UDC 003.26:512.5

Complexity of The Systems of Linear Restrictions over a Finite Field

Oleh Kurinnyi[1]

[1]*National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Institute of Physics and Technology*

Abstract

This paper continues the results obtained in [1]. In the previous paper, we formulated the problem of the unknown vector recovering from linear dependencies with this vector, which act as constraints on it. The next step, after finding out some algebraic and combinatorial properties, is to give basic estimates of complexity for the main problem as well as for related problems. Such related problems can be obtained by fixing some parameters of the main problem or applying constraints on the number of restrictions in the system. Such an analysis makes possible to arrange the problem of recovering an unknown vector based on partial information into the general computational complexity framework in order to approach existing theoretical results to its solution. The obtained theoretical results can be used in algebraic cryptanalysis of stream ciphers and cryptosystems based on linear codes.

Keywords: system of linear restrictions, finite field, computational complexity, SLR problem.

Introduction

The problem of recovering an unknown vector based on partial information is to recover that unknown vector, given some constraints on it, which are presented in the form of linear dependencies. In the context of computational complexity theory we often use a decision version of the initial problem, in which it is not necessary to restore the vector directly, but to provide an answer to whether it exists. That's why we will start with the formalization of such a problem and its varieties, which can be obtained by fixing certain input parameters, and we will also provide an estimation of the complexity of these problems by establishing their belonging to the complexity classes. In addition, we will formulate some partial cases of the decision problem and also give estimates of their complexity, construct polynomial probabilistic algorithms for one of these partial cases, as well as a probabilistic heuristic algorithm for finding several solutions of the system of linear restrictions.

1. Main notations

Recall the notation of the system of linear restrictions. The system of linear restrictions over a field $\mathbb{F}_{2^k}$ is a system of expressions of the form

$$\begin{cases} a_1^{(1)}x_1 + a_2^{(1)}x_2 + \ldots + a_n^{(1)}x_n \neq a_0^{(1)}, \\ a_1^{(2)}x_1 + a_2^{(2)}x_2 + \ldots + a_n^{(2)}x_n \neq a_0^{(2)}, \\ \qquad\qquad \ldots \\ a_1^{(m)}x_1 + a_2^{(m)}x_2 + \ldots + a_n^{(m)}x_n \neq a_0^{(m)}, \end{cases}$$

where $a_i^{(j)} \in \mathbb{F}_{2^k}$ for $i = \overline{0,n}$, $j = \overline{1,m}$, $x_t \in \mathbb{F}_{2^k}$ for $t = \overline{1,n}$, and $m > 1$. The short notation, that we will use, is $A \cdot x \neq a_0$, where A is a $m \times n$-matrix of coefficients in left-hand sides, a_0 is a $m \times 1$-vector of coefficients in right-hand sides and symbol «$\neq$» is used in untypical context and stands for «not equal in all components». The solution of the system of linear restrictions is a vector $x_0 \in \mathbb{F}_{2^k}^n$ such that $A \cdot x \neq a_0$. The solution set of the system of linear restrictions is a set of all solutions of the system.

We also recall the property of the system of linear restrictions [1] for the case of zero right-hand sides.

Claim 1. *Let $D \subseteq \mathbb{F}_{2^k}^n$ – the solution set of the system of linear restrictions $A \cdot x \neq \overline{0}$ over a finite field $\mathbb{F}_{2^k}$, then $|D|$ is divisible by $2^k - 1$.*

This claim makes possible to get several solutions from one known solution by multiplying it

by all non-zero elements of the field. Also, we will say that polynomial is identically equal to zero over field $\mathbb{F}_{2^k}$ if and only if it turns zero on all elements of field $\mathbb{F}_{2^k}$.

2. The main problems related to systems of linear restrictions

Let's formulate the SLR (short for System of Linear Restrictions) problem of the existence of a solution of a system of linear restrictions over a finite field.

Problem 1. (SLR) *Input.* The size of the field 2^k, the matrix A of size $m \times n$ over the field $\mathbb{F}_{2^k}$, the vector a_0 of size $m \times 1$.

It is necessary to find out whether there is a solution over the field $\mathbb{F}_{2^k}$ for the system of linear restrictions $A \cdot x \neq a_0$.

Output. «Yes» if the solution exists, «No» otherwise.

Now we'll formulate derived problems, in which either the size of the field or the number of nonzero elements in each restriction is fixed, as well other problem, in which these parameters are fixed simultaneously.

Problem 2. (SLR-$\mathbb{F}_{2^k}$) *Input.* The matrix A of size $m \times n$ over the field $\mathbb{F}_{2^k}$, the vector a_0 of size $m \times 1$.

