Simulation of quantum algorithms for classification of their complexity

Nikolay Raychev

Abstract - This article examines the quantum computational complexity in three fundamental aspects: quantum computations feasible in polynomial time, effective verification of the quantum proofs, as well as quantum interactive proof systems. On the basis of these concepts are defined the classes of quantum complexity, such as BQP, QMA and QIP, which contain computational issues of varying difficulty. The relationships between these classes and the classical complexity classes are presented. Since these concepts and the complexity classes are usually defined within the model of the quantum circuit, this article includes a section, which focuses on the basic properties of the quantum circuits, which are important when determining the quantum complexity. Two different, but closely related areas of study, are not discussed in this article: complexity of the quantum requests and the quantum communication complexity. These discussions are intended only to highlight the aspects of these topics, which are non-standard, require clarification, or have a relative importance for the quantum computational complexity.

Key words: Quantum computing, computational complexity, operators, gates

1. INTRODUCTION

In this article the binary alphabet {0,1} is designated with Σ, and all computational problems are assumed to be encoded by this alphabet. As usual, a function *is:* $\Sigma^* \rightarrow \Sigma^*$ is called *computable* for *polynomial time,* if there exists a deterministic quantum computation circuit that would calculate for polynomial time f(x) for each input $x \in \Sigma^*$, in this article are used related definitions of the terminology.

1. A function of the form *P:* **N** → **N (where N** = { 0,1,2,. , , }) is called polynomial - *bounded function, if* and only if there is a deterministic quantum computation circuit, that calculates the polynomial time $1^{f(n)}$ for input data 1^n for every $N \in$ **N.** These functions are upper-bounded polynomially, and are effectively computable.

2. A function of the specific form *A:* **N** → [0,1] is called *computable for polynomial time,* if and only if there is a deterministic quantum computation circuit that calculates for a polynomial time a binary *representation* of a(*n*) on input 1^n for *each* π ∈ **N.** The reference to the functions of this form is usually linked with restrictions on probabilities, which are functions of the length of an input string regarding a certain problem.

The depth of a classical or quantum circuit is the maximum number of operators, encountered on any path from an input qubit to an output qubit in the circuit. The depth of the circuit can be considered as a parallel time for running, or as a number of time units, which are necessary to be applied on the circuit in order for the operations to be parallelized in a way that corresponds to the topology of the circuit.

Many other complexity classes are examined on the basis of the quantum circuits, which are bounded on depth. In the classic case there is a very close connection between the space-bounded and depth-bounded computations. This close relationship is based on two main ideas: The first is that the space-bounded computations may be simulated effectively using bounded on depth circuits using parallel algorithms for matrix computations, and the second is that the bounded on depth Boolean circuits may be effectively simulated by space-bounded computations and depth packages of the circuit to be simulated.

For the quantum computations such close relationship is not known to exist. The space-bounded quantum computations can be effectively simulated from depth-bounded circuits. The opposite direction, which is efficient is a space-bounded simulation of a depth-bounded quantum circuit, but for now, such cases are not known and are less likely. Informally speaking, the depth-bounded quantum circuits are computationally powerful, while the space-bounded ones are not.

Quantum circuit.
The quantum circuit constitutes an acyclic network of quantum operators connected by wires: the operators represent quantum operations, and the wires qubits, on which these operations are carried out. The model of the quantum circuit is the most frequently studied model of quantum computation.

Quantum complexity classes.
A quantum complexity class represents a collection of computational problems which are solvable by a given quantum computational model that is subject to certain limitations of the resources. For example, BQP is a quantum complexity class of all problems, whose solutions can be found by a quantum computer for polynomial time.

Quantum proof.
The quantum proof is a quantum state, which plays the role of a certificate for a quantum computer, on which runs a procedure for verification. The quantum complexity class QMA is defined with this concept: It includes all the problems related to decisions, whose cases are effectively verifiable through quantum proofs.

Quantum interactive proof system.
A quantum interactive proof system expresses an interaction between the verifier and one or more proof links, it includes processes in processing and exchange of quantum information, at which the proof procedures are trying to convince the verifying links in the answer to some computational problems.

