International Transaction Journal of Engineering, Management, & Applied Sciences & Technologies

http://TuEngr.com

A STUDY OF THE CONVERGENCE OF THE BEZOUT COEFFICIENTS SEARCH ALGORITHM

Alisher R. Zhumaniezov [a, b*]

[a] *Department of Computer Science, Faculty of Electrical Engineering, Kazan Federal University, Kazan 420008, RUSSIAN FEDERATION.*

[b] *Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, CZECH REPUBLIC.*

ARTICLEINFO

Article history:
Received 05 September 2018
Received in revised form 26 November 2018
Accepted 07 December 2018
Available online
11 December 2018

Keywords:
Euclidean algorithm; extended Euclidean algorithm; k-ary algorithm for computing GCD; calculation of inverse elements modulo; parallel GCD computation.

ABSTRACT

Time is very valuable in modern technology era. An important indicator of the program's work is its computational speed. This article describes the optimization of Bezout coefficients search algorithm by introduction different optimization schemes. Among all schemes, the following are chosen: K-ary, approximating and parallel approximating optimization schemes. Bezout's equation is a representation of the greatest common divisor d of two integers A and B as a linear combination Ax+By = d, where x, and y are integers called Bezout's coefficients. Bezout's coefficients are counted using the extended version of the classical Euclidian Algorithm.

1. INTRODUCTION

In the age of modern technology, time is a very valuable resource. Therefore, an important indicator of the program's work is its computational speed. The process of increasing the speed goes in the following directions:

1. Increase the productivity of equipment. Achieved by using more powerful processors. The main difficulty in this approach is the great difficulty in manufacturing such processors. Another difficulty is the renewal of equipment on all machines, which entails additional costs.
2. Parallelizing the program. The distribution of common work between different threads allows you to reduce the total running time of the program. However, the Bezout coefficients search algorithm is iterative, and therefore not subject to parallelization.
3. Reduction of the asymptotic complexity of the algorithm used. This approach will be used in this work. It consists in reducing the number of elementary operations used by the algorithm.

The Bezout relation is the representation of the greatest common divisor of integers in the form

*Corresponding author (A.R.Zhumaniezov). Tel: 8(927)4068857. E-mail: myzerix58@gmail.com, of their linear combination with integer coefficients (Hasse, 1950). For the case of two integers A and B and their $GCD(A, B) = d$ — is the greatest common divisor, this relation is written in the following form (Jones and Jones, 1998):

$$Au + Bv = d \tag{1}$$

Moreover, the coefficients u and v are called the Bezout coefficients.

Finding the Bezout coefficients can be used to solve the following problems:

1. The solution of linear Diophantine equations in the form (Sushkevich, 1954):

$$Ax + By \; = \; C \tag{2}$$

2. The solution of first-degree comparisons in the form (https://brilliant.org/wiki/bezouts-identity):

$$Ax \equiv B(mod \;\; m) \tag{3}$$

3. The search for an inverse element in a field — is a particular case of equation (3):

$$Ax \equiv 1(mod \;\; m) \tag{4}$$

4. It is the basis for some cryptographic algorithms with a public key, such as RSA (Menezes et al, 1996).

Euclid's algorithm is an effective algorithm for finding the greatest common divisor of two integers (Euclid's Elements, 1949). This algorithm is iterative and is defined by:

$$GCD(A, B) = GCD(B, A \;\; mod \;\; B) \tag{5}$$

Extended Euclidean algorithm is a modification of the Euclidean algorithm, which allows to find Bézout coefficients (Yegorov, 1923). The main idea is to represent the resulting remainder at each iteration in the form of a linear combination of the original numbers A and B. For the "reverse" approach we use the following formula (Akritas, 1994):

$$\begin{gathered} u_i = v_{i+1} \\ v_i = u_{i+1} - v_{i+1} * q_i \end{gathered} \tag{6}$$

The main idea of optimizing the Bezout coefficients search algorithm will be to change the choice of a new number, which should lead to a faster convergence of the algorithm and, accordingly, to a decrease in the number of iterations.

The main disadvantage of this approach is an increasing of the load per iteration, which can lead to a decrease in the speed of the complete algorithm. Therefore, we must also optimally find new values for the next iteration.

At first, we estimate the convergence of the algorithms, giving them an estimate of the number of iterations. After that using the implementation in the language Python 3, we will test in practice the efficiency of the considered optimization schemes.

2. METHOD

2.1 K-ary extended Euclidean algorithm

The k-ary Euclidean algorithm was first published in Sorenson (Sorenson, 1990; Sorenson, 1994). Then, Weber and Jebelean improvements were proposed (Weber, 1995; Jebelean, 1993).