It is necessary to find out whether there is a solution over the field $\mathbb{F}_{2^k}$ for the system of linear restrictions $A \cdot x \neq a_0$.

Output. «Yes» if the solution exists, «No» otherwise.

Problem 3. (qSLR) *Input.* The size of the field 2^k, the matrix A of size $m \times n$ over the field $\mathbb{F}_{2^k}$ in which each row has no more that q non-zero elements, the vector a_0 of size $m \times 1$.

It is necessary to find out whether there is a solution over the field $\mathbb{F}_{2^k}$ for the system of linear restrictions $A \cdot x \neq a_0$.

Output. «Yes» if the solution exists, «No» otherwise.

The parameter q will be called the *arity* of the linear restriction, and the linear restriction will be *q-ary*.

Problem 4. (qSLR-$\mathbb{F}_{2^k}$) *Input.* The matrix A of size $m \times n$ over the field $\mathbb{F}_{2^k}$ in which each row has no more that q non-zero elements, the vector a_0 of size $m \times 1$.

It is necessary to find out whether there is a solution over the field $\mathbb{F}_{2^k}$ for the system of linear restrictions $A \cdot x \neq a_0$.

Output. «Yes» if the solution exists, «No» otherwise.

We present a set of statements which helps to estimate complexity of formulated problems.

Claim 2. *Problem 2SLR-$\mathbb{F}_4$ is $\mathcal{NP}$-complete.*

Proof. We make sure that this problem belongs to the $\mathcal{NP}$ class. The certificate for this problem is the vector $(x_1, x_2, \ldots, x_n)$. The length of this vector is n, which is bounded by a polynomial of the size of the input data. Verification of the certificate consists, in fact, substituting $(x_1, x_2, \ldots, x_n)$ into the system of linear restrictions and checking whether this vector satisfies all restrictions.

The $\mathcal{NP}$-hardness of this problem was proved in the work of Selezneva in 2017 [2] for representation in the multilinear form. The proof is given for the case of an arbitrary finite field and is based on the use of the qCOLOR problem, which consists in finding a coloring of an arbitrary undirected graph in q colors such that no two adjacent vertices are colored in the same color. In the case of the 2SLR-$\mathbb{F}_4$ problem we should use 4COLOR problem for reduction. The idea of the proof is as follows. We establish a one-to-one correspondence between the set of four colors and the elements of the $\mathbb{F}_4$ field. For the input graph $G = (V, E)$, we build a system with $|E|$ linear restriction and $|V|$ variables, in which each restriction corresponds to a pair of adjacent vertices $(u, v) \in E$ and has the form $x_u + x_v \neq 0$, where $x_u, x_v \in \mathbb{F}_4$ are variables that encode the color of the corresponding vertices u and v. If the graph is colored in 4 colors, then each pair of adjacent vertices (u, v) has a different color, so the corresponding linear restriction $x_u + x_v \neq 0$ is automatically fulfilled in the system. In the case when there is no coloring for the graph, the considerations are similar. ■

It follows from the proved statement that the general SLR problem is automatically $\mathcal{NP}$-complete.

Claim 3. *The qSLR problem is $\mathcal{NP}$-complete for any $q \geq 2$.*

Proof. The $\mathcal{NP}$-completeness of the 2SLR problem follows from claim 2, since 2SLR-$\mathbb{F}_4$ is a partial case of 2SLR. In order to prove the $\mathcal{NP}$-completeness of problems qSLR for $q \geq 3$, it is necessary to check the fulfillment of the condition 2SLR $\leq_p$ qSLR for all $q \geq 3$. This reduction holds for any $q \geq 3$, since any restriction that has arity 2, i.e. contains no more than 2 variables, also contains at most k variables, i.e. has k arity. ■

Claim 4. *Problem SLR-$\mathbb{F}_{2^k}$ is $\mathcal{NP}$-complete for any of the finite field $\mathbb{F}_{2^k}$ except $\mathbb{F}_2$.*

Proof. The $\mathcal{NP}$-completeness of the problem SLR-$\mathbb{F}_4$ follows from the claim 2, since the problem 2SLR-$\mathbb{F}_4$ is a partial case of SLR-$\mathbb{F}_4$. In order to prove $\mathcal{NP}$-completeness of problems SLR-$\mathbb{F}_{2^k}$ for all $k \geq 3$, we should check conditions SLR-$\mathbb{F}_4$ $\leq_p$ SLR-$\mathbb{F}_{2^k}$ for all fields $k \geq 3$. Let's fix some field $\mathbb{F}_{2^k}$, where $k \geq 3$. Let's construct the mapping $\varphi : \mathbb{F}_4 \to \mathbb{F}_{2^k}$, considering the fields $\mathbb{F}_4$ and $\mathbb{F}_{2^k}$ purely as sets of certain elements. This mapping must be injective (it cannot be subjective, since sizes of these sets are different); this can be achieved by writing the elements of these fields in a certain basis, sorting them in lexicographic order and matching them between $\mathbb{F}_4$ and the first four elements of $\mathbb{F}_{2^k}$. Let (A, a_0) is an instance problem SLR-$\mathbb{F}_4$, then for reduction it is necessary to replace all constants in the system (A, a_0) according to the mapping φ and add to the new system (A', a_0) all possible restrictions of the form $x_i \neq g$ for $i = \overline{1, n}$ and $g \in \mathbb{F}_{2^k} \setminus Im(\varphi)$. Number of new restrictions is $n \cdot (2^k - 4)$, i.e. is polynomial of the input length, because the field is a parameter of the problem and is not provided as an input. ■

