Quantum algorithm for non-local coordination

Nikolay Raychev

Abstract - In this report is considered an algorithm for quantum non-local coordination. The algorithm for non-local coordination have been implemented as a quantum logic circuit, which include paired qubits, which can be represented as unitary matrices. The solution can be found through the constraints, on which these matrices must comply. The proposed quantum circuit opens new perspectives for more efficient methods for non-local quantum coordination.

Index Terms— boolen function, circuit, composition, encoding, gate, quantum.

1 INTRODUCTION

In the recent years with the help of the geometric representation of the distribution of the probabilities the understanding for the quantum nonlocallity was significantly improved. The local distributions forms a closed convex set with a finite number of extreme points, the aspects of this polyhedron determine narrow BELL inequalities. In order to characterize the properties of the non-local quantum correlation, associated with the proposed circuit, it is important to classify all possible non-local correlations, each of which theoretically could be implemented from two or more sides, without allowing any communication.

In the proposed algorithm for non-local coordination as a substitute for the classical necessity of communication is used a quantum entanglement. The quantum entanglement cannot be used for direct transmission of information between distant sides. And yet it can be used to reduce the amount of the communication, requiring the processing of different distributed computational tasks. Here we present a simple logical quantum circuit with distributed logic. The proposed quantum circuit could be used for detection of other practical applications based on non-local synchronization.

Given that the quantum entanglement allows for rapid reduction of the required volume of classical information at communication, upon the implementation of some distributed computational tasks it is natural to be thought in the direction of full elimination of the need of communication. In other words, there are distributed tasks that would be impossible to be performed in the classical world, if the participants are not allowed to communicate, but these tasks can be performed without any form of communication, provided that they share the information by a preliminary quantum entanglement.

2 QUANTUM LOGARITHM FOR NON-LOCAL COORDINATION

This simulation demonstrates one of the possible applications of the quantum logical circuits. The purpose of the particular algorithm is two isolated one from another objects to determine the coordinates of the target position in a 3 × 3 matrix, provided that the first object has a information for the row, and the second - for the column.

The rules are as follows:

- Topological grid - 3 × 3 matrix
- Two participating sides (A and B), which are isolated from one another.
- A and B has by 2 attempts to find the target position.
- A receives information for the row, in which is located the target position.
- B receives information for the column, in which is located the target position.

Design

Quantum circuits:

- Lines = vectors/qubits
- Operators = Matrices

For obtaining the unknown and non-local values is necessary both sides to share the paired state of two pairs of qubits :

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|1,1,1,1\rangle - |0,1,0,1\rangle - |1,0,1,0\rangle|0,0,0,0\rangle)$$

The first two qubits belong to A, and the last two to B. Upon obtaining of their inputs χ and γ, A *and* B apply unitary transformations A_χ and B_γ, in accordance with the following matrices.

$$A_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} i & 0 & 0 & 1 \\ 0 & -i & 1 & 0 \\ 0 & i & 1 & 1 \\ 0 & 0 & 1 & i \end{bmatrix},$$

$$A_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} i & 1 & 1 & i \\ -i & 1 & -1 & i \\ i & 1 & -1 & -i \\ -i & 1 & 1 & -i \end{bmatrix},$$

$$A_3 = \frac{1}{\sqrt{2}}\begin{bmatrix} -1 & -1 & -1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & -1 & -1 & -1 \end{bmatrix}$$

$$B_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} i & -i & 1 & 1 \\ -i & -i & 1 & -1 \\ 1 & 1 & -i & i \\ -i & i & 1 & 1 \end{bmatrix},$$

$$B_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} -1 & i & 1 & i \\ 1 & i & 1 & -i \\ 1 & -i & 1 & i \\ -1 & -i & 1 & -i \end{bmatrix},$$

$$B_3 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 0 & 0 & 1 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 \end{bmatrix}$$

Pairing

Both participating sides *A and B* use two pairs of paired qubits as a resource for their circuits.

The generation of the superpositions is done identically with the analogous realization in the quantum logical circuits.