This algorithm is iterative, like the original algorithm. The main difference is to find a new pair for the next step. For this, we use the theorem proved in Weber (1995) and Jebelean (1993):

Theorem 1. Let $A, B > 0$ are natural numbers and k is a small positive number that is coprime to A and B. Then there are integers x and y, satisfying the relation $|x|, |y| \leq \left[\sqrt{k}\right] + 1$ such that:

$$A * x + B * y \equiv 0 \ \ (mod \ \ k) \tag{7}$$

Thus, it follows directly from the theorem that the number:

$$C = {}^{(A * x + B * y)}\!/_{k} \tag{8}$$

Is natural. In case that $GCD(C, k) = d \neq 1$ it is necessary to divide the number C by d. Then the transition to the next iteration has the form:

$$GCD(A, B) = GCD(B, C) \tag{9}$$

Note that when the algorithm works, additional multipliers may appear (Ishmukhametov, 2016). To eliminate them, you must start recursively $GCD(A, GCD(B, d))$, where d is the result obtained at the output of the k-ary algorithm.

Now we get optimization for the extended Euclidean algorithm. It was proposed in the (Ishmukhametov, 2016). In this algorithm, too, we will use the "reverse" approach. To construct the transition formula, we present the Bezout equation:

$$A_{i+1} * u_{i+1} + B_{i+1} * v_{i+1} = d \tag{10}$$

Now we substitute in this equation the transition formulas (8) and (9) and group terms. Thus, we have explicitly obtained the transition formula for the Bezout coefficients (Ishmukhametov, 2016):

$$\begin{aligned} u_i &= {}^{x_i * v_{i+1}}\!/_{k} \\ v_i &= u_{i+1} + {}^{y_i * v_{i+1}}\!/_{k} \end{aligned} \tag{11}$$

However, in this formula there is one drawback - the non-integer division into k. Since it is not guaranteed that the result of dividing will always be integer, this move our calculations into the field of real numbers, which is undesirable for us. Therefore, to stay in the field of integers, we introduce auxiliary variables u_i^* and v_i^* (Ishmukhametov, 2016). They are defined as follows:

$$\begin{aligned} u_i^* &= u_i * k^{n-i} \\ v_i^* &= v_i * k^{n-i} \end{aligned} \tag{12}$$

Then the recurrence relation for them will be written in the form (Ishmukhametov, 2016):

$$\begin{aligned} u_i^* &= x_i * v_{i+1}^* \\ v_i^* &= k * u_{i+1}^* + y_i * v_{i+1}^* \end{aligned} \tag{13}$$

In case there is an additional factor $r_i = 2^{t_i}$ for $k = 2^t$, then the relation (13) becomes:

$$\begin{aligned} u_i^* &= x_i * v_{i+1}^* \\ v_i^* &= k * r_i * u_{i+1}^* + y_i * v_{i+1}^* \end{aligned} \tag{14}$$

Since we can't divide by $k^{n-i} * r$, where $r = \prod_{i=r}^{n} r_i = 2^{\sum_{i=1}^{n} t_i} = 2^T$ for $k = 2^t$ because we leave the field of integers, we can replace it by an integer division modulo $A = A_1$. We obtain the intermediate coefficients u' and v' by formula for $k = 2^t$:

$$\begin{aligned} u' &= (u_1^* * k^{1-n} * 2^{-T}) \;\; mod \;\; A \\ v' &= (v_1^* * k^{1-n} * 2^{-T}) \;\; mod \;\; A \end{aligned} \qquad (15)$$

Then the Bezout equation takes the form:

$$A_1 * u' + B_1 * v' = d_1 \equiv d(mod \;\; A) \qquad (16)$$

Hence we obtain the final formula for the Bezout coefficients:

$$\begin{aligned} u &= u' - (d_1 \;\; div \;\; A) \\ v &= v' \end{aligned} \qquad (17)$$

2.2 Approximating extended Euclidean algorithm

Optimization of the k-ary algorithm was presented by Ishmukhametov and Rubtsova (2016). The main idea is the replacement of the algorithm for finding the coefficients x and y for the next iteration. For this $|x|$ is taken from the interval $(0; k)$ and y is taken close to $-{}^{A * x}/_{B}$.

At first we introduce the following notation:

$$q = A/B \;\; mod \;\; k \qquad (18)$$

$$r = A/B \qquad (19)$$

$$r - q = r_1 + k * s_1 \qquad (20)$$

$$r_0 = r_1/k \qquad (21)$$

Now we get the formula for C (Ishmukhametov and Rubtsova, 2016):

$$|C| = \left|\frac{A*x+B*y}{k}\right| = B\left|\frac{r*x+y}{k}\right| = B|r_0 x + (s_1 * x + s)| = B\left|\frac{u}{v}x + s_1 * x + s\right| \qquad (22)$$