It follows from the proved statement that the problem qSLR-$\mathbb{F}_{2^k}$ is $\mathcal{NP}$-complete for all $k \geq 2$ and for all of finite fields $\mathbb{F}_{2^k}$ except $\mathbb{F}_2$.

Claim 5. *Having the system of linear restrictions with n variables and m by restrictions over the field $\mathbb{F}_{2^k}$, in which each restriction contains no more than q variables, where $4 \leq q \leq n$, we can construct an equivalent system over $\mathbb{F}_{2^k}$, in which each the restriction contains $(q-1)$ variables, adding a polynomial from 2^k and m quantity of new restrictions and variables. In other words, qSLR-$\mathbb{F}_{2^k}$ $\leq_p$ $(q-1)$SLR-$\mathbb{F}_{2^k}$.*

Proof. Consider the linear restriction

$$a_1^{(1)}x_1 + a_2^{(1)}x_2 + \ldots + a_n^{(1)}x_n \neq a_0,$$

where q coefficients are not equal to zero. Let's choose a pair of indices i and j, where $i < j$, which corresponds to a pair of monomials $a_i^{(1)}x_i$ and $a_j^{(1)}x_j$ is restriction. We will make a replacement $a_{ij}^{(1)} = a_i^{(1)}x_i + a_j^{(1)}x_j$. In fact, by substituting a new variable in the restriction and by adding a new equation to the system, you can get an equivalent system of restrictions, but the system of restrictions cannot contain expressions that are equations. So the equation $a_{ij}^{(1)} + a_i^{(1)}x_i + a_j^{(1)}x_j = 0$ is needed to replace with a set of $2^k - 1$ restrictions of the form $a_{ij}^{(1)} + a_i^{(1)}x_i + a_j^{(1)}x_j \neq g$, where $g \in \mathbb{F}_{2^k}^*$. Then we should replace $a_i^{(1)}x_i + a_j^{(1)}x_j$ on $a_{ij}^{(1)}$ in the system. Thus, the number of terms in the restriction was reduced on 1 by adding one additional variable and $2^k - 1$ restrictions with three terms. We perform such a transformation with each restriction and as a result the system got $m \cdot (2^k - 1)$ new restrictions and m new variables. ■

Since in practice it is often necessary to find the solution itself, then weformulate the SLRS (short for SLR Search) problem of finding a solution of the system of linear restrictions.

Problem 5. (SLRS) *Input.* The size of the field 2^k, the matrix A of size $m \times n$ over the field $\mathbb{F}_{2^k}$, the vector a_0 of size $m \times 1$.

It is necessary to find at least one solution over the field $\mathbb{F}_{2^k}$ of the system of linear restrictions $A \cdot x \neq a_0$.

Output. The solution of the system of linear restrictions $A \cdot x \neq a_0$ or «$\perp$» if it does not exist.

Consider a claim that, in a certain sense, reduce the search problem solution to the problem of checking the existence of a solution.

Claim 6. *Problems of SLR-$\mathbb{F}_{2^k}$ and SLRS-$\mathbb{F}_{2^k}$ are Turing equivalent.*

Proof. In order to prove the Turing equivalence of two problems, it is necessary and sufficient to show that each of these problems is reducible by Turing to another.

SLR reduces to SLRS because having the answer from the SLR oracle, the algorithm returns «Yes» if the answer is an n-dimensional vector

and the answer is «No» if this answer consists of the symbol «$\perp$».

In order to prove the Turing reducibility of SLRS to SLR, it is necessary to use calls to the SLR oracle and restore a vector $(x_1, x_2, \ldots, x_n)$ component-wise. To restore the first component x_1 of the vector x, we transfer term $a_1^{(j)}x_1$ to the right-hand side for all $j = \overline{1,m}$ (according to properties of the system of linear restrictions from [1] this is not changes the solution set of the system) and gradually fix the value of x_1 with elements of the field $\mathbb{F}_{2^k}$. If the answer on some element d is «Yes», then we fix the component x_1 with this value d and modify the initial system of linear restrictions by replacing the right-hand sides $a_0^{(j)}$ on $a_0^{(j)} + a_1^{(j)}d$ for $j = \overline{1,m}$. We proceed to the search of the next component with modified matrix A' of dimension $m \times (n-1)$ and vector a'.

