. Hence to distinguish balanced from constant functions we apply H_n to $|f\rangle$ and then read the bits to see whether they are all zero or not.

The above measurement has 2^n natural outcomes (i.e., all n-bit strings) and we may ask if there are special balanced functions that yield the other outcomes $x \in B^n$ with certainty in the same way that constant functions lead to the outcome $0\ldots0$. For each $k \in B^n$ consider the function $f_k : B^n \to B$ given by $f_k(x) = k \cdot x$. It can be easily verified that each f_k is a balanced function for $k \neq 0\ldots0$ (giving a small subset of all possible balanced functions). We will see that the operation H_n is the Fourier transform on the additive group B^n (also known as the Walsh or Hadamard transform) and the functions f_k are the Fourier (Walsh or Hadamard) basis functions. For these functions we have $H_n|f_k\rangle = |k\rangle$ that follows readily by comparing last expressions and the fact that $H_n H_n = I$. Thus this QA can reliably distinguish the 2^n functions f_k after evaluating the function only *once*.

However, this finer use of the measurement outcomes does not represent an exponential advantage over classical computation since the classical evaluation of just n values of f_k on the inputs $10\ldots0$, $010\ldots0$, up to $0\ldots01$ will successively reveal the n bits of k.

Assume that for one of Boolean functions $f : \{0,1\}^n \to \{0,1\}$, it is «promised» that it is either constant or balanced (i.e., it has an equal number of 0 outputs), and consider the goal of determining which of the two properties the function actually has.

How many evaluations of f are required to do this? Any classical algorithm for this problem would require $2^{n-1}+1$ evaluations of f before determining the answer with certainty in the worst case. There is a QA that solves this problem with a single evaluation of f. The algorithm is presented in Fig. 1.19, where the control register is now composed of n q-bits, all initially in state $|0\rangle$. It denotes as $|00\ldots0\rangle$ and as in QA for Deutsch's problem, an auxiliary q-bit is employed, that is initially set to the state $|0\rangle - |1\rangle$ and is not altered during the computation.

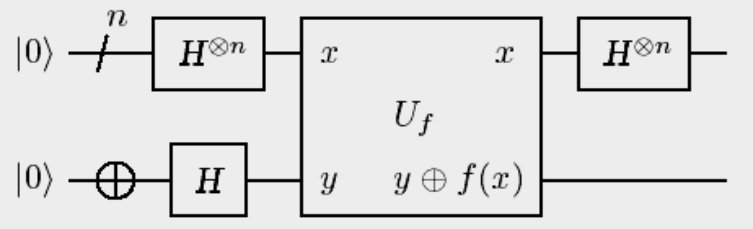

Figure 7. The quantum circuit for the Deutsch–Jozsa algorithm

Also, the n-q-bit Hadamard transform H is defined as $|x\rangle \xrightarrow{H} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle$, for all $x \in \{0,1\}^n$, where $x \cdot y = (x_1 \wedge y_1) \oplus \ldots \oplus (x_n \wedge y_n)$ (i.e. the scalar product modulo 2). This is equivalent to performing a one-q-bit Hadamard transform on each of the *n*-q-bits individually. The actual computation of the function f is by means of an *f*-controlled-NOT (the middle gate in Fig. 7) that acts as

$$|x\rangle|y\rangle \xrightarrow{f\text{-}c\text{-}N} |x\rangle|y \oplus f(x)\rangle .$$

This is similar to the above expression, except one that now $x \in \{0,1\}^n$.

Stepping through the execution of the network, the state after the first *n*-q-bit Hadamard transform is applied as $\sum_{x \in \{0,1\}^n} |x\rangle(|0\rangle - |1\rangle)$ that after the *f*-controlled-NOT gate, is $\sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle(|0\rangle - |1\rangle)$.

