УДК 004.415.2, 004.588

ALGORITHMIC REPRESENTATION OF THE QUANTUM OPERATORS AND FAST QUANTUM ALGORITHMS

Barchatova Irina[1], Hagiwara Tahiko[2], Ulyanov Sergey[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]*PhD, professor;*
Yamaha Motor Europe N.V.;
Polo Didattico e di Ricerca di Crema;
Via Bramante, 65-26013, Crema (CR), Italy.

[3]*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.

The simplest technique for simulating a quantum algorithm (QA) is described and is based on the direct matrix representation of the quantum operators. This approach is stable and precise, but it requires allocation of operator's matrices in the computer's memory. Since the size of the operators grows exponentially, this approach is useful for simulation of QAs with a relatively small number of qubits (e.g., approximately 11 qubits on a typical desktop computer). Using this approach it is relatively simple to simulate the operation of a QA and to perform fidelity analysis. Two special search algorithms: Shor's and Grover's QA are described.

Keywords: quantum operators, fast quantum algorithms, algorithmic representation.

АЛГОРИТМИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ КВАНТОВЫХ ОПЕРАТОРОВ И КВАНТОВЫХ АЛГОРИТМОВ

Бархатова Ирина Александровна[1], Хагивара Тахико[2], Ульянов Сергей Викторович[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 наук, профессор;*
Yamaha Motor Europe N.V.;
Поло дидаттико, Крема, факультет информационных технологий;
Италия, Крема, Via Bramante, 65-26013.

[3]*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.

Описана простая техника моделирования квантового алгоритма, основанная на прямом матричном представлении квантовых операторов. Такой подход является устойчивым и точным, но требует огромного объема оперативной памяти компьютера для вычисления матричного представления квантовых операторов. Так как простарнственно-временная размерность операторов возрастает экпоненциально, то такой подход может быть использован для моделирования кванто-

вых алгоритмов с относительно малым числом входных кубитов (т.е. примерно 11 кубитов для типовой конфигурации ПК). Используя этот подход, можно моделировать относительно просто квановые алгоритмы и достигать высокого качества результата. Даны примеры моделирования двух поисковых квантовых алгоритмов: алгоритм Шора и алгоритм Гровера.

Ключевые слова: квантовые операторы, быстрые квантовые алгоритмы, алгоритмическое представление.

Introduction

In one embodiment, a more efficient fast QA simulation technique is based on computing all or part of the operator matrices on an as needed current computational basis. Using this technique, it is possible to avoid storing all or part of the operator matrices. In this case, the number of qubits that can be simulated (e.g., the number of input qubits, or the number of qubits in the system state register) is affected by: (i) the exponential growth in the number of operations required to calculate the result of the matrix products; and (ii) the size of the state vector that is allocated in computer memory.

In one embodiment, using this approach it is reasonable to simulate up to 19 or more qubits on typical desktop computer, and even more on a system with vector architecture.

Due to particularities of the memory addressing and access processes in a typical desktop computer (such as, for example, a Pentium-based Personal Computer), when the number of qubits is relatively small, the compute-on-demand approach tends to be faster than the direct storage approach. The compute-on-demand approach benefits from a study of the quantum operators, and their structure so that the matrix elements can be computed more efficiently.

The study portion of the compute-on-demand approach can, for some QAs lead to a problem-oriented approach based on the QA structure and state vector behavior [1 – 3].

For example, in Grover's quantum search algorithm (QSA), the state vector always has one of the two different values: (i) one value corresponds to the probability amplitude of the answer; and (ii) the second value corresponds to the probability amplitude of the rest of the state vector. Using this assumption, it is possible to configure the algorithm using these two different values, and to efficiently simulate Grover's QSA.

In this case, the primary limit is a representation of the floating-point numbers used to simulate the actual values of the probability amplitudes. After the superposition operation, these probability amplitudes are very small ($\frac{1}{\sqrt{2^n}}$).

Thus, it is possible to simulate Grover's QSA with this approach simulating 1024 qubits or more without termination condition calculation and up to 64 qubits or more with termination condition estimation based on Shannon entropy.

Other QAs do not necessarily reduce to just two values. For those algorithms that reduce to a finite number of values, the techniques used to simplify the Gover's QSA can be used, but the maximum number of input qubits that can be simulated will tend to be smaller, because the probability amplitudes of other algorithms have relatively more complicated distributions. Introduction of an external excitation can decrease the possible number of qubits for some algorithms.

In some algorithms, the entanglement and interference operators can be bypassed (or simplified), and the output computed based only on a superposition of the initial states (and deconstructive interference of the final output patterns) representing the state of the designed schedule of control gains. For example, a particular case of Deutsch-Jozsa's and Simon algorithms can be made entanglement free by using pseudo-pure quantum states.

The disclosure that follows begins with a comparative analysis of the temporal complexity of several representative QAs. That analysis is followed by an introduction of the generalized approach in QA simulation and algorithmic representation of quantum operators. Subsequent portions describe the structure representation of the QAs applicable to low level programming on classical computer (PC), generalizations of the approaches and introduction of the general QA simulation tool based on fast problem-oriented QAs.

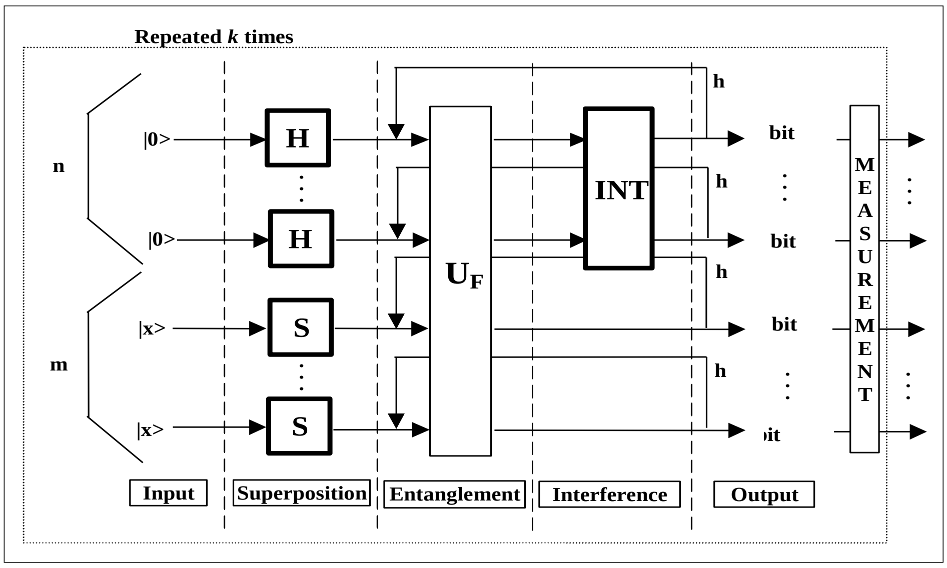
The simulation techniques are then applied to a quantum control algorithm.

The matrix-based approach can be efficiently realized for a small number of input qubits. The matrix approach is used above as a useful tool to illustrate complexity issues associated with QA simulation on classical computer.

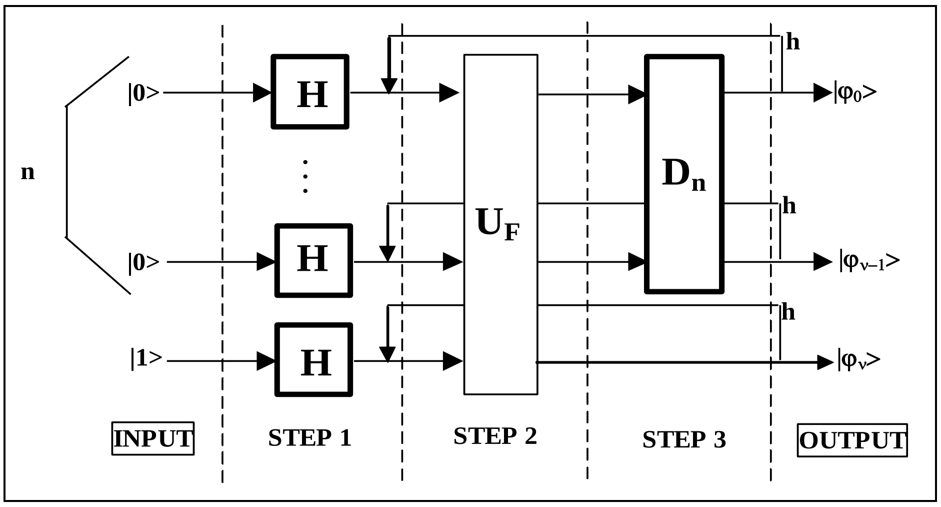
1. Structure of QA gate system design

As shown in Fig. 1 (a), a QA simulation can be represented as a generalized representation of a QA as a set of sequentially-applied smaller quantum gates. From the structural point of view, each QA is based on a particular set of quantum gates, but generally speaking each particular set can be divided into superposition operators, entanglement operators, and interference operators.

This division into superposition operators, entanglement operators, and interference operators permits a generalization of the design of a simulation and allows creation of a classical tool to simulate QAs. Moreover, local optimization of QA components according to specific hardware realization makes it possible to develop appropriate hardware accelerators for QA simulation using classical gates.



(a)



(b)

Figure 1. (a) Circuit representation of QA; (b) Quantum circuit of Grover's QSA

2. Generalized approach in QA simulation

In general, any QA can be represented as a circuit of smaller quantum gates as shown in Figs 1 (a, b). The circuit shown in the Fig. 1 (a) is divided into five general layers: (i) input; (ii) superposition; (iii) entanglement; (iv) interference; and (v) output.

Layer 1: *Input.* The quantum state vector is set up to an initial value for this concrete algorithm. For example, input for Grover's QSA is a quantum state $|\phi_0\rangle$ described as a tensor product

$$\begin{aligned}|\phi_0\rangle &= a_1|0\rangle\otimes\cdots\otimes|0\rangle\otimes|0\rangle + a_2|0\rangle\otimes\cdots\otimes|0\rangle\otimes|1\rangle + a_3|0\rangle\otimes\cdots\otimes|1\rangle\otimes|0\rangle \\ &+\cdots+a_n|1\rangle\otimes\cdots\otimes|1\rangle\otimes|1\rangle = 1|0\rangle\otimes\cdots\otimes|0\rangle\otimes|1\rangle = |0\cdots01\rangle,\end{aligned} \qquad (1)$$

where $|0\rangle=\begin{pmatrix}1\\0\end{pmatrix}$; $|1\rangle=\begin{pmatrix}0\\1\end{pmatrix}$; $\otimes$ denotes Kronecker tensor product operation.

Such a quantum state can be presented as shown on the Fig. 2 (a).

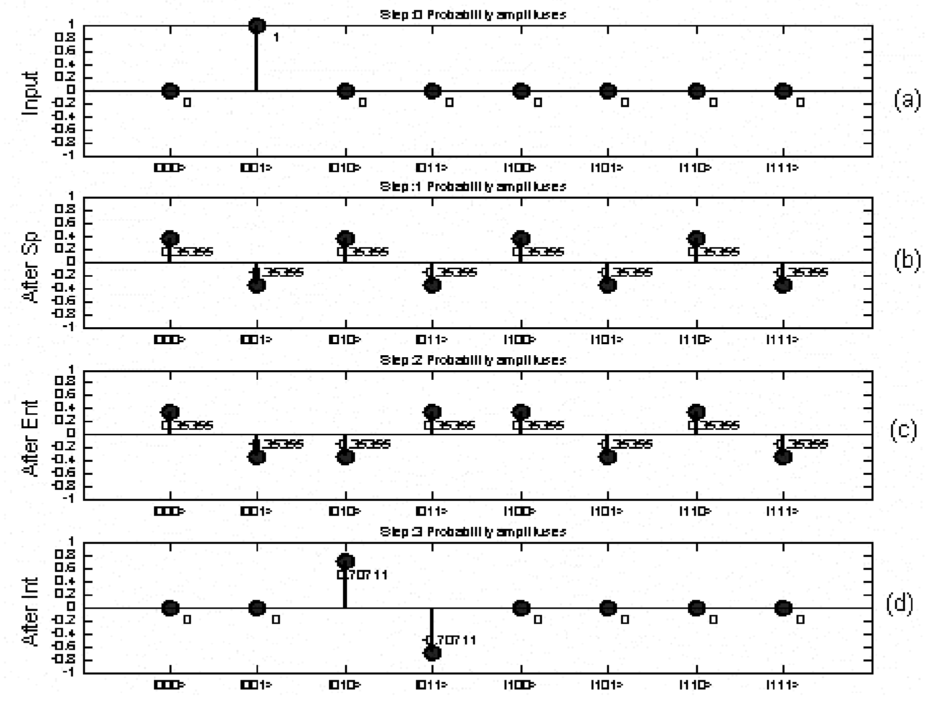


Figure 2. Dynamics of Grover's QSA probability amplitudes of state vector on each algorithm step

The coefficients a_i in the Eq. (1) are called probability amplitudes. Probability amplitudes can take negative and/or complex values. However, the probability amplitudes must obey the following constraint:

$$\sum_i a_i^2 = 1. \qquad (2)$$

The actual probability of the arbitrary quantum state $a_i|i\rangle$ to be measured is calculated as a square of its probability amplitude value $p_i = |a_i|^2$.

Layer 2: *Superposition*. The state of the quantum state vector is transformed by the Walsh-Hadamard operator so that probabilities are distributed uniformly among all basis states. The result of the superposition layer of Grover's QSA is shown in Fig. 2 (b) as a probability amplitude representation, and also in Fig. 3 (b) as a probability representation.