From a computational point of view, to restore one component we require at most $|\mathbb{F}_{2^k}| = 2^k$ oracle access operations. For all components of the vector we need $n \cdot 2^k$ operations to the oracle, which is polynomial according to the length of the input.

Correctness of the algorithm. The process of finding the vector x can be represented by in the form of a tree, where each vertex has $|\mathbb{F}_{2^k}|$ descendants, and the number of leaves of the tree is 2^{kn}. The search problem consists in finding a path to the tree leaf that corresponds to the value of the vector x, which is the solution of the system of linear restrictions, and the oracle, in fact, helps to prevent the search of all possible paths, i. e. search the path level by level, discarding the exponential number of wrong options on each level. Because on every steo we choose exactly such value of the component of the vector that the entire vector, including the unknowns component, is a solution, then we are guaranteed to find a solution. So, the solution cannot be restored if and only if on the first component we will receive all «No» answers from the oracle, which will indicate the absence solutions in the system. ■

3. The complexity of the SLR partial cases

Let's consider some partial cases of the SLR problem.

Claim 7. *The problem SLR-$\mathbb{F}_2$ belongs to the complexity class $\mathcal{P}$.*

Proof. If the field in the problem description is fixed and equal to $\mathbb{F}_2$, then for the system of linear restrictions

$$\begin{cases} a_1^{(1)}x_1 + a_2^{(1)}x_2 + \ldots + a_n^{(1)}x_n \neq y_1, \\ a_1^{(2)}x_1 + a_2^{(2)}x_2 + \ldots + a_n^{(2)}x_n \neq y_2, \\ \qquad\qquad \ldots \\ a_1^{(m)}x_1 + a_2^{(m)}x_2 + \ldots + a_n^{(m)}x_n \neq y_m, \end{cases}$$

where $a_i^{(j)}$, $y_j \in \mathbb{F}_2$ for $i = \overline{1,n}$, $j = \overline{1,m}$, it's possible to obtain an equivalent system of equations by substituting $\widetilde{y}_j = y_j + 1$ for $j = \overline{1,m}$:

$$\begin{cases} a_1^{(1)}x_1 + a_2^{(1)}x_2 + \ldots + a_n^{(1)}x_n = \widetilde{y}_1, \\ a_1^{(2)}x_1 + a_2^{(2)}x_2 + \ldots + a_n^{(2)}x_n = \widetilde{y}_2, \\ \qquad\qquad \ldots \\ a_1^{(m)}x_1 + a_2^{(m)}x_2 + \ldots + a_n^{(m)}x_n = \widetilde{y}_m. \end{cases}$$

There are polynomial algorithms for solving such systems, for example, the Gaussian elimination. It also can be used to calculate the rank of a matrix which defines a system of linear equations, and having the rank of the matrix, we can give the answer to the question of whether there are solutions for the input system of linear equations. Complexity of the Gaussian algorithm is cubic with respect to the length of the input data, so it bounded by a polynomial of the length of the input data. ■

Consider an approximate version of the SLR problem, in which it is not necessary to find a solution for all linear restrictions in the system, but can be found a partial solution that satisfies at least $1 \leq l \leq m$ linear restrictions.

Problem 6. (Max-SLR) *Input.* The size of the field 2^k, the matrix A of size $m \times n$ over the field $\mathbb{F}_{2^k}$, the vector a_0 of size $m \times 1$, number l, $1 \leq l \leq m$.

It is necessary to find out whether there is a solution over the field $\mathbb{F}_{2^k}$ for the system of linear restrictions that formed from at least l linear restrictions of the input system of linear restrictions $A \cdot x \neq a_0$.

Output. «Yes» if the solution exists, «No» otherwise.

Claim 8. *The Max-SLR problem is $\mathcal{NP}$-complete.*

Proof. We make sure that this problem belongs to the $\mathcal{NP}$ complexity class. The certificate for this problem, as in the case of SLR, is the vector $(x_1, x_2, \ldots, x_n)$. The length of this vector is n, which is bounded by a polynomial in size input data. To check the certificate we should substitute $(x_1, x_2, \ldots, x_n)$ in the system of linear restrictions and count the number of linear restrictions for which $(x_1, x_2, \ldots, x_n)$ is a solution.

