ISSN 0032-9460, Problems of Information Transmission, 2007, Vol. 43, No. 3, pp. 1–9. © Pleiades Publishing, Inc., 2007.
Original Russian Text © P.V. Trifonov, 2007, published in Problemy Peredachi Informatsii, 2007, Vol. 43, No. 3, pp. 28–38.

CODING THEORY

Interpolation in List Decoding of Reed-Solomon Codes

P. V. Trifonov[a]

Saint-Petersburg State Polytechnic University
petert@dcn.infos.ru

Received November 28, 2006

Abstract—The problem of efficient implementation of two-dimensional interpolation in the Guruswami-Sudan list decoding algorithm for Reed-Solomon codes is considered. It is shown that it can be implemented by computing the product of the ideals of interpolation polynomials constructed for subsets of interpolation points. A method for fast multiplication of zero-dimensional co-prime ideals product is proposed.

DOI: 10.1134/S003294600703????

1. INTRODUCTION

List decoding [1] enables one in many cases to increase the probability of successful data decoding after transmission over highly noisy channels. Efficient list decoding algorithms are available only for a very narrow class of error-correcting codes. Guruswami-Sudan algorithm allows one to perform list decoding of Reed-Solomon codes in polynomial time, which, however, turns out to be prohibitively high for practical applications. Construction of a bivariate interpolation polynomial passing with a given multiplicity through the points corresponding to the received symbols turns out to be the most complex step of the algorithm. In this paper a method for construction of such a polynomial is proposed, which in some cases allows one to decrease the computational complexity.

The paper is organized as follows. Section 2 presents an overview of the Guruswami-Sudan algorithm and related computational algorithms. The proposed interpolation method is described in Section 3. The main result of the paper is forumated in Theorem 2. The complexity of the proposed method is studied in Section 4.

2. LIST DECODING OF REED-SOLOMON CODES

2.1. Guruswami-Sudan algorithm

$(n, k+1, n-k)$ Reed-Solomon code over finite field $\mathbb{F}$ is a set of vectors $(f(x_1), \ldots, f(x_n))$, where $f(x)$ is a polynomial of degree at most k with coefficients in $\mathbb{F}$, and x_i are distinct elements of $\mathbb{F}$. List decoding consists in finding for any vector $Y = (y_1, \ldots, y_n)$ all polynomials (as well as the corresponding codewords) $f^{(j)}(x)$, such that $\deg f^{(j)}(x) \le k$ and their values coincide with elements of vector Y in at least τ positions, i.e. $\left|\{i | f^{(j)}(x_i) = y_i\}\right| \ge \tau$. Parameters n and k will be assumed fixed.

The main idea of the Guruswami-Sudan algorithm [2] is to construct a polynomial $Q(x, y)$ such that the solutions of the list decoding problem can be found among its functional roots $f^{(j)}(x) : Q(x, f^{(j)}(x)) = 0$, $j = 1, \ldots, u$, corresponding to the factors in the decomposition $Q(x, y) = (y - f^{(1)}(x))(y - f^{(2)}(x)) \cdots (y - f^{(u)}(x))Q'(x, y)$. Increasing the size of the list requires increasing the number of factors u in this decomposition. This in turn leads to increased number of points (x_i, y_i), such that a number of different factors $(y - f^{(j)}(x))$ simultaneously become zero. This requires construction of a polynomial $Q(x, y)$ with high root multiplicity.

Definition 1. j-th Hasse derivative $g^{[j]}(x_0)$ of polynomial $g(x) = \sum_{i=0}^{t} g_i x^i$ at point x_0 is the j-th coefficient of the "shifted" polynomial $g(x+x_0) = \sum_{i=0}^{t} g'_i x^i$. Alternatively, $g^{[j]}(x_0) = \frac{1}{j!} g^{(j)}(x_0)$, where $g^{(j)}(x)$ is the conventional formal derivative of $g(x)$.

This definition can be extended to the case of multivariate polynomials. A polynomial has a root of multiplicity r at point z, if all its Hasse derivatives of total order less than r at this point are equal zero. This will be denoted as $A(z) = 0^r$.

Definition 2. (a, b)-weighted degree of monomial $cx^i y^j$ equals $ai + bj$. (a, b)-weighted degree $\mathrm{wdeg}_{(a,b)} Q(x, y)$ of polynomial $Q(x, y)$ equals to the maximum of (a, b)-weighted degrees of its non-zero terms.