Finally, after the last Hadamard transform, the state is:

$$\sum_{x,y\in\{0,1\}^n}(-1)^{f(x)\oplus(x\cdot y)}|x\rangle\left(|0\rangle-|1\rangle\right).$$

Remark. The amplitude of $|00\ldots0\rangle$ is $\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}$, so if f is balanced then, for the state of the first n q-bits, the amplitude of $|00\ldots0\rangle$ is zero. Therefore, by measuring of the first n q-bits, it can be determined with certainty whether f is constant or balanced. Note that, as in Deutsch's example, this entails a single f-controlled-NOT operation. (This is a slight improvement of Deutsch-Jozsa original QA that involves two f-controlled-NOT operations.) Now let us summarize the computational steps of Deutsch-Jozsa QA.

Mathematical model of the Deutsch-Jozsa QA. Consider the subset of Boolean functions . with the property that f is either constant or balanced (i.e. it has an equal number of 0 outputs as 1 outputs). The Deutsch-Jozsa QA determines, for a given unknown function f, whether the function is constant or balanced. With a classical algorithm, this problem would require $2^{N-1}+1$ evaluations of f in the worst case whereas the QA solves it with a single evaluation by means of the following steps.

(i)	All q-bits are prepared in the initial state $\|0\rangle$; therefore for the n-q-bit register is in the state $\|00\ldots0\rangle$
(ii)	Perform an n-q-bit Hadamard transformation H: $$\|x\rangle\xrightarrow{H}\sum_{y\in\{0,1\}^n}(-1)^{x\cdot y}|y\rangle,\ x\in\{0,1\}^n,$$ where $x\cdot y=(x_1\wedge y_1)\oplus\ldots\oplus(x_n\wedge y_n)$ the scalar product modulo 2. This is equivalent to perform a one-bit Hadamard transformation to each q-bit individually
(iii)	Apply the f-controlled phase shift $U_f:|x\rangle\xrightarrow{U_f}(-1)^{f(x)}|x\rangle,\ x\in\{0,1\}^n$
(iv)	Perform another Hadamard transformation H
(v)	Measure the final state of the register. If the result is $\|00\ldots0\rangle$, the function f is constant; however, if the amplitude $a_{\|00\ldots0\rangle}$ of the state $\|00\ldots0\rangle$ is zero, the function f is balanced. This is because $a_{\|00\ldots0\rangle}=\frac{1}{2^n}\sum_{x\in\{0,1\}^n}(-1)^{f(x)}$.

Remark. It is reasonable to identify the functions f whose outputs can be mapped into each other by a bitwise NOT. For these functions, the gate operations in equation $|x\rangle\xrightarrow{U_f}(-1)^{f(x)}|x\rangle$ differ merely by a global phase factor (-1). We shall use the convention $f(00\ldots0)=0$. This reduces the number of gates that need to be implemented, by a factor of two and does not affect the exponential speed-up. It is interesting to note that the Deutsch-Jozsa QA does not involve entanglement for $n\le 2$ that is the $n=2$ version can be realized with uncoupled q-bits. On the other hand, the implementation of the algorithm, for $n>2$, involves entanglement (see in details Chapter 3) and therefore requires a set-up of coupled q-bits.

Example. To summarize, generalized by Deutsch and Jozsa, in its simplest form, the problem is stated as follows: «For an unknown given Boolean function $f(x)$, $f(x):\{0,1\}^n\to\{0,1\}$ determines whether it is constant or balanced by evaluating it only once», where «balanced» means equal number of variables for all function values. The original Deutsch problem corresponds to the case of $n=1$.