We will prove that this problem is $\mathcal{NP}$-hard. For this we will use the Max-XORSAT problem [3], but first let's recall it. The input of the problem is a formula presented in CNF, in which all disjunction operations in disjuncts are replaced by the XOR operation. The formula contains n variables $(x_1, x_2, \ldots, x_n)$ and m XOR expressions, which can be named XOR-expressions. An XOR-expression can include both a variable and its own negotiation. Also, natural number l is given as an input, that is upper bound for the number of XOR expressions. It is necessary to determine whether exists a vector that satisfies at least l XOR-expressions in the given formula.

This formula can be written in the form of a system of equations, equating each XOR expression to one, then the question will be whether it is possible find a vector that is a solution for at least l equations of this system. Note, that all negation operations in XOR expressions can be replaced by XOR variable with a 1. Then only variables will remain in the left parts of the equations; the right-hand sides of the equations will be zero in the case when XOR-expression contained an odd number of variables with negation, and one in all others cases.

So, the system will look like this:

$$\begin{cases} x_{i_1^{(1)}} + x_{i_2^{(1)}} + \ldots + x_{i_{k_1}^{(1)}} = y_1, \\ x_{i_1^{(2)}} + x_{i_2^{(2)}} + \ldots + x_{i_{k_2}^{(2)}} = y_2, \\ \ldots \\ x_{i_1^{(m)}} + x_{i_1^{(m)}} + \ldots + x_{i_{k_m}^{(m)}} = y_m, \end{cases}$$

where $1 \leq k_s \leq n$ for $s = \overline{1,m}$ is the number of variables in each XOR-expression, $i_j^{(s)} \in \{1, 2, \ldots, n\}$ for $1 \leq k_s \leq n$, $s = \overline{1,m}$ are indices that define variables in each XOR-expression (note that XOR-expression cannot contain variables with the same indices or a variable and its negation simultaneously), $y_s \in \{0, 1\}$ for $s = \overline{1,m}$ is a set of values that corresponding XOR-expression take.

We see that indices in this formula actually reflect the presence of some variable from the set $(x_1, x_2, \ldots, x_n)$ in some of the m equations. This formula can be simplified if we introduce artificial variables, each of which will be an indicator of whether some variable is present in some equation. Then this system takes the following form:

$$\begin{cases} a_1^{(1)}x_1 + a_2^{(1)}x_2 + \ldots + a_n^{(1)}x_n = y_1, \\ a_1^{(2)}x_1 + a_2^{(2)}x_2 + \ldots + a_n^{(2)}x_n = y_2, \\ \ldots \\ a_1^{(m)}x_1 + a_2^{(m)}x_2 + \ldots + a_n^{(m)}x_n = y_m, \end{cases}$$

where $a_i^{(j)} \in \{0, 1\}$ for $i = \overline{1,n}$, $j = \overline{1,m}$ are these artificial variables and $y_j \in \{0, 1\}$ for $j = \overline{1,m}$. Note that such a transformation affects only the representation of the system equations and does not change the solution set.

Since y_j for $j = \overline{1,m}$ takes only two values, we can denote $\widetilde{y}_j = y_j + 1$ for $j = \overline{1,m}$. Then the system takes such form:

$$\begin{cases} a_1^{(1)}x_1 + a_2^{(1)}x_2 + \ldots + a_n^{(1)}x_n \neq \widetilde{y}_1, \\ a_1^{(2)}x_1 + a_2^{(2)}x_2 + \ldots + a_n^{(2)}x_n \neq \widetilde{y}_2, \\ \ldots \\ a_1^{(m)}x_1 + a_2^{(m)}x_2 + \ldots + a_n^{(m)}x_n \neq \widetilde{y}_m. \end{cases} \quad (1)$$

We obtained a system of linear restrictions over the field $\mathbb{F}_2$. Therefore, the Max-XORSAT problem is actually a sub-case of the Max-SLR problem. The reduction function f which matches each instance of the Max-XORSAT problem with an instance of the Max-SLR problem is the transformation described above of the Boolean formula with XOR-expressions into the system of linear restrictions over $\mathbb{F}_2$. It is clear that the function f saves while mapping a set of instances of the Max-XORSAT problem with the answer «Yes» – it follows from the construction of the function f. The complexity of the function f calculation is bounded by a polynomial from the input data length since it only needs several iterations of matrix A and vector a_0. ■

We formulate a version of the SLR problem by bounding $m \leq 2^{k-1}$ the number of linear restrictions in the system. Note that formulated in a such way problem is a *promise* one [4]. This means that within this problem not all possible

input data must be answered «Yes» or «No» – to part of inputs for which $m > 2^{k-1}$, we can return nothing.

Problem 7. (RSLR) *Input.* The size of the field 2^k, the matrix A of size $m \times n$ over the field $\mathbb{F}_{2^k}$, the vector a_0 of size $m \times 1$, $m \leq 2^{k-1}$.