Determination of the subject and its importance

The inherent difficulty of the computational problems is a basic concept in the theory of the computational complexity. The difficulty usually is formalized in terms of the resources, required by different models of computation for solving a certain problem, as for example the number of the steps of the deterministic quantum computation circuit. Sets of models and resources are considered: deterministic, nondeterministic and probabilistic models; time and space restrictions; and interactions between models with different characteristics. Many interesting relationships between these different models and resource limitations are known.

The common thing between the most frequently studied computational models and resource limitations, is that they *are motivated by* the physics. This is quite natural, given the fact that the computers are physical devices, and their study motivates and directs the researches on the computational complexity.

The most frequently given example is for the class of functions, which are computable for a polynomial time, which ultimately derives from physical considerations; this is a mathematical abstraction of the class of functions which can be effectively computed by computer devices.

The quantum mechanics is a clear candidate for a physical theory that has a potential for impact on the computational complexity.

It can be said that the main purpose of the quantum theory of the computational complexity is to be expressed the effects of the quantum physics on the theory of the computational complexity. For this the purpose the quantum physics considers the complexity of the computational problems in regard to models of the quantum computations, classifications of the problems based on these models, as well as their relationships with the classical models and the complexity classes.

2. CLASSIFICATION OF THE COMPLEXITY

Verification of a quantum algorithm pretending for running in polynomial time for a 3-SAT, NP-complete problem

This chapter considers a quantum algorithm pretending for running in polynomial time for a 3-SAT, NP-complete problem, and proves why it does not actually work as described in theory.

The algorithm

The main idea behind the algorithm is to engage with the amplitudes of the variable assignments, which do not fulfill all clauses. The algorithm creates a single superposition of solutions, rotates a qubit, so that its OFF state is entangled with the assignments and is connected with this, how many clauses are fulfilled, then it measures that qubit. If the measurement does not look promising, the algorithm is repeated. Otherwise, it repeats the verification for rotation and measurement enough times in order to be sure before returning an answer.

The following diagram summarizes the algorithm:

<u>**The algorithm, described on paper, as a circuit**</u>

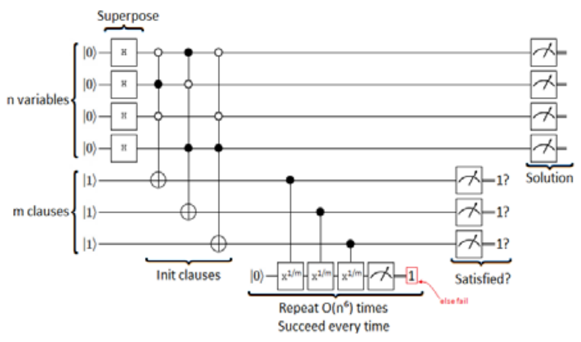

Figure 1

In this case the understanding of the problem does not require to be known what exactly does the gate $X^{\frac{1}{m}}$. Everything is connected with the measurements.

The Problem

It is assumed that the quantum algorithms receive their force for finding answers from their ability to contest decisively the wrong answers, but this algorithm is not doing this. In particular, it should be observed that, while performing complex repetitive activity, nothing is happening with the qubits holding the superposition of the variables assignments. The mixing of the amplitudes of possible assignments, so that they can contest decisively, imposes performance of operations on these qubits.

In order to see what is actually happening, a small change in the circuit must be made. Since the qubits of the variable assignments are not used during the complex activity, there is no need to be measured in the right side of the circuit. In fact, because the controls and measurements are moving, it is not necessary to wait for the qubits of the clauses to be initialized. The qubits of the assignments can be measured immediately after placing them in a superposition without changing the expected behavior of the circuit. The following results is obtained:

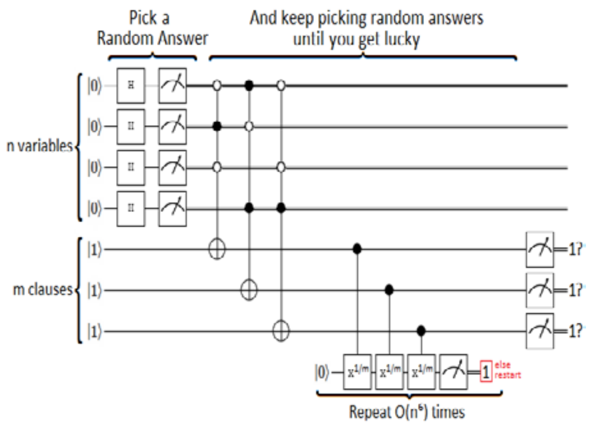

Figure 2

They just carry out a subsequent selection. When selecting an unsatisfying answer, the verification for repeated rotation and measurement leads to restart (in the usual case). When selecting a satisfying answer, the verifications pass after work $O(n^6)$ and finish.