We note that the original fraction $\frac{u}{v}$ has an arbitrary form, so we must find an approximation $\frac{m}{n} \approx \frac{u}{v}$ (for example an algorithm using Farey's sequence) on condition $n < k$. Then formula (22) takes the form:

$$|C| = B\left|\frac{u}{v}x + s_1 * x + s\right| \approx B\left|\frac{m}{n}x + s_1 * x + s\right| \qquad (23)$$

Then the minimum, which is equal to 0, will be achieved with $x = n$ and $s = -m - s_1 * n$. Now we obtain the final formulas for the transition:

$$\begin{aligned} x &= n \\ s &= -m - s_1 * n \\ y &= s * k - q * x \end{aligned} \qquad (24)$$

The basic steps of the algorithm are the same as for the k-ary extended algorithm. The only difference is the use of a different approach for choosing the coefficients x and y. For this, an algorithm using Farey's sequence is used, as in the usual approximating algorithm.

2.3 Parallel approximating extended Euclidean algorithm

When using Farey sequence, we first obtain an approximation segment, from which we then select one of the end points. Hence the second approach follows, when we take both end points (Ishmukhametov and Rubtsova, 2017). Then the transition to (9) is transformed into:

$$GCD(A,B) = GCD(C_1, C_2) \tag{25}$$

Then the transition to (13) is transformed into:

$$\begin{aligned} u_i^* &= x_i^1 * u_{i+1}^* + x_i^2 * v_{i+1}^* \\ v_i^* &= y_i^1 * u_{i+1}^* + y_i^2 * v_{i+1}^* \end{aligned} \tag{26}$$

And if there are additional factors r_i^1 and r_i^2 introduce $r_i = LCM(r_i^1, r_i^2)$ and additions to r_i:

$$\begin{aligned} r_i^{1\prime} &= r_i / r_i^1 \\ r_i^{2\prime} &= r_i / r_i^2 \end{aligned} \tag{27}$$

Then instead of (14) we obtain the following transition:

$$\begin{aligned} u_i^* &= x_i^1 * u_{i+1}^* * r_i^{1\prime} + x_i^2 * v_{i+1}^* * r_i^{2\prime} \\ v_i^* &= y_i^1 * u_{i+1}^* * r_i^{1\prime} + y_i^2 * v_{i+1}^* * r_i^{2\prime} \end{aligned} \tag{28}$$

All other formulas keep unchanged.

For parallelization, the portion of the algorithm that computes C_1 and C_2. Each of them can be calculated independently.

3. RESULTS AND DISCUSSION

3.1 K-ARY EXTENDED EUCLIDEAN ALGORITHM

We construct an upper bound for C:

$$|C| = {|A * x + B * y|}/{k} \leq {|A * x| + |B * y|}/{k} = {A * |x|}/{k} + {B * |y|}/{k} \leq {2 * A}/{\sqrt{k}} \tag{29}$$

From this estimate it follows that at each step the greater of the numbers decreases at least ${\sqrt{k}}/{2}$ times (Ishmukhametov and Rubtsova, 2016). Thus, we obtain an estimation for the number of iterations $T(A,B)$ for any pair A and B:

$$T(A,B) \leq \log_{\sqrt{k}/2}(A) + \log_{\sqrt{k}/2}(B) \tag{30}$$

Thus, we obtain a upper bound for the average number of iterations $\tau(A)$ for any A (Ishmukhametov and Rubtsova, 2016):

$$\tau(A) \leq 2 * \log_{\sqrt{k}/2}(A) = \frac{2}{\ln\left(\sqrt{k}/2\right)} \ln(A) = \frac{2}{\ln(k)/2 - \ln(2)} \ln(A) \approx \frac{4}{\ln(k)} \ln(A) \tag{31}$$

3.2 APPROXIMATING EXTENDED EUCLIDEAN ALGORITHM

The proof of the following theorem was presented by Ishmukhametov (2016):

Theorem 2. There is an integer $x, 1 \leq x \leq k$, such that:

$$\{r_0 x\} \leq \frac{3}{2k} \tag{32}$$

And the search for such x can be produced by an algorithm with complexity $O(\log_2(k))$.

From the theorem it follows directly that it is possible in an acceptable time to find such a pair of coefficients x and y, that:

$$|C| = \left|\frac{A*x+B*y}{k}\right| \leq \frac{3B}{2k} \tag{33}$$

That is, at each step there is a decrease in one number, at least in ${}^{2k}/_{3}$ times. Thus we obtain a estimation on the number of iterations $T(A, B)$ for any A and B from above (Ishmukhametov and Rubtsova, 2016):

$$T(A, B) \leq \log_{2k/_3}(A) + \log_{2k/_3}(B) \tag{34}$$

Then the average number of iterations $\tau(A)$ for any A will be limited from above:

$$\tau(A) \leq 2 * \log_{2k/_3}(A) = \frac{2}{\ln({}^{2k}/_{3})}\ln(A) == \frac{2}{\ln(k)-\ln(1.5)}\ln(A) \approx \frac{2}{\ln(k)}\ln(A) \tag{35}$$

3.3 PARALLEL APPROXIMATING EXTENDED EUCLIDEAN ALGORITHM

All calculations for this algorithm are the same as for the approximating. So, as a result, we get the formula (35).