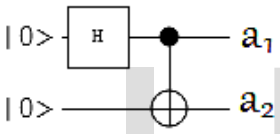


Figure 1: The generation of a superposition

The input of the circuit is initialized as switched off. Then in order to randomize the state of the first wire is applied the Hadamard operator. Both lines are connected by applying a XOR on the first bit and a controlled NOT operator on the second. In this way both wires are paired in a superposition containing all possible states. More precisely, the system ends up in the superposition $\frac{1}{\sqrt{2}}(|1,1\rangle|0,0\rangle)$ with both wires α_1 and α_2. Since the designed algorithm uses two pairs of paired qubits, the same logic is applied two times:

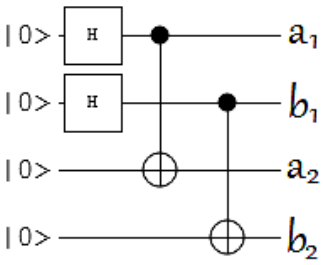


Figure 2: The generation of a superposition

The output of this circuit is the superposition:

$$\frac{1}{\sqrt{2}}(|1,1,1,1\rangle - |0,1,0,1\rangle - |1,0,1,0\rangle|0,0,0,0\rangle)$$

The first two qubits, α_1 and β_1 are intended for A, α_2 and β_2 are for B.

Used Operators

The operators correspond to the matrices. For the needs of the current logic circuit are used six different operators, three of which are 1-bit, and three 2-bit.

The three 1-bit operators are Hadamard, $\sqrt{\text{NOT}}$, and Splitter. The diagram below shows the used symbols and equivalent matrices for each of the operators:

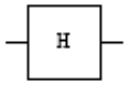
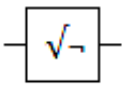
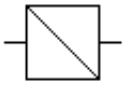
Name	*Hadamard*	$\sqrt{NOT}$	*Splitter*
Symbol			
Matrix	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & i \\ i & 1 \end{bmatrix}$

Table 1: 1-bit operators

Each one of these operators mixes the input states, the interference of two qubits in a superposition may lead to decoherention. To prevent this from happening, it is necessary to be checked the phases of the intermediate state: they are coordinated, when the initial state is 1 and are non-coordinated when the input parameter is 0. Upon the presence of two consecutive operators the interference is reflected in a different way, when it is applied on the second operator.

The implementation of the 2-bit operators in practice is more simple, since neither of them mixes the states. They all have classical equivalents. The swap operator swaps the states of both bits, the controlled NOT operator (*CNOT*) inverts the state of one of the bit, only if the other has the value 1. The decrementing operator reduces the value with 1. The next diagram shows the equivalent matrices for each of the operators:

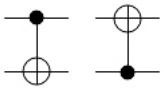
Name	Swap	*CNOT*	*Decrement*
Symbol			
Matrix	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$

Table 2: 2-bit operators

If two of these operators are applied consistently on one of the lines, the effect is equivalent to $M_1 \otimes M_2$, where the matrix M_1 is the second operator. When they are applied on separate lines, the sequence of the operators is irrelevant. When it is necessary, a 1-bit operator to be implemented for manipulation on 2 bits, can be used a tensor product with a single matrix, in order to extend

the equivalent matrix from 2 × 2 to 4 × 4. If the operator is intended to be applied on the first bit, the equivalent 2-bit operator is G⊗I, where G is the equivalent matrix of the 1-bit operator and I is the single matrix. If the operator is intended to be applied on the second bit, then the equivalent operator is I⊗G.

Composing a circuit from a tensor product

In order to multiply the effects from the examined matrices, we will unite them in a single matrix, equivalent to all circuits. For a circuit, which relates to the right column, composed of a Decrementing operator followed by $\sqrt{\text{NOT}}$ operator of the second line, which works in the following way:

$$
\begin{aligned}
(I \otimes SRN) \cdot DEC &= \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \right) \cdot M_{DEC} \\
&= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \cdot \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} & 0 \cdot \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \\ 0 \cdot \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} & 1 \cdot \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \end{bmatrix} \cdot M_{DEC} \\
&= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \\
&= \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 1 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

Figure 3: Tensor product

A tensor product is used for adjustment of the $\sqrt{\text{NOT}}$ operator and its application in respect of the second line. Then the resulting matrix is multiplied by the matrix of the Decrementing operator. The obtained 4 x 4 matrix represents the functionality of the circuit in its entirety.