The error in this algorithm is that the price of all failures and iterations is forgotten to be included in the time for running. Only the successes are counted out. But if exactly one of 2^n possible variables assignments satisfies all clauses, then the expected number of iterations before finding the assignment and before successful completion of the algorithm is $\Omega(2^n)$.
At the unsatisfying cases the estimated number is even worse. At a rough estimate around $2^{\Omega(n^6)}$ iterations, based on the fact that most of the assignments must satisfy a constant coefficient of clauses, the situation is the same as winning n^6 coin tosses.

In order to understand the operating time in practice, let's consider an example with 1000 variables, 1000 clauses and without satisfying assignment. To obtain an upper limit on the expected number of iterations needed to trigger a false positive value that allows the algorithm to complete, let's assume that there exist an assignment, corresponding to 999 of 1000 clauses and it is selected constantly.

The probability for this ideal assignment to pass a single verification is $sin^2\left(\frac{\pi}{2}\frac{999}{1000}\right)$~99.99%. But the probability of passing the verification quintillion times in a row is so small, that are used SI prefixes, introduced in the mid 60s to describe how many zeros are written after the decimal point before obtaining the actually useful digits. So ultimately this can lead to multiple iterations.

Simulation of quantum algorithms for classification of their complexity

In this article is given a brief guide for simulation of quantum circuits, a code for simulation of the algorithm for polynomial time is provided and the results of this simulation are used, in order to demonstrate that the algorithm really takes exponential time.

Simulation of a quantum circuit

The simulation of a quantum circuit is not a magic. The space of the states may be unknown and the effects of the operations can be counter-intuitive, but everything is defined well mathematically. The code is trivial. The hard part is the internalization and the understanding of the rules.

Quantum states

The type of state that can be supported by a quantum chain, is called "mixed state". The mixed state is a probability distribution of "pure states". A pure state is a superposition of classical states. A classic state is an assignment of Boolean values per each qubit: qubit 1 is OFF, qubit 2 is ON, etc.

These are many definitions at once, so let's examine them one by one. Programming representations are needed on all three levels (classical, clean, and mixed), if the simulation will be carried out.

The classical state is simply a bunch of bits. A very convenient way to store bit values is the integer bit mask. The class QuClassicalState is nothing more than a few useful methods around an integer value:

```
>>> print(QuClassicalState(5))
|00000101⟩
>>> print(QuClassicalState(5).bit(2))
True
```

The pure state, also called "superposition", is a weighted combination of classical states. The weight, associated with any state, is called "amplitude" and is fundamentally the square root of probability. If the magnitudes of all amplitudes are squared and these square roots are added, is obtained a total of 100% (otherwise, that is not a valid superposition). The class QuPureState uses a glossary to store the pure state: the switches are the classic states, and the values - the amplitudes.

```
>>> print(QuPureState({
      QuClassicalState(2): -0.8,
      QuClassicalState(7): 0.6j
   }))
-0.800*|00000010⟩  + 0.600j*|00000111⟩
```

The mixed state is also a weighted combination of states, but this time they are pure states instead of classical and the weights are probabilities instead of amplitudes. The class QuMixedState uses a glossary, to store the mixed state: the switches are the pure states, and the values - the probabilities.

```
>>> print(QuMixedState({
      QuPureState({QuClassicalState(4): -1}): 0.25,
      QuPureState({QuClassicalState(5):
math.sqrt(0.5),           QuClassicalState(6):          -
math.sqrt(0.5)}): 0.75
   }))
75.0%: 0.707*|00000101⟩  + -0.707*|00000110⟩
25.0%: -1.000*|00000100⟩
```

Thus the state of a quantum circuit is a probability distribution of superpositions of classical states. A convenient mathematical representation for this type of state is the matrix of the density.

Quantum operations

There are two types of operations that can be applied on quantum states: unitary operations and measuring operations. Roughly speaking, the unitary operations transform the classical states in pure states, while the measuring operations transform the pure states in mixed such.