Weighted degree can be used to define term ordering. Graded lexicographic ordering is defined as $cx^i y^j \prec dx^p y^q \Leftrightarrow (ai + bj < ap + bq) \vee (ai + bj = ap + bq) \wedge (cx^i y^j \prec_{lex} dx^p y^q)$. Lexicographic ordering is defined as $cx^i y^j \prec_{lex} dx^p y^q \Leftrightarrow (j < q) \vee (j = q) \wedge (i < p)$. Leading term $\mathrm{LT}\, Q(x, y)$ of polynomial $Q(x, y) = \sum q_{ij} x^i y^j$ is given by $\arg\max_{q_{ij} \neq 0} q_{ij} x^i y^j$. Multivariate polynomials can be ordered according to their leading terms. In what follows, weighted degree of a polynomial will denote its $(1, k)$-weighted degree, unless stated otherwise. Furthermore, graded lexicographic term ordering based on $(1, k)$-weighted degree will be used.

Guruswami-Sudan algorithm includes the following steps:

1. Construct a polynomial $Q(x, y)$ such that $\mathrm{wdeg}_{(1,k)} Q(x, y) \leq l$, and points (x_i, y_i) are its roots of multiplicity r:
$$Q^{[j_1, j_2]}(x_i, y_i) = 0, j_1 + j_2 < r, i = 1, \ldots, n \tag{2.1}$$

2. Find all polynomials $f^{(j)}(x)$ such that $\deg f^{(j)}(x) \leq k$ and $Q(x, f^{(j)}(x)) = 0$. Construct the corresponding codewords and return as a solution of the decoding problem those matching vector Y in at least τ positions.

Proof of algorithm correctness, as well as the expressions relating parameters l, r, τ, k, n, and extensions of the method to the case of weighted interpolation, are given in [2]. Weighted interpolation can be used for "soft" decoding of Reed-Solomon codes [3]. This case will not be considered here for the sake of simplicity.

2.2. Construction of the interpolation polynomial

Construction of the polynomial $Q(x, y)$ satisfying $n\frac{r(r+1)}{2}$ constraints (2.1) can be considered as a special case of Hermite interpolation. Since these constraints are linear with respect to $Q(x, y)$ coefficients, the interpolation problem can be solved by means of Gaussian elimination. However, the complexity of this method is proportional to the third degree of the number of equations, making it practically infeasible.

More efficient method for construction of the interpolation polynomial $Q(x, y)$, known as *iterative interpolation algorithm* (IIA), was proposed in [4]. The main idea of this algorithm is to construct $\rho+1$ interpolation polynomials of y-degree not higher than ρ satisfying the above constraints, instead of a single one. The value of ρ is selected in such way that the required polynomial satisfying $\mathrm{wdeg}_{(1,k)} Q(x, y) \leq l$ is guaranteed to appear among these polynomials. This algorithm can be interpreted as follows [5]. Let us represent a vector of polynomials $Q_j(x, y) = \sum_{i=0}^{\rho} q_{ij}(x) y^i, j = 0, \ldots, \rho$ as $\mathcal{Y}Q(x)$, where $\mathcal{Y} = (y^0, y^1, \ldots, y^\rho)$ and $Q(x) = ||q_{ij}(x)||$ is a matrix polynomial. Let us initialize $Q(x) = I$. Let us now process sequentially interpolation points (x_i, y_i) and corresponding

constraints (2.1), multiplying at each step $Q(x)$ by matrix

$$\Delta^{(i,j_1,j_2)} = \begin{pmatrix} 1 & 0 & \dots & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ -\frac{\Delta_0}{\Delta_{j_0}} & -\frac{\Delta_1}{\Delta_{j_0}} & \dots & x - x_i & \dots & -\frac{\Delta_\rho}{\Delta_{j_0}} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 & \dots & 1 \end{pmatrix},$$

where $\Delta_j = Q_j^{[j_1,j_2]}(x_i, y_i)$, $j_0 = \arg \min\limits_{j:\Delta_j \neq 0} Q_j(x, y)$, and minimal polynomial is selected according to the graded lexicographic ordering. Element $x - x_i$ is located in column j_0 and row j_0. Multiplication by this matrix causes order $[j_1, j_2]$ Hasse derivatives of all polynomials $Q_j(x, y)$ at point (x_i, y_i) to become zero. After processing of all interpolation points the obtained polynomials $Q_j(x, y)$ satisfy the equations (2.1), and their leading terms are given by $\operatorname{LT} Q_j(x, y) = x^{i_j} y^j$, $j = 0, \dots, \rho$ with degrees i_j being smallest possible. Finally, a polynomial with minimal weighted degree should be selected among the obtained ones, which represents the solution of the interpolation problem.

