Speed improvement of the quantum factorization algorithm of P. Shor by upgrade its classical part

Larissa Cherckesova[1,*], *Olga* Safaryan[1], *Pavel* Razumov[1], *Irina* Pilipenko[1], *Yuriy* Ivanov[2], *Ivan* Smirnov[3]

[1]Don state technical University, Rostov–on–Don, 1, Gagarin square, Russia
[2]Southern Federal University, Taganrog, Taganrog, 44, per. Nekrasovsky, Rostov region, Russia
[3]Institute of artificial intelligence problems, 118B, Artyom street, Donetsk, Ukraine

Abstract. This report discusses Shor's quantum factorization algorithm and ρ–Pollard's factorization algorithm. Shor's quantum factorization algorithm consists of classical and quantum parts. In the classical part, it is proposed to use Euclidean algorithm, to find the greatest common divisor (GCD), but now exist large number of modern algorithms for finding GCD. Results of calculations of 8 algorithms were considered, among which algorithm with lowest execution rate of task was identified, which allowed the quantum algorithm as whole to work faster, which in turn provides greater potential for practical application of Shor's quantum algorithm. Standard quantum Shor's algorithm was upgraded by replacing the binary algorithm with iterative shift algorithm, canceling random number generation operation, using additive chain algorithm for raising to power. Both Shor's algorithms (standard and upgraded) are distinguished by their high performance, which proves much faster and insignificant increase in time in implementation of data processing. In addition, it was possible to modernize Shor's quantum algorithm in such way that its efficiency turned out to be higher than standard algorithm because classical part received an improvement, which allows an increase in speed by 12%.

1 Introduction

Today, the field of quantum computations is gradually becoming the leader of information technology and calculations. Scientific researchers are paying increasing attention to super-modern computational model based on the notion of qubit (quantum bit) designed to supplant the long–known model that has been used in almost all computer hardware and based on the notion of bit, developed by A. Turing and J. von Neumann [1, 2].

Quantum bit is kind of quantum system, which before the measurement is in arbitrary linear superposition of two basic quantum states, and as a result of the measurement, it takes one of two possible values with certain probability. On one hand, this element is the same bit, because it takes two values, and on other, it is quantum, since these two values are

[*] Corresponding author: chia2002@inbox.ru

in superposition with each other. Recently, quantum computational model has attracted increasing attention of scientists and engineers, because its practical application in future will provide tremendous opportunities for solving problems that have not found effective solution algorithms in the bit computational model [3].

One of the clearest examples of such problems is the problem of the certain number factorization, the solution of which is to find the divisors of this number [4].

2 Purpose of Investigations

Based on the foregoing, it is necessary to prove the advantage of the quantum algorithm over the algorithms of other computational models. In addition, Shor's algorithm has quantum and classical parts, which makes it possible to increase the calculation speed of the algorithm as whole, by maximally modernizing and simplifying the latter.

Despite the advantages of the quantum algorithm, it has its drawbacks, since it has a classic part that runs on ordinary computers. The classical part is the weak link in the Shor's algorithm. Therefore, by modernizing and simplifying it as much as possible, it is possible to increase the speed of computation of the Shor's quantum algorithm.

3 Theoretical Preconditions and Research Methods

The supposedly large computational complexity of the factorization task lies at the heart of cryptographic strength of some public-key encryption algorithms, such as RSA. Moreover, the system is hacked unambiguously if at least one of the key parameters of the RSA algorithm is known.

The factoring is Pollard's ρ – method. Consider the following factorization Pollard's algorithm ρ – method:

Consider a sequence of integers x_n , such, that each next number is: $x_{i+1} = (x_n^2 - 1) \bmod N, n = 0,1,2...$. In this sequence x_0 is a small number.

At each step, we will calculate the value

$$d = GCD(n, | x_i - x_j |), where \quad j < i .$$

If $d \neq 1$, then the calculation ends. The found number d is a divisor of the number n. If n/d is not a prime number, then you can continue by taking the number n/d instead of n.

The main disadvantage of this method is the allocation of additional memory to store the previous values x_j.

Note that if $(x_i - x_j) \equiv 0 (\text{mod } p)$, then

$$(f(x_i) - f(x_j)) \equiv 0 (\text{mod } p) .$$

Therefore, if a pair of (x_i, x_j) gives us the solution, then a solution will be given by any pair of $(x_i + k, x_j + k)$ [5].

4 Shor's Quantum Algorithm

The essence of this method lies in the fact that the task of factorization is reduced to the task of finding the period of a function. When the function period is known, factorization is performed on a classical computer in polynomial time using the Euclidean algorithm. The quantum part of the algorithm is assigned to perform the search period function. And the classical part of the algorithm first prepares this function in a special way, and then checks the period found by the quantum part for sufficiency to solve the problem. If the period is found correctly (the algorithm is probabilistic, so it can find not what you want), then the problem is solved. If not, then the quantum part of the algorithm is executed again. In addition, since the validation of the solution for the factorization problem is very simple (multiplied two numbers and compared with the third one), then this part of the algorithm can be disregarded in general from the point of view of calculating complexity [6, 7].