A given unitary operation associates the result of a pure state with each allowed classical state. When applied on pure state, the operation is distributed linearly: it is applied on each classical state in the superposition and the amplitudes in the resulting pure states are scalable on the associated input amplitude of the state. The set of output superpositions are set equal to a single superposition through a concatenation, with the exception that matching classical states are opposed to each other (their amplitudes of each superposition are added).

When applied on a mixed state, the unitary operations simply are distributed directly on each pure state in the mixed state. (No steps are necessary for equalization or intervention at the level of the mixed state.)

Here is given an example for an unitary operation applied on a mixed state:

```
>>>   q_op_hadamard_on_first_bit   =   lambda   c:
QuPureState({
      c.q_with_bit(0, False): math.sqrt(0.5),
      c.q_with_bit(0, True): -math.sqrt(0.5) if c.bit(0)
else +math.sqrt(0.5)
   })
>>> input = QuMixedState({
      QuPureState({QuClassicalState(4): -1}): 0.25,
      QuPureState({QuClassicalState(5):
math.sqrt(0.5),           QuClassicalState(6):          -
math.sqrt(0.5)}): 0.75
   })
>>> print(input)
75.0%: 0.707*|00000101⟩  - 0.707*|00000110⟩
25.0%: -1.000*|00000100⟩
>>>
print(input.q_unitary_transform(q_op_hadamard_
on_first_bit))
75.0%:  0.500*|00000100⟩  + -0.500*|00000101⟩  + -
0.500*|00000110⟩  + -0.500*|00000111⟩
25.0%: -0.707*|00000100⟩  + -0.707*|00000101⟩
```

A given measuring operation makes a difference between the classical states, which constitute a pure state, dividing it into pieces. The probabilities of the resulting pieces are determined by the amount of the squared amplitudes of the states within this piece. Each piece after that becomes a separate branch at the level of the mixed state:

```
>>> q_value_of_first_bit = lambda c: c.bit(0)
>>> input = QuMixedState({
      QuPureState({QuClassicalState(4): -1}): 0.25,
      QuPureState({QuClassicalState(5): math.sqrt(0.5), QuClassicalState(6): -math.sqrt(0.5)}): 0.75
   })
>>> print(input.measure(q_value_of_first_bit))
37.5%: -1.000+0.000j*|00000110〉
37.5%: 1.000+0.000j*|00000101〉
25.0%: -1.000+0.000j*|00000100〉
```

The code supports also a subsequent selection, where a measurement is carried out, but is stated what the result will be. In practice, this would include the conducting of an experiment again and again, until the desired result is obtained. The code returns both the final re-normalized state, and the likelihood of success:

```
>>> q_value_of_first_bit = lambda c: c.bit(0)
>>> print(input.post_select(q_value_of_first_bit))
(0.3750000000000001,
QuMixedState({QuPureState({QuClassicalState(5): (1+0j)}): 1.0}))
```

Thanks to the possibility for storing states and performing operations is already available a simulation machine of base quantum circuit.

Simulation of the algorithm

Here is presented a part of the code of the algorithm, by creating useful values:

```
def q_simulate_younes_algo(q_anti_clauses):
  n = max(max(q_used_variables) for q_used_variables in q_anti_clauses) + 1
  m = len(q_anti_clauses)
  var_bits = range(n)
  q_clause_bits = range(n, n + m)
  ancilla_bit = n + m

  state = QuMixedState({QuPureState({QuClassicalState(0): 1}): 1})
```

Bits of variable submission are superpositioned and entangled bits of type "is the clause satisfied" are initialized:

```
  for i in var_bits:
    state = state.q_unitary_transform(hadamard_op(i))
  for j in range(m):
    state = state.q_unitary_transform(not_op(n + j))
    state = state.q_unitary_transform(
      q_controlled_by(not_op(n + j), q_anti_clauses[j]))
```

And the iterated test for rejection on the basis of the number of satisfied clauses is executed:

```
while True:
    [... track and output debug info ...]

    for j in q_clause_bits:
      q_op_mx = q_controlled_by(
          q_partial_x_rotation_op(ancilla_bit, m),
          {j: True})
      state = state.q_unitary_transform(q_op_mx)
    p_pass, state = state.post_select(bit_check_predicate(ancilla_bit))
    state = state.q_unitary_transform(not_op(ancilla_bit))
```