Layer 3: *Entanglement*. Probability amplitudes of the basis vector corresponding to the current problem are flipped while rest basis vectors left unchanged. Entanglement is typically provided by controlled-NOT

(CNOT) operations. Figures 2 (c) and 3 (c) show results of entanglement from the application of the operator to the state vector after superposition operation. An entanglement operation does not affect the probability of the state vector to be measured. Rather, entanglement prepares a state, which cannot be represented as a tensor product of simpler state vectors. For example, consider state ϕ_1 shown in the Fig. 2 (b) and state ϕ_2 presented on the Fig. 2 (c):

$$\begin{aligned}\phi_1 &= 0.35355\left(|000\rangle - |001\rangle + |010\rangle - |011\rangle + |100\rangle - |101\rangle + |110\rangle - |111\rangle\right)\\ &= 0.35355\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right)\left(|0\rangle - |1\rangle\right)\end{aligned}$$

$$\begin{aligned}\phi_2 &= 0.35355\left(|000\rangle - |001\rangle - |010\rangle + |011\rangle + |100\rangle - |101\rangle + |110\rangle - |111\rangle\right)\\ &= 0.35355\left(|00\rangle - |01\rangle + |10\rangle + |11\rangle\right)|0\rangle - 0.35355\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right)|1\rangle\end{aligned}$$

As shown above, the description of state ϕ_1 can be presented as a tensor product of simpler states, while state ϕ_2 (in the measurement basis $\{|0\rangle, |1\rangle\}$) cannot.

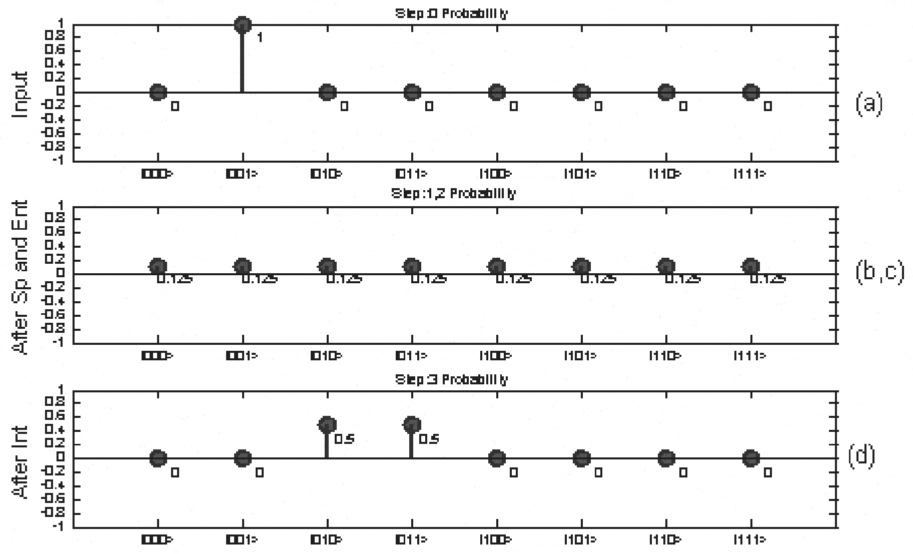


Figure 3. Dynamics of Grover's QSA probabilities of state vector on each algorithm step

Layer 4: *Interference*. Probability amplitudes are inverted about the average value. As a result, the probability amplitude of states «marked» by entanglement operation will increase.

Figs 2 (d) and 3 (d) show the results of interference operator application.

Fig. 2 (d) shows probability amplitudes and Fig. 3 (d) shows probabilities.

Layer 5: *Output*. The output layer provides the measurement operation (extraction of the state with maximum probability), followed by interpretation of the result. For example, in the case of Grover's QSA, the required index is coded in the first n bits of the measured basis vector.

Since the various layer of the QA are realized by unitary quantum operators, simulation of quantum operators depends on simulation of such unitary operators. Thus, in order to develop an efficient, simulation, it is useful to understand the nature of the QAs basic quantum operators.

3. Basic QA operators

The superposition, entanglement and interference operators are now considered from the simulation viewpoint. In this case, the superposition operators and the interference operators have more complicated structure and differ from algorithm to algorithm. Thus, it is first useful to consider the entanglement operators, since they have a similar structure for all QAs, and differ only by the function being analyzed.

In general, the superposition operator is based on the combination of the tensor products Hadamard H operators: $H=\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$ with identity operator I : $I=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$.

Remark. The simulation system of quantum computation is based on quantum algorithm gates (QAG). The design process of QAG includes the matrix design form of three quantum operators: superposition *(Sp)*, entanglement (U_F) and interference *(Int)*. In general form, the structure of a QAG can be described as follows:

$$QAG=\left[\left(Int\otimes{}^{n}I\right)\cdot U_F\right]^{h+1}\cdot\left[{}^{n}H\otimes{}^{m}S\right],$$

where I is the identity operator; the symbol $\otimes$ denotes the tensor product; S is equal to I or H and dependent on the problem description. One portion of the design process in QAG is the type-choice of the entanglement problem-dependent operator U_F that physically describes the qualitative properties of the function f .

The Hadamard Transform creates the superposition on classical states, and quantum operators such as *CNOT* create robust entangled states. The Quantum Fast Fourier Transform (QFFT) produces interference.

For most QAs the superposition operator can be expressed as

$$Sp=\left(\bigotimes_{i=1}^{n}H\right)\otimes\left(\bigotimes_{i=1}^{m}S\right), \tag{3}$$

where n and m are the numbers of inputs and of outputs respectively. The operator S depends on the algorithm and can be either the Hadamard operator H or the identity operator I . The numbers of outputs m as well as structures of the corresponding superposition and interference operators are presented in Table 1 for different QAs.

Table 1.Parameters of superposition and interference operators of main quantum algorithms

Algorithm	**Superposition**	m	**Interference**
Deutsch's	$H\otimes I$	1	$H\otimes H$
Deutsch-Jozsa's	${}^{n}H\otimes H$	1	${}^{n}H\otimes I$
Grover's	${}^{n}H\otimes H$	1	$D_n\otimes I$
Simon's	${}^{n}H\otimes{}^{n}I$	n	${}^{n}H\otimes{}^{n}I$
Shor's	${}^{n}H\otimes{}^{n}I$	n	$QFT_n\otimes{}^{n}I$

Superposition and interference operators are often constructed as tensor powers of the Hadamard operator, which is called the Walsh-Hadamard operator. Elements of the Walsh-Hadamard operator can be obtained as

$$\left[{}^{n}H\right]_{i,j}=\frac{(-1)^{i*j}}{2^{n/2}}\left[{}^{n-1}H\right]=\frac{1}{2^{n/2}}\begin{pmatrix}{}^{(n-1)}\mathbf{H} & {}^{(n-1)}\mathbf{H}\\ {}^{(n-1)}\mathbf{H} & -{}^{(n-1)}\mathbf{H}\end{pmatrix}, \tag{4}$$

where $i=0,1,\ j=0,1,\ \mathbf{H}$ denotes Hadamard matrix of order 2.

The rule in Eq. (4) provides way to speed up of the classical simulation of the Walsh-Hadamard operators, because the elements of the operator can be obtained by the simple replication described in Eq. (4) from the elements of the ${}^{n-1}H$ order operator. For example, consider the superposition operator of Deutsch's algorithm $n=1,\ m=1,\ S=I$:

$$[Sp]^{Deutsch}_{i,j}=\frac{(-1)^{i*j}}{2^{1/2}}\otimes I=\frac{1}{\sqrt{2}}\begin{pmatrix}(-1)^{0*0}I & (-1)^{0*1}I\\ (-1)^{1*0}I & (-1)^{1*1}I\end{pmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}I & I\\ I & -I\end{bmatrix} \tag{5}$$

As a further example, consider the superposition operator of Deutsch-Jozsa's and of Grover's algorithm, for the case $n=2,\ m=1,\ S=H$:

$$[Sp]^{Deutsch-Jozsa's,Grover's} = {}^2H\otimes H = \left(\frac{1}{\sqrt{8}}\right)^3\mathbf{H} = \frac{1}{\sqrt{8}}\begin{pmatrix} {}^2\mathbf{H} & {}^2\mathbf{H} \\ {}^2\mathbf{H} & -{}^2\mathbf{H}\end{pmatrix} = \frac{1}{\sqrt{8}}\begin{pmatrix} \mathbf{H} & \mathbf{H} & \mathbf{H} & \mathbf{H} \\ \mathbf{H} & -\mathbf{H} & \mathbf{H} & -\mathbf{H} \\ \mathbf{H} & \mathbf{H} & -\mathbf{H} & -\mathbf{H} \\ \mathbf{H} & -\mathbf{H} & -\mathbf{H} & \mathbf{H}\end{pmatrix}. \tag{6}$$

For yet another example, the superposition operator of Simon's and of Shor's algorithms $n=2, m=2, S=I$ can be expressed as:

$$[Sp]^{Simon,Shor}_{i,j} = {}^2H\otimes {}^2I = \frac{1}{2}\begin{pmatrix} (-1)^{0*0}\mathbf{H} & (-1)^{1*0}\mathbf{H} \\ (-1)^{1*0}\mathbf{H} & (-1)^{1*1}\mathbf{H}\end{pmatrix}\otimes {}^2I =$$

$$= \frac{1}{2}\begin{pmatrix} \mathbf{H} & \mathbf{H} \\ \mathbf{H} & -\mathbf{H}\end{pmatrix}\otimes {}^2I = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1\end{pmatrix}\otimes {}^2I = \frac{1}{2}\begin{pmatrix} {}^2I & {}^2I & {}^2I & {}^2I \\ {}^2I & -{}^2I & {}^2I & -{}^2I \\ {}^2I & {}^2I & -{}^2I & -{}^2I \\ {}^2I & -{}^2I & -{}^2I & {}^2I\end{pmatrix}$$

Interference operators are calculated for each algorithm according to the parameters listed in Table 1. The interference operator is based on the interference layer of the algorithm, which is different for various algorithms, and from the measurement layer, which is the same or similar for most algorithms and includes the m th -tensor power of the identity operator.

The interference operator of Deutsch's algorithm includes the tensor product of two Hadamard transformations, and can be calculated using Eq. (4) with $n=2$ as:

$$\left[Int^{Deutsch}\right]_{i,j} = {}^2H = \frac{(-1)^{i*j}}{2^{2/2}} = \frac{1}{2}\begin{pmatrix} (-1)^{0*0}\mathbf{H} & (-1)^{0*1}\mathbf{H} \\ (-1)^{1*0}\mathbf{H} & (-1)^{1*1}\mathbf{H}\end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1\end{pmatrix} \tag{7}$$

In Deutsch's algorithm, the Walsh-Hadamard transformation in the interference operator is used also for the measurement basis.

The interference operator of Deutsch-Jozsa's algorithm includes the tensor product of the n th power of the Walsh-Hadamard operator with an identity operator. In general form, the block matrix of the interference operator of Deutsch-Jozsa's algorithm can be written as from the n-1 order matrix as:

$$\left[Int^{Deutsch-Jozsa's}\right] = {}^nH\otimes I = \frac{1}{2^{n/2}}\begin{pmatrix} {}^{(n-1)}\mathbf{H} & {}^{(n-1)}\mathbf{H} \\ {}^{(n-1)}\mathbf{H} & -{}^{(n-1)}\mathbf{H}\end{pmatrix}\otimes I, \quad \text{where } \mathbf{H} = \begin{pmatrix} 1 & 1 \\ 1 & -1\end{pmatrix}, \tag{8}$$

Interference operator of Deutsch-Jozsa's algorithm $n=2, m=1$:

$$\left[Int^{Deutsch-Jozsa's}\right] = {}^2H\otimes I = \frac{1}{2}\begin{pmatrix} \mathbf{H} & \mathbf{H} \\ \mathbf{H} & -\mathbf{H}\end{pmatrix}\otimes I = \frac{1}{2}\begin{pmatrix} I & I & I & I \\ I & -I & I & -I \\ I & I & -I & -I \\ I & -I & -I & I\end{pmatrix}.$$

The interference operator of Grover's algorithm can be written as a block matrix of the following form:

$$\left[Int^{Grover}\right]_{i,j} = D_n\otimes I = \left(\frac{1}{2^{n/2}} - {}^nI\right)\otimes I = \left(-1 + \frac{1}{2^{n/2}}\right)\otimes I\Big|_{i=j},$$

$$\left(\frac{1}{2^{n/2}}\right)\otimes I\Bigg|_{i\neq j} = \frac{1}{2^{n/2}}\begin{cases}-I, i=j\\ I, i\neq j\end{cases}, \qquad (9)$$

where $i=0,...,2^n-1, j=0,...,2^n-1$, D_n refers to diffusion operator:

$$\left[D_n\right]_{i,j}=\frac{(-1)^{1\, AND\, (i=j)}}{2^{n/2}}.$$

For example, the interference operator for Grover's QSA, when $n=2, m=1$ is:

$$\left[Int^{Grover}\right]_{i,j}=D_2\otimes I=\left(\frac{1}{2^{2/2}}\ -{}^2I\right)\otimes I=\left(-1+\frac{1}{2}\right)\otimes I\Bigg|_{i=j},$$

$$\left(\frac{1}{2}\right)\otimes I\Bigg|_{i\neq j}=\begin{pmatrix}\left(-1+\frac{1}{2}\right)I & \frac{1}{2}I & \frac{1}{2}I & \frac{1}{2}I\\ \frac{1}{2}I & \left(-1+\frac{1}{2}\right)I & \frac{1}{2}I & \frac{1}{2}I\\ \frac{1}{2}I & \frac{1}{2}I & \left(-1+\frac{1}{2}\right)I & \frac{1}{2}I\\ \frac{1}{2}I & \frac{1}{2}I & \frac{1}{2}I & \left(-1+\frac{1}{2}\right)I\end{pmatrix}==\frac{1}{2}\begin{pmatrix}-I & I & I & I\\ I & -I & I & I\\ I & I & -I & I\\ I & I & I & -I\end{pmatrix} \qquad (10)$$

As the number of qubits increases, the gain coefficient will become smaller. The dimension of the matrix increases according to 2^n, but each element can be extracted using Eq. (9), without allocation of the entire operator matrix.