It is necessary to find out whether there is a solution over the field $\mathbb{F}_{2^k}$ for the system of linear restrictions $A \cdot x \neq a_0$.

Output. «Yes» if the solution exists, «No» otherwise.

Claim 9. *The RSLR problem belongs to the complexity class $\mathcal{RP}$.*

Proof. To prove this fact, it is necessary to show that exists polynomial probability algorithm B such that:

1) if the RSLR problem on the input $(2^k, A, a_0)$ answers «Yes», then the condition
$$\Pr\left[B(2^k, A, a_0) = 1\right] \geq \frac{1}{2}$$
holds;
2) if the RSLR problem on the input $(2^k, A, a_0)$ answers «No», then the condition
$$\Pr\left[B(2^k, A, a_0) = 1\right] = 0$$
holds.

Let's build such algorithm B.

Input. The size of the field 2^k, the matrix A of size $m \times n$ over the field $\mathbb{F}_{2^k}$, the vector a_0 of size $m \times 1$, $m \leq 2^{k-1}$.

1) Randomly generate $(r_1, r_2, \ldots, r_n)$, where $r_i \in \mathbb{F}_{2^k}$, $i = \overline{1, n}$.
2) Calculate $F(r_1, r_2, \ldots, r_n)$, where
$$F(x) = \prod_{i=1}^{m} [(a^{(i)}, x) + a_0^{(i)}].$$
3) If $F(r_1, r_2, \ldots, r_n) \neq 0$ then return 1, otherwise return 0.

In this case, we used the equivalent form of the problem of a system of linear restrictions from [1].

This algorithm is polynomial in the length of the input as it requires $m \cdot n$ multiplication operations in the field. We make sure that the conditions are met.

1) Assume that the RSLR on the input $(2^k, A, a_0)$ answers «No», then $F(x) \equiv 0$. In this case, the answer of the algorithm in all possible cases will be 0, since there is no vector, on which value of polynomial differs from zero, for identically equal to zero polynomial.
2) Assume that the RSLR at the input $(2^k, A, a_0)$ answers «Yes», then $F(x) \not\equiv 0$. In this case, the probability that $F(x)$ will be equal to zero can be estimated using the Schwarz-Ziepel lemma [4] (here we also use $m \leq 2^{k-1}$):
$$\Pr_{r_1, r_2, \ldots, r_n \in \mathbb{F}_{2^k}} [F(r_1, r_2, \ldots, r_n) = 0] \leq \frac{1}{2}.$$
We can calculate the probability of the opposite event:
$$\Pr_{r_1, r_2, \ldots, r_n \in \mathbb{F}_{2^k}} [F(r_1, r_2, \ldots, r_n) \neq 0] \geq \frac{1}{2}.$$
Since the answer is 1 only if the polynomial F is not equal to zero, then the probability that algorithm B will give an answer of 1, is greater than 0.5.

Table 1
Error matrix of B on input $x = (2^k, A, a_0)$

	$B(x) = 1$	$B(x) = 0$
RSLR returns «Yes»	$\geq \frac{1}{2}$	$\leq \frac{1}{2}$
RSLR returns «No»	0	1

Calculation of algorithm errors completes the proof (see table 1). ■

Moreover, the test given in claim 9 can be used repeatedly for a fixed number of times d which will reduce the error of algorithm exponentially. Such an iterative version of the algorithm B will be called SE (short for Solution Exists). Errors of the algorithm SE are given in table 2.

Table 2
Error matrix of SE on input $(2^k, A, a_0)$

	$B(x) = 1$	$B(x) = 0$
RSLR returns «Yes»	$\geq 1 - \left(\frac{1}{2}\right)^d$	$\leq \left(\frac{1}{2}\right)^d$
RSLR returns «No»	0	1

Let's analyze the complexity of this algorithm. Calculation the value of the polynomial on some input requires: $m \times n$ multiplication operations

in the field for calculation of all intermediate values of the left-hand sides of the system, as well as m operations of multiplication to find the total product. We get $m(n+1)$ multiplication operations in the field. This procedure must be repeated d times, so in general, $dm(n+1) = O(dmn)$ operations are needed, thereofore the given the algorithm is polynomial probabilistic.

Note that in partial cases we can find out whether solutions exist only according to the form of this system. For example, if the system contains such x_i, $1 \le i \le n$, that all coefficients of $a_i^{(j)} \neq 0$ for $j = \overline{1,m}$, then solutions to the system are guaranteed to exist. Vectors, in which on the i-th position a non-zero element and all other components are equal to zero, will be such solutions. For the case of multilinear forms the corresponding polynomial will contain this x_i in every factor. In general case such x_i for $1 \le i \le n$ does not always exist.