The selected 3-SAT example is used only for testing and at it the only solution is submission of True to all 11 variables:

```
q_simulate_younes_algo(q_anti_clauses=[
   # Force 0 true
   {0: False, 1: False, 2: False},
   {0: False, 1: True, 2: False},
   {0: False, 1: False, 2: True},
   {0: False, 1: True, 2: True},

   # Force 1 true
   {0: True, 1: False, 2: False},
   {0: True, 1: False, 2: True},

   # Force all true
   {0: True, 1: True, 2: False},
   {0: True, 1: True, 3: False},
   {0: True, 1: True, 4: False},
   {0: True, 1: True, 5: False},
   {0: True, 1: True, 6: False},
   {0: True, 1: True, 7: False},
   {0: True, 1: True, 8: False},
   {0: True, 1: True, 9: False},
   {0: True, 1: True, 10: False},
])
```

There are two important values for tracking, while the algorithm works: q_p_survived and q_p_correct. q_p_survived is the likelihood that the algorithm is not forced to be restarted, as should happen, if the subsequent selection after the verification "rotation on the basis of the number of satisfied clauses and anticipation of True" fails. q_p_correct is the likelihood that when measuring the bits in the clauses and variables bits in the current iteration will be obtained the right answer (all clauses are satisfied, all variables are true).

q_p_survived and q_p_correct act as multipliers on the time for operation of the algorithm. If it has to be restarted 99 of 100 times because of q_p_survived (which is 1 %), the algorithm will be executed about 100 times longer. If an incorrect answer is obtained 99 of 100 times because of q_p_correct (which is 1 %), the algorithm must be repeated ~100 times before seeing a good answer. Even more, the results of these two multipliers on the time for operation are accumulated, so that the actual quantity is the product q_p_correct*q_p_survived.

Since the shown test case has 11 variables and the algorithm starts by putting them in a single superposition, q_p_correct in the beginning is $\frac{1}{2^n} = \frac{1}{2048} \approx 0.049\%$. q_p_survived starts from 100%, since the subsequent selection happens later.

The searched result from the operation of the algorithm is q_p_correct to goes to 100%. It must do this more quickly rather than q_p_survived goes down, because q_p_correct*q_p_survived must be increased, if the time for operation should be reduced.

Results

Upon start of the stimulation code are obtained the following results:

```
iter 0;    q_p_survived: 100.0000%;   q_p_correct: 0.0488%;   q_p_correct*q_p_survived: 0.0488%
iter 10;   q_p_survived: 71.6915%;    q_p_correct: 0.0681%;   q_p_correct*q_p_survived: 0.0488%
iter 100;  q_p_survived: 25.2162%;    q_p_correct: 0.1936%;   q_p_correct*q_p_survived: 0.0488%
iter 200;  q_p_survived: 8.4309%;     q_p_correct: 0.5792%;   q_p_correct*q_p_survived: 0.0488%
iter 300;  q_p_survived: 2.8427%;     q_p_correct: 1.7177%;   q_p_correct*q_p_survived: 0.0488%
iter 400;  q_p_survived: 0.9801%;     q_p_correct: 4.9819%;   q_p_correct*q_p_survived: 0.0488%
iter 500;  q_p_survived: 0.3592%;     q_p_correct: 13.5918%;  q_p_correct*q_p_survived: 0.0488%
iter 600;  q_p_survived: 0.1523%;     q_p_correct: 32.0607%;  q_p_correct*q_p_survived: 0.0488%
iter 700;  q_p_survived: 0.0833%;     q_p_correct: 58.6048%;  q_p_correct*q_p_survived: 0.0488%
iter 800;  q_p_survived: 0.0603%;     q_p_correct: 80.9426%;  q_p_correct*q_p_survived: 0.0488%
iter 900;  q_p_survived: 0.0527%;     q_p_correct: 92.7231%;  q_p_correct*q_p_survived: 0.0488%
iter 1000;    q_p_survived: 0.0501%;    q_p_correct: 97.4508%;  q_p_correct*q_p_survived: 0.0488%
iter 1090;    q_p_survived: 0.0493%;    q_p_correct: 99.0362%;  q_p_correct*q_p_survived: 0.0488%
```

As was expected, q_p_correct increases over time, but q_p_survived has a tendency to decrease (because of the subsequent selection).