The complexity of this algorithm can be estimated as $O(nr^2C)$, where C is the total complexity of operations being performed while processing each constraint (2.1). It is proportional to the number of terms in polynomials $Q_j(x, y)$, which increases with the number of processed constraints. Hence $C = O(\rho nr^2)$, and the complexity of IIA is $O(n^2r^5)$. For high-rate codes it can be decreased by subtracting from Y a codeword matching Y in some $k + 1$ symbols [6]. This causes many matrices $\Delta^{(i,j_1,j_2)}$ to become independent of the received vector, allowing thus one to perform some computations beforehand. Furthermore, the modified list decoding problem can be solved by using an interpolation polynomial $Q(x, y)$ with smaller number of coefficients. The solution of the original list decoding problem can be easily recovered from the solution of the modified problem.

2.3. Finding functional roots of the polynomial

The second step of the Guruswami-Sudan algorithm requires finding all polynomials $f^{(j)}(x)$ such that $Q(x, f^{(j)}(x)) = 0$, $\deg f^{(j)}(x) \leq k$. An efficient method for solving this problem was presented in [7]. This algorithm allows finding coefficients of $f^{(j)}(x) = \sum\limits_i f_{ji} x^i$ in ascending order of i simultaneously for all j. Coefficients f_{j0} can be recovered as follows. Let us divide $Q(x, y)$ by the highest possible degree of x. Then $Q(x, f^{(j)}(x)) = 0$ implies $Q(0, f_{j0}) = Q(0, f^{(j)}(0)) = 0$. Now standard methods for finding roots of univariate polynomials can be used to find all values f_{j0}. Then $0 = Q(x, f^{(j)}(x)) = Q(x, f_{j0} + x\widetilde{f}^{(j)}(x)) = Q^{(j)}(x, \widetilde{f}^{(j)}(x))$. Coefficients of $\widetilde{f}^{(j)}(x)$ (i.e. the remaining coefficients of $f^{(j)}(x)$) can be found from $Q^{(j)}(x, y) = Q(x, f_{j0} + xy)$ in a similar way. The complexity of this method is $O((k + 1)n\rho \log^2 \rho)$ operations, which is much smaller than the complexity of the interpolation step.

3. FAST INTERPOLATION

The standard way to reduce the complexity of computational problems involving sequential processing of a number of objects is partitioning the set of objects ito a number of subsets, their independent processing and merging the results. It is assumed that processing of a number of small subsets is easier than processing of a large set, and the complexity of merging the results is reasonably small. This section presents application of this approach to the problem of construction of a polynomial satisfying (2.1). The proposed algorithm represents a development of the method proposed in [5].

3.1. Ideal of interpolation polynomials

Let $V = \{(x_i, y_i) | i = 1, \ldots, n\}$ be the set of interpolation points. It is possible to show [5] that any polynomial $Q(x, y)$ with y-degree not higher than ρ satisfying (2.1), can be represented as $Q(x, y) = \sum_{j=0}^{\rho} Q_j(x, y) p_j(x)$, where $Q_j(x, y)$ are the polynomials constructed by IIA for the set of points V, and $p_j(x) \in \mathbb{F}[x]$. Hence, these polynomials represent a basis of the module of interpolation polynomials $M(V) = \left\{ Q(x, y) | \operatorname{wdeg}_{(0,1)} Q(x, y) \le \rho, \forall (x_i, y_i) \in V : Q(x_i, y_i) = 0^r \right\}$. Let us introduce also the ideal of interpolation polynomials $I(V) = \langle Q_0(x, y), \ldots, Q_\rho(x, y) \rangle = \left\{ \sum_{j=0}^{\rho} Q_j(x, y) p_j(x, y) \middle| p_j(x, y) \in \mathbb{F}[x, y] \right\} \supset M(V)$.

Clearly, changing the processing order of interpolation points (x_i, y_i) in IIA leads to equivalent results. This allows one to suggest the following approach [5]. Let us partition the set of interpolation points V into two disjoint subsets V_0, V_1. For each of them construct a set of basis polynomials $Q_j^{(s)}(x, y), s = 0, 1$, for modules $M(V_0)$ and $M(V_1)$. Then any polynomial $Q(x, y)$ in module $M(V)$ can be represented as $Q(x, y) = \sum_{j=0}^{\rho} Q_j^{(0)}(x, y) p_j^{(0)}(x) = \sum_{j=0}^{\rho} Q_j^{(1)}(x, y) p_j^{(1)}(x)$. Hence $Q(x, y) \in M(V_0) \cap M(V_1)$. Since the ideals $I(V_s)$ are the supersets of modules $M(V_s)$, it is also true that $Q(x, y) \in I(V_1) \cap I(V_2)$. However, the existing ideal and module intersection algoriothms turn out to be extremely complex [8].