We can present the results of the problems complexity on figure 1 (for convenience, the figure assumes that $\mathcal{P} \neq \mathcal{NP}$). On this figure notation $\mathcal{NPC}$ stands for $\mathcal{NP}$-complete.

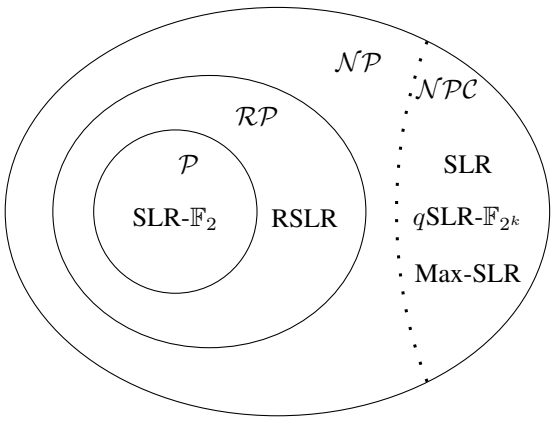

Figure 1: Problems complexity

4. Algorithms for finding solutions of the system of linear restrictions

To obtain a polynomial probability search algorithm in the case of $m \le 2^{k-1}$ the SE algorithm can be combined with the results of claim 5. Field size 2^k is a fixed parameter.

Algorithm 1. *Input.* The matrix A of size $m \times n$ over the field $\mathbb{F}_{2^k}$, the vector a_0 of size $m \times 1$, $m \le 2^{k-1}$.

Sort the elements of the field $\mathbb{F}_{2^k}$ in the polynomial basis as binary strings in ascending order. These bit strings will form an array B, where $|B| = 2^k$.

Fix $d \in \mathbb{N}$, then the error of SE one run will be 2^{-d}.

Loop for i from 1 to n:

1) Loop for j from 1 to 2^k:
   a) Form the matrix A' from the matrix A by removing in the left-hand sides of all restrictions terms with x_i. Form the vector a_0' from the vector a_0 by assigning $a_0'^{(j)}$ the value of $a_0^{(j)} + a_0^{(j)} + a_i^{(j)} B_j$ for $j = \overline{1,m}$.
   b) Calculate $d = SE(2^k, A, a_0)$.
   c) If $d = 1$, then put $x_i = B_j$, $A = A'$, $a_0 = a_0'$ and break the loop on j.
2) If $d = 0$, then return «$\perp$».

Return vector $(x_1, x_2, \ldots, x_n)$.

Output. Vector $(x_1, x_2, \ldots, x_n)$ or «$\perp$».

The given algorithm for finding solution can be simplified if the SE algorithm, in addition to the value 1, will also return a vector on which holds $F(x) \neq 0$. Then to find solution we will only need n checks for the first component. But in this case, algorithm 1 will lose its universality. Therefore, this algorithm is constructed so that instead of the SE algorithm it was possible to use an arbitrary polynomial algorithm for checking the solution existence with a one-sided error 2^{-d}.

So, the total complexity of the algorithm in the worst case is $O(n^2 m 2^k d)$ and remains polynomial of the input length. Because the value of d, which fixes the error rate, is set in advance, then it is possible to choose it much larger than the number of iterations of the probabilistic algorithm, to prevent error accumulating throughout the algorithm. In that case the error will not accumulate because the number of calls to SE is polynomial, and the function is 2^{-d} decreases exponentially with increasing d.

Note that such an algorithm can be used in the case of $m > 2^{k-1}$, but then it becomes empirical, since there's no theoretical reasoning for the correctness of its work. The possibility of using it caused by the fact that the algorithm SE of checking the existence of solutions has one-sided error, that is, it may accidentally reject some valid ones solutions, but never returns vectors that are not solutions. So there may be

a situation when a solution for the system of linear restrictions was existed and the solution search algorithm returned «$\perp$», but it can't be a situation when this algorithm returned a vector that is not a solution.

Consider an empirical algorithm that allows us to search several solutions of a system of linear restrictions. By $FOS(A, a_0)$ (short for Find One Solution) we will denote the polynomial probability algorithm 1, which finds one solution of the system of linear restrictions with high probability. We can apply the algorithm for finding one solution in the following way: after finding some solution, impose a restriction on it, and continue iteratively running the probabilistic algorithm to find the next solutions, adding the corresponding restrictions to the system for each of them. In this case, the running time of the algorithm is no longer limited by a polynomial, since the number of solutions of the system of linear restrictions can potentially be exponential, that is, not limited by a polynomial in the length of the input data. For the case of zero right-hand sides, statement 1 provides an insights about the structure of solutions, and more precisely, it states that the number of solutions of the system is $s \times (2^k - 1)$, where s is the number of equivalence classes according to the «proportionality» relation on the set of solutions of the system. Thus, for the case of zero right parts, after finding one solution, it is necessary to add its equivalence class to the set of solutions, that is, the set of vectors proportional to it.

