ISSN 0361-7688, Programming and Computer Software, 2025, Vol. 51, No. 2, pp. 109–116. © Pleiades Publishing, Ltd., 2025.
Russian Text © The Author(s), 2025, published in Programmirovanie, 2025, Vol. 51, No. 2.

On Binary Solutions to a System of Linear Equations Modulo Three

O. A. Zverkov[a,*] (ORCID: 0000-0002-8546-364X)
and A. V. Seliverstov[a,] (ORCID: 0000-0003-4746-6396)**

[a] *Institute for Information Transmission Problems (Kharkevich Institute), Russian Academy of Sciences, Moscow, 127051 Russia*

**e-mail: zverkov@iitp.ru*

***e-mail: slvstv@iitp.ru*

Received July 30, 2024; revised August 28, 2024; accepted September 19, 2024

Abstract—We consider the problem of finding a binary solution to a system of linear equations modulo three. In the case where the number of equations is less than a sufficiently slowly growing function of the number of variables, a new polynomial-time algorithm is proposed to recognize the existence of a binary solution to such a system. The algorithm is based on the note that, if the coefficient matrix contains nonzero columns proportional to each other, then the elimination of the corresponding variables preserves the property of having no binary solution to the system. In particular, every system of two equations in five variables permits the elimination of some variables that preserves the property of having no binary solution to the system. Based on these results, we propose an errorless heuristic algorithm, which is implemented in the Python programming language. The NumPy library is used to represent matrices and perform basic operations. The input is the augmented matrix of the system. The empirical runtime estimate is calculated using the implementation. It is experimentally shown that the algorithm is more efficient for sparse systems of equations. Obviously, the binary search method can find a binary solution to the system when one exists. This observation opens up the possibility of its practical use, in particular, for solving problems of mathematical biology.

Keywords: finite field, system of linear equations, and computer algebra system

DOI: 10.1134/S0361768824700919

1. INTRODUCTION

Let us denote the field of residues modulo three by $GF(3)$. The elements of $GF(3)$ are denoted by numbers from set $\{0,1,2\}$. Operations in this field are addition and multiplication modulo three, in particular, both expressions -1 and $1/2$ are equal to element 2. We consider systems of algebraic equations over the field $GF(3)$. In this paper, a polynomial-time algorithm is proposed for a special case of the problem of incidence of a subspace and points with coordinates from set $\{0, 1\}$. This problem is considered computationally difficult in the worst case. In contrast to [1, 2], here, we consider systems of fewer equations.

The proposed algorithm is implemented in Python. Due to the availability of libraries such as NumPy, which are used in this implementation to represent matrices and perform basic operations, Python provides a good tradeoff between development velocity and performance.

Systems of algebraic equations over a finite field admit geometric interpretation. Since the 19th century, many finite geometries that use only finite numbers of points and lines have been proposed [3, 4]. In each of them, any statement about the mutual arrangement of points and lines is resolvable, because the complete enumeration of a finite number of variants is feasible [5, 6]. However, the computational complexity can be high, which explains the importance of finding special cases computable in polynomial time. On the other hand, calculations over finite fields often prove less difficult than calculations over the field of rational numbers because, over a finite field, the length of each coefficient of equations is limited [7].

Over the field $GF(3)$, each affine line has exactly three points. An affine plane corresponds to the Hesse configuration [8]. On this plane, there are 9 points and 12 lines. In the three-dimensional affine space, there are 27 points, 117 straight lines, and 39 planes. Each pair of distinct points defines one straight line, and each straight line can be defined by three pairs of points. Therefore, in the n-dimensional affine space, there are 3^n points and $3^{n-1}(3^n-1)/2$ lines. Due to duality, the n-dimensional projective space contains $(3^{n+1}-1)/2$ points and the same number of hyper-

planes. Therefore, in the affine space, the number of hyperplanes is $(3^{n+1}-1)/2-1$.

A solution to a system of equations in which the value of each variable belongs to set {0, 1} is called the (0, 1)-solution or the binary solution. Recognizing the existence of a (0, 1)-solution to a system of linear equations over $GF(3)$ is an NP-complete problem. However, for a single equation, this problem is easy to solve: only a linear equation of form $x_k = 2$ has no (0, 1)-solution, whereas every linear equation that depends nontrivially on two or more variables has one. Moreover, the problem of finding (0, 1)-solutions can also be solved in polynomial time for systems with a fixed number of linear equations. For instance, we can use the reducibility of the problem over $GF(3)$ to a similar problem over the field of rational numbers, for which many algorithms were proposed. The descriptions of these algorithms can be found in [9, 10]. In contrast, we consider systems in which the number of equations is bounded not by a constant but by a monotonically increasing function of the number of variables. Geometrically, the problem consists in recognizing the incidence of a subspace and vertices of a cube. The decomposition of an NP-complete problem into several NP-complete and low-complexity problems is consistent with recent results [11].