4. SUMMARY

4.1 4-BIT MODULE

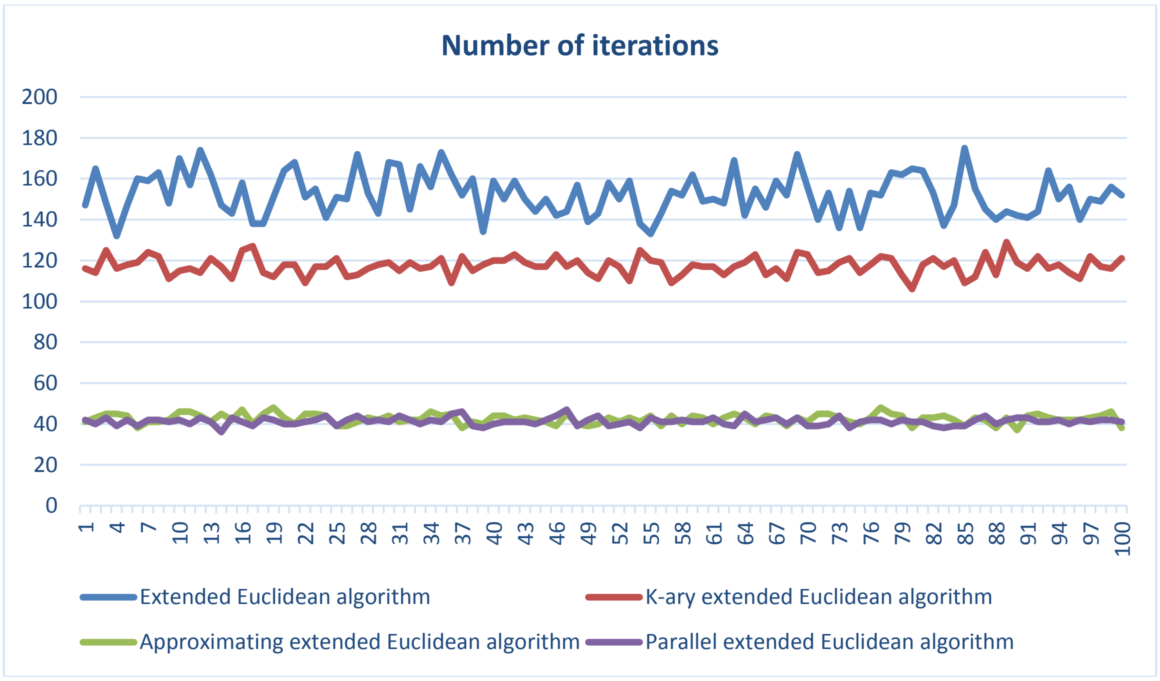


Figure 1: Comparison of the number of steps with the original algorithm (binary, 4 bit).

k-ary: As can be seen from Figure 1, the number of steps became slightly less than 1.5 times less compared to the original algorithm. However, of all the algorithms considered, this is the worst result.

Approximating: Figure 1, the number of iterations has decreased more than 3 times in comparison with the original algorithm. The result was close to parallel.

Parallel: Figure 1, the number of iterations has decreased more than 3 times compared to the original algorithm. The result was close to approximating.