Observe that $\det \Delta^{(i,j_1,j_2)} = \delta_{i,j_1,j_2}(x - x_i), \delta_{i,j_1,j_2} \ne 0$. Hence, for any extension of the original field $\mathbb{F}$ matrix polynomial $Q(x)$, obtained as a product of some matrices $\Delta^{(i,j_1,j_2)}$, is singular only for $x = x_i$ [9]. More specifically, the determinant of $Q(x)$ equals

$$\det Q(x) = \prod_{i=1}^{n} (x - x_i)^{r(r+1)/2} \tag{3.2}$$

Left nullspace of matrix $Q(x_i)$ is a finite-dimensional one. This implies that the set of vectors $\{\mathcal{Y} = (1, y, y^2, \ldots, y^\rho) | \mathcal{Y} Q(x_i) = 0\}$ is finite, i.e. the set of common roots of $Q_j(x, y)$ is finite. This holds for any algebraic extension of the original field. Hence, the ideal $\langle Q_0(x, y), \ldots, Q_\rho(x, y) \rangle$ is zero-dimensional [8]. Furthermore, $\{(x, y) | Q(x, y) = 0, Q(x, y) \in I(V)\} = V$. If $V_1 \cap V_2 = \varnothing$, the ideals $I(V_1)$ and $I(V_2)$ are coprime, i.e. $I(V_1) + I(V_2) = \{Q^{(1)}(x, y) + Q^{(2)}(x, y) | Q^{(1)}(x, y) \in I(V_1), Q^{(2)}(x, y) \in I(V_2)\} = \mathbb{F}[x, y]$. In this case the Chinese remainder theorem implies that the intersection of the ideals coincides with their product [10]. The product of ideals I_1 is I_2 is the set $I_1 I_2 = \left\{ \sum_{t=1}^{m} P_t(x, y) S_t(x, y) | P_t(x, y) \in I_1, S_t(x, y) \in I_2, m \ge 0 \right\}$. It is known [8] that

$$I(V_1) I(V_2) = \left\langle Q_u^{(1)}(x, y) Q_v^{(2)}(x, y),\ u, v = 0, \ldots, \rho \right\rangle. \tag{3.3}$$

Observe that such basis of the ideal may not contain the required interpolation polynomial. Finding it requires invoking the Buchberger algorithm for construction of the Gröbner basis of ideal $I(V_1)I(V_2)$, which is guaranteed to contain the minimal interpolation polynomial [8, 11]. Furthermore, the basis (3.3) contains $(\rho + 1)^2$ elements, which is much more than the number of elements in the basis produced by IIA. Hence, multiplying the ideals using the classical rule (3.3) with subsequent construction of Gröbner basis is not an optimal approach.

3.2. Generating functions of ideal bases

The interpolation complexity can be reduced by exploiting the properties of the interpolation polynomial ideals constructed for different subsets of V. In this section only the case of bivariate

polynomials will be considered, but the proposed method can be extended to the case of arbitrary number of variables.

Definition 3 (see [12]). Let $D^{[j_1,j_2]}$ be the differential operator corresponding to computation the Hasse derivative of order j_1 with respect to x and order j_2 with respect to y. Let

$$\sigma_{x^l y^m}(D^{[j_1,j_2]}) = \begin{cases} D^{[j_1-l,j_2-m]}, & j_1 \geq l \wedge j_2 \geq m \\ 0 & \text{otherwise.} \end{cases}$$

Subset G of the space of differential operators $\mathbb{D}$ is closed if for any pair $(l,m) \in \mathbb{N}^2$ and any $\delta \in G : \sigma_{x^l y^m}(\delta) \in G$.

Observe that the statement of the interpolation problem makes use of the closed set of Hasse derivatives $D^{[j_1,j_2]}$, where $j_1 + j_2 < r$.

Theorem 1 ([12, Theorem 2.8]). *Each zero-dimensional ideal $I \subset \mathbb{F}[x,y]$ is uniquely defined by giving a set of points $t_1, \ldots, t_n$ in $\mathbb{F}^2$, and for each such point a closed subspace $G_i = \mathrm{span}_{\mathbb{F}}(\delta_{i,1}, \ldots, \delta_{i,s_i}) \subset \mathrm{span}_{\mathbb{F}}(\mathbb{D})$ such that $f \in I$, if and only if for any i,j $\delta_{i,j}(t_i)(f) = 0$, where $\delta_{i,j}(t_i)(f)$ equals to the value of the differential operator $\delta_{i,j}$ applied to f at point t_i.*

This theorem enables on to perform the analysis of the ideals by studuing the set of points where ideal elements become zero, and behaviour of Hasse derivatives at these points.

Definition 4. Let $\{Q_0(x,y), \ldots, Q_\rho(x,y)\} \subset \mathbb{F}[x,y]$ be a basis of ideal I. Its generating function is defined as $Q(x,y,z) = \sum_{i=0}^{\rho} Q_i(x,y) z^i$.