Probabilistic algorithms are also used to solve similar combinatorial problems [12]. The distribution of sums of random (0, 1)-variables was considered by Yashunskii [13]. The distribution of entries in matrices of a given rank over a finite field [14], as well as the distribution of ranks of random matrices with a given number of nonzero entries [15, 16], was also investigated. The ratio between the number of nondegenerate matrices and the number of all matrices over the field $GF(3)$ decreases monotonically with increasing order n; however, above the asymptotic value [17], it is equal to the product

$$\prod_{k=1}^{\infty}\left(1-\frac{1}{3^k}\right) = 0.560126\ldots$$

On the other hand, the so-called generic algorithms or errorless heuristics are discussed, which give correct answers on a large number of inputs among inputs of a given length and satisfy easily verifiable constraints. However, such an algorithm may return a warning about its inapplicability. In any case, the algorithm is errorless. The formal definitions can be found in Rybalov's works [18, 19].

Section 2 provides the theoretical substantiation of the algorithm and related results. Section 3 discusses the implementation of the algorithm and the computational results. Section 4 provides brief conclusions.

2. THEORETICAL RESULTS

Suppose that a system of linear equations in variables $x_1, \ldots, x_n$ contains more than one equation, and some equation depends nontrivially on variable x_k. We say that a new system of linear equations is derived from the original system by eliminating variable x_k if the new system does not depend on variable x_k and the original system is equivalent to the union of the new system and exactly one equation (which depends on x_k) corresponding to the linear combination of equations of the original system. The elimination of a variable corresponds to the projection onto a coordinate subspace. Let us consider the following example:

$$\begin{cases} x_1 + x_2 = 1 \\ x_1 - x_2 + x_3 + x_4 = 0. \end{cases}$$

The elimination of variable x_3 (or x_4) gives one equation $x_1 + x_2 = 1$. Each of its (0, 1)-solutions can be extended to the (0, 1)-solution to a system of two equations. Indeed, if two variables x_3 and x_4 run independently through values from set {0, 1}, then their sum $x_3 + x_4$ takes on all three values from the field $GF(3)$. This is also true for their difference $x_3 - x_4$. Therefore, simultaneous elimination of two variables, if possible, allows us to reduce the number of equations without violating the existence of (0, 1)-solutions.

The elimination of a variable may lead to a system with a larger number of (0, 1)-solutions as compared to the original system. The following result holds only over $GF(3)$, but not over fields with a larger number of elements.

Theorem 1. *Suppose that we have natural numbers n and m that satisfy inequalities $n \geq 5$, $m \geq 2$, and $m \leq \log_3(2n-1)$, as well as a system of m linear equations in n variables over field $GF(3)$. Suppose also that, for each index $1 \leq k \leq n$, there is an equation that depends nontrivially on variable x_k. If this system has no (0, 1)-solution, then there is index $k \leq n$ such that the elimination of variable x_k again leads to a system that has no (0, 1)-solution. Moreover, this system can be found in polynomial time $O(mn\log_2(n+1))$.*

Proof. The system of equations can be written in matrix form $A\mathbf{x} = \mathbf{b}$, where A is an $m \times n$ matrix of coefficients for the linear terms of the equations, while $\mathbf{x}$ and $\mathbf{b}$ are columns of n variables and m numbers, respectively. By the condition of the theorem, there are no zero columns in matrix A.

For $2 \leq m \leq \log_3(2n-1)$, matrix A has two linearly dependent columns. Indeed, the number of possible different nonzero columns is $3^m - 1$. This set is divided into $(3^m-1)/2$ pairs of linearly dependent columns. Therefore, condition $n \geq (3^m+1)/2$ ensures that A

has two linearly dependent columns. Let us denote the numbers of these columns by j and k. To find j and k, we can search through $n(n-1)/2$ variants while checking the linear dependence of the corresponding columns.