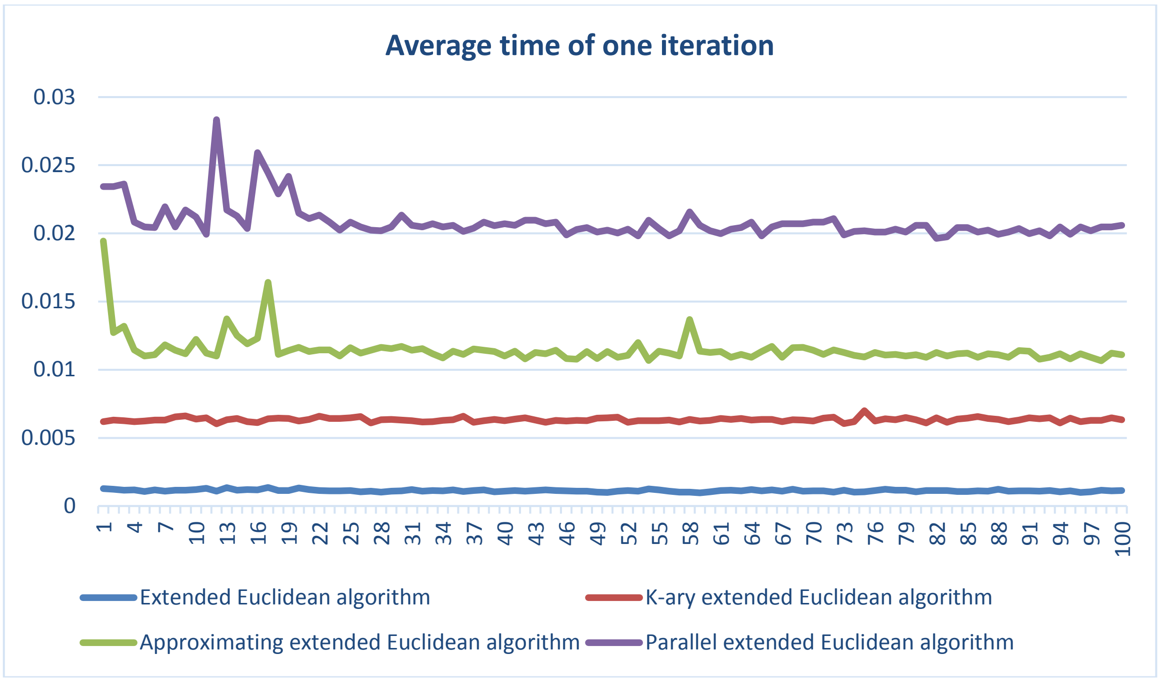


Figure 2: Comparison of the running time of the step with the original algorithm (binary, 4 bit).

From Figure 2, k-ary: the amount of work on one iteration increased about 6 times compared to the original algorithm. This leads to a time lag in comparison with the original algorithm. It is also clear that the running time at the step is very close to the approximating algorithm.

Approximating: Figure 2, the amount of work on one iteration increased approximately 10 times compared to the original algorithm. This leads to a time lag in comparison with the original algorithm.

Parallel: As can be seen from Figure 2, the amount of work on one iteration increased approximately 20 times compared to the original algorithm. This leads to a time lag in comparison with the original algorithm. This also leads to a lag in comparison with the approximating algorithm.

4.2 8-BIT MODULE

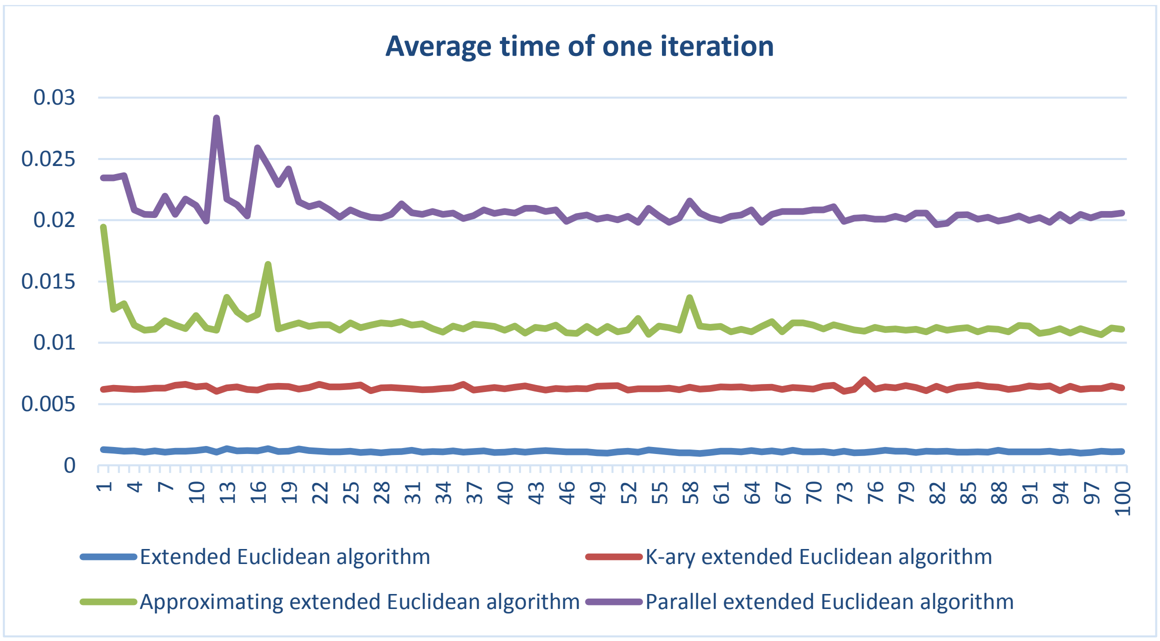


Figure 4: Comparison of the number of steps with the original algorithm (binary, 8-bit).

k-ary: As can be seen from Figure 3, the number of steps became approximately 2 times less compared to the original algorithm. However, of all the algorithms considered, this is the worst result.

Approximating: Figure 3, the number of iterations has decreased by approximately 6 times in comparison with the original algorithm. The result was close to parallel.

