On finding optimal quantum query algorithms using numerical optimization

Māris Ozols,
Laura Mančinska

University of Latvia,
29 Rainis Boulevard,
Riga LV-1459, Latvia

Abstract

We propose a method that can be used to construct a quantum query algorithm for the given Boolean function. This method is based on numerical optimization. We apply it to all 3 and 4 argument Boolean functions. We also show how one quantum query algorithm can be modified to compute other Boolean functions.

1. Quantum query algorithms

A query algorithm computes Boolean function by querying its arguments. The *complexity* of query algorithm is the number of queries made. A *quantum* query algorithm can query all arguments in a superposition. We consider oracle matrices of the following type:

$$O = \begin{pmatrix} (-1)^{x_1} & 0 & \cdots & 0 \\ 0 & (-1)^{x_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & (-1)^{x_n} \end{pmatrix}$$

Quantum query algorithm is a sequence of unitary transformations:

$$Q = U_m \cdot O \cdot U_{m-1} \cdot \ldots \cdot U_1 \cdot O \cdot U_0 \qquad (1)$$

and the final amplitude distribution is $Q\,|0\rangle$.

2. General $n\times n$ unitary matrix

One can use the so called Givens rotations to transform any unitary matrix U to a diagonal form

$$D = U \cdot \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} G_{ij} \qquad (2)$$

where D is diagonal unitary matrix, i.e. $d_{kl} = \delta_{kl} \exp(i\varphi_k)$. Givens rotation G_{ij} is an $n\times n$ identity matrix modified at positions (i,i), (i,j), (j,i) and (j,j). General Givens rotation is determined by a general 2×2 unitary matrix:

$$\begin{pmatrix} g_{ii} & g_{ij} \\ g_{ji} & g_{jj} \end{pmatrix} = \begin{pmatrix} e^{i(\delta+\sigma+\tau)}\cos\theta & e^{i(\delta+\sigma-\tau)}\sin\theta \\ -e^{i(\delta-\sigma+\tau)}\sin\theta & e^{i(\delta-\sigma-\tau)}\cos\theta \end{pmatrix}$$

If we multiply (2) from the right had side by the adjoints of G_{ij}, we obtain a formula for a general $n\times n$ unitary matrix U.

3. General quantum query algorithm

If we independently replace each of the $U_0, \ldots, U_m$ in (1) with a general unitary matrix, we obtain a general quantum query algorithm. We can obtain any specific quantum query algorithm $Q(x_1, x_2, \ldots, x_n, m)$ by substituting each of the $U_0, \ldots, U_m$ with an appropriate unitary matrix. $Q(x_1, x_2, \ldots, x_n, m)$ is a unitary matrix that depends on the input and on the number of queries made. The corresponding final amplitude distribution is

$$|\psi(x_1, x_2, \ldots, x_n, m)\rangle = Q(x_1, x_2, \ldots, x_n, m)\,|0\rangle$$

The result of computation is obtained by measuring $|\psi(x_1, x_2, \ldots, x_n, m)\rangle$ in some basis B. In order to obtain only 0 or 1 as the output, we divide the basis vectors of B into two parts – B_0 and B_1. Without the loss of generality we can assume that the measurement is performed in the standard basis and B_0 consists of the first b vectors of the standard basis.

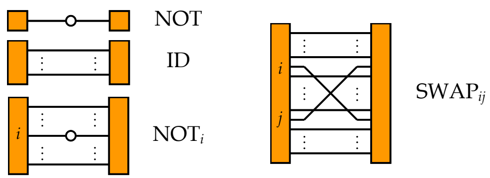
Definition Query algorithm *computes* a Boolean function f if it returns the correct answer with probability $> 1/2$ for each input.

By varying parameters b $(1 \le b \le n\text{-}1)$ and m $(1 \le m \le n\text{-}1)$ we obtain different query algorithm templates. For each template we perform a numerical optimization to find the best algorithm of this form. To obtain the best algorithm we maximize the worst case success probability.

4. NPN-equivalence

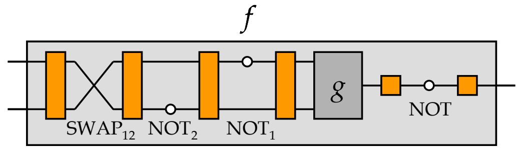
Definition The following logic gates are called *trivial gates*:

- NOT - negation,
- ID - identity transformation,
- NOT_i - inversion of i-th argument,
- SWAP_{ij} - swapping of i-th and j-th arguments.



Definition Two Boolean functions f and g are *NPN-equal* if a circuit for f can be made out of trivial gates and a circuit for g.

Example Boolean functions $f(x_1, x_2) = x_1 \vee x_2$ and $g(x_1, x_2) = x_2 \wedge x_1$ are *NPN-equal*:



The number of NPN-equivalence classes of Boolean functions of exactly n variables $F(n)$ (Sloane's A001528) is significantly less than the number of all Boolean functions:

n	0	1	2	3	4	5
$F(n)$	1	1	2	10	208	615 904
2^{2^n}	2	4	16	256	65 536	4 294 967 296

Theorem All NPN-equal Boolean functions have the same quantum query complexity.

5. Results

We computed all NPN-equivalence classes of three and four argument Boolean functions. We took a representative from each class and applied the method described in Section 3 to it. For three argument functions we found one NPN-equivalence class with quantum query complexity less than the deterministic one:

$$f = x_1 \Leftrightarrow x_2 \Leftrightarrow x_3,$$

Among four argument functions we found seven such classes:

$f_1 = x_1 \oplus x_2 \oplus x_3 \oplus x_4,$

$f_2 = (!x_1 \wedge !x_2 \wedge x_3 \wedge x_4) \vee (!x_1 \wedge x_2 \wedge !x_3 \wedge x_4) \vee (!x_1 \wedge x_2 \wedge x_3 \wedge !x_4) \vee (x_1 \wedge !x_2 \wedge !x_3 \wedge x_4) \vee (x_1 \wedge !x_2 \wedge x_3 \wedge !x_4) \vee (x_1 \wedge x_2 \wedge !x_3 \wedge !x_4),$

$f_3 = x_1 \Leftrightarrow x_2 \Leftrightarrow x_3 \Leftrightarrow x_4,$

$f_4 = (x_1 \Leftrightarrow x_2 \Leftrightarrow x_3) \vee (!x_1 \wedge x_3 \wedge x_4) \vee (x_1 \wedge !x_3 \wedge !x_4),$

$f_5 = (x_1 \Leftrightarrow x_2 \Leftrightarrow x_3 \Leftrightarrow x_4) \vee (!x_1 \wedge !x_2 \wedge x_3 \wedge x_4) \vee (x_1 \wedge x_2 \wedge !x_3 \wedge !x_4),$

$f_6 = (x_1 \Leftrightarrow x_2 \Leftrightarrow x_3) \vee (x_1 \Leftrightarrow x_2 \Leftrightarrow x_4) \vee (x_1 \Leftrightarrow x_3 \Leftrightarrow x_4),$

$f_7 = (x_1 \Leftrightarrow x_2) \vee (x_1 \wedge x_3 \wedge x_4) \vee (x_2 \wedge !x_3 \wedge !x_4).$