Original system $A\mathbf{x} = \mathbf{b}$ is equivalent to system $B\mathbf{x} = \mathbf{c}$, where nonzero entries in columns j and k of $m \times n$ matrix B are located only in one row, the number of which is denoted by ℓ. Here, matrix B is obtained from matrix A by elementary operations on rows, while an entry of column $\mathbf{c}$ is equal to the corresponding linear combination of entries of column $\mathbf{b}$. If the system of equations obtained by removing the ℓth equation from that system has a (0, 1)-solution, then it has a (0, 1)-solution for some (0, 1)-values of variables x_j and x_k. Therefore, the entire system also has a (0, 1)-solution because the choice of values of x_j and x_k allows the ℓth equation to hold for any estimate of the remaining variables. Removing the ℓth equation corresponds to eliminating each variable x_j and x_k. □

Let us make a note. Suppose that, in the five-dimensional space over $GF(3)$, there is a three-dimensional subspace that does not pass through any (0, 1)-point. The image of the projection onto a coordinate hyperplane, which corresponds to the elimination of a variable, is a subspace that also does not pass through any (0, 1)-point.

The following result makes it easy to check whether a subspace of low codimension passes through some (0, 1)-point.

Theorem 2. *There is a polynomial-time algorithm that receives a system of m linear equations in n variables over $GF(3)$ as an input and, provided that inequality $m \le \log_3\log_3(2n-1)$ holds, accepts the input if and only if the system has a* (0, 1)*-solution.*

Proof. The algorithm tries to eliminate variables in a loop in accordance with Theorem 1. All variables not included in a new system are also eliminated. If successful, the next step leads to a system with fewer equations. In this case, the new system has a (0, 1)-solution if and only if the original system has a (0, 1)-solution. Upon performing less than m steps, the process terminates in one of two possible cases: either there is only one equation left or the resulting system depends on a small number of variables.

If only one equation is left, then the input is rejected for an equation of type $x_k = 2$ and is accepted for an equation of the other types.

If k variables are left and the system contains several equations without possibility of further reduction, then 2^k cases are analyzed. Let us estimate k from above. Since the remaining number of equations does not exceed m, inequality $\log_3(2k-1) < m$ holds. However, $m \le \log_3\log_3(2n-1)$ according to the condition of applicability of the algorithm. Therefore, inequalities $(2k-1) < \log_3(2n-1)$ and $k \le 0.5\log_3(2n-1) < 0.3155\log_2(2n-1)$ hold. Hence, the number of different (0, 1)-estimates of the remaining k variables is less than $(2n-1)^{0.3155}$. □

If condition $m \le \log_3\log_3(2n-1)$ of Theorem 2 is violated, then the algorithm always gives a correct answer; however, its runtime may be longer. Nevertheless, in many cases, among inputs with given values of m and n, the runtime of the algorithm is short even under weaker constraint $m \le \log_3(2n-1)$.

Given the values of m and n that satisfy inequality $m \le \log_3(2n-1)$, almost any system of m equations in n variables over the field $GF(3)$ has many (0, 1)-solutions. This is another heuristic probabilistic algorithm for solvability checking, whereby randomly selected (0, 1)-points are checked. If a (0, 1)-solution is found, then it really exists; if no (0, 1)-solution is found, then the algorithm returns a failure. On the other hand, under this condition, the existence of a (0, 1)-solution can always be checked by a deterministic algorithm in quasi-polynomial time $n^{O(\log n)}$.

Theorem 3. *There is an algorithm that receives a system of m linear equations in n variables over field $GF(3)$ as an input and, in time $n^{O(m)}$, accepts the input if and only if the system has a* (0, 1)*-solution.*

Proof. Each (0, 1)-solution to a system of equations in n variables over field $GF(3)$ is extended to a (0, 1)-solution to a system of equations in $n + m\lceil\log_3 n\rceil$ variables over the field of rational numbers. Conversely, each solution to the new system is shortened to the solution to the original system. Each system has m equations. Here, the jth equation of form $a_{j0} + a_{j1}x_1 + \cdots + a_{jn}x_n = 0$ over field $GF(3)$ corresponds to equation $a_{j0} + a_{j1}x_1 + \cdots + a_{jn}x_n = 3y_{j1} + 9y_{j2} + \cdots + 3^k y_{jk}$ over the field of rational numbers, where $k = \lceil\log_3 n\rceil$ and each new variable y_{j1}, ..., y_{jk} occurs only once. The coefficients of the new system are integers the absolute values of which do not exceed $3n$. In turn, (0, 1)-solutions to this system coincide with (0, 1)-solution to the new system, generally speaking, is not the solution to the original system. Therefore, solutions to one equation equal to the linear combination of equations to the system in which coefficients are bounded from above by $n^{O(m)}$. Then, the search for (0, 1)-solutions can be performed by dynamic programming [9] in time not exceeding $n^{O(m)}$. □