Thus, to evaluate $f(x)$ on a quantum computer, some unitary transformation $U_{f(x)}$ is necessary, and the Deutsch-Jozsa QA uses an f -controlled-NOT operations that is defined by

$$|x\rangle|y\rangle \xrightarrow{U_{f(x)}} |x\rangle \underbrace{|y \oplus f(x)\rangle}_{\boxed{Target}},$$
$$\boxed{Control}$$

$$x \in \{0,1\}^n = 0,1,\ldots,2^n - 1;\ y, f(x) \in \{0,1\}^m = 0,1,\ldots,2^m - 1,$$

where the first q-bits (quantum register) are the control and the second q-bits are the target. If we set the target q-bit to a one-q-bit superposition state, $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, we have

$$|x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \xrightarrow{U_{f(x)}} (-1)^{f(x)}|x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

Using this Deutsch-Jozsa QA and the Hadamard transformation H, we obtain

$$|0\rangle|1\rangle \xrightarrow{H^2} \xrightarrow{U_{f(x)}} \xrightarrow{H^2} (-1)^{f(0)} \underbrace{|f(0) \oplus f(1)\rangle}_{\boxed{Control}} |1\rangle$$

This transformations show that stating from the state $|0\rangle|1\rangle$, only one function evaluation suffices to obtain the state $|f(0) \oplus f(1)\rangle$ for the control that is $|0\rangle(|1\rangle)$ when $f(x)$ *is* constant (balanced) function solving the $n = 1$ Deutsch problem. This may be the simplest example of quantum parallelism, an important aspect of the power of quantum computation.

Remark. Recently, *Collins et all* (1998) showed, for f -controlled-NOT transformation, that if the target register in the input is restricted to the subspace spanned by $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, then there is no entanglement between the control and the target registers in the output, the state of the target does not change; therefore, the target can be redundant. In this refined QA, the transformation $U_{f(x)}$ is simply given by $|x\rangle \xrightarrow{U_{f(x)}} (-1)^{f(x)}|x\rangle$. Of course, this algorithm is experimentally applicable when we can directly create the phase $(-1)^{f(x)} = \pm 1$ for the state $|x\rangle$.

We will now give an algorithm to create arbitrary phase $e^{i\pi\phi(x)}$, rather than only ± 1, and to apply this algorithm to solve an extended Deutsch problem where the function maps

$$f(x): \{0,1\}^n \to \{0,1\}^m.$$

Generalized Deutsch problem: *Phase-creation algorithm*. Recalling that the input states for the target q-bit of the Deutsch-Jozsa QA is the Hadamard transform of $|1\rangle$, and that the Hadamard transform is the simplest (modulo 2) case of a QFT (see Appendix 1), we will replace this q-bit (the target) by an m -q-bit quantum register, $|u\rangle$ that is the QFT modulo 2^m of $|00\ldots01\rangle$:

$$\underbrace{|00\ldots01\rangle}_{m} \xrightarrow{F_{2^m}} |u\rangle = \frac{1}{\sqrt{2^m}} \sum_{y \in \{0,1\}^m} \exp\left\{\frac{i2\pi y}{2^m}\right\}|y\rangle.$$

Note that $|u\rangle$ is in fact unentangled, and is given by

$$|u\rangle = \frac{1}{\sqrt{2^m}} \left(|0\rangle + e^{\frac{i2\pi}{2}}|1\rangle\right) \otimes \left(|0\rangle + e^{\frac{i2\pi}{2^2}}|1\rangle\right) \otimes \cdots \otimes \left(|0\rangle + e^{\frac{i2\pi}{2^m}}|1\rangle\right)$$

Next, extending the definition of f -controlled-NOT transformation, we will define a unitary transformation $U_{\phi(x)}$ by a state-shift operation on each state $|y\rangle$ of the target register $|u\rangle$; y is shifted by an integer value $2^{m-1}\phi(x)$ that is determined by the state $|x\rangle$ of the control register:

$$\begin{array}{|c|c|c|} \hline |x\rangle|y\rangle & \xrightarrow{U_{\phi(x)}} & |x\rangle\left|y-2^{m-1}\phi(x)\bmod 2^m\right\rangle \\ \hline \phi(x) & \in & \frac{1}{2^m}\{0,1\}^m \\ \hline & = & 0,\frac{1}{2^{m-1}},\frac{2}{2^{m-1}},\ldots,\frac{2^m-1}{2^{m-1}} \\ \hline \end{array}, \quad x\in\{0,1\}^n, \quad y\in\{0,1\}^m.$$