The circuits that build the solution:

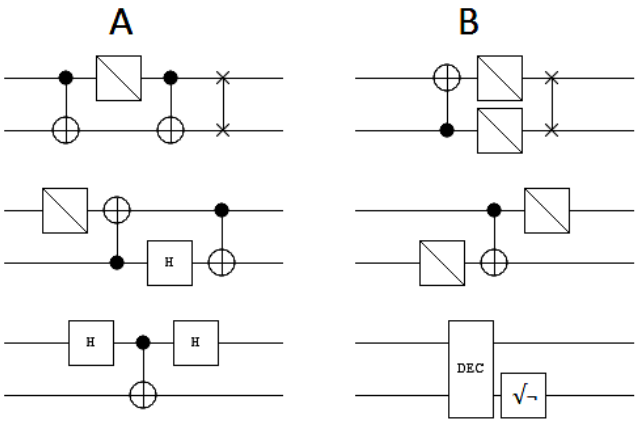


Figure 4: Circuits

For evaluation of the coordinates of the target position in a 3 × 3 matrix are implemented 6 quantum logic circuits, one for each row and one for each column. This assessment is done in two stages. Initially a pairing of two pairs of qubits is carried out, then in the context of the prepared superpositions are assessed the parameters of the exchanged signals, at the second stage is assessed the position.

Complete diagram of the solution

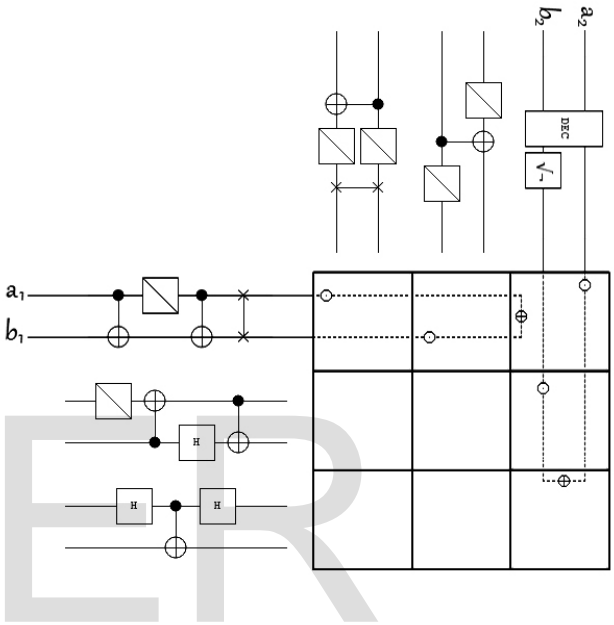


Figure 5: Complete diagram of the solution

On the diagram are shown the six quantum logic circuits, which are used in the solution, one for each row and one for each column. In this case, the target position is located on the first row, last column. The inputs to the selected circuits are paired in advance qubits. Qubit α_1 is paired with qubit α_2 and similarly β_1 with β_2. The qubits are in a superposition, so that the result of each circuit is contained in α_1 and β_1 or in α_2 and β_2. After each of the two sides *A and B* run their respective paired qubits through the circuit selected by them, they measure both lines. For each line/bit, they will receive 1 or 0. Then *A and B* use these results to find the coordinates of the probable location for the position sought.

A and B use the following logic:

1. They check the value from the first line - if it is 1, they mark the line as probable
2. They check the value from the second line by the same logic
3. The third line is marked as probable only if on one of the already checked lines is obtained a result 0.

This quantum logic ensures the finding of the solution sought.

Implementation The diagram below shows the implementation of the algorithm, presented as a quantum logic circuit:

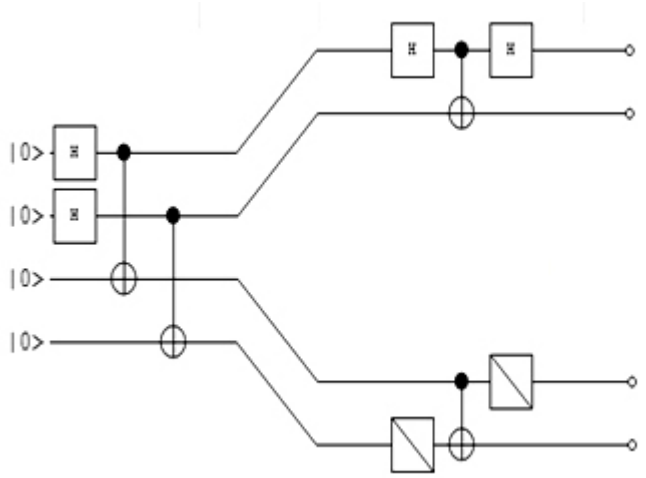


Figure 6: Implementation of the algorithm

Below is shown also the implemented code for simulation of the upper circuit and the measurement of the result:

```
public static Distribution<GlobalState> Localization (int refRow, int refCol) {

// Both sides A and B pair by 2 qubits

var globalState = QubitsSuperposition();

// A run its qubits for the selected row of the quantum circuit

var tempState = ARunCircuit(globalState, refRow);

// B run its qubits for the selected column of the quantum circuit

var finalState = BRunCircuit(tempState, refCol);

// Measurement of the output states from the quantum circuits

return Measure(finalState);

}
```