It is possible to additionally modify the system to equations while preserving the number of variables and the property to having a (0, 1)-solution. This allows us to expand the scope to our algorithm. However, in this case, the (0, 1)-solution of the new system,

generally speaking, is not the solution of the original system. Therefore, this approach is convenient for checking the existence of a (0, 1)-solution but not for finding it.

Theorem 4. *Suppose that we have a system of linear equations in n variables over the field $GF(3)$*

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\ \cdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n = b_m \end{cases}$$

and an integer $1 \le s \le n$. This system has a (0, 1)-solution if and only if there is a (0, 1)-solution to a new system where, for each $1 \le j \le m$ in the jth equation, coefficient a_{js} of variable x_s is replaced by the linear combination of coefficients

$$c_j = 2b_j - \sum_{k=1}^{n} a_{jk}.$$

Proof. Let us consider an auxiliary system in which the linear term of new variable y is added into each equation:

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n + c_1 y = b_1 \\ \cdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n + c_m = yb_m, \end{cases}$$

where coefficients c_j are defined in the condition of the theorem. One of the solutions to this system is obtained when all variables are equal to element 2. For $y = 0$, each (0, 1)-solution extends some (0, 1)-solution to the original system. For $y = 1$, each (0, 1)-solution is obtained from some (0, 1)-solution for $y = 0$ by simultaneously replacing the other variables $x_k \to 1 - x_k$. Therefore, the auxiliary system has a (0, 1)-solution if and only if the original system has a (0, 1)-solution. Moreover, the set of (0, 1)-solutions to the auxiliary system is divided into pairs of antipodal solutions that pass into each other under the simultaneous inversion of values of all variables. Therefore, if a (0, 1)-solution exists, then the auxiliary system has a pair of antipodal (0, 1)-solutions both for $x_s = 0$ and $x_s = 1$.

Next, we fix value $x_s = 0$ and obtain a system of n variables x_1, ..., x_{s-1}, y, x_{s+1}, ..., x_n. Let us replace the name of variable y with x_s. The new system of n variables has a (0, 1)-solution if and only if the original system has a (0, 1)-solution, although the solution to one system, generally speaking, is not a solution to the other system. □

It should be noted that the geometric meaning of the transformation from Theorem 4 is in the projective transformation, whereby a hyperplane that passes through a point in the affine part with coordinates $(2,\cdots,2)$ and is not incident to any (0, 1)-point becomes an improper hyperplane. This transformation serves as an involution. The composition of several such transformations does not lead to essentially new systems.

Let us consider another easily verifiable condition for the existence of a (0, 1)-solution to a system of linear equations over $GF(3)$.

Theorem 5. *Given a system of linear equations over field $GF(3)$, suppose that this system has a solution where all variables, except one, are equal to 2. If the system has no (0, 1)-solution, then the elimination of some variable leads to a new system that also has no (0, 1)-solution.*

Proof. Without loss of generality, we can assume that the original system of equations has solution $(0,2,\cdots,2)$, where all variables, except for the first one, are equal to 2. Suppose that the solution to the system is a point with coordinates $(2,a_2,\cdots,a_n)$, where, for $j \ge 2$, all values of a_j belong to set {0, 1}. Then, the (0, 1)-solution to the system is a point with coordinates $(1,1-a_2,\cdots,1-a_n)$. □

3. IMPLEMENTATION AND DISCUSSION

Function `has01solution`(M) is implemented. Its input is nonempty matrix M with m rows over field $GF(3)$. Let us denote the last column of this matrix, called the auxiliary one, by **b**. By A, we denote the submatrix located in the first n columns, except for the last one. Column **b** contains free terms of equations, while matrix A consists of coefficients of linear terms. If matrix A is empty, then linear parts of all equations vanish, and equations themselves turn into equality $0 = 0$ or false formulas $0 = 1$ or $0 = 2$. Matrix M serves as an augmented matrix of a system of equations. Matrix M is modified in such a way that its last column always contains free terms, while its other columns contain the coefficients of the linear terms of the equations of a new system, which has a (0, 1)-solution if and only if the original system of equations has a (0, 1)-solution. In this case, the numbers of rows and columns never increase. The following steps are performed in a loop until matrix M stabilizes or an additional termination condition is satisfied, under which the existence or absence of a (0, 1)-solution is easily verified.