Using abovementioned expressions, we obtain

$$\begin{array}{|c|c|c|} \hline |x\rangle|u\rangle & = & \frac{1}{\sqrt{2^m}}|x\rangle\sum_{y\in\{0,1\}^m}\exp\left\{\frac{i2\pi y}{2^m}\right\}|y\rangle \\ \hline & \xrightarrow{U_{\phi(x)}} & \frac{1}{\sqrt{2^m}}|x\rangle\sum_{y\in\{0,1\}^m}\exp\left\{\frac{i2\pi y}{2^m}\right\}\left|\left|y-2^{m-1}\phi(x)\bmod 2^m\right\rangle\right\rangle \\ \hline & = & \frac{1}{\sqrt{2^m}}\exp\{i\pi\phi(x)\}|x\rangle \\ & & \sum_{y\in\{0,1\}^m}\exp\left\{i2\pi\left[\frac{y}{2^m}-\frac{\phi(x)}{2}\right]\right\}\left|\left|y-2^{m-1}\phi(x)\bmod 2^m\right\rangle\right\rangle \\ \hline & = & \exp\{i\pi\phi(x)\}|x\rangle|u\rangle \\ \hline \end{array}$$

that shows that state-shift operations on states $|y\rangle$ of the target register $|u\rangle$ result in the creation of a phase $\exp\{i\pi\phi(x)\}$ for the control register $|x\rangle$, because of the property of the Fourier-transformed state of the target. This is an explicit expression of what *Cleve et all* (1998) proposed for the creation of arbitrary interference patterns.

If we put $m=1$ in the abovementioned expressions, we obtain the Deutsch-Jozsa QA, and if we put $n\geq 2,$ and $m=2,$ we obtain four phases, $1,i,-1,$ and $-i$, for $|x\rangle$ corresponding to $\phi(x)=0,\frac{1}{2},1,$ and $\frac{3}{2}$, respectively.

Remark. Note that since $2^{m-1}\phi(x)$ is an integer, the smallest phase created is $\exp\left\{i\frac{\pi}{2^{m-1}}\right\}$. We see that in the output the target register $|u\rangle$ does not entangle with the control register $|x\rangle$, and that the state of the target register does not change by the transformation $U_{\phi(x)}$. Therefore, just as in the Collins QA, the target register can also be redundant in this phase-creation algorithm. If we experimentally create the phase $\exp\{i\pi\phi(x)\}$ for the state $|x\rangle$, this algorithm is simply described as $|x\rangle\xrightarrow{U_{\phi(x)}}e^{i\pi\phi(x)}|x\rangle$.

Extended Deutsch problem. Here we will return to an extension of the Deutsch's problem: «*For an unknown given* 2^m*-valued function* $\phi(x),\phi(x):\{0,1\}^n\to\frac{1}{2^{m-1}}\{0,1\}^m, n\geq m$, *determine whether it is constant or balanced by evaluating it only once*». In this problem, the numbers of constant and balanced functions are 2^m and $\frac{2^n}{\left(2^{n-m}\right)^{2^m}}$, respectively. Following the procedure given by Deutsch and Jozsa to solve the Deutsch's

problem, where $n \geq 2$, we can prove that this extended problem, where $n \geq m \geq 2$, can also be solved if we use the phase-creation algorithm, as follows. Walsh transformation for an n-q-bit state $|v\rangle$ is given by

$$|v\rangle \xrightarrow{W} \frac{1}{\sqrt{2^n}} \sum_{w \in \{0,1\}^n} (-1)^{\bar{v} \cdot \bar{w}} |w\rangle,$$

where $\bar{v} \cdot \bar{w}$ is the sum modulo 2 of the bitwise products of v and w. This transformation is equivalent to performing a one-q-bit Hadamard transformation on each of the n q-bits individually. When we apply the phase-creation transformation to an equally weighted superposition state $\sum |x\rangle$ that is produced by the Walsh transformation, we obtain