Remark. Since $D_nD_n^*=I$, D_n is unitary and is therefore a possible quantum state transformation. While the matrix D_n is clearly unitary it can to have the decomposition form $D_n=-H_nR_n^1H_n$, where $R_n^1\left[i,j\right]=0$, if $i\neq j$, $R_1^1\left[1,1\right]=-1$ and $R_n^1\left[i,i\right]=+1$, if $1<i\leq N$.

In concrete form the operator D_n (*diffusion – inversion about average*) in Grover algorithm is decomposed as

D_n	$=$	$\frac{1}{\sqrt{2^n}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}^{\otimes n}\cdot\begin{pmatrix}-1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & \ddots & 0\\ 0 & 0 & 0 & 1\end{pmatrix}^{\otimes n}\cdot\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}^{\otimes n}$

and can be accomplished with $O(n)=O(\log(N))$ quantum gates. It means that from the viewpoint of efficient computation the form as in Eq. (9) is more preferable.

The interference operator of Simon's algorithm is prepared in the same manner as the superposition (as well as superposition operators of Shor's algorithm) and can be described as follows from Eq. (3) and Eq. (6):

$$\left[Int^{Simon}\right]_{(i,j)}={}^nH\otimes{}^mI=\frac{(-1)^{(i*j)}}{2^{n/2}}{}^{(n-1)}\mathbf{H}\otimes{}^mI,\quad \text{where } \mathbf{H}=\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}.$$

In general, the interference operator of Simon's algorithm coincides with the interference operator of Deutsch-Jozsa's algorithm Eq. (8), but for each block of the operator matrix includes ~~m~~ tensor products of the identity operator.

The interference operator of Shor's algorithm uses the Quantum Fourier Transformation operator (QFT), calculated as:

$$\left[QFT_n\right]_{i,j}=\frac{1}{2^{n/2}}e^{J(i*j)\frac{2\pi}{2^n}},\qquad(11)$$

where $J=\sqrt{-1}$, $i=0,...,2^n-1$ and, $j=0,...,2^n-1$.

When $n=1$ then:

$$QFT_n\Big|_{n=1}=\frac{1}{2^{\frac{1}{2}}}\begin{pmatrix}e^{J*(0*0)2\pi/2^1} & e^{J*(0*1)2\pi/2^1}\\ e^{J*(1*0)2\pi/2^1} & e^{J*(1*1)2\pi/2^1}\end{pmatrix}=\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}=H\qquad(12)$$

Eq. (11) can also be presented in harmonic form using the Euler formula:

$$\left[QFT_n\right]_{i,j}=\frac{1}{2^{n/2}}\left(\cos\left((i*j)\frac{2\pi}{2^n}\right)+J\sin\left((i*j)\frac{2\pi}{2^n}\right)\right)\qquad(13)$$

For some applications, the harmonic form of Eq. (13) is preferable.

General case. For most algorithms, the superposition operator can be expressed as: $Sp=\left(\bigotimes_{i=1}^{k_1}H\right)\otimes\left(\bigotimes_{i=1}^{k_2}S\right)$, where k_1 and k_2 are the numbers of the inclusions of H and of S into the corresponding tensor products. Values of k_1,k_2 depend on the concrete algorithm and can be obtained from Table 2.

Table 2. Parameters of superposition and of interference operators of QAs

Algorithm	k_1	k_2	S	Interference
Deutsch's	1	1	I	$H\otimes H$
Deutsch-Jozsa's	$n-1$	1	H	${}^{k_1}H\otimes I$
Grover's	$n-1$	1	H	$D_{k_1}\otimes I$
Simon's	$n/2$	$n/2$	I	${}^{k_1}H\otimes^{k_2}I$
Shor's	$n/2$	$n/2$	I	$QFT_{k_1}\otimes^{k_2}I$

Operator S, depending on the algorithm, may be the Walsh-Hadamard operator H or the identity operator I. It is convenient to automate the process of the calculation of the tensor power of the Walsh-Hadamard operator as follows:

$$\left[{}^nH\right]_{i,j}=\frac{(-1)^{i*j}}{2^{n/2}}=\frac{1}{2^{n/2}}\begin{cases}1,\text{ if } i*j \text{ is even}\\ \text{-1, if } i*j \text{ is odd}\end{cases}\qquad(14)$$

where $i=0,1,...,2^n$, $j=0,1,...,2^n$. The tensor power of the identity operator can be calculated as

$$\left[{}^nI\right]_{i,j}=1|_{i=j}\;0|_{i\neq j},\qquad(15)$$

where $i=0,1,...,2^n$, $j=0,1,...,2^n$. Then any superposition operator can be presented as a block matrix of the following form:

$$\left[Sp\right]_{i,j}=\frac{(-1)^{i+j}}{2^{k_1/2}}\otimes{}^{k_2}S,\qquad(16)$$

where $i=0,...,2^{k_1}-1$, $j=0,...,2^{k_1}-1$ denote the blocks; ${}^{k_2}S$ is a k_2 tensor power of the corresponding operator. In this case n denotes the total number of qubits in the algorithm, including measurement qubits, and qubits necessary for encoding of the function. The actual number of input bits in this case is k_1. The actual number of output bits in this case is k_2. Operators used as S are presented in the Table 2 for all typical models of QAs.

Let us consider examples.

For the superposition operator of Deutsch's algorithm: $n=2, k_1=1, k_2=1, S=I$:

$$\left[Sp\right]^{Deutsch}_{i,j}=\frac{(-1)^{i+j}}{2^{1/2}}\otimes I=\frac{1}{\sqrt{2}}\begin{pmatrix}(-1)^{0*0}I & (-1)^{0*1}I\\ (-1)^{1*0}I & (-1)^{1*1}I\end{pmatrix}=\frac{1}{\sqrt{2}}\begin{bmatrix}I & I\\ I & -I\end{bmatrix} \quad (17)$$

The superposition operator of Deutsch-Jozsa's and of Grover's algorithm is, $n=3, k_1=2, k_2=1, S=H$:

$$\left[Sp\right]^{Deutsch-Jozsa's,Grover's}_{i,j}=\frac{(-1)^{i+j}}{2^{2/2}}\otimes H=$$
$$=\frac{1}{2}\begin{pmatrix}(-1)^{0*0}H & (-1)^{0*1}H & (-1)^{0*2}H & (-1)^{0*3}H\\ (-1)^{1*0}H & (-1)^{1*1}H & (-1)^{1*2}H & (-1)^{1*3}H\\ (-1)^{2*0}H & (-1)^{2*1}H & (-1)^{2*2}H & (-1)^{2*3}H\\ (-1)^{3*0}H & (-1)^{3*1}H & (-1)^{3*2}H & (-1)^{3*3}H\end{pmatrix}=\frac{1}{2}\begin{pmatrix}H & H & H & H\\ H & -H & H & -H\\ H & H & -H & -H\\ H & -H & -H & H\end{pmatrix} \quad (18)$$

The superposition operator of Simon's and of Shor's algorithms are, $n=4, k_1=2, k_2=2, S=I$:

$$\left[Sp\right]^{Simon,Shor}_{i,j}=\frac{(-1)^{i+j}}{2^{2/2}}\otimes^2 I,$$
$$=\frac{1}{2}\begin{pmatrix}(-1)^{0*0}({}^2I) & (-1)^{0*1}({}^2I) & (-1)^{0*2}({}^2I) & (-1)^{0*3}({}^2I)\\ (-1)^{1*0}({}^2I) & (-1)^{1*1}({}^2I) & (-1)^{1*2}({}^2I) & (-1)^{1*3}({}^2I)\\ (-1)^{2*0}({}^2I) & (-1)^{2*1}({}^2I) & (-1)^{2*2}({}^2I) & (-1)^{2*3}({}^2I)\\ (-1)^{3*0}({}^2I) & (-1)^{3*1}({}^2I) & (-1)^{3*2}({}^2I) & (-1)^{3*3}({}^2I)\end{pmatrix}=\frac{1}{2}\begin{pmatrix}{}^2I & {}^2I & {}^2I & {}^2I\\ {}^2I & -{}^2I & {}^2I & -{}^2I\\ {}^2I & {}^2I & -{}^2I & -{}^2I\\ {}^2I & -{}^2I & -{}^2I & {}^2I\end{pmatrix}. \quad (19)$$

The interference blocks implement the interference operator which, in general, is different for all algorithms. By contrast, the measurement part tends to be the same for most of the algorithms. The interference blocks compute the k_2 tensor power of the identity operator.

Interference operator of Deutsch's algorithm. This interference operator of Deutsch's algorithm is a tensor product of two Walsh-Hadamard transformations, and can be calculated in general form using Eq. (14) with $n=2$:

$$\left[Int^{Deutsch}\right]_{i,j}={}^2H=\frac{(-1)^{i*j}}{2^{2/2}}=\frac{1}{2}\begin{pmatrix}1 & 1 & 1 & 1\\ 1 & -1 & 1 & -1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\end{pmatrix}. \quad (20)$$

Note that in Deutsch's algorithm, the Walsh-Hadamard transformation in interference operator is used also for the measurement basis.

Interference operator of Deutsch-Jozsa's algorithm. The interference operator of Deutsch-Jozsa's algorithm is a tensor product of k_1 power of the Walsh-Hadamard operator with an identity operator. In general form, the block matrix of the interference operator of Deutsch-Jozsa's algorithm can be written as:

$$\left[Int^{Deutsch-Jozsa's}\right]_{i,j}=\frac{(-1)^{i*j}}{2^{\frac{k_1}{2}}}\otimes I, \quad (21)$$

where $i=0,...,2^{k_1}-1, j=0,...,2^{k_1}-1.$

Interference operator of Deutsch-Jozsa's algorithm for $n=3, k_1=2, k_2=1$:

$$\left[Int^{Deutsch-Jozsa's}\right]_{i,j} = \frac{(-1)^{i*j}}{2^{\frac{2}{2}}} \otimes I =$$

$$= \frac{1}{2}\begin{pmatrix} (-1)^{0*0}I & (-1)^{0*1}I & (-1)^{0*2}I & (-1)^{0*3}I \\ (-1)^{1*0}I & (-1)^{1*1}I & (-1)^{1*2}I & (-1)^{1*3}I \\ (-1)^{2*0}I & (-1)^{2*1}I & (-1)^{2*2}I & (-1)^{2*3}I \\ (-1)^{3*0}I & (-1)^{3*1}I & (-1)^{3*2}I & (-1)^{3*3}I \end{pmatrix} = \frac{1}{2}\begin{pmatrix} I & I & I & I \\ I & -I & I & -I \\ I & I & -I & -I \\ I & -I & -I & I \end{pmatrix}. \tag{22}$$

Interference operator of Grover's algorithm: The interference operator of Grover's algorithm can be written as a block matrix of the following form:

$$\left[Int^{Grover}\right]_{i,j} = D_{k_1} \otimes I = \left(\frac{1}{2^{k_1/2}} - {}^{k_1}I\right) \otimes I = \left(-1 + \frac{1}{2^{k_1/2}}\right) \otimes I\Bigg|_{i=j}, \ \left(\frac{1}{2^{k_1/2}}\right) \otimes I\Bigg|_{i\neq j} = \frac{1}{2^{k_1/2}}\begin{cases} -I, i=j \\ I, i \neq j \end{cases}, \tag{23}$$

where $i = 0,...,2^{k_1}-1, j = 0,...,2^{k_1}-1$, D_{k_1} refers to diffusion operator:

$$\left[D_{k_1}\right]_{i,j} = \frac{(-1)^{1AND(i=j)}}{2^{k_1-1}}.$$

Thus, the interference operator of Grover's algorithm for $n=3, k_1=2, k_2=1$ is constructed as follows:

$$\left[Int^{Grover}\right]_{i,j} = D_2 \otimes I = \left(\frac{1}{2^{2/2}} - {}^{2}I\right) \otimes I = \left(-1 + \frac{1}{2}\right) \otimes I\Bigg|_{i=j}, \left(\frac{1}{2}\right) \otimes I\Bigg|_{i\neq j} =$$

$$= \left(\frac{1}{2}\right) \otimes I\Bigg|_{i\neq j} = \begin{pmatrix} \left(-1+\frac{1}{2}\right)I & \frac{1}{2}I & \frac{1}{2}I & \frac{1}{2}I \\ \frac{1}{2}I & \left(-1+\frac{1}{2}\right)I & \frac{1}{2}I & \frac{1}{2}I \\ \frac{1}{2}I & \frac{1}{2}I & \left(-1+\frac{1}{2}\right)I & \frac{1}{2}I \\ \frac{1}{2}I & \frac{1}{2}I & \frac{1}{2}I & \left(-1+\frac{1}{2}\right)I \end{pmatrix} = \frac{1}{2}\begin{pmatrix} -I & I & I & I \\ I & -I & I & I \\ I & I & -I & I \\ I & I & I & -I \end{pmatrix}. \tag{24}$$

Note that as the number of qubits increases the gain coefficient becomes smaller and the dimension of the matrix increases according to 2^{k_1}. However, each element can be extracted using Eq. (23), without constructing the entire operator matrix.