Initially the system is initialized in the state $|0,0,0,0\rangle$, with the aim of: limiting the size. Then are applied both Hadamard and CNOT operators, which create two pairs of paired qubits. In this way the system is placed in the state: $(|0,0,0,0\rangle|0,1,0,1\rangle|1,0,1,0\rangle|1,1,1,1\rangle)\frac{1}{\sqrt{2}}$. Next step is both upper and lower circuits respectively of *A and B* to be isolated. This does not affect the state of the system. Then, *B* and *A* again run their circuits. Since the operations are applied to different lines, the order in which this will happen is irrelevant. If it is necessary to be used 4-qubit superpositions instead of 2-qubit, a tensor product with single matrices is applied. Since: $(A \otimes I).(I \otimes B) = A \otimes B$, it is possible their circuits to be implemented together by a tensor product. The state of the system, after applying $A_3 \otimes B_1$ is: $(i|1,0,0,0\rangle - |0,1,0,0\rangle - |1,0,1,0\rangle i|0,1,1,0\rangle i|0,0,0,1\rangle - |1,1,0,1\rangle - |0,0,1,1\rangle i|1,1,1,1\rangle)\ t\frac{1-i}{4}$, And finally, both sides *A and B* carry out a measurement of their parts of the system. They can be in any of the displayed in the upper superpositions state with equal probability. They can be in the superposition $|1,0,0,0\rangle$ which is $|0,0\rangle$ for *A* and $|\mathbf{1},\mathbf{0}\rangle$ for *B*.

Matrix of the Constraints

It must be taken into account, that at the input of a circuit, in this case, this is a superposition over two pairs of paired qubits, is applied a scale factor to each column based on the state's amplitude. Then the imposition at the output is determined by summing in rows. For simplification purposes is applied: columns = inputs and rows = outputs. The circuit used is initialized in a superposition: $(|\mathbf{0},\mathbf{0},\mathbf{0},\mathbf{0}\rangle|\mathbf{0},\mathbf{1},\mathbf{0},\mathbf{1}\rangle|\mathbf{1},\mathbf{0},\mathbf{1},\mathbf{0}\rangle|\mathbf{1},\mathbf{1},\mathbf{1},\mathbf{1}\rangle)\frac{\mathbf{1}}{\sqrt{\mathbf{2}}}$. This superposition only uses 4 of the 16 possible states (all the other have an amplitude of 0). This means that all columns except the four columns corresponding to the states in the used input superposition will be multiplied by zero. That's why they don't affect the result. The remaining four columns all contribute equally. It is also known that only certain results are insignificant. This would be useful if the amplitudes of the significant outputs are scrambled, but the amplitudes of any losing output *must* be zero. Otherwise, a consistently winning strategy is not available. So that the most obscured rows actually correspond to the *winning* outputs, which should not be limited additionally, and the non-obscured rows are the losing outputs. Since one and the same input is always used, columns which are significant are always the same: 1st, 6th, 11th, and 16th. The significance of these columns is extracted and analyzed from the tensor product of the *A and B* matrices. In the first column is a tensor product from the first selected column of *A* and the first selected column of the *B* matrix. The sixth column is a tensor product from the second columns of the *A and B* matrices, the 11-th from the third columns, and the 16th from fourth columns. The columns, which are significant in the 16 × 16 matrix are generated entirely by pairs of columns with the same position from the used 4 × 4 *A and B* matrices. This means that summing of the significant columns is in fact only a computational product of the rows *A and B*. This allows to be rewritten all significant restrictions in the form $A_{row,\ r=x} \cdot B_{row,\ r=y} = 0$. Since there are 9 matrix combinations and 8 insignificant rows per combination, this means 72 significant equations which must be implemented over the 96-th complex numbers, composing the $6^{\text{-th}}$ matrices. Solving this system of equations will allow to be found matrices that represent a solution. The matrices presented above are a particular example of a solution of the system.

Result

All the information required to understand the solution is non-local. The implementation (the simulation of) the solution is just a matter of multiplying the initial vector with the correct (significant) matrices and verification whether the resulting states contain only significant (non-zero) amplitudes.