The set of zeros of the generating function of the ideal I basis includes $V(I) \times \mathbb{F}$, where $V(I)$ is the affine variety (set of zeroes) of ideal I, as well as some other points, which depend on the basis being used, and the order of elements in it.

Theorem 2. *Let $I_s = \left\langle Q_0^{(s)}(x,y), \ldots, Q_\rho^{(s)}(x,y) \right\rangle, s = 0,1$ be zero-dimensional coprime ideals. Then ideals*

$$I' = \left\langle \sum_{j=0}^{\rho} Q_{i-j}^{(0)}(x,y) Q_j^{(1)}(x,y), i = 0, \ldots, 2\rho \right\rangle \tag{3.4}$$

and $I = I_0 I_1$ coincide.

Proof. Observe that the basis of ideal I', shown in the statement of the theorem, corresponds to the product (linear convolution) of I_1 and I_2 basis generating functions. Let $V(I_s) = \{(x_i, y_i)\}$ and G_i be the set of zeroes of I_s, and the corresponding closed sets of differential operators. Then for any point $(x_i, y_i) \in V_s$ the generating function of ideal I_s basis can be represented as

$$Q^{(s)}(x,y,z) = \sum_l Q_l^{(s)}(x,y) z^l = \sum_{(j_1,j_2): D^{[j_1,j_2]} \notin G_i} (x - x_i)^{j_1} (y - y_i)^{j_2} P_{ij_1j_2}^{(s)}(z),$$

where $P_{ij_1j_2}^{(s)}(z)$ are some polynomials such that $P_{ij_1j_2}^{(s)}(z) \neq 0$. Since ideals $I_s, s = 0,1$ are coprime, $(x_i, y_i) \in V_s$ implies $Q^{(1-s)}(x_i, y_i, z) \neq 0$. Multiplying the generating functions, one obtains

$$Q(x,y,z) = \sum_{(j_1,j_2): D^{[j_1,j_2]} \notin G_i} (x - x_i)^{j_1} (y - y_i)^{j_2} P_{ij_1j_2}^{(s)}(z) Q^{(1-s)}(x,y,z), (x_i, y_i) \in V_s, \tag{3.5}$$

for $s = 0,1$, and $P_{ij_1j_2}^{(s)}(z) Q^{(1-s)}(x_i, y_i, z) \neq 0, (x_i, y_i) \in V_s$. Hence, the algebraic multiplicity of common roots of polynomials in I' is the same as in the original ideals. Hence,

$$Q \in I' \Leftrightarrow (\forall (x_i, y_i) \in V_0 \cup V_1) \delta_{ij}(x_i, y_i)(Q) = 0, \tag{3.6}$$

where the differential operators δ_{ij} are exactly the same as for ideals I_0 and I_1. Since all products $Q_i^{(0)}(x,y)Q_j^{(1)}(x,y)$ used in the classical ideal multiplication method (3.3) belong both to I_0 and I_1, they satisfy (3.6). Hence, they belong to I', i.e. $I \subset I'$. Inclusion $I' \subset I$ is obvious.

Expression (3.4) enables one not only to reduce the basis of the ideal product, but to employ also the existing fast linear convolution algorithms [13, 14]. To the best of author knownledge, the problem of fast ideal multiplication has not been considered up to now.

3.3. Recovering the interpolation polynomial

The Guruswami-Sudan algorithm requires finding an interpolation polynomial $Q(x,y)$ such that $\mathrm{wdeg}_{(1,k)} Q(x,y) \le l$. It is known that any Gröbner basis always contains the minimal interpolation polynomial [11]. If the Gröbner basis is constructed with respect to the graded lexicographic ordering, this polynomial should satisfy the weighted degree constraint imposed by the Guruswami-Sudan algorithm. Hence, the basis (3.4) should be transformed into a Gröbner one. In general, this requires application of the Buchberger algorithm, which has quite high complexity. A drawback of the above described method is that even if the bases of ideals being multiplied are Gröbner ones, the basis obtained according to (3.4) does not in general preserve this property.

Recall that the original problem was finding the basis of the module of interpolation polynomials of y-degree not higher than ρ. If the polynomials $Q_i(x,y) = \sum_{j=0}^{2\rho} \widehat{q}_{ji}(x) y^j$ represent a basis of the ideal, all required interpolation polynomials can be obtained as