Interference operator of Simon's algorithm The interference operator of Simon's algorithm is prepared in the same manner as the superposition operators of Shor's and of Simon's algorithms and can be described as follows (see Eqs. (16), (19)):

$$\left[Int^{Simon}\right]_{i,j} = {}^{k_1}H \otimes^{k_2} I = \frac{(-1)^{i*j}}{2^{k_1/2}} \otimes {}^{k_2}I$$

$$= \frac{1}{2^{k_1/2}}\begin{pmatrix} (-1)^{0*0}\cdot{}^{k_2}I & \cdots & (-1)^{0*j}\cdot{}^{k_2}I & \cdots & (-1)^{0*\left(2^{k_1}-1\right)}\cdot{}^{k_2}I \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ (-1)^{i*0}\cdot{}^{k_2}I & \cdots & (-1)^{i*j}\cdot{}^{k_2}I & \cdots & (-1)^{i*\left(2^{k_1}-1\right)}\cdot{}^{k_2}I \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ (-1)^{\left(2^{k_1}-1\right)*0}\cdot{}^{k_2}I & \cdots & (-1)^{\left(2^{k_1}-1\right)*j}\cdot{}^{k_2}I & \cdots & (-1)^{\left(2^{k_1}-1\right)*\left(2^{k_1}-1\right)}\cdot{}^{k_2}I \end{pmatrix}. \tag{25}$$

In general, the interference operator of Simon's algorithm is similar to the interference operator of Deutsch-Jozsa's algorithm Eq. (21), but each block of the operator matrix Eq. (25) is a k_2 tensor product of the identity operator.

Each odd block (when the product of the indexes is an odd number) of the Simon's interference operator Eq. (25), has a negative sign. Actually if $i = 0,2,4,\ldots 2^{k_1} - 2$ or $j = 0,2,4,\ldots 2^{k_1} - 2$ the block sign is positive, otherwise the block sign is negative. This rule is applicable also for Eq. (21) of the Deutsch-Jozsa's algorithm interference operator.

Then it is convenient to check if one of the indexes is an even number instead of calculating the product. Thus Eq. (25) can be reduced as:

$$\left[Int^{Simon}\right]_{i,j} = {}^{k_1}H \otimes {}^{k_2} I = \frac{(-1)^{i*j}}{2^{k_1/2}} \otimes {}^{k_2}I = \frac{1}{2^{k_1/2}} \begin{cases} {}^{k_2}I, & \text{if } i \text{ is odd or if } j \text{ is odd} \\ -{}^{k_2}I, & \text{if } i \text{ is even and } j \text{ is even} \end{cases}. \tag{26}$$

Interference operator of Shor's algorithm. The interference operator of Shor's algorithm uses the Quantum Fourier Transformation operator (QFT), calculated as:

$$\left[QFT_{k_1}\right]_{i,j} = \frac{1}{2^{k_1/2}} e^{J(i*j)\frac{2\pi}{2^{k_1}}}, \tag{27}$$

where: J — imaginary unit, $i = 0,\ldots,2^{k_1} - 1$, $j = 0,\ldots,2^{k_1} - 1$. With $k_1 = 1$:

$$QFT_{k_1}\Big|_{k_1=1} = \frac{1}{2^{\frac{1}{2}}} \begin{pmatrix} e^{J*(0*0)2\pi/2^1} & e^{J*(0*1)2\pi/2^1} \\ e^{J*(1*0)2\pi/2^1} & e^{J*(1*1)2\pi/2^1} \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = H. \tag{28}$$

Eq. (27) can be also presented in harmonic form using Euler's formula:

$$\left[QFT_{k_1}\right]_{i,j} = \frac{1}{2^{k_1/2}} \left(\cos\left((i*j)\frac{2\pi}{2^{k_1}} \right) + J \sin\left((i*j)\frac{2\pi}{2^{k_1}} \right) \right). \tag{29}$$

In general, entanglement operators are part of a QA when the information about the function being analyzed is coded as an input-output relation. Thus, it is useful to develop a general approach for coding binary functions into corresponding entanglement gates.

Consider the arbitrary binary function: $f : \{0,1\}^n \to \{0,1\}^m$, such that:

$$f(x_0,\ldots,x_{n-1}) = (y_0,\ldots,y_{m-1}).$$

In order to create unitary quantum operator, which performs the same transformation first transform the irreversible function f into a reversible function F, as follows: $F : \{0,1\}^{m+n} \to \{0,1\}^{m+n}$, such that:

$$F(x_0,\ldots,x_{n-1},y_0,\ldots,y_{m-1}) = (x_0,\ldots,x_{n-1}, f(x_0,\ldots,x_{n-1}) \oplus (y_0,\ldots,y_{m-1})),$$

where $\oplus$ denotes addition modulo 2.

For the reversible function F, it is possible to design an entanglement operator matrix using the following rule:

$$\left[U_F\right]_{i^B, j^B} = 1 \text{ iff } F(j^B) = i^B,\ i,j \in \Big[\underbrace{0,..,0}_{n+m};\underbrace{1,..,1}_{n+m};\Big],$$

where B denotes binary coding. The resulting entanglement operator is a block diagonal matrix, of the form:

$$U_F = \begin{pmatrix} M_0 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & M_{2^n-1} \end{pmatrix}. \tag{30}$$

Each block $M_i, i = 0,...,2^n - 1$ includes m tensor products of I or of C operators, and can be obtained as follows:

$$M_i = \bigotimes_{k=0}^{m-1} \begin{cases} I, \text{iff } F(i,k) = 0 \\ C, \text{iff } F(i,k) = 1 \end{cases}, \tag{31}$$

where C represents the NOT operator, defined as: $C = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

The entanglement operator is a sparse matrix. Using sparse matrix operations it is possible to accelerate the simulation of the entanglement. Each row or column of the entanglement operation has only one position with non-zero value. This is a result of the reversibility of the function F. For example, consider the entanglement operator for a binary function with two inputs and one output: $f:\{0,1\}^2 \to \{0,1\}^1$, such that: $f(x) = 1|_{x=01} 0|_{x \neq 01}$.

The reversible function F in this case is: $F:\{0,1\}^3 \to \{0,1\}^3$, such that:

(x,y)	$(x, f(x) \oplus y)$
$00,0$	$00, 0 \oplus 0 = 0$
$00,1$	$00, 0 \oplus 1 = 1$
$01,0$	$01, 1 \oplus 0 = 1$
$01,1$	$01, 1 \oplus 1 = 0$
$10,0$	$10, 0 \oplus 0 = 0$
$10,1$	$10, 1 \oplus 0 = 1$
$11,0$	$11, 0 \oplus 0 = 0$
$11,1$	$11, 1 \oplus 0 = 1$

The corresponding entanglement block matrix can be written as:

$$U_F = \begin{array}{c} \\ |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{array} \begin{array}{c} \begin{array}{cccc} \langle 00| & \langle 01| & \langle 10| & \langle 11| \end{array} \\ \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & \boxed{C} & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{pmatrix} \end{array}.$$

Fig. 2 (c) shows the result of the application of this operator in Grover’s QSA. Entanglement operators of Deutsch and of Deutsch-Jozsa’s algorithms have the general form shown in the above equation.

As a further example, consider the entanglement operator for a binary function with two inputs and two outputs: $f:\{0,1\}^2 \to \{0,1\}^2$, such that: $f(x) = 10|_{x=01,11} 00|_{x \neq 01,11}$ and

$$U_F = \begin{array}{c} \\ |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{array} \begin{array}{c} \begin{array}{cccc} \langle 00| & \langle 01| & \langle 10| & \langle 11| \end{array} \\ \begin{pmatrix} I \otimes I & 0 & 0 & 0 \\ 0 & \boxed{C \otimes I} & 0 & 0 \\ 0 & 0 & I \otimes I & 0 \\ 0 & 0 & 0 & \boxed{C \otimes I} \end{pmatrix} \end{array}.$$

The entanglement operators of Shor’s and of Simon’s algorithms have the general form shown in the above equation.

4. Results of classical QA gate simulation

Analyzing the quantum operators leads to the following simplifications for increasing the performance of classical QA simulations:

(a) All quantum operators are symmetrical around main diagonal matrices;

(b) The state vector is a sparse matrix;

(c) Elements of the quantum operators need not be stored, but rather can be calculated when necessary using Eqs. (6), (12), (30) and (31);

(d) The termination condition can be based on the minimum of Shannon entropy of the quantum state, calculated as:

$$H = -\sum_{i=0}^{2^{m+n}} p_i \log p_i \,. \tag{32}$$

Calculation of the Shannon entropy is applied to the quantum state after the interference operation. The minimum of Shannon entropy in Eq. (32) corresponds to the state when there are few state vectors with high probability (states with minimum uncertainty are intelligent states). Selection of an appropriate termination condition is important since QAs are periodical.

Fig. 4 shows results of the Shannon information entropy calculation for the Grover's algorithm with 5 inputs.

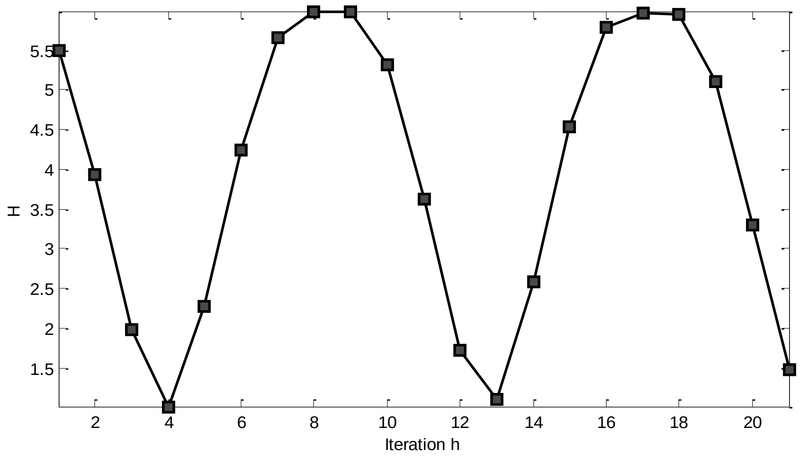


Figure 4. Shannon entropy simulation of Grover's QSA dynamics with five inputs

Fig. 4 shows that for five inputs of the Grover's QSA an optimal number of iterations, according to minimum of the Shannon entropy criteria for successful result, is exactly four. With more iterations the probability of obtaining a correct answer will decrease and the algorithm may fail to produce a correct answer.

The theoretical estimation for 5 inputs gives $\frac{\pi}{4}\sqrt{2^5} = 4.44$ iterations. The Shannon entropy-based termination condition provides the number of iterations. More detailed description of the information-based termination condition is presented below.

Simulation results of a fast Grover QSA are summarized in Table 3.

The number of iterations for the fast algorithm is estimated according to the termination condition based on minimum of Shannon entropy of the quantum intelligent state vector.

The following approaches were used in the simulations listed in Table 3.

In Approach 1, the quantum operators are applied as matrices, elements of quantum operator matrices are calculated dynamically according to Eqs. (6), (12), (4.30) and (31).

As shown in Fig. 5, the classical hardware limit of this approach to simulation on a desktop computer is around 20 or more qubits, caused by an exponential temporal complexity.

Table 3. Temporal complexity of Grover's QSA simulation on 1.2 GHz computer with two CPUs

n	**Number of iterations** *h*	**Temporal complexity**, *seconds*	
		Approach 1 (one iteration)	*Approach* 2 (*h* iterations)
10	25	0.28	~0
12	50	44	~0
14	100	99.42	~0
15	142	489.05	~0
16	201	2060.63	~0
20	804	-	~0
30	2375	-	0.016
40	853.549	-	4.263
50	26.353.589	-	12.425

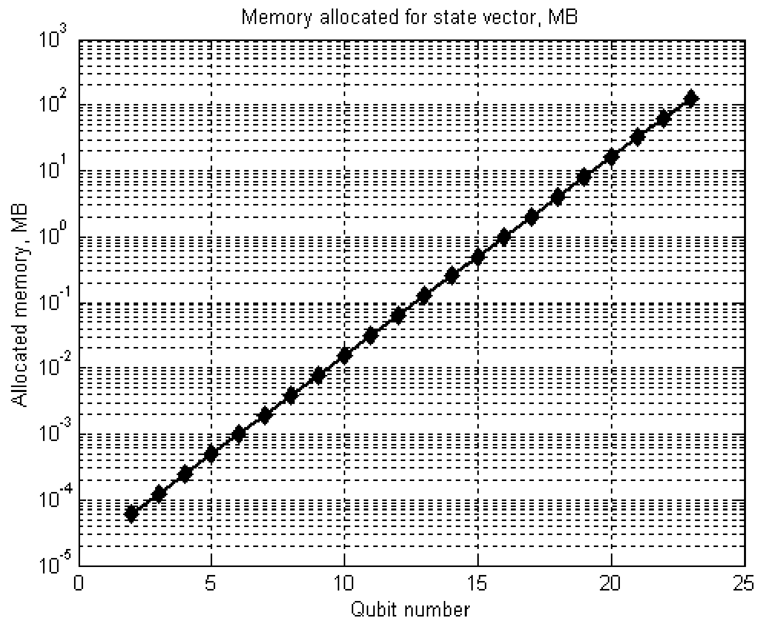


Figure 5. Spatial complexity of Grover QA simulation

In Approach 2, the quantum operators are replaced with classical gates. Product operations are removed from the simulation as described above. The state vector of probability amplitudes is stored in compressed form (only different probability amplitudes are allocated in memory).

Fig. 6 shows that with the second approach, it is possible to perform classical efficient simulation of Grover's QSA on a desktop computer with a relatively large number of inputs (50 qubits or more).

