Eur. Phys. J. Plus (2016) **131**: 38
DOI 10.1140/epjp/i2016-16038-2

THE EUROPEAN PHYSICAL JOURNAL PLUS

Regular Article

Quantum search of a real unstructured database

Bogusław Broda[a,b]
Department of Theoretical Physics, Faculty of Physics and Applied Informatics, University of Łódź, 90-236 Łódź, Pomorska 149/153, Poland

Received: 19 November 2015 / Revised: 21 December 2015
Published online: 19 February 2016

Abstract. A simple circuit implementation of the oracle for Grover's quantum search of a real unstructured classical database is proposed. The oracle contains a kind of quantumly accessible classical memory, which stores the database.

1 Introduction

Quantum counterparts of some classical algorithms can substantially speed up various computational tasks [1]. In particular, the famous Grover search algorithm [2], which is a quantum counterpart of classical search, achieves quadratic speed-up in terms of oracle queries. Namely, if N is the number of elements in a search space, to find a single distinguished element with probability $\mathcal{O}(1)$, in the quantum case one should query the oracle only $\mathcal{N}_{\text{Grover}} = \mathcal{O}(\sqrt{N})$ times, rather than $\mathcal{N}_{\text{classical}} = \mathcal{O}(N)$ times, as classically expected.

Originally, the Grover search algorithm was called the database search algorithm, but the word "database" was later dropped. To retain the term "database search" one should distinguish two qualitatively distinct categories of databases [3–5]: the real (actual or explicit) database, which is a database in conventional meaning, and the virtual (abstract or implicit) database, which is a search space in the meaning of [1]. Roughly speaking, the real database represents data stored in a physical memory device, whereas the virtual one is not actually a database at all. Evidently, most of the existing literature is devoted to quantum search of virtual databases ([6] is one of the few notable exceptions), instead in the present work we will be exclusively focused on the quantum search of real unstructured classical databases. We should stress that we leave aside all the potentially important and interesting issues concerning practical aspects of such quantum searches or their possible usefulness (for a discussion on such matters, see first of all [3], and possibly also [4–6]).

More precisely, the aim of our work is to present a simple model, in the form of a quantum circuit, which implements the standard Grover search algorithm for the searching of a real unstructured classical database. Actually, the only unknown part of the circuit is the oracle, which is, in the framework of the general Grover search, a black box, but here it has to be explicitly defined. In particular, the oracle should contain, as its main component, a kind of quantumly accessible memory, which stores the database. The architecture of such a memory will be proposed.

2 Grover's search for real databases

The starting point of the standard Grover search algorithm is the equal superposition state [1]

$$|\psi\rangle = H^{\otimes n} |0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle \,, \tag{1}$$

where H is the Hadamard operation, and $N = 2^n$ is the dimension of the Hilbert space —the number of elements in a search space. A particular instance of the search problem can conveniently be represented by an oracle function f, which takes as input an integer x, in the range from 0 to $N-1$. By definition, $f(x) = 1$, if x is a solution to the search problem, and $f(x) = 0$, if x is not a solution to the search problem. Next, as a result of repeated executions of the Grover search operation, the state $|\psi\rangle$ goes towards the uniform superposition of solution states [1].

[a] e-mail: bobroda@uni.lodz.pl
[b] http://merlin.phys.uni.lodz.pl/BBroda

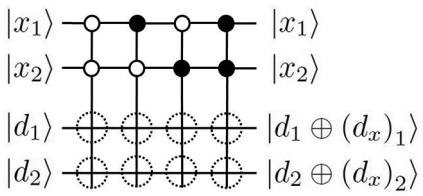

Fig. 1. Architecture of the physical memory device entering the oracle. The memory consists of an array of $N = 2^{n=2} = 4$ columns (registers), which are generalized Toffoli gates. Each column stores a single $m = 2$ bit number. In the column k ($k = 1, 2, 3, 4$), a vertical sequence of white and black dots is fixed, and it binary represents the number $k-1$, where the white dot corresponds to 0, whereas the black dot corresponds to 1, respectively. The vertical sequence of dotted $\oplus$'s is database dependent, and it binary represents the number (record) d_{k-1}.The dotted $\oplus$ denotes the possibility of the presence of the actual $\oplus$ in a given place. More precisely, 1 corresponds to $\oplus$, whereas 0 corresponds to the lack of $\oplus$ (symbolically "$\cdot$"), respectively. This circuit realizes the `LOAD` operation.