Parallel: Figure 3, the number of iterations has decreased by approximately 6 times in comparison with the original algorithm. The result was close to approximating.

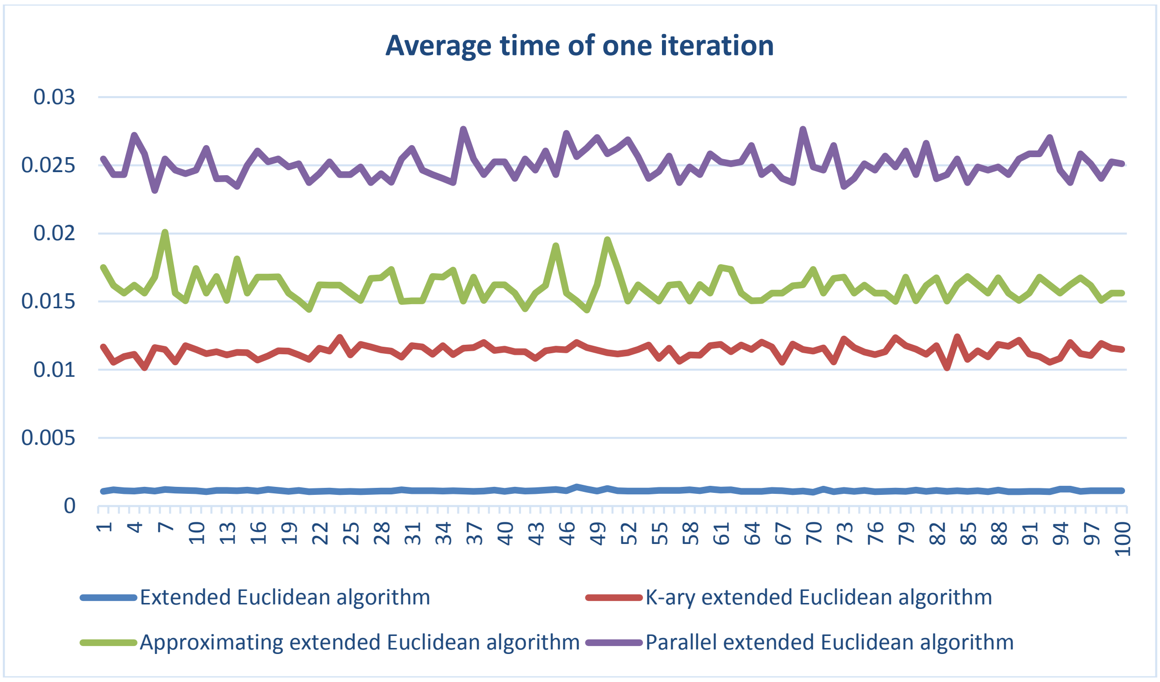


Figure 4: Comparison of the running time of the step with the original algorithm (binary, 8-bit).

k-ary: As can be seen from Figure 4, the amount of work on one iteration increased by about 10 times in comparison with the original algorithm. This leads to a time lag in comparison with the original algorithm. It is also clear that the running time at the step is very close to the approximating algorithm.

Figure 4, Approximating: the amount of work on one iteration increased approximately 15 times compared to the original algorithm. This leads to a time lag in comparison with the original algorithm.

Figure 4, Parallel: the amount of work on one iteration increased approximately 20 times compared to the original algorithm. This leads to a time lag in comparison with the original algorithm. This also leads to a lag in comparison with the approximating algorithm.

k-ary: from Figure 5, the number of steps became approximately 3 times less compared to the original algorithm. However, of all the algorithms considered, this is the worst result.

Approximating: Figure 5, the number of iterations has decreased by about 7 times compared to the original algorithm. The result was close to parallel.

Parallel: Figure 5, the number of iterations has decreased by about 7 times compared to the original algorithm. The result was close to approximating.