To implement such an algorithm, it is necessary to be able to search a restriction for the found solution. Such a restriction can be a vector orthogonal to the found solution. The task of finding an orthogonal vector is not difficult, you can use the deterministic algorithm 2, which we will call OS (short for Orth Search).

Algorithm 2. *Input.* Vector $x \in \mathbb{F}_{2^k}^n$.

1) If $x = \overline{0}$, then return the vector $u = (1, 0, \ldots, 0)$.
2) If $x \neq \overline{0}$, then there is at least one non-zero component. We will denote the position of this component i, $1 \leq i \leq n$.
3) Put $I = \{1, 2, \ldots, n\}$ – the set of indices of the vector x. Calculate
$$s = \sum_{j \in I \setminus \{i\}} x_j.$$
4) Return a vector u of the following form:
$$u = (1, 1, \ldots, 1, x_i^{-1} \cdot s, 1, \ldots, 1),$$
where $x_i^{-1} \cdot s$ is contained on the position with index i.

Output. Vector $u \neq \overline{0}$ such that $(u, x) = 0$.

We verify that $u \neq \overline{0}$ is indeed orthogonal to x:

$$(u, x) = x_1 + \ldots + x_i^{-1} s \cdot x_i + \ldots + x_n = s + s = 0.$$

Note that the proposed method of constructing an orthogonal vector not the only one.

Having the procedure for finding an orthogonal vector, we can formulate algorithm 3 for finding several solutions of the system of linear restrictions.

Algorithm 3. *Input.* The matrix A of size $m \times n$ over the field $\mathbb{F}_{2^k}$, the vector a_0 of size $m \times 1$, $m \leq 2^{k-1}$.

Initialize $D = \varnothing$.

Repeat:

1) $d = FOS(A, a_0)$.
2) If d is not equal to «$\perp$», then:
   a) If $a_0 = \overline{0}$, then $D = \{c \cdot d, c \in \mathbb{F}_{2^k}^*\}$, otherwise $D_0 = \{d\}$.
   b) Update $D = D \cup D_0$.
   c) Find the vector $u = OS(d)$.
   d) Update matrix A' by adding the vector u to it. Update vector a_0 by adding the element 0 to it.

Until d is not equal to «$\perp$».

Output. A subset D of the solution set of the input system of linear restrictions $A \cdot x \neq a_0$.

This algorithm is heuristic, because when we add a restriction at each step, we reject not only 1 solution (or $2^k - 1$ solutions in the case of $a_0 = \overline{0}$), but at least 1, – it may happen that in the solution set there is a solution that is orthogonal to the restriction added to the system. During the practical application of this algorithm we established empirically that it shows the best results when the power of the solution set is small. For example, in the case of the field $\mathbb{F}_8$, $n = 5$ and the solution number approximately 1-10, the algorithm found all these solutions. So it is advisable to use the algorithm in a situation when the system of linear restrictions has a sufficient number of linear restrictions and its solution set consists of small number of vectors.

Conclusions

In this paper, we formulated problems of checking the existence of solution and finding solution for the system of linear restrictions, as well as related problems, in which certain input parameters are fixed, and also we estimated complexity of all these given problems. We proved the Turing equivalence of decision and search problems when a field size is fixed. Also we formulated several partial cases of the original problem of checking the solution existence for systems of linear restrictions. We established belonging to the appropriate complexity class for all these partial cases. The obtained results allow us to build a polynomial probabilistic search algorithm for at least one solution of the system of linear restrictions in the case of $m \leq 2^{k-1}$. Algorithm was implemented and tested on specific input data sets. Based on this algorithm, we built another heuristic search algorithm for finding several solutions of the system of linear restrictions. During the testing, it was found that the best results this algorithm shows in the case when the power of the system solution set is approximately 1-10 vectors.

References

[1] O. Kurinnyi, "System of linear restrictions over a finite field," Theoretical and Applied Cybersecurity, vol. 4, 2023.

[2] S. Selezneva, "Complexity of the satisfiability problem for multilinear forms over a finite field," Moscow University Computational Mathematics and Cybernetics, vol. 41, pp. 81–88, 2017.

[3] C. Moore and S. Mertens, The Nature of Computation. Oxford University Press, 1 ed., 2011. ISBN: 978-0-19-923321-2.

[4] O. Goldreich, Computational Complexity: A Conceptual Perspective. Cambridge University Press, 1 ed., 2008. ISBN: 978-0-521-88473-0.