Fig. 6 shows also that with allocation of the state vector in computer memory, this approach permits simulation 26 qubits on a conventional PC with 1GB of RAM. By contrast, Fig. 5 shows memory required for Grover's algorithm simulation when the entire state vector is stored in memory. Adding one qubit doubles the computer memory needed for simulation of Grover's QSA when state vector is allocated completely in memory.

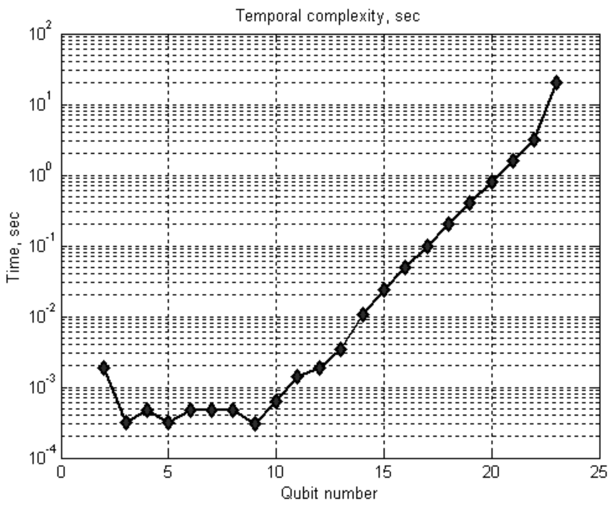


Figure 6. Temporal complexity of Grover's QSA

5. Information criteria for solution of the QSA-termination problem

Quantum algorithms come in two general classes: algorithms that rely on a Fourier transform, and algorithms that rely on amplitude amplification. Typically, the algorithms include a sequence of trials. After each trial, a measurement of the system produces a desired state with some probability determined by the amplitudes of the superposition created by the trial. Trials continue until the measurement gives a solution, so that the number of trials and hence, the running time are random.

The number of iterations needed, and the nature of the termination problem (i.e., determining when to stop the iterations) depends in art on the information dynamics of the algorithm. An examination of the dynamics of Grover's QSA algorithm starts by preparing all m qubits of the quantum computer in the state $|s\rangle = |0\ldots0\rangle$. An elementary rotation in the direction of the sought state $|x_0\rangle$ with property $f(x_0)=1$ is achieved by the gate sequence:

$$Q = -\left[\underbrace{\left(I_s H^{\otimes 2m}\right)\cdot I_{x_0}}_{k\ times}\right]\cdot H^{\otimes 2m}, \tag{33}$$

where the phase inversion I_s with respect to the initial state $|s\rangle$ is defined by $I_s|s\rangle = -|s\rangle, I_s|s\rangle = |s\rangle\,(x \neq s)$. The controlled phase inversion I_{x_0} with respect to the sought state $|x_0\rangle$ is defined in an analogous way. Because the state $|x_0\rangle$ is not known explicitly but only implicitly through the property $f(x_0)=1$, this transformation is performed with the help of the quantum oracle. This task can be achieved by preparing the ancillary of the quantum oracle in the state $|a_0\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right)$ as the unitary and Hermitian transformation: $U_F : |x,a\rangle \to |x, f(x)\oplus a\rangle$.

Thus, $|x\rangle$ is an arbitrary element of the computational basis and $|a\rangle$ is the state of an additional ancillary qubit. As a consequence, one obtains the required properties for the phase inversion I_{x_0}, namely:

$$|x, f(x)\oplus a_0\rangle \equiv |x, 0\oplus a_0\rangle = \frac{1}{\sqrt{2}}\left[|x,0\rangle - |x,1\rangle\right] = |x,a_0\rangle, \quad for\ x \neq x_0$$

$$|x, f(x)\oplus a_0\rangle \equiv |x, 1\oplus a_0\rangle = \frac{1}{\sqrt{2}}\left[|x,1\rangle - |x,0\rangle\right] = -|x,a_0\rangle, \quad for\ x = x_0.$$

In order to rotate the initial state $|s\rangle$ into the state $|x_0\rangle$ one can perform a sequence of n such rotations and a final Hadamard transformation at the end, i.e., $|s_{fin}\rangle = HQ^n|s_{in}\rangle$. The optimal number n of repetitions of the gate Q in Eq. (33) is approximately given by

$$n = \frac{\pi}{4\arcsin\left(2^{-\frac{m}{2}}\right)} - \frac{1}{2} \approx \frac{\pi}{4}\sqrt{2^m}, \left(2^m \gg 1\right). \tag{34}$$

The matrix D_n, which is called the diffusion matrix of order *n*, is responsible for interference in this algorithm. It plays the same role as QFT$_n$ (Quantum Fourier Transform) in Shor's algorithm and of ^nH in Deutsch-Jozsa's and Simon's algorithms. This matrix is defined as

$$\left[D_n\right]_{i,j} = \frac{(-1)^{1AND(i=j)}}{2^{n/2}}, \tag{35}$$

where $i = 0,...,2^n - 1$, $j = 0,...,2^n - 1$ n is a number of inputs.

The gate equation of Grover's QSA circuit is the following:

$$G^{Grover} = [(D_n \otimes I) \cdot U_F]^h \cdot ({}^{n+1}H). \tag{36}$$

The diagonal matrix elements in Grover's QSA-operators (as shown, for example, in Eq. (37) below) are connected to a database state to itself and the off-diagonal matrix elements are connected to a database state and to its neighbors in the database. The diagonal elements of the diffusion matrix have the opposite sign from the off-diagonal elements.

The magnitudes of the off-diagonal elements are roughly equal, so it is possible to write the action of the matrix on the initial state (see Table 4).

Table 4. Diffusion matrix definition

D_n	|0..0>	|0..1>	…	|i>	…	|1..0>	|1..1>
|0..0>	$-1+1/2^{n-1}$	$1/2^{n-1}$	…	$1/2^{n-1}$	…	$1/2^{n-1}$	$1/2^{n-1}$
|0..1>	$1/2^{n-1}$	$-1+1/2^{n-1}$	…	$1/2^{n-1}$	…	$1/2^{n-1}$	$1/2^{n-1}$
…	…	…	…	…	…	…	…
|i>	$1/2^{n-1}$	$1/2^{n-1}$	…	$-1+1/2^{n-1}$	…	$1/2^{n-1}$	$1/2^{n-1}$
…	…	…	…	…	…	…	…
|1..0>	$1/2^{n-1}$	$1/2^{n-1}$	…	$1/2^{n-1}$	…	$-1+1/2^{n-1}$	$1/2^{n-1}$
|1..1>	$1/2^{n-1}$	$1/2^{n-1}$	…	$1/2^{n-1}$	…	$1/2^{n-1}$	$-1+1/2^{n-1}$

For example:

$$\begin{pmatrix} -a & b & b & b & b & b \\ b & -a & b & b & b & b \\ b & b & -a & b & b & b \\ b & b & b & -a & b & b \\ b & b & b & b & -a & b \\ b & b & b & b & b & -a \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ -1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \frac{1}{\sqrt{N}} = \begin{pmatrix} -a+(N-3)b \\ -a+(N-3)b \\ +a+(N-1)b \\ -a+(N-3)b \\ -a+(N-3)b \\ -a+(N-3)b \end{pmatrix} \frac{1}{\sqrt{N}}, \tag{37}$$

where $a = 1 - b$, $b = \frac{1}{2^{n-1}}$. If one of the states is marked, i.e., has its phase reversed with respect to that of the others, the multimode interference conditions are appropriate for constructive interference to the marked state, and destructive interference to the other states. That is, the population in the marked bit is amplified. The form of this matrix is identical to that obtained through the inversion about the average procedure in Grover's QSA. This operator produces a contrast in the probability density of the final states of the database of $\frac{1}{N}\left[a+(N-1)b\right]^2$ for the marked bit versus $\frac{1}{N}\left[a-(N-3)b\right]^2$ for the unmarked bits; where N is the number of bits in the data register.

Grover's algorithm gate in Eq. (36) is optimal and it is, thus, an efficient search algorithm. Thus, software based on the Grover algorithm can be used for search routines in a large database. Grover's QSA includes a number of trials that are repeated until a solution is found. Each trial has a predetermined number of iterations, which determines the probability of finding a solution. A quantitative measure of success in the database search problem is the reduction of the information entropy of the system following the search algorithm.

Entropy $S^{Sh}(P_i)$ in this example of a single marked state is defined as

$$S^{Sh}(P_i) = -\sum_{i=1}^{N} P_i \log P_i\,, \tag{38}$$

where P_i is the probability that the marked bit resides in orbital i.

In general, the Von Neumann entropy is not a good measure for the usefulness of Grover's algorithm. For practically every value of entropy, there exist states that are good initializes and states that are not. For example, $S\left(\rho_{(n-1)-mix}\right) = \log_2 N - 1 = S\left(\rho_{\left(\frac{1}{\log_2 N}\right)-pure}\right)$, but when initialized in $\rho_{(n-1)-mix}$, the Grover algorithm is not good at guessing the market state.

Another example may be given using pure states $H|0\rangle\langle 0|H$ and $H|1\rangle\langle 1|H$. With the first, Grover finds the marked state with quadratic speed-up. The second is practically unchanged by the algorithm.

The information intelligent measure $\Im_T(|\psi\rangle)$ of the state $|\psi\rangle$ with respect to the qubits in *T* and to the basis $B = \{|i_1\rangle \otimes \cdots \otimes |i_n\rangle\}$ is

$$\Im_T(|\psi\rangle) = 1 - \frac{S_T^{Sh}(|\psi\rangle) - S_T^{VN}(|\psi\rangle)}{|T|}. \tag{39}$$

The intelligence of the QA state is maximal if the gap between the Shannon and the von Neumann entropy in Eq. (39) for the chosen resultant qubit is minimal. Information QA-intelligent measure $\Im_T(|\psi\rangle)$ and interrelations between information measures $S_T^{Sh}(|\psi\rangle) \ge S_T^{VN}(|\psi\rangle)$ are used together with entropic relations of the step-by-step natural majorization principle for solution of the QA-termination problem.

From Eq. (39) it can be seen that for pure states

$$\max \Im_T(|\psi\rangle) \;\mapsto\; 1 - \min\left(\frac{S_T^{Sh}(|\psi\rangle) - S_T^{VN}(|\psi\rangle)}{|T|}\right) \mapsto \min S_T^{Sh}(|\psi\rangle), \quad S_T^{VN}(|\psi\rangle) = 0\,. \tag{40}$$

From Eq. (39) the principle of Shannon entropy minimum is described as follows.

According to Eq. (40), the Shannon entropy shows the lower bound of quantum complexity of the QA. It means that the criterion in Eq. (40) includes both metrics for design of an intelligent QSA: (i) minimal quantum query complexity; and (ii) optimal termination of the QSA with a successful search solution.

The Shannon information entropy is used for optimization of the termination problem of Grover's QSA. A physical interpretation of the information criterion begins with an information analysis of Grover's QSA based on using of Eq. (4.39). Thus Eq (4.39) gives a lower bound on the amount of entanglement needed for a successful search and of the computational time.

A QSA that uses the quantum oracle calls $\{O_s\}$ as $I - 2|s\rangle\langle s|$ calls the oracle at least $T \ge \left(\frac{1-P_e}{2\pi} + \frac{1}{\pi \log N}\right)\sqrt{N}$ times to achieve a probability of error P_e. The information system includes the

N — state data register. Physically, when the data register is loaded, the information is encoded as the phase of each orbital. The orbital amplitudes carry no information. While state-selective measurement gives as result only amplitudes, the information is hidden from view, and therefore, the entropy of the system is maximum: $S_{init}^{Sh}(P_i) = -\log(1/N) = \log N$.

The rules of quantum measurement ensure that only one state will be detected each time.

If the algorithm works perfectly, the marked state orbital is revealed with unit efficiency, and the entropy drops to zero. Otherwise, unmarked orbitals may occasionally be detected by mistake. The entropy reduction can be calculated from the probability distribution, using Eq. (38). The minimum Shannon entropy criteria is used for successful termination of Grover's QSA and realized in this case in digital circuit implementation.

Fig. 7 shows the results of entropy analysis for Grover's QSA according to Eq. (32), for the case where $n = 7,\ f(x_0) = 1$.

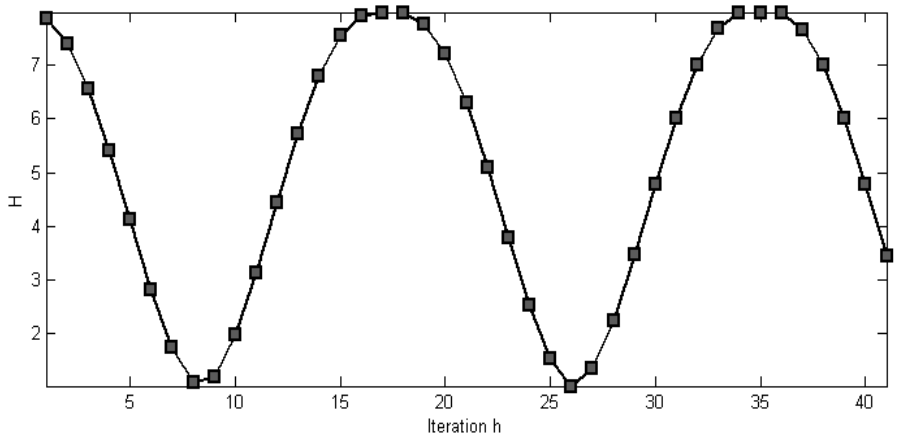


Figure 7. Shannon entropy simulation of QSA with 7- inputs

Fig. 7 shows that minimum Shannon entropy is achieved on the 8th iteration (the minimum value of the Shannon entropy is 1). A theoretical estimation for this case is $\frac{\pi}{4}\sqrt{2^7} \approx 9$ iterations. On the ninth iteration, the probability of the correct answer already becomes smaller, and as a result, measurement of the wrong basis vector may happen.