$$
\begin{aligned}
Q(x,y) &= \sum_{j=0}^{\rho} y^j \sum_{i=0}^{\rho} q_{ji}(x) p_i(x) = \sum_{i=0}^{2\rho} Q_i(x,y) P_i(x,y) = \sum_t y^t \sum_{j=0}^{2\rho} y^j \sum_{i=0}^{2\rho} \widehat{q}_{ji}(x) \widehat{p}_{ti}(x) \\
&= \begin{pmatrix} 1 \\ y \\ y^2 \\ \vdots \\ y^{\rho} \\ y^{\rho+1} \\ \vdots \end{pmatrix}^T \begin{pmatrix} q_{0,0}(x) & \dots & q_{0,\rho}(x) \\ q_{1,0}(x) & \dots & q_{1,\rho}(x) \\ \dots & \dots & \dots \\ q_{\rho,0}(x) & \dots & q_{\rho,\rho}(x) \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \end{pmatrix} \begin{pmatrix} p_0(x) \\ p_1(x) \\ \vdots \\ p_{\rho}(x) \end{pmatrix} \qquad (3.7) \\
&= \begin{pmatrix} 1 \\ y \\ y^2 \\ \vdots \end{pmatrix}^T \underbrace{\begin{pmatrix} \overbrace{\begin{matrix} \widehat{q}_{0,0}(x) & \dots & \widehat{q}_{0,2\rho}(x) \\ \widehat{q}_{1,0}(x) & \dots & \widehat{q}_{1,2\rho}(x) \\ \vdots & \ddots & \vdots \\ \widehat{q}_{\rho,0}(x) & \dots & \widehat{q}_{\rho,2\rho}(x) \\ \vdots & \ddots & \vdots \\ \widehat{q}_{2\rho,0}(x) & \dots & \widehat{q}_{2\rho,2\rho}(x) \\ 0 & \dots & 0 \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \end{matrix}}^{B_0} & \overbrace{\begin{matrix} 0 & \dots & 0 \\ \widehat{q}_{0,0}(x) & \dots & \widehat{q}_{0,2\rho}(x) \\ \vdots & \ddots & \vdots \\ \widehat{q}_{\rho-1,0}(x) & \dots & \widehat{q}_{\rho-1,2\rho}(x) \\ \vdots & \ddots & \vdots \\ \widehat{q}_{2\rho-1,0}(x) & \dots & \widehat{q}_{2\rho-1,2\rho}(x) \\ \widehat{q}_{2\rho,0}(x) & \dots & \widehat{q}_{2\rho,2\rho}(x) \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \end{matrix}}^{B_1} & \begin{matrix} 0 & \dots \\ 0 & \dots \\ \dots & \ddots \\ \widehat{q}_{\rho-2,0}(x) & \dots \\ \vdots & \ddots \\ \widehat{q}_{2\rho-2,0}(x) & \dots \\ \widehat{q}_{2\rho-1,0}(x) & \dots \\ \widehat{q}_{2\rho,0}(x) & \dots \\ \dots & \ddots \end{matrix} \end{pmatrix}}_{\mathcal{Q}(x)} \begin{pmatrix} \widehat{p}_{00}(x) \\ \vdots \\ \widehat{p}_{0,2\rho}(x) \\ \widehat{p}_{10}(x) \\ \vdots \\ \widehat{p}_{1,2\rho}(x) \\ \vdots \end{pmatrix}.
\end{aligned}
$$

Hence, one should construct an intersection of the infinite-dimensional module generated by polynomials $y^t Q_i(x,y)$ with the module of polynomials with y-degree at most ρ. This can be done by

transforming the semi-infinite matrix $\mathcal{Q}(x)$ with elements $\widehat{q}_{ij}(x)$ into the upper-triangular form, and dropping rows and columns with numbers higher than ρ. Obviously, this matrix polynomial is highly structured. It can be seen that its blocks B_i are identical, and need to be triangularized only once. Having done this, one should strip from the obtained matrix columns containing non-zero elements in rows with numbers higher than ρ, since these columns cannot be used to construct polynomials $Q(x,y)$ with $\mathrm{wdeg}_{(0,1)} Q(x,y) \le \rho$. The polynomials corresponding to the remaining columns should be used to derive a set of linearly independent polynomials being the basis of the desired module. Let $R(x,y) = \sum_{i=0}^{t} y^i r_i(x)$, $S(x,y) = \sum_{i=0}^{t} y^i s_i(x)$ be two polynomials (columns) from the remaining set having the same y-degree. Let us derive via the extended Euclidean algorithm polynomials $u_1(x), u_2(x), v_1(x), v_2(x)$ such that

$$\begin{aligned} r_t(x)u_1(x) + s_t(x)v_1(x) &= (r_t(x), s_t(x)), \\ r_t(x)u_2(x) + s_t(x)v_2(x) &= 0. \end{aligned}$$

Then the polynomials $R(x,y)$ and $S(x,y)$ can be replaced with $R'(x,y) = R(x,y)u_1(x) + S(x,y)v_1(x)$ and $S'(x,y) = R(x,y)u_2(x) + S(x,y)v_2(x)$. Observe that $S'(x,y)$ has y-degree stictly less than t. After a few such steps one obtains a set of polynomials $Q_i(x,y)$ being a basis of the module of interpolation polynomials. Observe that it is not necessary to do these operations until all "excessive" columns become zero. One can use the degree of the determinant of the square submatrix consisting of first $\rho + 1$ columns of $\mathcal{Q}(x)$ matrix as a stopping criterion. Equation (3.2) implies that the degree of the determinant of a matrix polynomial being a basis of the module of interpolation polynomials should be equal to the number of equations. The degree of the determinant of an upper-triangular matrix can be obtained by summing up the degrees of elements on its main diagonal.