$\underbrace{\lvert 00\ldots 0\rangle}_{n}$	$\xrightarrow{W}$	$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} \lvert x\rangle$
	$\xrightarrow{U_{\phi(x)}}$	$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} e^{i\pi\phi(x)} \lvert x\rangle$
	$\xrightarrow{W}$	$\frac{1}{2^n} \sum_{x,w \in \{0,1\}^n} e^{i\pi[\phi(x)+\bar{x}\cdot\bar{w}]} \lvert w\rangle$

The amplitude of the state $|00\ldots 0\rangle$ in the output is $\frac{1}{2^n} \sum_{x \in \{0,1\}^n} e^{i\pi\phi(x)}$ that is nonzero (either $1, i, -1,$ or -1 for the case of $m = 2$) if $\phi(x)$ is constant, whereas it is zero if $\phi(x)$ is balanced.

Remark. When we solve the extended Deutsch problem, we create phases $e^{i\pi\phi(x)}$ for equally weighted superposition states $|x\rangle$. Experimentally this process can be carried out either indirectly using an additional register $|u\rangle$, or, if possible, directly without using it.

Example. For extended Deutsch's problem where the phases are arbitrary, let us consider the simplest case of $n = m = 2$, where four functions are constant and 24 functions are balanced. Two typical examples of the balanced functions are

$\phi_1(00) = 0$	$\phi_1(01) = \frac{1}{2}$	$\phi_1(10) = 1$	$\phi_1(11) = \frac{3}{2}$
$\phi_2(00) = 0$	$\phi_2(01) = \frac{1}{2}$	$\phi_2(10) = \frac{3}{2}$	$\phi_2(11) = 1$

When we apply the transformation $U_{\phi(x)}$ of these functions to the superposition state that is the Walsh transform of $|00\rangle$, we obtain either an entangled state or a superposition state:

$(\lvert 0\rangle - \lvert 1\rangle) \otimes (\lvert 0\rangle + i\lvert 1\rangle)$	*for* U_{ϕ_1}
$\lvert 00\rangle + i\lvert 01\rangle - i\lvert 01\rangle - \lvert 11\rangle$	*for* U_{ϕ_2}

The matrices for these transformations are given by

$$U_{\phi_1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -i \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix},$$

$$U_{\phi_1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & -i & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \left[\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right].$$

If we note that:

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = ie^{-i\pi I_z} = ie^{-i\pi I_x} e^{-i\pi I_y}, \quad \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} = \frac{1+i}{\sqrt{2}} e^{-i\left(\frac{\pi}{2}\right)I_z} = \frac{1+i}{\sqrt{2}} e^{-i\left(\frac{\pi}{2}\right)I_x} e^{-i\left(\frac{\pi}{2}\right)I_y} e^{i\left(\frac{\pi}{2}\right)I_x},$$

and

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \frac{1+i}{\sqrt{2}} e^{-i\pi I_z S_z},$$

where I and S are spin operators for the two-spin system, we see that these transformations (and also all other transformations) consists of terms of the forms $\exp\{\pm i\pi I_\beta\}, \exp\left\{\pm i\frac{\pi}{2} I_\beta\right\} (\beta = x, y)$, and $\exp\{\pm i\pi I_\beta S_z\}$ (up to the overall phases).

