On finding optimal quantum query algorithms using numerical optimization

Laura Mančinska, **Māris Ozols**
Institute of Mathematics and Computer Science, University of Latvia
29 Rainis Boulevard, Riga LV-1459, Latvia
e-mail: *marozols@yahoo.com*

In this work we examine how numerical optimization can be used to create optimal quantum query algorithms [1]. For this purpose we created a program in *Mathematica 5.2* which constructs a general quantum query algorithm and then finds optimal values of parameters using numerical optimization. We applied our program to all 3 and 4 argument Boolean functions. To decrease the amount of computation (there are 2^{2^N} N-argument functions, see Table 1) we introduced the notion of symmetric functions.

Definition. *Trivial reductions* are following transformations of Boolean function: argument inversion, result inversion, swapping of two arguments.

Definition. *Primitive reduction* is a pair of sequences of trivial reductions - one sequence applied to arguments of function, other - to result.

Definition. Two Boolean functions are said to be *primitively equivalent*, if there exists a primitive reduction between them.

Theorem. If we have optimal quantum query algorithm for some Boolean function f, then we can transform it for any other primitively equivalent function g and obtained algorithm will also be optimal.

Thus we do not have to examine all Boolean functions, but only those which are not primitively equivalent. We are not interested in *degenerated* functions (which have arguments that does not influence the value of function). Number of such functions $F(N)$ is shown in Table 1.

To construct a general quantum query algorithm, we must first specify general $N \times N$ unitary matrix. We use a method similar to QR-factorization to construct it from simple two-level matrices [2]:

$$\prod_{i=1}^{N-1} \prod_{j=i+1}^{N} G_{ij}, \tag{1}$$

where G_{ij} is $N \times N$ identity matrix modified at positions $\left(\begin{smallmatrix} g_{ii} & g_{ij} \\ g_{ji} & g_{jj} \end{smallmatrix}\right)$. We replace elements at these positions either with general 2×2 unitary matrix (4 unknown real parameters: δ, σ, τ, and θ):

$$U = \begin{pmatrix} e^{i(\delta+\sigma+\tau)} \cos\theta & e^{i(\delta+\sigma-\tau)} \sin\theta \\ -e^{i(\delta-\sigma+\tau)} \sin\theta & e^{i(\delta-\sigma-\tau)} \cos\theta \end{pmatrix}, \tag{2}$$

N	1	2	3	4
$F(N)$	1	2	10	208
2^{2^N}	4	16	256	65536

Table 1: Number of different (up to primitive reduction) nondegenerated Boolean functions F(N).

or rotation matrix (1 unknown real parameter θ):

$$R = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}. \tag{3}$$

In the first case we obtain general unitary matrix, in the second - general orthogonal matrix. For N-argument function as oracle transformation we use

$$O = \begin{pmatrix} (-1)^{x_1} & 0 & \cdots & 0 \\ 0 & (-1)^{x_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & (-1)^{x_N} \end{pmatrix}. \tag{4}$$

Therefore general quantum query algorithm with L queries will have final amplitude distribution $G_L \cdot O \cdot G_{L-1} \cdot \ldots \cdot G_1 \cdot O \cdot G_0 \cdot |0\rangle$ which has $c\frac{n(n+1)}{2}$ unknown parameters (c is either 4 or 1). We varied the number of questions and the number of amplitudes being measured. In this way we found a class of 3-argument Boolean functions - f_1^3 and seven classes of 4-argument functions: $f_1^4, \ldots, f_7^4$, which can be computed by quantum query algorithm with fewer questions than in deterministic case:

$$f_1^3 = x_1 \Leftrightarrow x_2 \Leftrightarrow x_3, \quad f_1^4 = x_1 \oplus x_2 \oplus x_3 \oplus x_4,$$

$$f_2^4 = (!\,x_1 \wedge !\,x_2 \wedge x_3 \wedge x_4) \vee (!\,x_1 \wedge x_2 \wedge !\,x_3 \wedge x_4) \vee (!\,x_1 \wedge x_2 \wedge x_3 \wedge !\,x_4) \vee (x_1 \wedge !\,x_2 \wedge !\,x_3 \wedge x_4) \vee (x_1 \wedge !\,x_2 \wedge x_3 \wedge !\,x_4) \vee (x_1 \wedge x_2 \wedge !\,x_3 \wedge !\,x_4),$$

$$f_3^4 = x_1 \Leftrightarrow x_2 \Leftrightarrow x_3 \Leftrightarrow x_4,$$

$$f_4^4 = (x_1 \Leftrightarrow x_2 \Leftrightarrow x_3) \vee (!\,x_1 \wedge x_3 \wedge x_4) \vee (x_1 \wedge !\,x_3 \wedge !\,x_4),$$

$$f_5^4 = (x_1 \Leftrightarrow x_2 \Leftrightarrow x_3 \Leftrightarrow x_4) \vee (!\,x_1 \wedge !\,x_2 \wedge x_3 \wedge x_4) \vee (x_1 \wedge x_2 \wedge !\,x_3 \wedge !\,x_4),$$

$$f_6^4 = (x_1 \Leftrightarrow x_2 \Leftrightarrow x_3) \vee (x_1 \Leftrightarrow x_2 \Leftrightarrow x_4) \vee (x_1 \Leftrightarrow x_3 \Leftrightarrow x_4),$$

$$f_7^4 = (x_1 \Leftrightarrow x_2) \vee (x_1 \wedge x_3 \wedge x_4) \vee (x_2 \wedge !\,x_3 \wedge !\,x_4).$$

References

[1] Ronald de Wolf "Quantum Computing and Communication Complexity", Institute for Logic, Language and Computation (2001)

[2] George Cybenko "Reducing Quantum Computations to Elementary Unitary Operations", *Computing in Science & Engineering*, **3** (2001), N27, pages 27–32