Application of the Shannon entropy termination condition is presented below for different input qubit numbers of Grover's QSA. For efficient termination of QAs that give the highest probability of successful result, the Shannon entropy is minimal for the step $m+1$. This is the principle of minimum Shannon entropy for termination of a QA with the successful result. This result also follows from the principle of QA maximum intelligent state. For this case:

$$\max \mathcal{J}_T(|\psi\rangle) = 1 - \min \frac{H_T^{Sh}(|\psi\rangle)}{|T|}, \quad S_T^{vN}(|\psi\rangle) = 0 \text{ (for } \textit{pure} \text{ quantum state).}$$

Thus, the principle of maximal intelligence of QAs includes as particular case the principle of minimum Shannon entropy for QA-termination problem solution.

6. Structure and acceleration method of quantum algorithm simulation

The analysis of the quantum operator matrices that was carried out in the previous sections forms the basis for specifying the structural patterns giving the background for the algorithmic approach to QA modeling on classical computers. The allocation in the computer memory of only a fixed set of tabulated (pre-defined) constant values instead of allocation of huge matrices (even in sparse form) provides computational efficiency. Various elements of the quantum operator matrix can be obtained by application of an appropriate algorithm based on the structural patterns and particular properties of the equations that define the matrix elements. Each representation algorithm uses a set of table values for calculating the matrix elements. The calculation of the tables of the predefined values can be done as part of the algorithm's initialization.

6.1. Algorithmic representation of the Grover's QA

Figs 8 (a--c) are flowcharts showing realization of such an approach for simulation of superposition (Fig. 8 (a)), entanglement (Fig. 8 (b)) and interference (Fig. 8 (c)) operators in Grover's QSA.

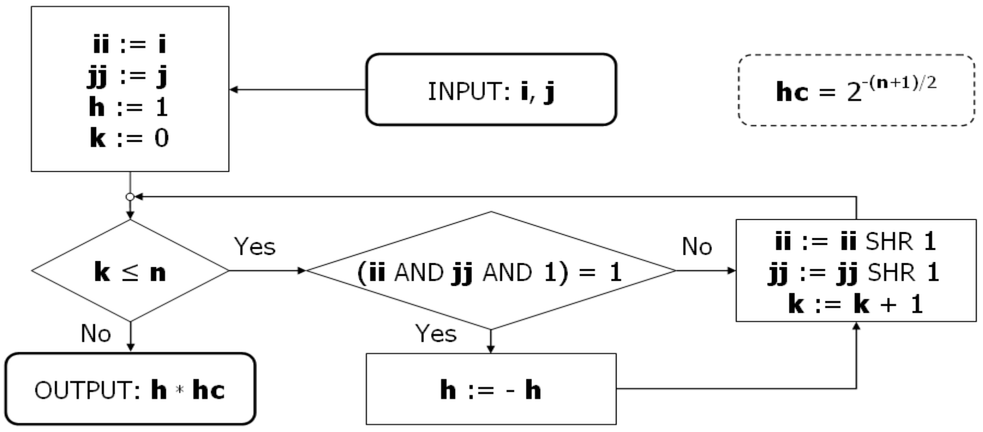


Figure 8 (a). Superposition operator representation algorithm for Grover's QSA

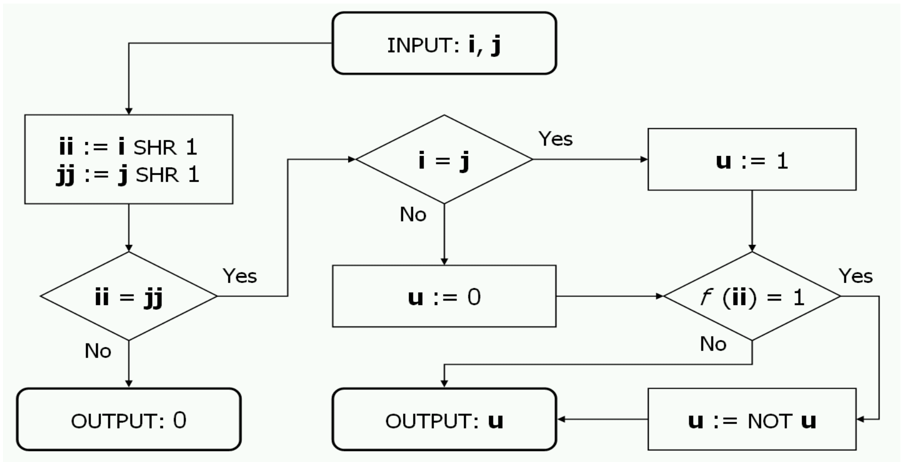


Figure 8 (b). Entanglement operator representation algorithm for Grover's QSA

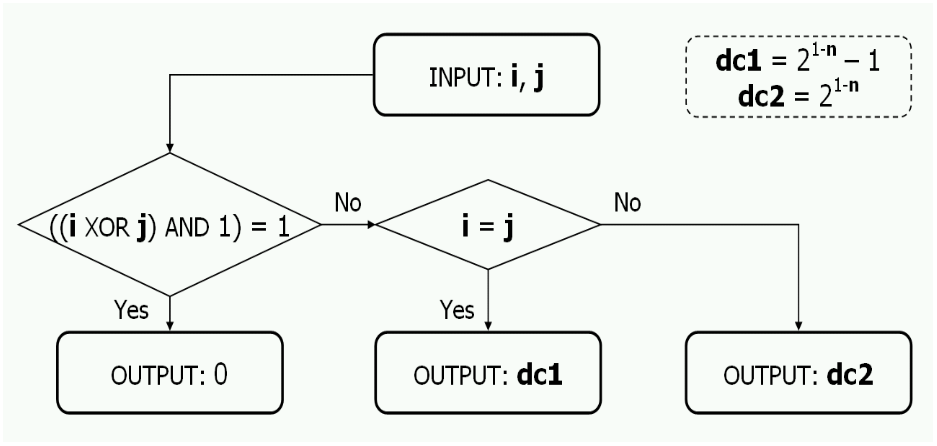


Figure 8 (c). Interference operator representation algorithm for Grover's QSA

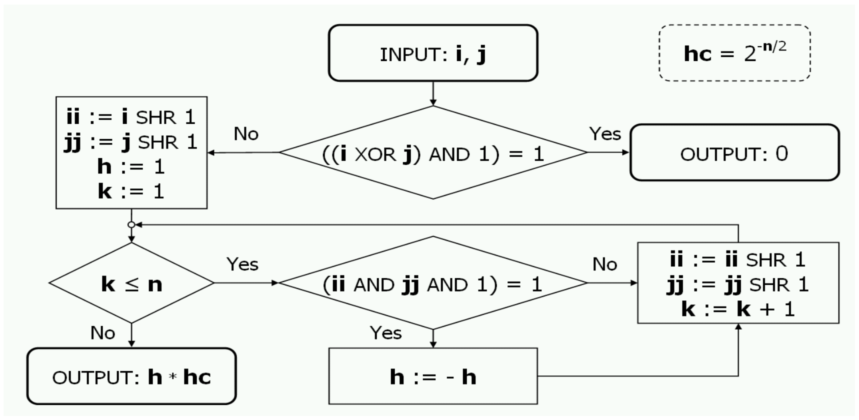


Figure 8 (d). Interference operator representation algorithm for Deutsch-Jozsa's QA

Here n is a number of qubit, i and j are the indexes of a requested element, hc=$2^{-(n+1)/2}$, $dc1 = 2^{1-n} - 1$ and $dc2 = 2^{1-n}$ are the table values.

In Fig. 8 (a), the *i,j* values are specified and provided to an initialization block with loops control variables *ii*:= *i*, *jj*:= 0, and *k*:= 0 are initialized, and calculation variable *h*:= 1 is initialized. The process then proceeds to a decision block. In the decision block, if *k* is less than or equal to *n*, then the process advances to another decision block; otherwise, the process advances to an output block, where the output *h*hc* is computed (where $hc = 2^{-(n+1)/2}$). In the decision block, if (*ii* and *jj* and 1) = 1, then the process advances to a block $h := -h$; otherwise, the process advances to another block and passes to the next iteration without probability amplitude inversion. Alternatively, the process sets *h*:= − *h* and proceeds to the next iteration. By setting *ii*:= *ii SHR 1*, *jj*:= *jj SHR* 1, and *k*:= *k* + 1 (where SHR is a shift right operation), and then the process continues until all probability amplitudes are assigned.

In Fig. 8 (b), the inputs *i, j* in an input block are initialized as *ii*:= *i SHR* 1, and *jj*:= *SHR* 1 and then are passed to the end test.

If the end test is not succeeding, means the inputs *i* and *j* are pointing to the marked elements, the process of the probability amplitude inversion of the marked states in this case is performed.

In Fig. 8 (c), interference operator of Grover's QSA can be substituted by a simple logic which outputs 0 *if* ((*i* XOR *j*) AND 1) = 1 then regarding nonzero elements, *if i* = *j* then the process outputs *dc*1, otherwise the process outputs *dc*2, where $dc1 = 2^{1-n} - 1$ and $dc2 = 2^{1-n}$.

As described above, the superposition and entanglement operators for Deutsch-Jozsa's QA are the same with superposition and entanglement operators for Grover's QSA (Figs 8 (a), and 8 (b), respectively). The interference operator representation algorithm for Deutsch-Jozsa's QA is shown in Fig. 8 (d), where $hc = 2^{-n/2}$.

The entanglement operator for the Simon QA is shown in Fig. 8 (e). Here m is an output dimension, $ec1 = 2^{m} - 1$ and $ec2 = 2^{m-1}$ are the table values.

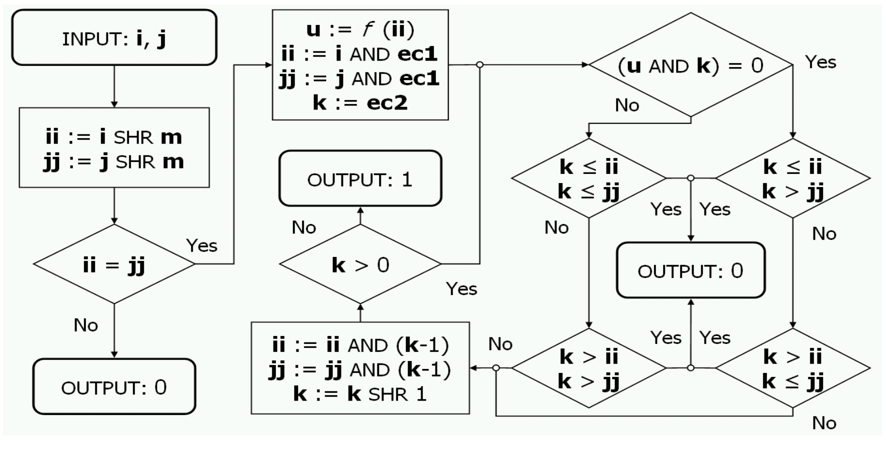


Figure 8 (e). Entanglement operator representation algorithm for Simon's and Shor's QA

Superposition and interference operators for the Simon QA are identical (see Table 1) and are shown by flowchart in Fig. 8 (f).

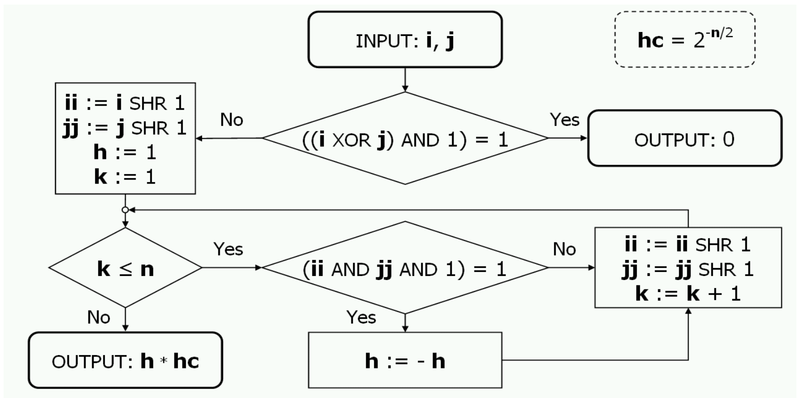


Figure 8 (f): Superposition and interference operator representation algorithm for Simon's QA

Fig. 8 (g) is a flowchart showing calculation of the interference operator from the Shor QA.

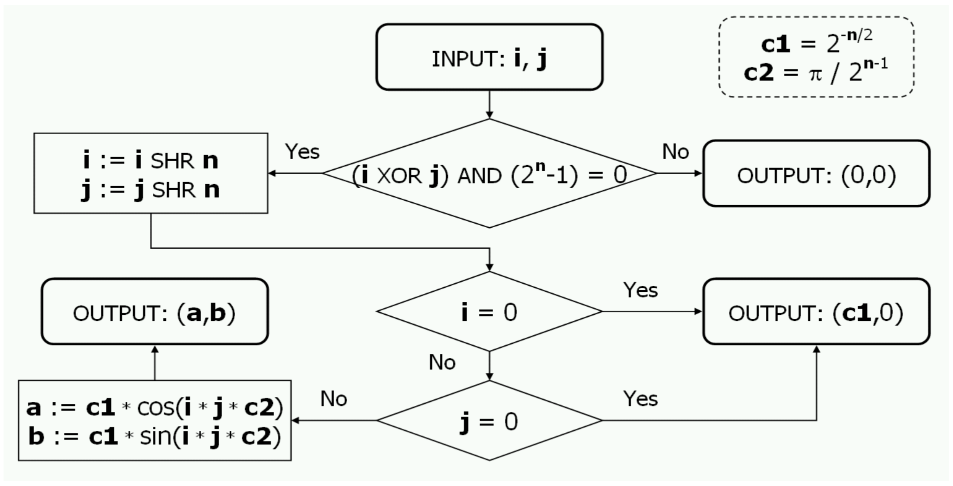


Figure 8 (g). An interference operator representation algorithm for Shor's QA

The Shor interference operator is relatively more complex, as explained above. Superposition and entanglement operators for the Shor algorithm are the same as the Simon's QA operators shown in Fig. 8 (f) and Fig. 8 (e). The Shor interference operator is based on the Quantum Fourier Transformation (QFT) with table values c1 = 2 – n/2 and c2 = π/2 n – 1.