The matrices that correspond to the significant circuits have the property to transform the initially paired qubits without the amplitude to lose states. The system with the matrices of the restrictions must be solvable when the matrices are unitary (through the laws of the quantum mechanics), but unsolvable when the matrices are stochastic (i.e. the classical physics).

3 Conclusion

The quantum logarithm for non-local coordinated strategy for Localization shows the ability to detect consequently correct solutions, which can not be consequently achieved through the laws of the classical physics.

The algorithms for searching are implemented as quantum logic circuits, which include paired qubits, which can be represented as unitary matrices. The solution can be found through the restrictions, on which these matrices must comply. The classical simulation of the developed quantum algorithm for a solution of this task achieves on average 8/9 successful solutions, which is: ~ 88%. Upon presence of a quantum computer, the success rate of this algorithm would probably reach ~100 %.

Refferences

[1] M. A. Nielsen and I. L. Chuang, Quantum Computation and Quantum Information (Cambridge University Press, Cambridge, United Kingdom, 2000).
[2] I. L. Chuang, N. Gershenfeld and M. Kubinec, "Experimental Implementation of Fast Quantum Searching", Physical Review Letters, 80(15), 3408-3411, 1998.
[3] R. Cleve and J. Watrous, "Fast Parallel Circuits for the Quantum Fourier Transform", Proceedings of IEEE Symposium on the Theory of Computing, pp. 526-535, 2000.
[4] W. Cooley and J. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series", Math. Of Computation, 19:297-301, 1965.
[5] D. Coppersmith, "An Approximate Fourier Transform Useful in Quantum Factoring", IBM Research Report RC 19642, 1994.
[6] X. Deng, T. Hanyu and M. Kameyama, "Quantum Device Model Based Super Pass Gate for Multiple-Valued Digital Sys-tems", Proceedings of Intl. Symp. Multiple-Valued Logic, pp. 130-138, 1995.
[7] D. Deutsch, "Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer", Proc. Royal Soc. A 400:97-117, 1985.
[8] N. Gershenfeld and I. L. Chuang, "Bulk Spin-Resonance Quantum Computing", Nature, vol. 404, pp.350-356, 1997.
[9] R. Duan, Z. Ji, Y. Feng and M. Ying, "A Relation Between Quantum Operations and the Quantum Fourier Transform", Quantum Physics Archive, arxiv: quant-ph/0304145
[10] L. Hales, "Quantum Fourier Transform and Extensions of the Abelian Hidden Subgroup Problem", Ph. D. Dissertation, UC Berkeley, 2002.
[11] S. L. Hurst, D. M. Miller and J. Muzio, "Spectral Techniques in Digital Logic", Academic Press, London, 1985.
[12] R. Jozsa, "Quantum Algorithms and the Fourier Transform", Proceedings of Royal Society of London, 454:323-337, 1997.
[13] Nikolay Raychev. Dynamic simulation of quantum stochastic walk. In International jubilee congress (TU), 2012.
[14] Nikolay Raychev. Classical simulation of quantum algorithms. In International jubilee congress (TU), 2012.
[15] Nikolay Raychev. Interactive environment for implementation and simulation of quantum algorithms. CompSysTech'15, DOI: 10.13140/RG.2.1.2984.3362, 2015
[16] Nikolay Raychev. Unitary combinations of formalized classes in qubit space. In International Journal of Scientific and Engineering Research 04/2015; 6(4):395-398, 2015.
[17] Nikolay Raychev. Functional composition of quantum functions. In International Journal of Scientific and Engineering Research 04/2015; 6(4):413-415, 2015.
[18] Nikolay Raychev. Logical sets of quantum operators. In International Journal of Scientific and Engineering Research 04/2015; 6(4):391-394, 2015.
[19] Nikolay Raychev. Controlled formalized operators. In International Journal of Scientific and Engineering Research 05/2015; 6(5):1467-1469, 2015.
[20] Nikolay Raychev. Controlled formalized operators with multiple control bits. In International Journal of Scientific and Engineering Research 05/2015; 6(5):1470-1473, 2015.
[21] Nikolay Raychev. Connecting sets of formalized operators. In International Journal of Scientific and Engineering Research 05/2015; 6(5):1474-1476, 2015.
[22] Nikolay Raychev. Indexed formalized operators for n-bit circuits. In International Journal of Scientific and Engineering Research 05/2015; 6(5):1477-1480, 2015.