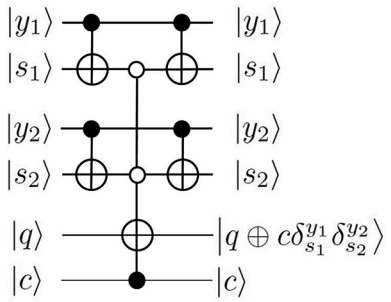

Fig. 2. For $c = 1$, the circuit computes the s-dependent oracle function $f_{(s_1 s_2)}(y_1 y_2) = \delta_{s_1}^{y_1}\delta_{s_2}^{y_2}$. Otherwise, *i.e.* for $c = 0$, the circuit computes the constant function $f_{(s_1 s_2)}(y_1 y_2) = 0$.

In the case of a real unstructured database search, for definiteness, we will assume, as a starting point, a very natural scheme proposed in chapt. 6.5 of [1]. Namely, suppose we have a database containing N records, each of length m bits. We will label these records $d_0, \ldots, d_{N-1}$. The aim is to determine where a particular m bit string, s, is in the database. First, let us consider the result of the "`LOAD`" operation defined in [1]

$$|x\rangle |0\rangle^{\otimes m} |s\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}} \xrightarrow{\texttt{LOAD}} |x\rangle |d_x\rangle |s\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}}, \tag{2}$$

which reveals the data d_x located at the address x. In the next step, the second and third registers should be compared, and if they are the same, then a bit flip is applied to register 4; otherwise nothing is changed. The effect of this operation is [1]

$$|x\rangle |d_x\rangle |s\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}} \to \begin{cases} -|x\rangle |d_x\rangle |s\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}}, & \text{if } d_x = s, \\ |x\rangle |d_x\rangle |s\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}}, & \text{if } d_x \neq s. \end{cases} \tag{3}$$

The part of the circuit (actually, of the oracle) performing the operation (2) is a kind of memory storing the database. The memory is supposed to store classical data, in principle, but it should be possible to access it quantumly. The proposed example architecture of such a memory is depicted in fig. 1. The memory consists of an array of N columns (registers), which are generalized Toffoli gates. Each column stores a single m bit number. The upper part of the circuit plays the role of a converter, and it translates the number x ($x = 0, 1, \ldots N-1$) from binary to unary numeral system, "activating" the column (register) $k = x + 1$, whereas the lower part is a proper memory store. The unitary transformation performed by the `LOAD` operation in fig. 1, which depends on the classical database $\{d_x\}_{x=0}^{N-1}$, is (compare to [6–8])

$$|x\rangle |d\rangle \to |x\rangle |d \oplus d_x\rangle . \tag{4}$$

In turn, the part of the oracle which performs the operation (3), and formally corresponds to the s-dependent oracle function

$$f_{(s)}(y) = \delta_s^y, \tag{5}$$

is implemented by the circuit presented in fig. 2. Strictly speaking, the function (5) is computed, provided $c = 1$, otherwise, *i.e.* for $c = 0$, $f_{(s)}(y) = 0$. One should note that for the proper functioning of the memory (oracle), the second register in (2) should be restored to its initial value $|0\rangle^{\otimes m}$ after each memory usage (Grover's iteration). To

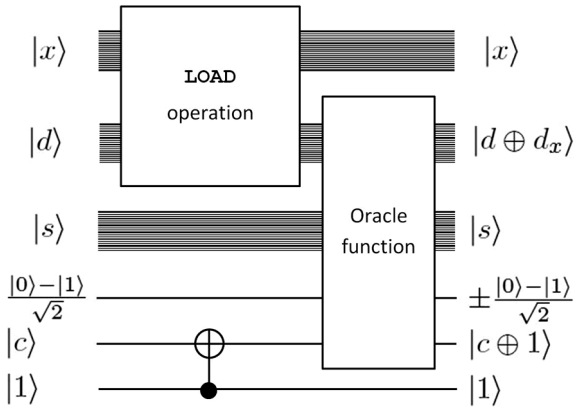

Fig. 3. Block scheme of the whole oracle. The oracle consists of the physical memory device proposed in fig. 1, the oracle function circuit presented in fig. 2, and an additional auxiliary CNOT gate controlling the actual execution of the oracle function $f_{(s)}(\cdot)$.