The time required for calculating the elements of an operator's matrix during a process of applying a quantum operator is generally small in comparison to the total time of performing a quantum step. Thus, the time burden created by exponentially increasing memory usage tends to be less, or at least similar to, the time burden created by computing matrix elements as needed. Moreover, since the algorithms used to compute the matrix elements tend to be based on fast bit-wise logic operations, the algorithms are amenable to hardware acceleration.

Table 5 shows comparisons of the traditional and as-needed matrix calculation when the memory used for the as-needed algorithm (Memory* denotes memory used for storing the quantum system state vector).

*The results shown in Table 5 are based on the results of testing the software realization of Grover QSA simulator on a personal computer with Intel Pentium III 1 GHz processor and 512 Mbytes of memory. Only one iteration of the Grover QSA was performed.

Table 5 shows that significant speed-up is achieved by using the algorithmic approach as compared with the prior art direct matrix approach. The use of algorithms for providing the matrix elements allows considerable optimization of the software, including the ability to optimize at the machine instructions level. However, as the number of qubits increases, there is an exponential increase in temporal complexity, which manifests itself as an increase in time required for matrix product calculations.

Table 5. Different approaches comparison: Standard (matrix based) and algorithmic based approach

Qubits	**Standard**		**Calculated Matrices**	
	Memory, MB	*Time*, s	*Memory**	*Time*, s
1	1	0.03	≈0	≈0
8	18	4	0.008	0.0325
11	1048	1411	0.064	2.3
16	--	--	2	4573
24	--	--	512	$3*10^8$
64	--	--	--	--

Use of the structural patterns in the quantum system state vector and use of a problem-oriented approach for each particular algorithm can be used to offset this increase in temporal complexity. By way of explanation, and not by way of limitation, the Grover algorithm is used below to explain the problem-oriented approach to simulating a QA on a classical computer.

6.2. Problem-oriented approach based on structural pattern of QA state vector

Let n be the input number of qubits. In the Grover algorithm, half of all 2^{n+1} elements of a vector making up its even components always take values symmetrical to appropriate odd components and, therefore, need not be computed.

Odd 2^n elements can be classified into two categories:

– The set of *m* elements corresponding to truth points of input function (or oracle); and

– The remaining $2^n - m$ elements.

The values of elements of the same category are always equal.

As discussed above, the Grover QA only requires two variables for storing values of the elements. Its limitation in this sense depends only on a computer representation of the floating-point numbers used for the state vector probability amplitudes. For a double-precision software realization of the state vector representation algorithm, the upper reachable limit of q-bit number is approximately 1024.

Fig. 9 shows a state vector representation algorithm for the Grover QA.

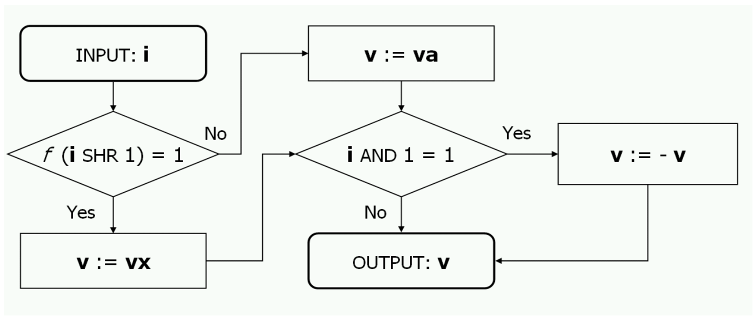


Figure 9. State vector representation algorithm for Grover' quantum search

In Fig. 9, i is an element index, *f* is an input function, *vx* and *va* corresponds to the elements' category, and *v* is a temporal variable. The number of variables used for representing the state variable is constant. A constant number of variables for state vector representation allow reconsideration of the traditional schema of quantum search simulation.

Classical gates are used not for the simulation of appropriate quantum operators with strict one-to-one correspondence but for the simulation of a quantum step that changes the system state. Matrix product operations are replaced by arithmetic operations with a fixed number of parameters irrespective of qubit number.

Fig. 10 shows a generalized schema for efficient simulation of the Grover QA built upon three blocks, a superposition block *H*, a quantum step block UD and a termination block *T*.

Fig. 10 also shows an input block and an output block. The *UD* block includes a *U* block and a *D* block. The input state from the input block is provided to the superposition block. A superposition of states from the superposition block is provided to the *U* block. An output from the *U* block is provided to the *D* block. An output from the *D* block is provided to the termination block. If the termination block terminates the iterations, then the state is passed to the output block; otherwise, the state vector is returned to the U block for iteration.

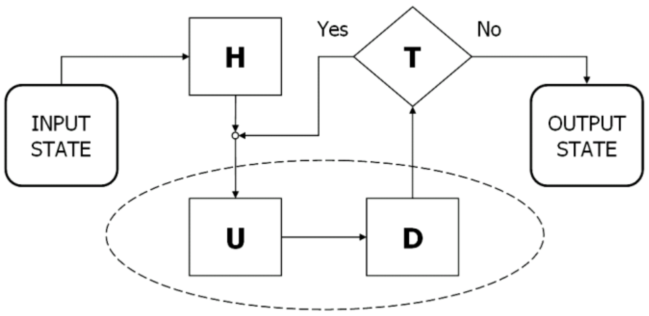


Figure 10. Generalized schema of simulation for Grover' QSA

As shown in Fig. 11, the superposition block *H* for Grover QSA simulation changes the system state to the state obtained traditionally by using *n* + 1 times the tensor product of Walsh-Hadamard transformations. In the process shown in Fig. 10, *vx:= hc, va:= hc,* and *vi:= 0,* where *hc* = $2^{-(n+1)/2}$ is a table value.

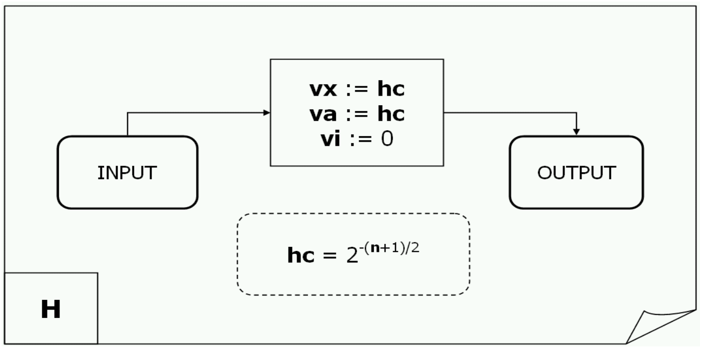


Figure 11. Superposition block for Grover's QSA

The quantum step block UD that emulates the entanglement and interference operators is shown on Figs 12 (a – c).

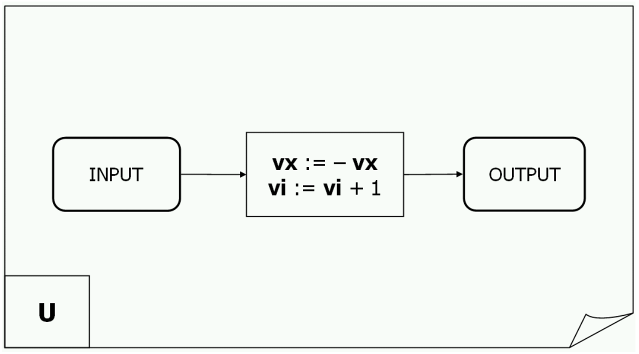


Figure 12 (a). Emulation of the entanglement operator application of Grover's QSA

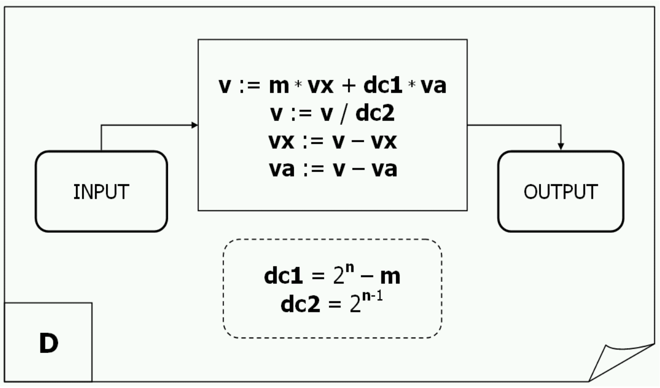


Figure 12 (b). Emulation of interference operator application of Grover's QSA

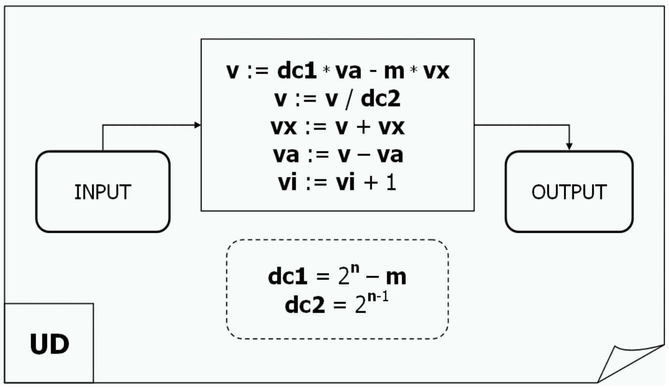


Figure 12 (c). Quantum step block for Grover' quantum search

The *UD* block reduces the temporal complexity of the quantum algorithm simulation to linear dependence on the number of executed iterations. The *UD* block uses recalculated table values $dc1 = 2^n - m$ and $dc2 = 2^{n-1}$.

In the *U* block shown in Fig. 12 (a), *vx*:= - *vx* and *vi*:= *vi* + 1.

In the *D* block shown in Fig. 12 (b), *v*:= *m***vx*+*dc*1**va*, *v*:= *v*/*dc*2, *vx*:= *v* - *vx*, and *va:*= *v* - *va* in the *UD* block shown in Fig. 12 (c), *v:= dc1*va = m*vx, v:=v/dc*2, *vx*:=*v* + *vx, va*:= *v* - *va*, and *vi*:= *vi* + 1.

The termination block *T* is general for all QAs, independently of the operator matrix realization. Block *T* provides intelligent termination condition for the search process. Thus, the block *T* controls the number of iterations through the block *UD* by providing enough iteration to achieve a high probability of arriving at a correct answer to the search problem. The block *T* uses a rule based on observing the changing of the vector element values according to two classification categories. The *T* block during a number of iterations, watches for values of elements of the same category monotonically increase or decrease while values of elements of another category changed monotonically in reverse direction. If after some number of iteration the direction is changed, it means that an extremum point corresponding to a state with maximum or minimum uncertainty is passed. The process can using direct values of amplitudes instead of considering Shannon entropy value, thus, significantly reducing the required number of calculations for determining the minimum uncertainty state that guarantees the high probability of a correct answer.

The termination algorithm realized in the block T can be used one or more of five different termination models:

Model 1: Stop after a predefined number of iterations;

Model 2: Stop on the first local entropy minimum;

Model 3: Stop on the lowest entropy within a predefined number of iterations;

Model 4: Stop on a predefined level of acceptable entropy; and/or

Model 5: Stop on the acceptable level or lowest reachable entropy within the predefined number of iterations.

Note that models 1 — 3 do not require the calculation of an entropy value.

Figs 13 — 15 show the structure of the termination condition blocks *T*.