The described procedure can be considered as an instance of Gauss or Buchberger algorithms. Computations can be simplified if the blocks of $\mathcal{Q}(x)$ matrix are initially obtained in an upper-triangular form. Observe that if the polynomials $Q_j^{(s)}(x,y)$ represent Gröbner bases of I_s ideals with respect to lexicographic ordering, they satisfy the condition $\mathrm{wdeg}_{(0,1)} Q_j^{(s)}(x,y) = j$. This property is preserved for polynomials obtained via (3.4). Hence, if the bases of interpolation polynomials for subsets V_s were obtained for the case of lexicographic ordering, matrix $\mathcal{Q}(x)$ consists of upper-triangular blocks. Furthermore, it is sufficient to construct polynomials

$$Q_i(x,y) = \sum_{j=0}^{\rho} Q_{i-j}^{(0)}(x,y) Q_j^{(1)}(x,y),\ i = 0, \ldots, \rho, \tag{3.8}$$

since the remaining ones do not affect the basis of the required module.

Construction of the interpolation polynomial module basis in the form of columns of an upper-triangular matrix is equivalent to finding a Gröbner basis of the corresponding ideal with respect to lexicographic ordering. However, the Guruswami-Sudan algorithm requires finding a polynomial with the minimum weighted degree which is guaranteed to appear only in the Gröbner basis constructed with respect to graded lexicographic ordering. Transformation of a Gröbner basis of a zero-dimensional ideal from one ordering to another can be performed by means of algorithms given in [15, 16]. For long codes one can employ the method given in [17].

Hence, the proposed interpolation algorithm involves the following steps:

1. Partitioning of the set of interpolation points into a number of disjoint subsets V_s.
2. Construction of bases of modules of interpolation polynomials having points in V_s as roots of multiplicity r with respect to lexicographic ordering. This step can be performed in parallel for different V_s using either the IIA, or recursively with the proposed method.

3. Construction of the ideal product basis according to (3.8). At this step one can employ fast convolution algorithms. In the implementation of the fast algorithm it is desired to take into account non-uniformity of $Q_i^{(s)}(x, y)$ degrees.
4. Correction of the obtained ideal basis with respect to lexicographic ordering.
5. Transition to graded lexicographic ordering by employing the algorithms given in [15, 16].

This method can be used jointly with the "re-encoding" approach proposed in [6]. Furthermore, some complexity reduction can be achieved by exploiting the algebraic structure of Gröbner bases with respect to lexicographic ordering.

4. EFFICIENCY ANALYSIS

Obtaining an analytical complexity estimate for the proposed method turns out to be non an easy problem. The main difficulties arise in the analysis of ideal bases generating functions multiplication complexity. Indeed, the tri-variate polynomials being multiplied have an "upper-triangular" form. Hence, elementary multiplications of polynomials with smaller number of variables, which are used in fast convolution algorithms, have different complexity. Furthermore, the complexity estimate of Gröbner basis transformation algorithm [15, 16] was obtained only for the worst case.

Due to described difficulties the performance of the proposed method was studied experimentally. The algorithm was implemented in C++ programming language and execution time of its different stages was recorded. All experiments were carried out on a PC equipped with AMD Athlon 64 X2 2,2GHz CPU. The results are presented in the following table.

Interpolation time, seconds

Code	Construction of $I(V_0), I(V_1)$ bases	Ideal multiplication	Module reduction	IIA over $V_0 \cup V_1$
(32,17)	0,045	0,068	0,1109	0,61
(32,20)	0,018	0,02	0,046	0,19
(256,239)	0,071	0,04	0,137	0,39

It can be seen that the total execution time of the proposed algorithm is slightly less compared to the case of standard IIA [4] applied to all points (the last column). Furthermore, the total time required for merging the results of interpolation subproblems (third and fourth columns) is much higher than the time required for solving these subproblems. This means that further improvement of the proposed method is needed.

5. CONCLUSIONS

In this paper a method for solving the interpolation problem in the Guruswami-Sudan algorithm was proposed. The method is based on partitioning of the original problem into a number of subproblems with subsequent merging of their solutions. It was shown that this can be done by constructing the basis of the product of interpolation polynomial ideals constructed for subsets of interpolation points. A method for finding a basis of the product of zero-dimensional co-prime ideals was proposed, which requires fewer operations than the standard rule.

