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

We propose a method how 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, to find the functions with quantum query complexity smaller than the deterministic one. We also show that a given query algorithm constructed for some particular function can be modified to compute other Boolean functions.

1. QUANTUM QUERY ALGORITHMS

A *query* algorithm computes Boolean function $f(x_1, x_2, \ldots, x_n)$ by querying its arguments $x_i \in \{0, 1\}$. The *complexity* of query algorithm is the number of queries made to determine the value of f. A *quantum* query algorithm queries all arguments in a superposition [1]. This is achieved by applying an oracle matrix that is a function of input variables. We consider the following type of oracle matrices:

$$O(\vec{x}) = \begin{pmatrix} (-1)^{x_1} & 0 & \ldots & 0 \\ 0 & (-1)^{x_2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & (-1)^{x_n} \end{pmatrix}, \tag{1}$$

where $\vec{x} = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n$ is the input that is queried. *Quantum query algorithm* is a sequence of unitary transformations:

$$Q(\vec{x}) = U_m \cdot O(\vec{x}) \cdot U_{m-1} \cdot \ldots \cdot U_1 \cdot O(\vec{x}) \cdot U_0, \tag{2}$$

where U_i's are arbitrary unitary matrices (therefore Q is also unitary). The final amplitude distribution for input $\vec{x}$ is

$$|\psi(\vec{x})\rangle = Q(\vec{x})\,|0\rangle\,. \tag{3}$$

2. GENERAL $n \times n$ UNITARY MATRIX

One can use the *Givens rotations* [2] to find a diagonal form D ($d_{kl} = \delta_{kl} e^{i\varphi_k}$) of any unitary matrix U

$$D = U \cdot \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} G_{ij}. \tag{4}$$

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:

$$\begin{pmatrix} g_{ii} & g_{ij} \\ g_{ji} & g_{jj} \end{pmatrix} = \begin{pmatrix} e^{i(\delta+\sigma+\tau)} \cos\theta & e^{i(\delta+\sigma-\tau)} \sin\theta \\ -e^{i(\delta-\sigma+\tau)} \sin\theta & e^{i(\delta-\sigma-\tau)} \cos\theta \end{pmatrix}, \tag{5}$$

where $\delta, \sigma, \tau, \theta \in \mathbb{R}$ and g_{ii}, g_{ij}, g_{ji}, and g_{jj} are the corresponding elements of matrix G_{ij}. If we multiply (4) from the right had side by the adjoints of G_{ij}, we obtain a formula for a general $n \times n$ unitary matrix U. To specify it, we need n parameters φ_k for the diagonal matrix D and $n(n-1)/2$ quadruplets $(\delta, \sigma, \tau, \theta)$ for matrices G_{ij}. In total $n + 4n(n-1)/2 = 2n^2 - n \approx 2n^2$ parameters.

3. GENERAL QUANTUM QUERY ALGORITHM

If we replace U_i's in (2) with independent general unitary matrices (each matrix has its own parameters), we obtain a general quantum query algorithm Q with m queries. There are $m+1$ matrices U_i therefore $(2n^2 - n)(m+1) \approx 2mn^2$ parameters are required to specify Q.

The result of computation is obtained by measuring the state $|\psi(\vec{x})\rangle$ given by (3) in some basis $\mathcal{B}$. In order to obtain only 0 or 1 in the output, we divide the basis vectors of $\mathcal{B}$ into two parts $\mathcal{B}_0$ and $\mathcal{B}_1$. Without the loss of generality we can assume that the measurement is performed in the standard basis and $\mathcal{B}_0$ consists of the first b vectors of the standard basis.

4. FINDING AN OPTIMAL ALGORITHM

By varying b $(1 \leq b \leq n-1)$ and m $(1 \leq m \leq n-1)$ one obtains different query algorithm templates (the case $m \geq n$ is not interesting, as the deterministic complexity does not exceed n). For each template one must perform a numerical optimization to find the best algorithm of this form.

Definition. Query algorithm *computes a Boolean function f with probability P*, if for each input $\vec{x}$ it returns the correct value of $f(\vec{x})$ with probability at least P. P is *the worst case success probability.*

The best algorithm is obtained by maximizing the worst case success probability. We say that the quantum query algorithm has an advantage over the deterministic one for a particular function, if $m < n$ and $P > 1/2$.