Shor's factorization algorithm:

1. Choose a random number a, less than $M : a < M$. Calculate $GCD(a,M)$ by Euclidean algorithm.
2. If $GCD(a,M)$ is not equal 1, then there is a nontrivial divisor of M, so the algorithm terminates (degenerate case).
3. Otherwise, it is necessary to use the quantum subroutine of the period function search $f(x) = a^x \bmod M$.
4. If the period r found is odd, then go back to step 1, and select another number a.
5. If $a^{r/2} \equiv M - 1 (\bmod M)$, then go back to step 1 and choose another number a.
6. Finally, define two values of $GCD(a^{r/2} \pm 1, M)$, which are the non–trivial divisors of M.

It is important to choose the number of qubits that will be used in the quantum scheme. It is necessary to choose such number of qubits q in order that inequality $M^2 \le 2^q < 2M^2$ be fulfilled. If this equality is fulfilled, it is ensured that a given number of qubits is enough for a sufficient number of times to fulfill the function for which a period is sought so that constructive interference will work [8-12].

5 Implementing Shor's Quantum Algorithm

Before implementation, it is necessary to define some auxiliary functions and constants.

```
numberToFactor ::Int
numberToFactor  = 21
simpleNumber ::Int
simpleNumber = 2
nofAncillas ::Int
nofAncillas = 5
nofWorkingOubits::Int
nofWorkingOubits = 4
nofOubits ::Int
nofOutbits = nofWorkingOubits+nofAncillas
periodicFunction::Int -> Int
periodicFunction x=simpleNumber^x’mod’numberToFactor
```

Constant *numberToFactor* specifies the number that must be decomposed into prime factors. Constant *simpleNumber* is mutually prime number with the previous constant. Constants: *nofAncillas*, *nofWorkingQubits* and *nofQubits* represent the number of auxiliary and working qubits, as well as the total number of qubits in the quantum scheme [9].

Function $periodicFunction$ is just that periodic function, the task of finding the period of which is fulfilled by the quantum subroutine of Shor's algorithm. Next, we implement the quantum scheme in the form of function.

```
quantunFactoring :: Matrix (Complex Double) – > IO String
quantunFactoring o =
 initial |> gateHn nofWorkingOubitd < ++ >
       gateIn nofAncillas
     |> o
     |> qft nofWorkingOubits < ++ >
       gateIn nofAncillas
     >>> (measure . fromVector nofOubits )
```

Comparative analysis of algorithm data in practice. The quantum algorithm for factoring Shore, as mentioned earlier, can be divided into two parts - the classical and the quantum. Consider the weakest point of this algorithm, namely the classical part, which can be modified and modernized.

The classic part. In the classical part, it is proposed to use the Euclidean algorithm to find the GCD, but, moreover, there are a fairly large number of algorithms for finding the greatest common divisor of a pair of numbers.

Below are the results of calculations of each of these algorithms, among which it is required to identify the algorithm with the lowest speed to complete the task, which will allow the quantum algorithm as a whole to work somewhat faster, which in turn will provide greater potential for practical application of the quantum Shor's algorithm [10].

As the studied ones, eight most common algorithms for finding GCD were selected, and their calculation speed was checked on numbers of different complexity.

The numbers gcd 1 – gcd 8 correspond to following names:
1. Search from an arbitrary number.
2. Search from the minimum number.
3. With decomposition into dividers.
4. Euclidean algorithm is recursive.
5. Euclidean algorithm iterative.
6. Binary algorithm recursive.
7. Binary algorithm iterative.
8. Binary algorithm iterative shift.

The results of the calculations are presented in Table 1.

Table 1. Calculations of Classical Algorithms.

N	**10-100**	**100-1000**	**1000-10000**	**10000-100 000**	**100 000 – 1 000000**
gcd 1	0.0040	0.0402	0.3531	1.7404	7.7908
gcd 2	0.0030	0.0305	0.2490	1.3123	7.4236
gcd 3	0.0028	0.0134	0.0436	0.1089	0.4675
gcd 4	0.0068	0.0150	0.0273	0.0277	0.0454
gcd 5	0.0010	0.0017	0.0025	0.0029	0.0036
gcd 6	0.0030	0.0064	0.0085	0.0099	0.0124

gcd 7	0.0012	0.0018	0.0020	0.0022	0.0026
gcd 8	0.0009	0.0014	0.0016	0.0018	0.0020

The most effective algorithm is gcd 8 (binary with iterative shift), the second in efficiency is gcd 7 (binary iterative) and the third is gcd 5 (Euclidean algorithm iterative). The gcd 5 algorithm (Euclidean algorithm iterative) is superior to gcd 7 (binary iterative algorithm) when working with numbers less than 1000, but it is undoubtedly inferior to large numbers.