Unfortunately q_p_survived*q_p_correct does not increase; it remains constant. This means that the change in the number of the iterations is simply an exchange of restarts at correctness with restarts at subsequent selection, without any improvements of the overall running. No matter what number of iterations is selected, the algorithm will require around 2^n repeated attempts before passing the verifications for subsequent selection and returning a correct answer.

This is in fact also the expected result on the basis of the fact that the measurements can be made before performing the complex activity - the algorithm is equivalent to a random guessing, but made in a much more complex way.

3. CONCLUSION

If we did not have to pay for the iterations, if we have worked in Post BQP instead of in BQP, the algorithm would have to work for a polynomial time. But unfortunately we MUST pay for the iterations. The subsequent selection is not for free. The algorithm is an obfuscated algorithm for subsequent selection. The entire optimized force comes from the restarting, when the things are not perfect, instead of disturbance or entanglement, or deduction, or something else, working in practice. It would be great, if the subsequent selection was for free, but unfortunately in BQP the reality is not like that.

REFFERENCES

[1] A review of ion trap work is in R. Blatt and D. Wineland, Nature, 2008, 453, 1008.
[2] A. M. Turing, Proc. London Math. Soc, 1936, 42, 230.
[3] R. P. Feynman, ''The Feynman lectures on computation'', Addison Wesley (1996).
[4] R. P. Feynman, Found Phys., 1986, 16, 507.
[5] P. Benioff, Phys. Rev. Lett., 1982, 48, 1581; P. Benioff, J. Stat Phys., 1980, 22, 563.
[6] M. A. Nielsen, I. L. Chuang, ''Quantum Computation and Quantum Information'' (CUP, 2000).
[7] A. Einstein, B. Podolsky and A. Rosen, Phys. Rev., 1935, 47, 777.
[8] See J. Kempe, Contemp. Phys., 2003, 44, 307, and refs. therein.
[9] A. P. Hines and P. C. E. Stamp, Phys. Rev A, 2007, 75, 062321.
[10] T. Fujisawa et al., Nature, 2002, 419, 278; T. Hayashi et al., Phys. Rev.Lett., 2003, 91, 226804; K. Ono et al., Science, 2002, 297, 1313; J. R. Petta et al., Science, 2005, 309, 2180; F. H. L. Koppens et al., Nature, 2006, 442, 776.
[11] Y. Nakamura et al., Nature, 1999, 398, 786; C. H. van der Wal et al., Science, 2000, 290, 773; D. Vion et al., Science, 2002, 296, 886; M. Steffen et al., Science, 2006, 313, 1423.
[12] F. Jelzko et al., Phys. Rev. Lett., 2004, 92, 76401; F. Jelzko et al., Phys. Rev. Lett., 2004, 93, 130501; T. Gaebel et al., Nature Phys., 2006, 2, 408; M. V. G. Dutt et al., Science, 2007, 316, 1312.
[13] Nikolay Raychev. Dynamic simulation of quantum stochastic walk. International jubilee congress (TU), 2012.
[14] Nikolay Raychev. Classical simulation of quantum algorithms. International jubilee congress (TU), 2012.
[15] Unitary combinations of formalized classes in qubit space. International Journal of Scientific and Engineering Research 04/2015; 6(4):395-398. DOI: 10.14299/ijser.2015.04.003, 2015.
[16] N. V. Prokof'ev and P. C. E. Stamp, J. Phys CM, 1993, 5, L663.
[17] N. V. Prokof'ev and P. C. E. Stamp, J. Low Temp. Phys, 1996, 104, 143.
[18] Nikolay Raychev. Functional composition of quantum functions. International Journal of

Scientific and Engineering Research 04/2015; 6(4):413-415. DOI:10.14299/ijser.2015.04.004, 2015. 7.

[19] Nikolay Raychev. Logical sets of quantum operators. International Journal of Scientific and Engineering Research 04/2015; 6(4):391-394. DOI:10.14299/ijser.2015.04.002, 2015.

[20] P. C. E. Stamp and I. S. Tupitsyn, Phys. Rev. B, 2004, 69, 014401.

[21] I. S. Tupitsyn et al., to be published.

[22] Nikolay Raychev. Controlled formalized operators. In International Journal of Scientific and Engineering Research 05/2015; 6(5):1467-1469. DOI:10.14299/ijser.2015.05.003, 2015.