1. Remove zero columns from matrix A.

2. If some row of matrix A contains one nonzero entry, located in the jth column, then the following cases are analyzed.

(a) If the entry of column **b** in this row is zero, then delete the jth column.

(b) If the entries in the jth column and column **b** in this row are not zero and coincide, then replace column **b** by difference **b** – **a**, where **a** is the jth column in matrix A, and then delete the jth column.

(c) If the entries in the jth column and column $\mathbf{b}$ in this row are both nonzero and distinct, then terminate the algorithm when there are no (0, 1)-solutions.

This step is repeated until the matrix is stabilized.

3. Remove zero rows from matrix M.

4. If M is empty or auxiliary column $\mathbf{b}$ in M is zero, then terminate the algorithm when there is a (0, 1)-solution.

5. If matrix A is empty or contains a zero row, then terminate the algorithm when there is no (0, 1)-solution.

6. Search for two linearly dependent columns in matrix A. If these columns are found, then denote the number of one of them by j. Otherwise, calculate column $\mathbf{c} = 2\mathbf{b} - \sum A_i$, where A_i is the ith column in matrix A and all columns are summed up. If column $\mathbf{c}$ is not zero, then search for a column linearly dependent on column $\mathbf{c}$ in matrix A. If this column in A is found, then denote its number by j and replace the other column in matrix A with column $\mathbf{c}$. Otherwise, terminate the algorithm.

7. Search for a nonzero entry in the jth column. Denote the number of its row by k. Multiply the kth row of augmented matrix M by the entry in the kth row and jth column. For each index i in augmented matrix M, subtract the kth row multiplied by the entry in the jth column and ith row from the ith row. As a result, the jth column and kth row become zero. Hereinafter, A and $\mathbf{b}$ are associated with augmented matrix M. Go to the first step.

Function `has01solution`(M) returns either an empty matrix or an augmented matrix of a new system of linear equations that has a (0, 1)-solution if and only if the original system of equations has a (0, 1)-solution. If matrix M is empty, then the original system of equations has a (0, 1)-solution. If auxiliary column $\mathbf{b}$ in matrix M is zero, then the corresponding system of equations is homogeneous; therefore, there is a zero solution. Otherwise, if matrix A is empty or has a zero row, although the corresponding row in augmented matrix M is not zero, then the new system contains false equality $0 = 1$ or $0 = 2$. If the system includes equation $x_j = 2$ or $2x_j = 1$, then there are no (0, 1)-solutions. Otherwise, additional calculations not implemented in this function are required. For instance, we can try to estimate the remaining variables. On the other hand, this result can be regarded as an ambiguous answer of the generic algorithm.

Searching for linearly dependent columns among n nonzero columns requires $O(n\log_2(n+1))$ inter-column comparisons. This search is reduced to sorting $2n$ columns, both the original ones and those multiplied by two. The search can be implemented using the `numpy.unique` method. However, a more practical implementation is based on hashing with the use of the `set` container built in Python, which implies the consideration of only a small number of columns, given a sufficiently large number of them (see [20, 21]). The total execution time is bounded from above by a function of form $\text{poly}(m)n\log_2(n+1)$.

Let us consider an example. The system of equations of two variables x_1 and x_2

$$\begin{cases} 2x_1 + 2x_2 = 1 \\ x_1 + 2x_2 = 1 \end{cases}$$

corresponds to the augmented matrix

$$M = \begin{pmatrix} 2 & 2 & 1 \\ 1 & 2 & 1 \end{pmatrix}.$$

Neither of its first two columns is a linearly dependent one. However, step 6 of the algorithm can be performed. The column

$$\mathbf{c} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

is proportional to the first column of matrix M. By replacing the second column, we obtain the matrix

$$\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \end{pmatrix}.$$

With the first two columns being proportional to one another, the elimination of variables leads to a new augmented matrix of one element 2, which corresponds to false equality $0 = 2$. Therefore, there are no