The most rational is the choice of the gcd 8 algorithm (binary iterative algorithm with a shift), the reason for which is that it is designed to solve very time-consuming tasks. In this regard, this algorithm is unlikely to be applied from a practical point of view for numbers with a small range smaller than 1000.

Thus, the use of this algorithm will allow you to calculate the GCD of two numbers 29% faster than using the standard Euclidean algorithm. It is also worth noting that in the algorithm when forming the number a, there is an operation to generate a random number that is not fast enough and cyber-safe in the calculation. The standard operation of exponentiation is also not fast enough in this algorithm. We propose to use an additive chain algorithm, since it works faster and meets our requirements.

In the algorithm, when generating a random number a, the operation of generating a random number is used, which is not fast enough and cyber–safe in the calculation. Therefore, it was decided to replace this operation on $M-n$ where $n<M$. The calculated result of the average efficiency of the algorithm was 72%.

The standard operation of exponentiation is also not fast enough in this algorithm. After analyzing all the existing algorithms, it was decided to use the algorithm of additive chains, the average efficiency of which is 53% higher compared to the standard algorithm

Having performed the modernization of the classical part of Shor's algorithm, testing was performed to calculate the secret exponent of d RSA cipher index, where d takes values in the range from 3 to 3000.

The graph shows the number of operations performed by algorithms to decipher the value of d.

The result of calculations using algorithms: 1) Pollard's ρ–method; 2) Shor's; and 3) modified Shor's algorithm are presented in the Figure 1:

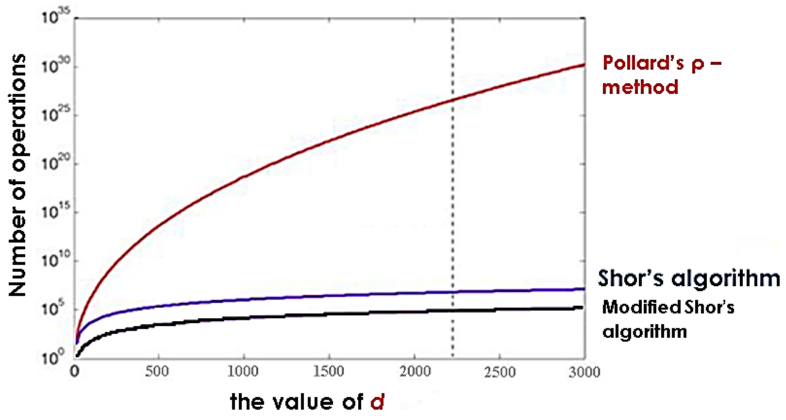

Fig. 1. Algorithm Comparison.

Based on the results of this experiment, there is an advantage of Shor's quantum algorithm, modified by upgrading its classical part with an iterative binary algorithm with the shift. It allows to the algorithm operating on 12% faster.

Thus, comparing the algorithms of Shor's and Pollard's ρ – method, it is easy to notice that the quantum algorithm performs the data processing and calculations much faster and, analyzing the graph, the computation time increases slightly depending on the increase in the processed number.

6 Conclusion

The studies performed in this paper show the superiority of quantum computing algorithms over modern non-quantum algorithms, whose productive power is significantly lower. The tested modified quantum algorithm showed high performance in front of the standard algorithm.

The results of testing two algorithms show that, depending on the increase in the value of the processed number, the elapsed time and the number of calculations on them increases.

The Shor's quantum algorithm improved by the authors showed its efficiency due to the modernization of the classical part, which works 50% faster than the standard algorithm.

References

1. Dushkin R V 2014 *Quantum Computation and Functional Programming* (M) p 318
2. Shor P W 1994 *Proceedings of the 35th annual IEEE symposium of foundations of Computer Science* 20 – 22
3. Nielson MA, Chuang I I 2000 *Quantum Computation and Information* (Cambridge: Cambridge Univ Press)
4. Monz T et al 2016 *Science* **351** 1068–1070
5. Hirvensalo M 2004 *Quantum Computing* (Berlin: Springer)
6. Johnsa R 1997 *Quantum preprint archive 9707033*
7. Lucero E et al 2012 *Nature Physics* **8** 719–723
8. Markov L, Saeedi M 2012 *Quantum Information and Computation* **12** 0361–0394
9. Kosyakov M S 2014 *Introduction to distributed computing NRU* (SPb: ITMO) p 155
10. Zheltov S 2014 *Effective computing in the CUDA architecture in information security applications* (M: RSHU Institute of ISST) p 145
11. V Yu Alpatov and M I Balzannikov 2020 IOP Conf. Ser.: Mater. Sci. Eng. 913 032057. doi:10.1088/1757-899X/913/3/032057
12. V Yu Alpatov and S M Petrov 2020 IOP Conf. Ser.: Mater. Sci. Eng. 913 042037. doi:10.1088/1757-899X/913/4/042037