[23] Nikolay Raychev. Controlled formalized operators with multiple control bits. In International Journal of Scientific and Engineering Research 05/2015; 6(5):1470-1473. DOI:10.14299/ijser.2015.05.001, 2015.

[24] Nikolay Raychev. Connecting sets of formalized operators. In International Journal of Scientific and Engineering Research 05/2015; 6(5):1474-1476. DOI:10.14299/ijser.2015.05.002, 2015.

[25] N. V. Prokof'ev, P. C. E. Stamp, pp. 347 371 in ''Quantum Tunneling of Magnetisation: QTM'94 '', ed. L. Gunther, B. Barbara (Kluwer, 1995).

[26] A. Morello et al., Phys. Rev. Lett., 2004, 93, 197202; A. Morello and J. de Jongh, Phys. Rev., 2007, B76, 184425.

[27] The fluctuation dissipation theorem is explained in, e.g., P. M. Chaikin, T. C. Lubensky, ''Principles of Condensed Matter Physics'', C.U.P. (1995).

[28] M. Dube´ and P. C. E. Stamp, Chem. Phys., 2001, 268, 257. 29 Nikolay Raychev. Indexed formalized operators for n-bit circuits. International Journal of Scientific and Engineering Research 05/2015; 6(5):1477-1480, 2015..

[30] Nikolay Raychev. Encoding and decoding of additional logic in the phase space of all operators. International Journal of Scientific and Engineering Research 07/2015; 6(7): 1356-1366. DOI:10.14299/ijser.2015.07.003, 2015.

[31] A. O. Caldeira and A. J. Leggett, Ann. Phys., 1983, 149, 374.

[32] A. J. Leggett, Phys. Rev., 1984, B30, 1208.

[33] R. P. Feynman and F. L. Vernon, Ann. Phys., 1963, 24, 118.

[34] A. Morello and P. C. E. Stamp, Phys. Rev. Lett., 2006, 97, 207206.

[35] Nikolay Raychev. Ensuring a spare quantum traffic. International Journal of Scientific and Engineering Research 06/2015; 6(6):1355-1359. DOI:10.14299/ijser.2015.06.002, 2015.

[36] Nikolay Raychev. Quantum circuit for spatial optimization. International Journal of Scientific and Engineering Research 06/2015; 6(6):1365-1368. DOI:10.14299/ijser.2015.06.004, 2015.

[37] P. W. Anderson, Phys. Rev., 1958, 109, 1492.

[38] M. Schechter and P. C. E. Stamp, Phys. Rev. Lett, 2005, 95, 267208; M. Schechter and P. C. E. Stamp, Phys. Rev. B, 2008, 78, 054438.

[39] Nikolay Raychev. Measure of entanglement by Singular Value decomposition. International Journal of Scientific and Engineering Research 07/2015; 6(7): 1350-1355. DOI:10.14299/ijser.2015.07.004, 2015.

[40] D. Collison, C. D. Garner, C. M. McGrath, J. F. W. Mosselmans, M. D. Roper, J. M. W. Seddon, E. Sinn and N. A. Young, J. Chem. Soc. Dalton Trans., 1997, 4371 4376.

[41] Nikolay Raychev. Quantum algorithm for spectral diffraction of probability distributions. International Journal of Scientific and Engineering Research 08/2015; 6(7): 1346-1349. DOI:10.14299/ijser.2015.07.005, 2015.

[42] Nikolay Raychev. Algorithm for switching 4 -bit packages in full quantum network with multiple network nodes. International Journal of Scientific and Engineering Research 09/2015; 6(8):1289. DOI:10.14299/ijser.2015.08.004, 2015.

[43] J. Lehmann, A. Gaita Ario, E. Coronado and D. Loss, Nature Nanotech., 2007, 2, 312 317; J. Lehmann, A. Gaita Ario, E. Coronado and D. Loss, J. Mat. Chem., DOI: 10.1039/b810634g.

[44] Nikolay Raychev. Reply to "The classical-quantum boundary for correlations: Discord and related measures". Abstract and Applied Analysis 11/2014; 94(4): 1455-1465, 2015.