5. NPN-EQUIVALENCE

We observed that similar Boolean functions can be computed with similar query algorithms. For example, if an algorithm for function f is available, it is not hard to compute the negation of f. The same stands for functions $f(x_1, x_2)$ and $f(x_2, x_1)$. The notion of similarity can be made formal and is called *NPN-equivalence* [3–5]. The idea is to use simple logic gates to transform one Boolean function to other.

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

- NOT - negation,
- ID - identity transformation,

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

Table 1: The number of NPN-equivalence classes of Boolean functions of exactly n variables $F(n)$ compared to the number of all Boolean functions.

- NOT_i - negation 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, because $f = \mathrm{SWAP}_{12} \circ \mathrm{NOT}_2 \circ \mathrm{NOT}_1 \circ g \circ \mathrm{NOT}$.

Lemma. The NPN-equivalence of Boolean functions is *equivalence relation* (reflexive, symmetric, transitive).

Theorem. All NPN-equal Boolean functions have the same quantum query complexity and a quantum query algorithm, that is designed for one of these functions, with slight modifications can be used for others.

It means that in order to have a full set of query algorithms for all n-argument boolean functions one has to construct only an algorithm for each of the equivalence classes. We are interested only in Boolean functions with *exactly n variables* - it means that the functions depends on all variables (for example, $f(x_1, x_2) = x_1$ does not). Let us denote the number of NPN-equivalence classes of such functions by $F(n)$. It is significantly smaller than the total number of all n-argument Boolean functions (see Table 1 for comparison). $F(n)$ corresponds to Sloane's A001528.

6. RESULTS

We computed representatives for all NPN-equivalence classes of three and four argument Boolean functions and applied the method described in Section 4 to them. In case of three argument functions we found one NPN-equivalence class for which the quantum query complexity is smaller than the deterministic one:

$$f = x_1 \Leftrightarrow x_2 \Leftrightarrow x_3. \tag{6}$$

Among four argument functions we found seven such classes:

$$
\begin{aligned}
f_1 =& x_1 \oplus x_2 \oplus x_3 \oplus x_4, \\
f_2 =& (!\, x_1 \wedge !\, x_2 \wedge x_3 \wedge x_4) \vee (!\, x_1 \wedge x_2 \wedge !\, x_3 \wedge x_4) \vee (!\, x_1 \wedge x_2 \wedge x_3 \wedge !\, x_4) \vee \\
& (x_1 \wedge !\, x_2 \wedge !\, x_3 \wedge x_4) \vee (x_1 \wedge !\, x_2 \wedge x_3 \wedge !\, x_4) \vee (x_1 \wedge x_2 \wedge !\, x_3 \wedge !\, x_4), \\
f_3 =& x_1 \Leftrightarrow x_2 \Leftrightarrow x_3 \Leftrightarrow x_4, \\
f_4 =& (x_1 \Leftrightarrow x_2 \Leftrightarrow x_3) \vee (!\, x_1 \wedge x_3 \wedge x_4) \vee (x_1 \wedge !\, x_3 \wedge !\, x_4), \\
f_5 =& (x_1 \Leftrightarrow x_2 \Leftrightarrow x_3 \Leftrightarrow x_4) \vee (!\, x_1 \wedge !\, x_2 \wedge x_3 \wedge x_4) \vee (x_1 \wedge x_2 \wedge !\, x_3 \wedge !\, x_4), \\
f_6 =& (x_1 \Leftrightarrow x_2 \Leftrightarrow x_3) \vee (x_1 \Leftrightarrow x_2 \Leftrightarrow x_4) \vee (x_1 \Leftrightarrow x_3 \Leftrightarrow x_4), \\
f_7 =& (x_1 \Leftrightarrow x_2) \vee (x_1 \wedge x_3 \wedge x_4) \vee (x_2 \wedge !\, x_3 \wedge !\, x_4).
\end{aligned}
$$

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, pp. 27-32.

[3] Zeljko Zilic, Zvonko Vranesic, *Using BDDs to Design ULMs for FPGAs*, Proc. 4th International Symposium on FPGAs, Monterey CA, Feb. 1996, pp. 97–103.

[4] Anas Al-Rabadi, Martin Zwick, *Enhancements to Crisp Possibilistic Reconstructability Analysis*, International Journal of General Systems, August 2004, Vol. 33 (4), pp. 361-382.

[5] Jerry Chih-Yuan Yang, Giovanni De Micheli, *Spectral Techniques for Technology Mapping*, Stanford University, Stanford, CA, USA (1994).