4.3 16-BIT MODULE

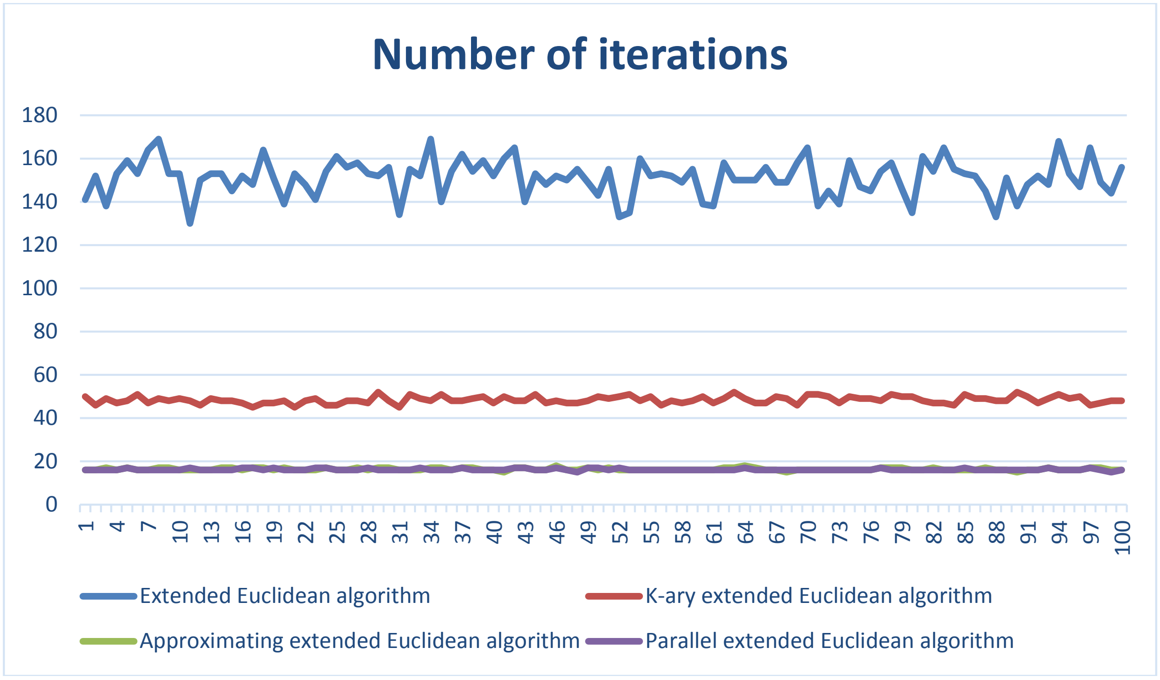


Figure 5: Comparison of the number of steps with the original algorithm (binary, 16-bit).

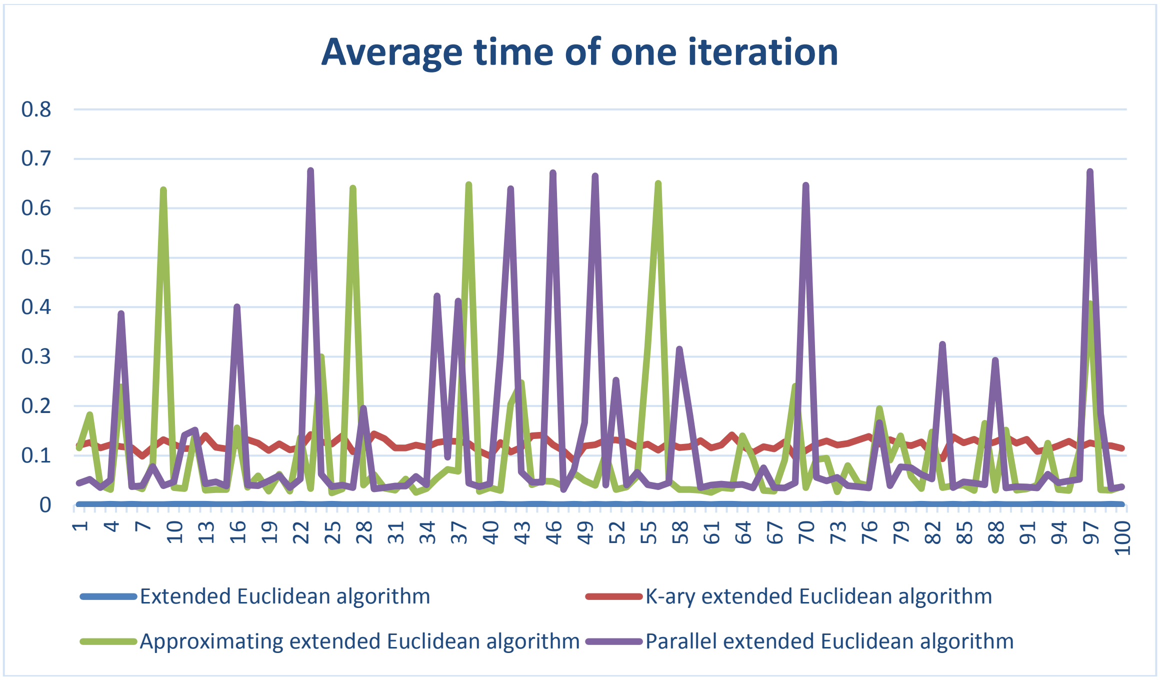


Figure 6: Comparison of the running time of the step with the original algorithm (binary, 16-bit).

k-ary: shown in Figure 6, the amount of work on one iteration has greatly increased in comparison with the original algorithm. This leads to a time lag in comparison with the original algorithm.

Approximating: Figure 6, the amount of work on one iteration has an extremely strong variance. Because of this, it becomes difficult to evaluate the attitude to the original algorithm. It also becomes difficult to compare with other schemes.