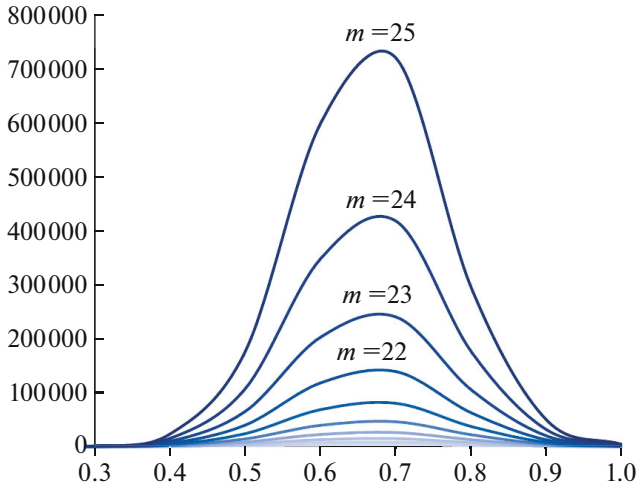

Fig. 1. The X-axis represents probability p that a randomly selected element is not zero. The Y-axis represents the median for the number of randomly generated nonzero columns, among which there are no linearly dependent ones, but the next column is linearly dependent on one of the previous columns. The number of entries in a column is denoted by m.

Table 1. For the number m of rows, the median and 99th percentile are shown for randomly generated nonzero columns among which there are no linearly dependent columns, but the next column is linearly dependent on some of the previous columns. The upper bound for the number of pairwise independent columns is also shown

m	50%	99%	100%
1	1	1	1
2	2	4	4
3	4	9	13
4	7	18	40
5	13	32	121
6	22	57	364
7	39	99	1093
8	67	172	3280
9	117	300	9841
10	202	520	29524
11	351	902	88573
12	605	1561	265720
13	1050	2708	797161
14	1823	4687	2391484
15	3163	8123	7174453
16	5450	14127	21523360
17	9467	24447	64570081
18	16423	42124	193710244
19	28435	73695	581130733
20	49176	126316	1743392200

Table 2. For the number m of rows and the probability p that a randomly selected entry is not zero, the median is shown for the number of randomly generated nonzero columns among which there are no linearly dependent ones, but the next column is linearly dependent on one of the previous columns

	p				
m	0.1	0.2	0.3	0.4	1.0
10	6	12	25	55	26
11	7	15	34	82	37
12	8	18	45	122	53
13	9	21	60	183	75
14	10	26	81	271	107
15	11	31	109	409	150
16	12	38	147	608	213
17	13	46	198	911	303
18	14	55	267	1371	426
19	16	67	360	2059	603
20	17	80	485	3094	848
21	19	98	658	4659	1204
22	21	118	895	7010	1699
23	23	144	1210	10611	2409
24	25	174	1642	15886	3420
25	27	212	2228	23958	4816
26	30	257	3027	36115	6831
27	33	313	4131	54066	9640
29	36	379	5598	81713	13658
29	40	463	7616	123263	19301
30	44	563	10313	186122	27250

(0, 1)-solutions. During the execution of the program, the elimination of variables first leads to the matrix

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix},$$

from which zero columns and zero rows are then removed. In this example, the algorithm terminates in two passes of the main loop, which is consistent with to the theoretical estimate of computational complexity.

To estimate the efficiency of the algorithm, matrices with entries independently and uniformly distributed over the field $GF(3)$ were generated. For the number m of rows, Table 1 shows the median and 99th percentile for n first randomly generated nonzero columns, among which there were no linearly dependent ones, but the next column turned out to be linearly dependent on some of the previously generated columns. For each number m, 100000 series of columns were used. The error in the median is approximately 1%. The last column of the table shows the upper bound for the number of pairwise independent columns: $(3^m - 1)/2$. The median for the largest number of columns in a series is close to the value of the function $(4/5) \cdot (26/15)^m$.

Table 1 shows that even the 99th percentile is significantly below the upper bound for the number of pairwise independent columns. That is why the algorithm remains efficient on average for systems in which the number of equations significantly exceeds the bound from Theorem 2. However, even Theorem 4 does not allow us to improve the bound from Theorem 2 in the worst case.

