Initialisation of quantum registers based on probability distribution

Jarosław Adam Miszczak*

Institute of Theoretical and Applied Informatics,
Polish Academy of Sciences,
Bałtycka 5, 44-100 Gliwice, Poland
`miszczak@iitis.gliwice.pl`

Abstract. Quantum computation is about generating interesting probability distributions using subtle rules of quantum mechanics. In this paper we provide a procedure which, for a given list of integer numbers, generates quantum circuit preparing superposition of those numbers and thus provides algorithmic way of preparing initial quantum state. This procedure may be regarded as a generalisation of classical variable initialisation and can be used in the implementation of the high level quantum programming language.

1 Motivation

Quantum algorithms [1–4] and communication protocols [5, 6] are described using a language of quantum circuits [7]. While this method is convenient for simple algorithms, it is very hard to operate on abstract data types using this notation [8].

This lack of data types and control structures motivated development of the quantum pseudocode [9, 10] and quantum programming languages [11–14].

Unfortunately existing languages describing quantum computation are based on mathematical formalism of quantum theory *i.e.* state vectors or density matrices and they do not provide sufficient level of abstraction which can be seen in classical programming languages.

Main contribution of this paper is to present how specific quantum state can be constructed from the specification of the measurement results *i.e.* using only information for classical device controlling quantum machine.

We describe an algorithm for generating a quantum circuit which prepares flat superposition of the list of integers. This can be understand as a quantum version of variable initialisation in programming languages. Presented algorithm describe generation of code for quantum machine (*i.e.* quantum gates) by classical controlling device.

Generated circuits can be interpreted as implementations of the variant of the quantum searching algorithm [1, 2]. For a given classical input, we output unitary operation which encodes it in the probabilities of the measurement outcomes.

2 Preparing flat superposition

We start by introducting notation. In what follows we will use numbers which can be written using M bits. By a_i^k we denote i-th bit of the number a^k.

As an analogue of a classical bits, qubits are described by systems with 2 base states. We introduce gate G, which can be used to accomplish the following task:

Problem 1. *Starting from the state $|0\rangle$ prepare the superposition $\sqrt{p}|0\rangle + \sqrt{1-p}|1\rangle$ with $0 \le |p| \le 1$.*

It is easy to see that the desired state can be reached by using R_y rotation [7]

$$R_y(\theta) = \begin{pmatrix} \cos(\theta/2) & \sin(\theta/2) \\ -\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}, \tag{1}$$

with parameter

$$\theta = 2\arctan\sqrt{\frac{p}{1-p}}. \tag{2}$$

This rotation is equivalent to

$$G^<(p) = R_y\left(2\arctan\sqrt{\frac{p}{1-p}}\right) \tag{3}$$

$$= \begin{pmatrix} \sqrt{1-p} & \sqrt{p} \\ -\sqrt{p} & \sqrt{1-p} \end{pmatrix}, \tag{4}$$

and we have clear probabilistic interpretation of the parameter p as a probability of measuring system in the state $|1\rangle$.

Since for $p = 1$ all we need is to perform $\text{Not} = \sigma_x$ gate, in what follows we use quantum gate defined as

$$G(p) = \begin{cases} G^<(p), & 0 \le p < 1 \\ \sigma_x, & p = 1 \end{cases}. \tag{5}$$

Since more than one qubit is needed to perform useful quantum computation, we can formulate our task as follows

Problem 2. *For a given set of integers $A = \{a^1, a^2, \ldots, a^K\}$ output unitary operation $R(A)$ such that*

$$R(A)|0\ldots0\rangle = \sum_{i=1}^{K} |a^i\rangle. \tag{6}$$

Here A may be regarded as a probability distribution P on the set $\{0, \ldots, 2^M - 1\}$ such that

$$\text{P}(a) = \begin{cases} \frac{1}{K}, & a \in A \\ 0, & a \notin A \end{cases}, \tag{7}$$

where K is a number of elements in A.

Such defined task can be regarded as a variant of the quantum searching algorithm [2]. In this case our goal is to prepare probability distribution, not only amplify chosen probability.

One should note that the procedure defined below uses only classical data to generate unitary evolution. As such it does not require any information about the quantum state [15].

3   Generating unitary matrix

In this section we describe a procedure which can be used to solve **Problem 2**.

It is convenient to divide our procedure into two parts – first we process bits on the first position and then the other bits.

We assume that the initial state of the systems is $|\phi_0\rangle = |0\ldots0\rangle \in \mathbb{C}^{2^M}$.

3.1   Part 1 – first qubit

Operation for the first qubit depends only on the information in the first bits $a_1^1, \ldots, a_1^K$ of the input numbers . If by t_1 we denote the number of occurrence of 1 at the first position, the first operation is defined as rotation

$$R_1 = G(t_1/K) \otimes \mathbb{1}^{M-1}. \tag{8}$$

The resulting state is

$$|\psi_1\rangle = R_1|0\ldots0\rangle \tag{9}$$

$$= \left(R_y \left(2 \arctan \sqrt{\frac{t_1}{K - t_1}} \right) |0\rangle \right) \otimes |\underbrace{0\ldots0}_{M-1}\rangle \tag{10}$$

$$= (\sqrt{1 - t_1/K}|0\rangle + \sqrt{t_1/K}|1\rangle) \otimes |\underbrace{0\ldots0}_{M-1}\rangle. \tag{11}$$

3.2   Part 2 – qubits $2, \ldots, M$

Unitary gates to be performed on the k-th qubit depend on the information in the bits $1, \ldots, k-1$. At the k-th step of the procedure we process k-th bits of the input numbers.