All proposed algorithms were implemented in software. Numerical experiments indicate that application of these algorithms enables one to reduce the computational complexity of the interpolation step of the Guruswami-Sudan algorithm. However, further research is needed in order to achieve more significant complexity reduction.

The proposed method was presented at 10-th Internation Workshop on Algebraic and Combinatorial Coding Theory [18] and at the seminar of A.A. Harkevich Institute for Information Transmission Problems, Russian Academy of Science (Moscow). The author thanks the organizers and participants of the workshop and the seminar. Furthermore, the author thanks Prof. A.I. Generalov (Saint-Petersburg State University) for fruitful discussion of theorem 2.

REFERENCES

1. Elias, P. List Decoding for Noisy Channels. Tech. Rep. 335. Research Laboratory of Electronics, MIT, 1957.
2. Guruswami, V., and Sudan, M. Improved Decoding of Reed-Solomon and Algebraic-Geometric Codes. *IEEE Trans. Inform. Theory.* 1999. vol. 45, no. 6. pp. 1757–1767.
3. Koetter, R., and Vardy, A. Algebraic Soft-Decision Decoding of Reed-Solomon Codes. *IEEE Trans. Inform. Theory.* 2003. vol. 49, no. 11. pp. 2809–2825.
4. Nielsen, R. R., and Hoholdt, T. Decoding Reed-Solomon Codes Beyond Half the Minimum Distance. *Proc. of the Int. Conf. on Coding Theory and Cryptography.* Mexico: Springer-Verlag, 1998.
5. Ma, J., Trifonov, P., and Vardy, A. Divide-and-Conquer Interpolation for List Decoding of Reed-Solomon Codes. *Proc. of IEEE Int. Sympos. on Inform. Theory.* Chicago, USA: 2004. June 27 – July 2. P. 387.
6. Koetter, R., Ma, J., Vardy, A., and Ahmed, A. Efficient Interpolation and Factorization in Algebraic Soft-Decision Decoding of Reed-Solomon Codes. *Proc. of IEEE Int. Sympos. on Inform. Theory.* Yokohama, Japan, 2003, June 29 – July 4. P. 365.
7. Roth, R., and Ruckenstein, G. Efficient Decoding of Reed-Solomon Codes Beyond Half the Minimum Distance. *IEEE Trans. Inform. Theory.* 2000. vol. 46, no. 1. pp. 246–257.
8. Cox, D., Little, G., and O'Shea, D. Ideals, varieties and algorithms. Springer-Verlag, 1992.
9. Kailath, T. Linear Systems. Englewood Cliffs, NJ: Prentice Hall, 1985.
10. Becker, T., and Weispfenning, V. Gröbner Bases. A Computational Approach to Commutative Algebra. New York: Springer, 1993.
11. Sauer, T. Polynomial Interpolation of Minimal Degree and Gröbner Bases. Gröbner Bases and Applications *London Mathematical Society Lecture Notes.* vol. 251 Cambridge University Press, 1998. pp. 483–494.
12. Marinari, M. G., Moller, H. M., and Mora, T. Gröbner Bases of Ideals Defined By Functionals With An Application to Ideals of Projective Points. *Applicable Algebra in Engineering, Communication and Computing.* 1993. vol. 4, no. 2. pp. 103–145.
13. Blahut, R. Fast Algorithms for Digital Signal Processing. Addison-Wesley, 1984.
14. Blahut, R. Theory and Practice of Error Control Codes. Addison-Wesley, 1984.
15. Faugere, J.-C., Gianni, P., Lazard, D., and Mora, T. Efficient Computation of Zero-Dimensional Gröbner Bases By Change of Ordering. *J. of Symbolic Computation.* 1993. vol. 16, no. 4. pp. 329–344
16. Basiri, A., and Faugere, J.-C. Changing the Ordering of Gröbner Bases With LLL: Case of Two Variables. *Proc. of the Int. Sympos. on Symbolic and algebraic computation.* Philadelphia, PA, USA: 2003. August 3 – 6. pp. 23–29.
17. Alekhnovich, M. Linear Diophantine Equations Over Polynomials and Soft Decoding of Reed-Solomon Codes. *IEEE Trans. On Inform. Theory.* 2005. July. vol. 51, no. 7. pp. 2257–2265.
18. Trifonov, P. On the Interpolation Step in the Guruswami-Sudan List Decoding Algorithm for Reed-Solomon Codes. *Proc. of Int. Workshop on Algebraic and Combinatorial Coding Theory.* Zvenigorod, Russia: 2006. September 3–9. pp. 269–272.