This can be compared to the case of the Deutsch's problem ($n=2, m=1$), where transformations $U_{f(x)}$ consist of terms of the forms $\exp\{\pm i\pi I_\beta\}$ and $\exp\{\pm i2\pi I_\beta S_z\}$. In the case of $n=3$, if we denote the third spin by K, the terms of $U_{f(x)}$ are of the forms $\exp\{\pm i\pi I_z\}, \exp\left\{\pm i\frac{\pi}{2} I_z\right\}$ and $\exp\{-i2\pi I_z S_z K_z\}$, where the last one is equivalent to the following expression: $e^{-i\left(\frac{\pi}{2}\right)I_x} e^{-i\pi I_z K_z} e^{-i\left(\frac{\pi}{2}\right)I_y} e^{-i\pi I_z S_z} e^{i\left(\frac{\pi}{2}\right)I_y} e^{i\pi I_z K_z} e^{i\left(\frac{\pi}{2}\right)I_x}$, and, consequently, the terms of $U_{\phi(x)}$ are of the forms as small as $\exp\left\{\pm i\frac{\pi}{2^m} I_\beta\right\}$ and $\exp\left\{\pm i\frac{\pi}{2^{m-1}} I_z S_z\right\}$. In NMR, it is possible to create directly the phase $\exp\{i\pi\phi(x)\}$ for the states $|x\rangle$ that are in equally weighted superpositions. This means that indirect phase creation for use of an additional target register $|u\rangle$ in practice is unnecessary.

Quantum computational algorithm for generalized Deutsch-Jozsa problem. A more general formulation of the Deutsch-Jozsa problem consists in summations of the M th roots of unity that allow to distinguish constant functions from functions that are many to one and onto evenly spaced ranges. Mathematically, the Deutsch-Jozsa problem is to determine whether a Boolean function $f : \mathbb{Z}_{2^n} \to \mathbb{Z}_2$ is non-constant or non-balanced where f is said to be balanced if $f(x) = 0$ for exactly half of the input values. The Deutsch-Jozsa's QA can solve this problem by a single evaluation of f on quantum computer efficiently without error but that requires exhaustive search to solve it deterministically without error in a classical setting.

From quantum computing standpoint, this QA consists of the successive application of the operators $W_n \otimes I$, U_f and $I \otimes W_n$ to two quantum registers with their initial state being $|0^n\rangle \otimes W_1|1\rangle$ where W_j is the j-q-bit Walsh-Hadamard operator and U_f is the function-evaluation operator defined by $|x\rangle \otimes |y\rangle \to |x\rangle \otimes |y \oplus f(x)\rangle$.

Remark. Actually the original Deutsch-Jozsa QA can solve the generalized Deutsch-Jozsa problem by slightly modifying the initial state of the second register. For simplicity, we are assuming that N, K and M are powers of 2, i.e., $N = 2^n, K = 2^k$ and $M = 2^m$ for some positive integers n, k and m.

Remark. The required generalization is obtained by modifying the unitary transformation $|x\rangle \to (-1)^{f(x)}|x\rangle$ to $U_f : |x\rangle \to \omega_M^{f(x)}|x\rangle$ written in terms of the M -th roots of unity denoted

$$\omega_M = \exp\left\{\frac{2\pi i}{M}\right\}.$$

The general algorithm then becomes

$$\begin{array}{|l|c|l|}\hline W_n U_f W_n |0^n\rangle & = & \frac{1}{\sqrt{N}} W_n U_f \sum_{x=0}^{N-1} |x\rangle \\ \hline & = & \frac{1}{\sqrt{N}} W_n \sum_{x=0}^{N-1} \omega_M^{f(x)} |x\rangle \\ \hline & = & \sum_{y=0}^{N-1} \left\{ \frac{1}{N} \sum_{x=0}^{N-1} (-1)^{x\cdot y} \omega_M^{f(x)} \right\} |y\rangle \\ \hline \end{array} \quad (2)$$

The resulting state becomes $\frac{1}{N}\sum_{x,y=0}^{N-1}(-1)^{x\cdot y + f(x)}|y\rangle \otimes W_1|1\rangle$ where $x\cdot y$ stands for the scalar product modulo 2 in $\mathbb{Z}_{2^n}$, i.e., $x\cdot y \equiv \sum_{j=0}^{n-1} x_j y_j \pmod 2$ for $x = \sum_{j=0}^{n-1} x_j 2^j$ and $y = \sum_{j=0}^{n-1} y_j 2^j$ with $x_j, y_j \in \mathbb{Z}_2$. At this stage discarding the second one-q-bit register we perform a measurement on the first n-q-bit register and conclude that f is non-balanced if the outcome is $|0^n\rangle$ and that f is non-constant otherwise.