this end, making use of the formula $|d \oplus d_x \oplus d_x\rangle = |d\rangle$, an additional `LOAD` operation should be performed, which must not be followed by the computation of the oracle function ($f_{(s)}(0)$, in this case). Therefore, the oracle should consist of the circuits defined in figs. 1 and 2, as well as an additional auxiliary CNOT gate controlling the actual execution of the oracle function $f_{(s)}(\cdot)$ (see, fig. 3). Consequently, the total number of oracle queries is effectively doubled, but only odd queries contribute to the evolution of the state $|\psi\rangle$, whereas even ones restore the second register of the memory to its proper initial value $|0\rangle^{\otimes m}$. Thus, the whole Grover algorithm should be initialized with the following set of initial states (fig. 3):

$$\begin{aligned} &|\psi\rangle = H^{\otimes n}\,|0\rangle^{\otimes n}\,,\\ &|d\rangle = |0\rangle^{\otimes m}\,,\\ &|s\rangle = |s_m \cdots s_1\rangle\,,\\ &\frac{|0\rangle - |1\rangle}{\sqrt{2}},\\ &|c\rangle = |0\rangle\,,\\ &|1\rangle\,. \end{aligned} \tag{6}$$

3 Conclusions

In the present work, we have proposed (fig. 3) a simple quantum circuit implementation of the oracle for the standard Grover algorithm adapted for searching an unstructured real classical N-record ($N = 2^n$) database. The oracle contains a kind of quantumly accessible classical memory, which stores the database. It appears that for the proper functioning of the memory, implying the restoring of the initial value $|d\rangle = |0\rangle^{\otimes m}$, the number of oracle queries should be doubled, *i.e.* $\mathcal{N} = 2\mathcal{N}_{\mathrm{Grover}}$. But one should note that since the oracle is now explicitly defined, the number $\mathcal{N}$ loses its primary role as a measure of computational complexity.

As a byproduct of our construction (fig. 3), we have proposed the architecture of a physical memory device (fig. 1), which could be compared to the qRAM (quantum Random Access Memory) introduced and extensively discussed in [7–10]. Actually, only the `LOAD` operation, necessary for the realization of the Grover algorithm, has been defined and quantumly implemented in our circuit. The lacking `STORE` operation, performed before the Grover algorithm starts, and consisting in setting the dotted $\oplus$'s (fig. 1) in appropriate positions, actual $\oplus$ or "$\cdot$", respectively, could possibly be implemented classically. It is interesting to note that our quantum memory is not so demanding (no exponentially large tensor products) as anticipated for conventional implementations in [7,9]. Namely, our memory operates on tensor products of only $\mathcal{O}(n)$ states, as the bucket-brigade architecture of qRAM promises, rather than on exponent of this number. Incidentally, the circuit of the bucket-brigade qRAM proposed in [8] operates on exponential number of states (input states are tensored with $|1\rangle \otimes |0\rangle^{\otimes 2^n - 1}$). In turn, our memory addressing `LOAD` operation requires an exponential number $N = 2^n$ of generalized Toffoli gates being executed at each oracle call. The addressing scheme proposed in subsect. 6.5 of [1] requires $\mathcal{O}(N \log N)$ quantum switches. Instead, the bucket brigade qRAM scheme [7–10] is supposed to activate only $\mathcal{O}(\log^2 N)$ switches in physical implementations [7], but the proposed circuit [8] still consists of $\mathcal{O}(N)$ gates.

The author has been supported by the University of Łódź.

References

1. M.A. Nielsen, I.L. Chuang, *Quantum Computation and Quantum Information* (Cambridge University Press, 2010).
2. L.K. Grover, Phys. Rev. Lett. **79**, 325 (1997).
3. C. Zalka, Phys. Rev. A **62**, 052305 (2000).
4. C.P. Williams, Comput. Sci. Eng. **3**, 44 (2001).
5. G.F. Viamontes, I.L. Markov, J.P. Hayes, Comput. Sci. Eng. **7**, 62 (2005).
6. D. Reitzner, M. Ziman, Eur. Phys. J. Plus **129**, 1 (2014).
7. V. Giovannetti, S. Lloyd, L. Maccone, Phys. Rev. Lett. **100**, 160501 (2008).
8. S. Arunachalam, V. Gheorghiu, T. Jochym-O'Connor, M. Mosca, P.V. Srinivasan, in *10th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2015), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2015*, edited by S. Beigi, R. Koenig, *Leibniz International Proceedings in Informatics (LIPIcs)*, Vol. **44** (2015) pp. 226–244, ISBN 978-3-939897-96-5, ISSN 1868-8969, `http://drops.dagstuhl.de/opus/volltexte/2015/5559`.
9. V. Giovannetti, S. Lloyd, L. Maccone, Phys. Rev. A **78**, 052310 (2008).
10. F.-Y. Hong, Y. Xiang, Z.-Y. Zhu, L.-Z. Jiang, L.-N. Wu, Phys. Rev. A **86**, 010306 (2012).