The difference in computational complexity in the worst and average cases is even more noticeable for sparse matrices. In the next experiment, series of random columns were generated in which all entries were not zero with fixed probability p and the choice between two nonzero values was equiprobable. For columns of $m \le 25$ entries, Fig. 1 shows the experimental p-dependence for the number n of the first randomly generated nonzero columns, among which there are no linearly dependent ones, but the next col-

Table 3. For m rows and n columns, the median is shown in seconds of program runtime under the condition of a certain answer

	n			
m	10^5	10^6	10^7	10^8
2	0.01	0.1	1	14
3	0.02	0.2	2	23
4	0.03	0.3	3	35
5	0.05	0.5	5	48
6	0.06	0.6	6	62
7	0.08	0.8	8	79
8	0.09	0.9	10	97
9	0.11	1.1	12	117
10	0.13	1.3	14	139
11	0.15	1.6	16	162
12	0.17	1.8	18	187
13	0.2	2	21	214
14	0.23	2.2	24	243
15	0.28	2.5	26	273
16	0.33	2.9	28	304
17	0.4	3.2	32	334
18	0.49	3.6	35	369
19	0.67	4.1	39	405
20	0.9	4.7	43	445
21	1.18	5.5	47	479
22	1.46	6.5	52	518
23	1.8	8.1	58	563
24	2.06	10.4	63	609

umn turns out to be linearly dependent on one of the previously generated columns.

In Table 2, the experimental p-dependence for this median is shown for large values of m but small values of p and $p = 1$. For $p = 0.1$ and $m \le 30$, the proposed method is efficient on average for systems of equations with a small number of variables. Thus, our method can be used to solve applied problems in mathematical biology.

To empirically estimate the runtime of the program at different numbers of rows m and columns n in random matrix A of coefficients of linear terms of equations, the runtime medians were calculated under the condition of obtaining a certain answer, whereby the existence or absence of a (0, 1)-solution is determined. The results are presented in Table 3.

The program listing and examples are available at http://lab6.iitp.ru/-/havoc.

4. CONCLUSIONS

The polynomial-time algorithm that makes it possible to check the existence of a (0, 1)-solution to a system with a sufficiently small number of equations over field $GF(3)$ has been substantiated and implemented. The reported results are consistent with the generally accepted hypothesis about the high computational complexity of the problems of recognizing (0, 1)-solutions to systems of linear equations, because the modification of the problem by elimination of variables encounters an obstacle in the worst case. However, for a system with a small number of equations over the field $GF(3)$, the computational complexity turns out to be low in a typical case. It has been experimentally shown that the algorithm is much more efficient for sparse systems of equations. Moreover, the binary search method allows one to find a (0, 1)-solution to the system, if any, even though the enumeration of all (0, 1)-solutions may be very difficult. This opens up the possibility of practical use in solving the applied problems that can easily be reduced to finding a (0, 1)-solution to a system of linear algebraic equations. For instance, the paper [22] considered some examples of reducibility of combinatorial problems to the problem of solvability of a system of linear equations over an additive semigroup of natural numbers.

Computer algebra systems support calculations over the residue field modulo a prime. This allows the new program to be integrated into a data processing pipeline, in particular, for solving mathematical biology problems.

ACKNOWLEDGMENTS

This work was carried out using the computing resources of the Joint Supercomputer Center of the Russian Academy of Sciences (JSCC RAS).

FUNDING

This work was supported by the Russian Science Foundation, grant no. 24-44-00099, https://rscf.ru/project/24-44-00099.

CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

REFERENCES

1. Seliverstov, A.V., Generalization of the subset sum problem and cubic forms, *Comput. Math. Math. Phys.*, 2023, vol. 63, no. 1, pp. 48–56. https://doi.org/10.1134/s0965542523010116
2. Boykov, A.A. and Seliverstov, A.V., On a cube and subspace projections, *Vestn. Udmurt. Univ., Mat. Mekh. Komp'yuternye Nauki*, 2023, vol. 33, no. 3, pp. 402–415. https://doi.org/10.35634/vm230302