Parallel: see Figure 6, the amount of work on one iteration has an extremely strong variance. Because of this, it becomes difficult to evaluate the attitude to the original algorithm. It also becomes difficult to compare with other schemes.

5. CONCLUSION

The theoretical and practical questions of the Bezout coefficients search are considered. Developed: a program containing the implementation of all the considered optimization schemes for the extended Euclidean algorithm. Also, in the dissertation, to evaluate the efficiency of optimization of the extended Euclidean algorithm, experiments were performed and the results are shown.

In the course of this work, the following conclusions were made on the schemes considered:

k-ary — showed the worst result among all the considered schemes. At small values of the module, the number of steps turned out to be greater than that of the original one, at large there was a significant gain. The amount of work on one iteration with a small module is less than that of other schemes, but more than the original algorithm. This leads to a total loss in time for the entire algorithm in comparison with the original.

Approximating — showed the best result among all the considered schemes. The number of steps is several times less than the original algorithm. For large values of the module, the number of steps coincides in parallel with the number of steps. The amount of work on one iteration with a small module is less than that of a parallel module, but more than the original algorithm. However, for large values, the dispersion becomes too large, which complicates the analysis. One of the reasons may be the nonuniform convergence of the algorithm on Farey's fractions.

Parallel — the number of steps is several times less than the original algorithm. For large values of the module, it coincides with the number of steps with the approximating step. The amount of work on one iteration with a small module is the largest in comparison with other algorithms. However, for large values, the dispersion becomes too large, which complicates the analysis. One of the reasons may be the nonuniform convergence of the algorithm on Farey's fractions.

The further direction of the study may be the reduction of the work time spent on one iteration. Among the possible approaches: the disclosure of internal function calls, the use of a lower-level language, the use of a faster algorithm for selecting coefficients.

6. ACKNOWLEDGEMENTS

The work is performed according to the Russian Government Program of Competitive Growth of Kazan Federal University.

7. REFERENCES

Hasse, H. (1950). «Vorlesungen Uber Zahlentheorie», *Berlin*, p. 59.

Jones, G. A., Jones, J. M. (1998). §1.2. Bezout's Identity. Elementary Number Theory. Berlin: *Springer-Verlag*, p. 302.

Sushkevich, A. K. (1954). The Theory of Numbers. Elementary course., Kharkov, Kharkov *University Press*, 204 p.

https://brilliant.org/wiki/bezouts-identity/ [accessed Oct 2016]

Menezes, A., van Oorschot, P., Vanstone, S. (1996). Handbook of Applied Cryptography. *CRC Press*, p. 816.

Euclid’s Elements V. 2. (1949). translated from Greek and commented by Mordukhai-Boltovsky D. D. Edited by Vygotsky M. Y. and Veselovsky I.N.., M., GITTL, p. 511.

Yegorov, D. F. (1923). The Elements of Number Theory., M., *Petrograd: Gosizdat*, p. 202.

Akritas, A. (1994). The principles of Computer Algebra with Appendix: translated from English, M., Mir, p. 544.

Sorenson, J. (1990). The k-ary GCD Algorithm, Universitet of Wisconsin-Madison, *Tecn.Report*, pp. 1–20.

Sorenson, J. (1994). Two fast GCD Algorithms, J.Alg. 16, No. 1, pp. 110–144

Weber, K. (1995). The accelerated integer GCD algorithm, *ACM Trans.Math.Software*, 21, No. 1, pp. 1–12.

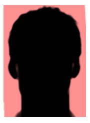
Jebelean, T. A. (1993). Generalization of the Binary GCD Algorithm, Proc. Of Intern.Symp.on Symb.and Algebr. Comp.(ISSAC’93), pp. 111-116.

Ishmukhametov, S. T. (2016). Calculation of Bezout Coefficients for k-ary for GCD Algorithm., KFU, *Kazan, Russia*, p. 6.

Ishmukhametov, S. T. (2016). Rubtsova P.G. About Approximating k-ary GCD Algorithm , KFU, *Kazan, Russia*, p. 4.

Ishmukhametov, S. T., Rubtsova, R. G. (2017). A parallel computation of the GCD of natural numbers//Параллельные вычислительные технологии – XI международная конференция, ПаВТ'2017, г. Казань, 3–7 апреля 2017 г.- Челябинск: Издательский центр ЮУрГУ. pp.120-129

Ishmukhametov. S. T. (2016). An Approximating k-ary GCD Algorithm, *Lobachevskii Journal of Mathematics*, Volume 37, Issue 6, pp. 723-729



Alisher Zhumaniezov obtained his master degree from Czech Technical University in Prague (Study Program: Open Informatics, Field of Study: Software Engineering). He is associated with Department of Computer Science, Faculty of Electrical Engineering, Kazan Federal University, Kazan, RUSSIA. His research interests encompass algorithm engineering.