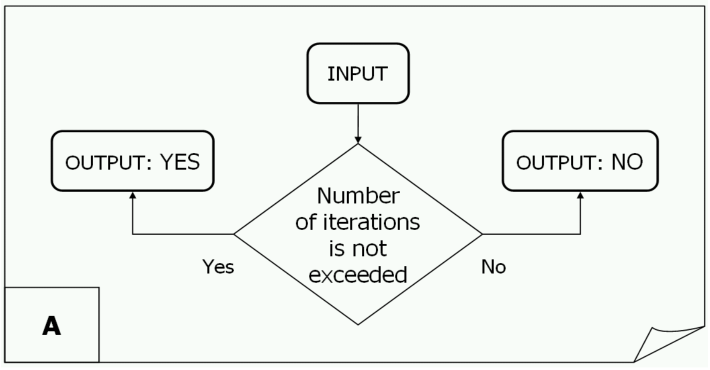


Figure 13. Termination block for method 1

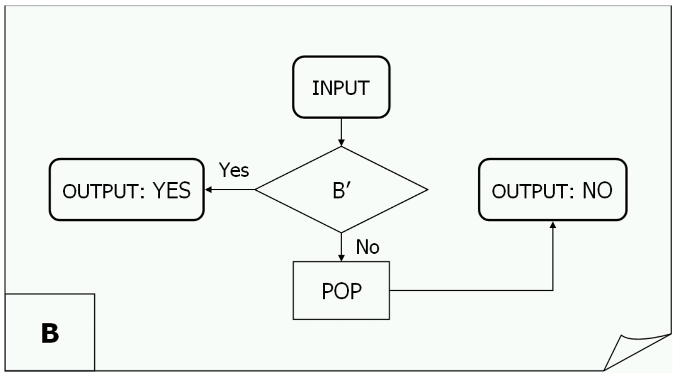


Figure 14. Component B for the termination block

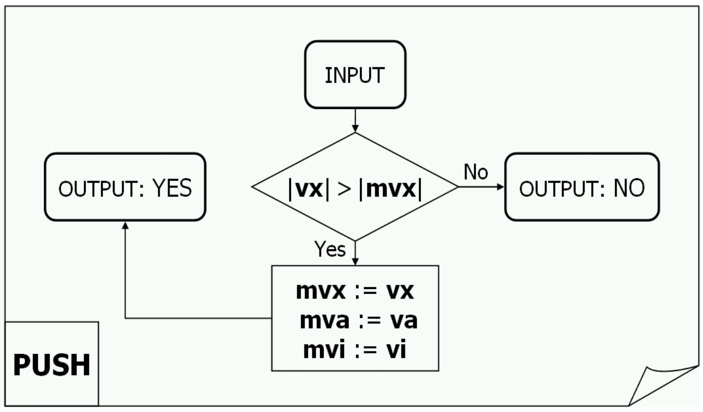


Figure 15 (a): Component PUSH for the termination block

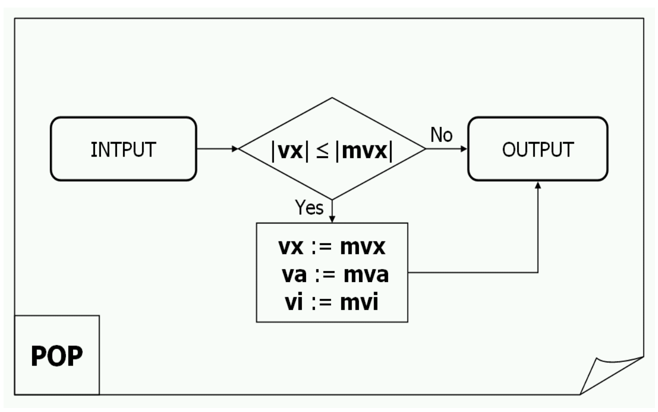


Figure 15 (b). Component POP for the termination block

Since time efficiency is one of the major demands on such termination condition algorithm, each part of the termination algorithm is represented by a separate module, and before the termination algorithm starts, links are built between the modules in correspondence to the selected termination model by initializing the appropriate functions' calls.

Table 6 shows components for the termination condition block T for the various models. Flow charts of the termination condition building blocks are provided in Figs 13 – 15.

Table 6. Termination block construction

Model	**T**	**B'**	**C'**
1	A	--	--
2	B	PUSH	--
3	C	A	B
4	D	--	--
5	C	A	E

The entries *A, B, PUSH, C, D, E*, and *PUSH* in Table 6 correspond to the flowcharts in Figs 13 — 18 respectively.

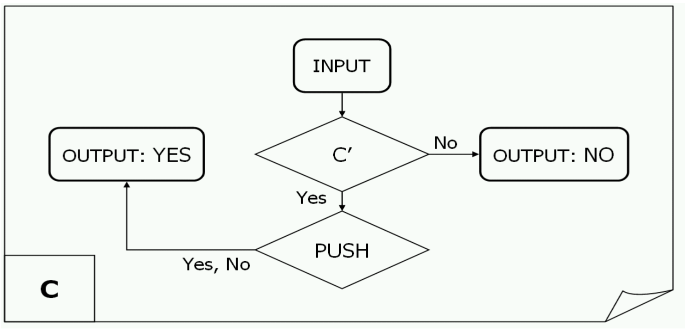


Figure 16. Component C for the termination block

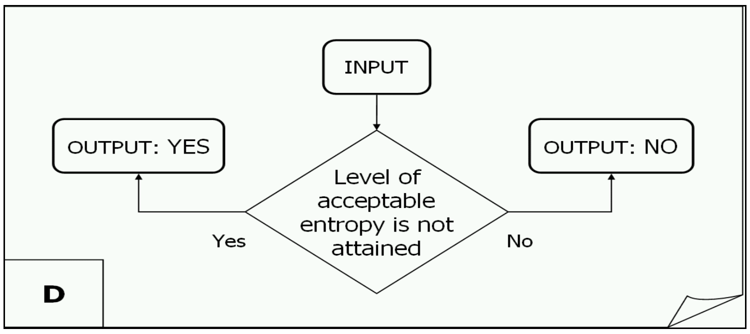


Figure 17. Component D for the termination block

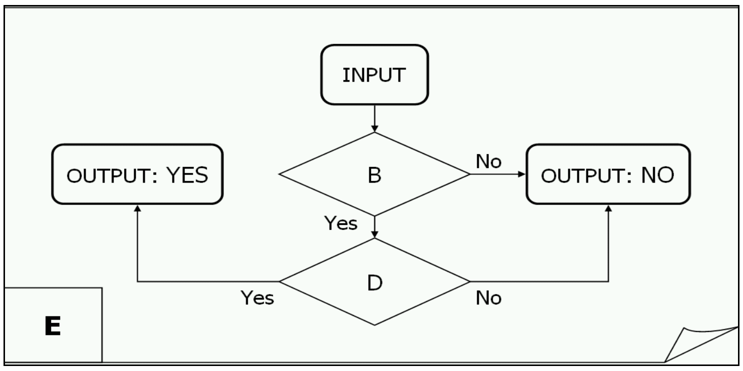


Figure 18. Component E for the termination block

In *model* 1, only one test after each application of quantum step block *UD* is needed. This test is performed by block *A*. So, the initialization includes assuming *A* to be *T*, i.e., function calls to *T* are addressed to block *A*. Block *A* is shown in Fig. 13.

As shown in Fig. 13, the *A* block checks to see if the maximum number of iterations has been reached, if so, then the simulation is terminated, otherwise, the simulation continues.

In *model* 2, the simulation is stopped when the direction of modification of categories' values are changed. Model 2 uses the comparison of the current value of *vx* category with value *mvx* that represents this category value obtained in previous iteration:

(i) If *vx* is greater than *mvx*, its value is stored in *mvx*, the *vi* value is stored in *mvi*, and the termination block proceeding to the next quantum step;

(ii) If *vx* is less than *mvx*, it means that the *vx* maximum is passed and the process needs to set the current (final) value of *vx* := *mvx*, *vi* := *mvi*, and stop the iteration process. So, the process stores the maximum of *vx* in *mvx* and the appropriate iteration number *vi* in *mvi*. Here block *B*, shown in Fig. 14 is used as the main block of the termination process.

The block PUSH, shown in the Fig. 15 (a) is used for performing the comparison and for storing the *vx* value in *mvx* (*case a*). A POP block, shown in Fig. 15 (b) is used for restoring the *mvx* value (*case b*). In the PUSH block of Fig. 15 (a), *if* |*vx*| > |*mvx*|, then *mvx*:= *vx*, *mva*:= *va*, *mvi*:= *vi*, and the block returns true; otherwise, the block returns false.

In the POP block of Fig. 15 (b), if |*vx*| <= |*mvx*|, *then vx:= mvx, va:= mva*, and *vi*:= *mvi*.

The *model* 3 termination block checks to see that a predefined number of iterations do not exceed (using block *A* in Fig. 13):

– If the check is successful, then the termination block compares the current value of *vx* with *mvx*. If *mvx* is less than, it sets the value of *mvx* equal to *vx* and the value of *mvi* equal to *vi*. If *mvx* is less using the PUSH block, then perform the next quantum step;

– If the check operation fails, then (if needed) the final value of *vx* equal to *mvx*, *vi* equal to *mvi* (using the POP block) and the iterations are stopped.
– The *model* 4, the termination block uses a single component block *D*, shown in Fig. 17.

The *D* block compares the current Shannon entropy value with a predefined acceptable level. If the current Shannon entropy is less than the acceptable level, then the iteration process is stopped; otherwise, the iterations continue.

The *model* 5 termination block uses the *A* block to check that a predefined number of iterations do not exceeded. If the maximum number is exceeded, then the iterations are stopped. Otherwise, the *D* block is then used to compare the current value of the Shannon entropy with the predefined acceptable level. If acceptable level is not attained, then the PUSH block is called and the iterations continue. If the last iteration was performed, the POP block is called to restore the *vx* category maximum and appropriate *vi* number and the iterations are ended.

Fig. 19 shows measurement of the final amplitudes in the output state to determine the success or failure of the search.

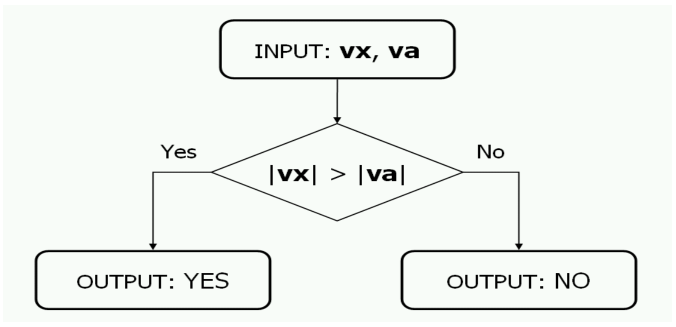


Figure 19. Final measurement emulation

If |*vx*| > |*va*|, then the search was successful; otherwise, the search was not successful.

Table 7 lists results of testing the optimized version of Grover QSA simulator on personal computer with Pentium 4 processor at 2GHz.

Table 7. High probability answers for Grover QSA

Qbits	**Iterations**	**Time**
32	51471	0.007
36	205887	0.018
40	823549	0.077
44	3294198	0.367
48	13176794	1.385
52	52707178	267
56	210828712	20.308
60	843314834	81.529
64	3373259064	328.274

The theoretical boundary of this approach is not the number of qubits, but the representation of the floating-point numbers. The practical bound is limited by the front side bus frequency of the personal computer. Using the above algorithm, a simulation of a 1000 qubit Grover QSA requires only 96 seconds for 10^8 iterations (see below).

The above approach can be used for simulation of the Deutsch-Jozsa's QA. The general schema of Deutsch-Jozsa's QA simulation is shown on Fig. 20, where an input state is provided to a quantum HUD block, which generates an output state.

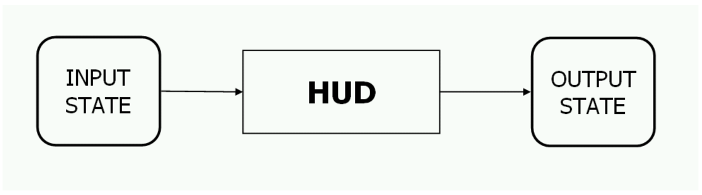


Figure 20. Generalized schema of simulation for Deutsch-Jozsa's QA

The structure of the HUD block is shown in Fig. 21, where the input is provided to an initialization block.

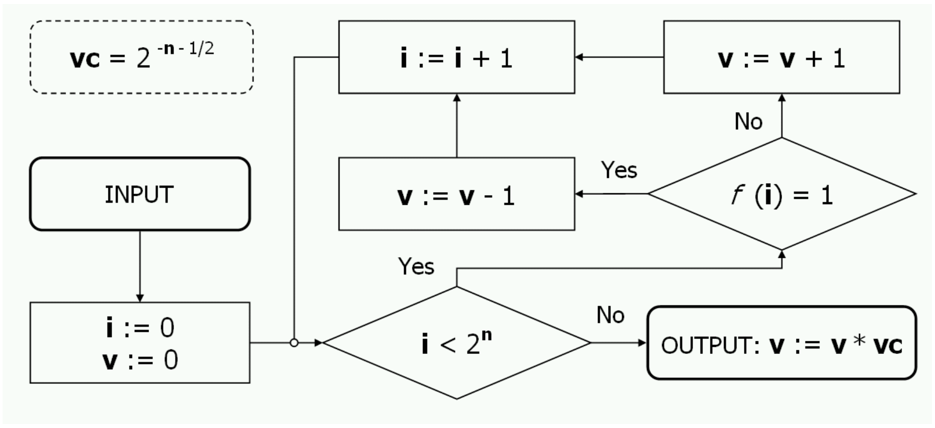


Figure 21. Quantum block HUD for Deutsch-Jozsa's QA

The initialization block sets *i*:= 0 and *v*:= 0, and then the process advances to a decision block. In the decision block, if $i < 2^n$, then the process advances to a decision block 3; otherwise, the process advances to an output block which outputs *v:= v*vc*, where $\mathrm{vc} = 2^{-n-1/2}$.

The quantum block HUD is applied only once to obtaining of the final state. Here *v* represents the vector |0…00> amplitude, f is an input function of order n, vc = $2^{-n\ -1/2}$ is a table value.

After applying the block HUD, the value of *v* is considered in correspondence with Table 8.

Table 8. Possible answers for Deutsch-Jozsa's problem

Value of v	**Answer**
0	*f* is balanced
$\frac{1}{\sqrt{2}}$	*f* is constant 0
$-\frac{1}{\sqrt{2}}$	*f* is constant 1
Otherwise	*f* is something else

References

1. Ulyanov S.V., Litvintseva L.V., Ulyanov I.S. et all. Quantum information and quantum computational intelligence: Classically efficient simulation of fast quantum algorithms (SW&HW implementations). – Note del Polo Ricerca. – Milano: Universita degli Studi di Milano Publ, 2004. – Vol. 79. – URL: http://www.qcoptimizer.com.
2. Ulyanov S.V. Efficient simulation system of quantum algorithms on classical computer based on fast algorithm: Patent US 2006/0224547 A1. – 2006.
3. Ulyanov S.V. Fast algorithm for efficient simulation of quantum algorithm gates on classical computer // Systemics, Cybernetics and Informatics. – 2004. – Vol. 2. – № 3. – Pp. 63-68.