Example. Let S_y be the inner summation in the final state (2) of the last expression: $S_y = \frac{1}{N}\sum_{x=0}^{N-1}(-1)^{x\cdot y}\omega_M^{f(x)}$. If f is constant, then we obtain:

$$S_y = \frac{1}{N}\omega_M^{f(0)}\sum_{x=0}^{N-1}(-1)^{x\cdot y} = \begin{cases} 0 & when\ y \neq 0 \\ \omega_M^{f(0)} & when\ y = 0 \end{cases}$$

If f is evenly distributed, then for $y = 0$ we have: $S_0 = \frac{1}{K}\sum_{j=0}^{K-1}\omega_M^{j\mu+t} = \frac{1}{K}\omega_M^t\sum_{j=0}^{K-1}\omega_M^j = 0$. Thus if the outcome of the measurement is $|0^n\rangle$ then f is not evenly distributed and otherwise f is non-constant. Here, the properties of summations of the roots of unity allow a generalization of the Deutsch-Jozsa algorithm to distinguish between functions promised to be either constant or evenly distributed. Further, this algorithm requires only a single evaluation of f. When we employ a multi-q-bit oracle that performs a functional evaluation by $U_f : |x\rangle \otimes |y\rangle \to |x\rangle \otimes |y \oplus f(x)\rangle$, by slightly modifying the initial state of the auxiliary register in the original Deutsch-Jozsa algorithm we can solve the generalized Deutsch-Jozsa evenly distributed problem.

References

1. Lo H.-K., Popescu S. and Spiller T. (Eds). Introduction to quantum computing and information. – World Scientific Publ. Co. – 1998.
2. Gruska J. Quantum computing // Advanced Topics in Computer Science Series, McGraw-Hill Companies. – London, 1999.
3. Pittenberg A.O. An introduction to quantum computing and algorithms. – Progress in Computer Sciences and Applied Logic. – Birkhauser, 1999. – Vol. 19.
4. Berman G.P., Doolen G.D., Mainieri R. and Tsifrinovich V.I. Introduction to quantum computers // World Scientific Publ. Co. – 1999.

5. Ulyanov S.V., Ghisi F., Kurawaki I. and Litvintseva L.V. Simulation of quantum algorithms on classical computer. – Note del Polo Ricerca, Università degli Studi di Milano (Polo Didattico e di Ricerca di Crema). – Milan, 1999. – Vol. 32.

6. Nielsen M.A. and Chuang I.L. Quantum Computation and Quantum Information. – Cambridge Univ. Press, UK, 2000.

7. Hirvensalo M. Quantum computing // Natural Computing Series, Springer-Verlag. – Berlin, 2001.

8. Hardy Y. and Steeb W.-H. Classical and quantum computing with C++ and Java Simulations. – Birkhauser Verlag, Basel. – 2001.

9. Calude C.S. and Paun G. Computing with cells and atoms: An introduction to quantum, DNA and membrane computing. – N.Y.: Taylor&Francis, 2001.

10. Kitaev A.Yu., Shen A.H., Vyaly M.N. Classical and quantum computation. – N.Y.: AMS, 2002.

11. Brylinski F.K. and Chen G. (Eds). Mathematics of quantum computation // Computational Mathematics Series. – CRC Press Co, 2002.

12. Ulyanov S.V., Litvintseva L.V., Ulyanov I.S. and Ulyanov S.S. Quantum information and quantum computational intelligence: Quantum decision making and search algorithms // Note del Polo Ricerca, Università degli Studi di Milano (Polo Didattico e di Ricerca di Crema). – Milan, 2005. – Vol. 84-85.