[45] Nikolay Raychev. Mathematical approaches for modified quantum calculation. International Journal of Scientific and Engineering Research

09/2015; 6(8):1302. doi:10.14299/ijser.2015.08.006, 2015.
[46] N. Aliaga Alcalde, R. S. Edwards, S. O. Hill, W. Wernsdorfer, K. Folting and G. Christou, J. Am. Chem. Soc., 2004, 126, 12503 12516.
[47] R. Bagai, W. Wernsdorfer, K. A. Abboud and G. Christou, J. Am. Chem. Soc., 2007, 129, 12918 12919.
[48] C. M. Ramsey, E. del Barco, S. Hill, S. J. Shah, C. C. Beedle and D. Hendrickson, Nature Physics, 2008, 4, 277 281.
[49] F. K. Larsen, E. J. L. McInnes, H. El Mkami, J. Overgaard, S. Piligkos, G. Rajaraman, E. Rentschler, A. A. Smith, G. M. Smith, V. Boote, N. Jennings, G. A. Timco and R. E. P. Winpenny, Angew. Chem. Int. Ed. Eng., 2003, 115, 105 109.
[50] M. Affronte, F. Troiani, A. Ghirri, S. Carretta, P. Santini, V. Corradini, R. Schuecker, C. Muryn, G. Timco and R. E. Winpenny, Angew. Chem. Int. Ed. Eng., 2003, 115, 105 109.
[51] Nikolay Raychev. Theoretically optimal computing frontiers for rapid multiplication through decomposition. International Journal of Scientific and Engineering Research 09/2015; 6(8):1318, 2015..
[52] Nikolay Raychev. Quantum computing models for algebraic applications. International Journal of Scientific and Engineering Research 09/2015; 6(8):1281, 2015..
[53] R. Raussendorf, Phys. Rev. A, 2005, 052301.
[54] C. S. Lent, B. Isaksen and M. Lieberman, J. Am. Chem. Soc., 2003, 1056 1063.
[55] O. Waldmann, H. U. Gdel, T. L. Kelly and L. K. Thompson, Inorg. Chem., 2006, 45, 3295.
[56] Nikolay Raychev. Indexed cluster of controlled computational operators. International Journal of Scientific and Engineering Research 09/2015; 6(8):1295, 2015.
[57] Nikolay Raychev. Quantum multidimensional operators with many controls. International Journal of Scientific and Engineering Research 09/2015; 6(8):1310. DOI:10.14299/ijser.2015.08.007, 2015.
[58] Special issue of Chem. Rev., , 1998, 98, pp. 1 390, edited by C. L. Hill.
[59] Nikolay Raychev. Algorithm for switching 4-bit packages in full quantum network with multiple network nodes. International Journal of Scientific and Engineering Research 08/2015; 6(8): 1289-1294. DOI: 10.14299/ijser.2015.08.004, 2015.
[60] A. Muller, P. Kogerlera and A. W. M. Dressb, Coord. Chem. Rev., 2001, 222, 193 218.
[61] Nikolay Raychev. Reply to "Flexible flow shop scheduling: optimum, heuristics and artificial intelligence solutions". Expert Systems 2015; 25(12): 98-105, 2015.
[62] Nikolay Raychev. Bilaterally Symmetrical Transformation between Independent Operators and Rotations. Journal of Quantum Information Science, 5, 79-88. doi: 10.4236/jqis.2015.53010, 2015.
[63] S. Caillieux, D. de Caro, L. Valade, M. Basso Bert, C. Faulmann, I. Malfant, H. Casellas, L. Ouahab, J. Fraxedas and A. Zwick, J. Mater. Chem., 2006, 13, 2931 2936.
[64] Nikolay Raychev. Formalized Operators with Phase Encoding. Journal of Quantum Information Science, 5, 114-126. doi: 10.4236/jqis.2015.53014.
[65] Y. Wang, X. Wang, C. Hu and C. Shi, J. Mater.Chem., 2002, 12
[66] Nikolay Raychev. Multi-functional formalized quantum circuits. International Journal of Scientific and Engineering Research 10/2015; 6(9):1304-1310. DOI:10.14299/ijser.2015.09.004, 2015.
[67] Nikolay Raychev. Application of the Raychev's formalized Circuits. International Journal of Scientific and Engineering Research 10/2015; 6(9):1297-1304. DOI:10.14299/ijser.2015.09.003, 2015.
[68] Nikolay Raychev. Analysis of the complexity of the formalized circuits of Raychev. International Journal of Scientific and Engineering Research 10/2015; 6(9): 1289-1296. DOI:10.14299/ijser.2015.09.002, 2015.