To generate appropriate operation we introduce

- $c_k(\alpha_j)$ – number of occurrences of the control α_j for the bit number k,
- $t_k(\alpha_j)$ – number of occurrences of 1 at the k-th bit of the input numbers for control α_j.

Using information in $c_k(\alpha_j)$ and $t_k(\alpha_j)$ we can define controlled gates

$$X_k(\alpha_j) = |\alpha_j\rangle\langle\alpha_j| \otimes G\left(\frac{t_k(\alpha_j)}{c_k(\alpha_j)}\right) \tag{12}$$

$$+ \left(\mathbb{1}^k - |\alpha_j\rangle\langle\alpha_j|\right) \otimes \mathbb{1}, \tag{13}$$

which represent rotations in subspaces defined by the operators $|\alpha_j\rangle\langle\alpha_j|$.

Here $\mathbb{1}^n$ is the identity operation on n qubits and α_j are the appropriate controls.

Operation R_k generated in the k-th step of our procedure reads

$$R_k = \left(\prod_{\alpha_j} X_k(\alpha_j)\right) \otimes \mathbb{1}^{M-k-1}. \tag{14}$$

Quantum gate R, which is an output of the procedure, is defined as

$$R = \prod_{l=1}^{M} R_l. \tag{15}$$

4   Discussion

It is easy to see by induction that operation R produces appropriate output gate, required to obtain state defined in 2. First step affects only first qubit and the rotation in the n-th step is performed only in the subspace characterised by the projection operators $|\alpha_j\rangle\langle\alpha_j|$.

The proposed procedure does not require to operate on all qubits and does not depends on information from the state vector [15]. Clearly it is easy to prepare desired state by starting from the any other base state and this modification requires at most M additional σ_x gates in the first step.

Gate R_l generated in the l-th step is composed of at most K rotations, each one controlled by the state of at most $l-1$ qubits. This operation can be decomposed using $2^{(l-1)}$ CNOT gates (see however [16]), thus output gate R can be decomposed using $\frac{K}{2}(2^{M+1} - 1)$ CNOT gates.

It is possible to reduce the number of CNOT gates needed by observing that qubit k is independent from the rest of the system during computation, when in the all input numbers k-th bit is identical.

If the input numbers have equal bits on m control positions (*i.e.* bits $1, \ldots, k-1$), number of CNOT gates required to perform operation R_k equals $K2^{k-1-m}$, and in this situation realization of R requires on average $\frac{K}{2^{m+1}}\left(2^{M+1} - 1\right)$ CNOT gates.

5   Final remarks

Presented algorithm generates a quantum circuit for preparing flat superposition of the list of integers and thus show that such operation in high level programming languages can be implemented unitarily. This introduces means for using

integer numbers as quantum data types [17]. Similar concepts were developed using fuzzy numbers [18].

Generation of code for quantum machine by classical controlling device is one of the translations phase for quantum language compiler [14]. Presented algorithm allows for optimisation of this procedure and thus could be used for code generation in quantum programming language compiler.

6 Acknowledgements

References

1. Grover, L.K.: Quantum computers can search rapidly by using almost any transformation. Phys. Rev. Lett. **80** (1998) 4329–4332
2. Grover, L.K.: Quantum mechanics helps in searching for a needle in a haystack. Phys. Rev. Lett. **79** (1997) 325–328
3. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press (1994) 124–134
4. Mosca, M.: Quantum Computer Algorithms. PhD thesis, Wolfson College, University of Oxford (1999)
5. Bennett, C.H., Brassard, G.: Quantum cryptography:public-key distribution and coin tossing. In: Proceedings of IEEE International Conference on Computers, Systems and Signal Processing. (1984) 175–179
6. Brassard, G., Broadbent, A., Tapp, A.: Quantum pseudo-telepathy. Found. Phys. **35** (2005) 1877–1907
7. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press (2000)
8. Shor, P.W.: Progress in quantum algorithms. Quantum Information Processing **3**(1-5) (2004)
9. Knill, E.: Conventions for quantum pseudocode. Technical Report LAUR-96-2724, Los Alamos National Laboratory (1996)
10. Knill, E.H., Nielsen, M.A.: Theory of quantum computation. In: Encyclopedia of Mathematics, Supplement III. Kluwer (2002)
11. Bettelli, S., Serafini, L., Calarco, T.: Toward an architecture for quantum programming. Eur. Phys. J. D **25**(2) (2003) 181–200
12. Mauerer, W.: Semantics and simulation of communication in quantum programming. Master's thesis, University Erlangen-Nuremberg (2005)
13. Oemer, B.: Structured Quantum Programming. PhD thesis, Technical University of Vienna (2003)
14. Svore, K.M., Cross, A.W., Chuang, I.L., Aho, A.V., Markov, I.L.: A layered software architecture for quantum computing design tools. IEEE Computer (January 2006) 74–83
15. Mottonen, M., Vartiainen, J.J., Bergholm, V., Salomaa, M.M.: Transformation of quantum states using uniformly controlled rotations. Quantum Information & Computation **5**(6) (2005)

16. Ralph, T.C., Resch, K.J., Gilchrist, A.: Efficient toffoli gates using qudits. Phys. Rev. A **75** (2007) 022313
17. Vedral, V., Barenco, A., Ekert, A.: Quantum networks for elementary arithmetic operations. Phys. Rev. A **54** (1996) 147
18. D'Hooghe, B., Pykacz, J., Zapatrin, R.R.: Quantum computation of fuzzy numbers. Int. J. Theor. Phys. **43**(6) (2004) 1423–1432