13. Stenholm S. and Suominen K.-A. Quantum approach to informatics // Wiley- Interscience. J. Wiley&Sons, Inc. – 2005.

14. Marinescu D.C. and Marinescu G.M. Approaching quantum computing. – Pearson Prentice Hall, New Jersey, 2005.

15. Benenti G., Casati G., Strini G. Principles of quantum computation and information. – Singapore: World Scientific. – 2004. – Vol. I.; – 2007. – Vol. II.

16. Janzing D. Computer science approach to quantum control // Habilitation: Univ. Karlsruhe (TH) Publ. Germany. – 2006.

17. Jaeger G. Quantum Information: An Overview. – N.Y.: Springer Verlag, 2007.

18. Kaye P., Laflamme R. and Mosca M. An introduction to quantum computing. – N.Y.: Oxford University Press, 2007.

19. McMahon D. Quantum computing explained // Wiley Interscience. A J. Wiley Sons, Inc. – 2008.

20. Lanzagorta M. and Uhlmann J. Quantum computer science // Morgan & Claypool Publ. – Series: SYNTHESIS LECTURES ON QUANTUM COMPUTING (Lecture #2), 2009.

21. Nakahara M. and Ohmi T. Quantum computing: From Linear Algebra to Physical Realizations // Taylor & Francis. – 2008.

22. Chen G., Kauffman L., and Lomonaco S. J. Mathematics of Quantum Computation and Quantum Technology – N.Y.: Chapman Hall/CRC (Applied Mathematics and Nonlinear Science Series), 2008.

23. Chen G., Church D.A., Englert B.-G., Henkel C., Rohwedder B., Scully M.O. and Zubairy M.S. Quantum Computing Devices: Principles, Designs, and Analysis. – N.Y.: Chapman Hall/CRC (Applied Mathematics and Nonlinear Science Series), 2008.

24. McMahon D. Quantum computing explained. – N.J.: John Wiley & Sons, 2008.

25. Yanofsky N.S. and Mannucci M.A. Quantum Computing for Computer Scientists. – UK: Cambridge University Press, 2008.

26. Chen G. and Diao. Mathematical Theory of Quantum Computation. – N.Y.: Chapman Hall/CRC (Applied Mathematics and Nonlinear Science Series), 2009.

27. Kholevo A.S. Quantum systems, channels, and information. – M.: МЦНМО. – 2010 (in Russian).

28. Lavor C., Manssur L.R.U. and Portugal R. Grover's algorithm: Quantum database search // arXiv:quant-ph/0301079v1 16 Jan 2003.

29. Lomonaco S.J. (Jr) A lecture on Grover's quantum search algorithm (Version 1.1) // arXiv:quant-ph/0010040v2 18 Oct 2000.

30. Lavor C., Manssur L.R.U. and Portugal R. Shor's algorithm for factoring large integers // arXiv: quant-ph/0303175v1 29 Mar 2003.

31. Galindo A. and Martin-Delgado M.A. Information and computation: Classical and quantum aspects // Review of Modern Physics. – 2002. – Vol. 74. – № 2. – Pp. 347-423.

32. Batty M., Braunstein S.L., Duncan A.J. and Rees S. Quantum algorithms in group theory // arXiv: quant-ph/0310133v1, 21 Oct 2003. – P. 52.

33. Quantum Algorithms: Shor's algorithm, Grover's algorithm, Quantum logic, Quantum algorithm, Quantum Fourier transform, Deutsch-Jozsa algorithms. – Books LLC. – 2010.

34. Rieffel E. G. and Polak W. H. Quantum Computing: A gentle introduction. – B.: The MIT Press, 2011.

35. Ohya M. and Volovich I. Mathematical foundations of quantum information and computation and its applications to nano- and bio-systems. – N.Y.: Springer Verlag, 2011.