3. Kárteszi, F., *Introduction to Finite Geometries*, Budapest: Akadémian Kiadó, 1976.
4. Feng, T. and Lu, J., New families of flag-transitive linear spaces, *Finite Fields Their Appl.*, 2023, vol. 87, p. 102156. https://doi.org/10.1016/j.ffa.2022.102156
5. Stoichev, S.D. and Gezek, M., Unitals in projective planes of order 25, *Mathematics in Computer Science*, 2023, vol. 17, no. 1, p. 5. https://doi.org/10.1007/s11786-023-00556-9
6. Kogabaev, N.T., Systems of Diophantine equations over finite configurations, *Sib. Math. J.*, 2023, vol. 64, no. 2, pp. 325–337. https://doi.org/10.1134/s0037446623020076
7. Bayramov, R.E., Blinkov, Yu.A., Levichev, I.V., Malykh, M.D., and Melezhik, V.S., Analytical study of cubature formulas on a sphere in computer algebra systems, *Comput. Math. Math. Phys.*, 2023, vol. 63, no. 1, pp. 77–85. https://doi.org/10.1134/s0965542523010050
8. Hesse, O., Über die Elimination der Variabeln aus drei algebraischen Gleichungen vom zweiten Grade mit zwei Variabeln, *J. Reine Angew. Math.*, 1844, vol. 1844, no. 28, pp. 68–96. https://doi.org/10.1515/crll.1844.28.68
9. Cacchiani, V., Iori, M., Locatelli, A., and Martello, S., Knapsack problems—An overview of recent advances. Part I: Single knapsack problems, *Comput. Oper. Res.*, 2022, vol. 143, p. 105692. https://doi.org/10.1016/j.cor.2021.105692
10. Cacchiani, V., Iori, M., Locatelli, A., and Martello, S., Knapsack problems—An overview of recent advances. Part II: Multiple, multidimensional, and quadratic knapsack problems, *Comput. Oper. Res.*, 2022, vol. 143, p. 105693. https://doi.org/10.1016/j.cor.2021.105693
11. Zhang, L., Quweider, M., Khan, F., and Lei, H., Splitting NP-complete sets infinitely, *Inf. Process. Lett.*, 2024, vol. 186, p. 106472. https://doi.org/10.1016/j.ipl.2024.106472
12. Vyalyi, M.N., Testing the satisfiability of algebraic formulas over the field of two elements, *Probl. Inf. Transm. (Engl. Transl.)*, 2023, vol. 59, no. 1, pp. 57–62. https://doi.org/10.1134/s0032946023010052
13. Yashunskii, A.D., On sums of Bernoulli random variables modulo 3, *Math. Notes*, 2022, vol. 111, nos. 1–2, pp. 166–169. https://doi.org/10.1134/s0001434622010205
14. Sanna, C., On the distribution of the entries of a fixed-rank random matrix over a finite field, *Finite Fields Their Appl.*, 2024, vol. 93, p. 102333. https://doi.org/10.1016/j.ffa.2023.102333
15. Balakin, G.V., The distribution of the rank of random matrices over a finite field, *Theory Probab. Its Appl.*, 1968, vol. 13, no. 4, pp. 594–605. https://doi.org/10.1137/1113076
16. Kruglov, V.I. and Mikhailov, V.G., On the rank of random matrix over prime field consisting of independent rows with given numbers of nonzero elements, *Matematicheskie Voprosy Kriptografii*, 2020, vol. 11, no. 3, pp. 41–52. https://doi.org/10.4213/mvk331
17. Cooper, C., On the distribution of rank of a random matrix over a finite field, *Random Structures and Algorithms*, 2000, vol. 17, nos. 3–4, pp. 197–212. https://doi.org/10.1002/1098-2418(200010/12)17:3/4<197::aid-rsa2>3.0.co;2-k
18. Rybalov, A.N., Generic polynomial algorithms for the knapsack problem in some matrix semigroups, *Sib. Elektron. Mat. Izv.*, 2023, vol. 20, no. 1, pp. 100–109. https://doi.org/10.33048/semi.2023.20.009
19. Rybalov, A.N., Generically undecidable and hard problems, *Prikl. Diskretnaya Mat.*, 2024, no. 63, pp. 109–116. https://doi.org/10.17223/20710410/63/7
20. Nayak, S. and Patgiri, R., A review on role of bloom filter on DNA assembly, *IEEE Access*, 2019, vol. 7, pp. 66939–66954. https://doi.org/10.1109/access.2019.2910180
21. Bille, P., Gørtz, I.L., and Stordalen, T., Predecessor on the ultra-wide word RAM, *Algorithmica*, 2024, vol. 86, no. 5, pp. 1578–1599. https://doi.org/10.1007/s00453-023-01193-1
22. Rybalov, A.N., On the generic complexity of solving equations over natural numbers with addition, *Prikl. Diskretnaya Mat.*, 2024, no. 64, pp. 72–78. https://doi.org/10.17223/20710410/64/6

Translated by Yu. Kornienko