Linear Diophantine equations over polynomials and soft decoding of Reed-Solomon codes

Michael Alekhnovich *

October 20, 2004

Abstract

We generalize the classical Knuth-Schönhage algorithm computing GCD of two polynomials for solving arbitrary linear Diophantine systems over polynomials in time, quasi-linear in the maximal degree. As an application, we consider the following weighted curve fitting problem: given a set of points in the plain, find an algebraic curve (satisfying certain degree conditions) that goes through each point the prescribed number of times. The main motivation for this problem comes from the coding theory, namely it is ultimately related to the list decoding of Reed-Solomon codes.

We present a new fast algorithm for the weighted curve fitting problem, based on the explicit construction of Groebner basis. This gives another fast algorithm for the soft decoding of Reed-Solomon codes different from the procedure proposed by Feng, which works in time $(\frac{w}{r})^{O(1)} n \log^2 n$, where r is the rate of the code, and w is the maximal weight assigned to a vertical line.

1. Preliminaries

Let $\mathbf{F}$ be an arbitrary field. This paper considers the following optimization problem:

Problem 1
INPUT: An $m \times \ell$ matrix P consisting of polynomials $P_{ij} \in \mathbf{F}[x]$.
SOLUTION: a tuple of polynomials $f_1, ..., f_\ell \in \mathbf{F}[x]$ and integers $d_1, ..., d_m$ s.t.

$$\begin{array}{c} \deg\left(P_{11}f_1 + P_{12}f_2 + ... + P_{1\ell}f_\ell\right) = d_1 \\ \deg\left(P_{21}f_1 + P_{22}f_2 + ... + P_{2\ell}f_\ell\right) = d_2 \\ ... \\ \deg\left(P_{m1}f_1 + P_{m2}f_2 + ... + P_{m\ell}f_\ell\right) = d_m \end{array} \tag{1}$$

GOAL:

$$\begin{array}{c} \max(d_1, d_2, ..., d_m) \to min \\ \max(d_1, d_2, ..., d_m) \neq 0 \end{array}$$

One can see that this problem is indeed a generalization of GCD of two polynomials P_1 and P_2 as the latter can be expressed as:

$$\begin{array}{c} \deg\left(P_1 f_1 + P_2 f_2\right) \to \min \\ P_1 f_1 + P_2 f_2 \neq 0 \end{array} \tag{2}$$

*Laboratory for Computer Science, MIT, mishka@theory.lcs.mit.edu

If f_1, f_2 is the solution of (2) then $P_1 f_1 + P_2 f_2$ is the greatest common divisor of P_1 and P_2. Increasing the number of polynomials involved, one can express various problems of simultaneous approximation: given polynomials $P_1, P_2, ..., P_m$ find $g_1, ..., g_m, g$ of the smallest degree such that

$$g_1 P_1 + g_2 P_2 + ... + g_m P_m = g.$$

The main suggested application, however, concerns algebraic interpolation problems which we will discuss below. Meanwhile, let us focus on the complexity of finding the optimal solution. It is easy to see that the problem (1) can be stated as a sequence of (ordinary) homogeneous linear systems over the coefficients of $f_1, ..., f_\ell$ and thus can be solved in polynomial time by the Gaussian procedure. However, in the case when the degree of P_{ij} is large w.r.t. m and ℓ, we show a faster way to deal with this problem. The following result is a generalization of the fast Knuth-Schönhage algorithm for GCD of two polynomials (see, for example, [AHU74] or Chapter 3 of [BCS96]). In what follows we denote by $M(n)$ the minimal number of arithmetic operations necessary to multiply two polynomials of degree n. If the ground field supports Fast Fourier Transform then $M(n) \leq n \log n$, otherwise $M(n) \leq n \log n \log \log n$ (see [BCS96], page 32 and pages 34-38).

Theorem 1.1 *There exists an algorithm that solves Problem 1 in time $(\ell m)^{O(1)} M(n) \log n$, where n is the maximal degree of P_{ij}.*

In the second part of the paper we show an application of this result, which concerns the list decoding of Reed-Solomon codes. In general, list decoding evolves around the following basic question: what information can be learned from the corrupted codeword, when the number of possible errors is too big for the unique decoding? In this case one can hope to generate a small "list" of candidates for being the correct "transmitted" codeword. These ideas lead to an interesting and rich theory of list decoding, which has been rapidly developed during the recent years. We do not give any further general information on list decoding and confine ourselves with one particular problem, however, we advise the interested reader to see [Gur01] for an excellent survey on the subject. The (weighted) list decoding of Reed-Solomon codes can be described by the following

Problem 2 (Weighted polynomial reconstruction)
INPUT: N points $(x_1, y_1), ..., (x_N, y_N)$, with natural weights $\mathrm{wt}(x_1, y_1), ..., \mathrm{wt}(x_N, y_N)$ and integer parameters k and W.
OUTPUT: All polynomials p of degree at most k s.t. $\sum_{i:p(x_i)=y_i} \mathrm{wt}(x_i, y_i) \geq W$.

Here polynomials p of degree at most k are considered as codewords. The intuitive meaning of weights is the measure of "confidence" that the "transmitted" p has $p(x_i) = y_i$. Thus, the goal is to find all codewords, for which the overall agreement is greater than the specified bound. A priori there are no requirements for the points to be distinct, in particular different points can have the same x-coordinate.

The weighted polynomial reconstruction problem plays an important role in the list decoding ([GS99]) and soft decoding algorithms ([KV00]), also it is used for decoding of the codes that result from the concatenation of Reed-Solomon with smaller inner codes ([GI01]). All these decoding procedures inherit the complexity of Problem 2, thus it is an interesting and important goal to design fast solutions for this problem.

It was shown in a series of works by Sudan [Sud97] and Guruswami, Sudan [GS99] that Problem 2 can be solved in time polynomial in the sum of the weights provided $W > \sqrt{k \sum_{i=1}^{N} \mathrm{wt}^2(x_i, y_i)}$ (this in particular implies that there can be at most polynomially many solutions). The main idea of their analysis is to consider the following interpolation subproblem:

Problem 3 (Weighted curve fitting)
INPUT: N points $(x_1, y_1), ..., (x_N, y_N)$ with natural integer weights $\mathrm{wt}(x_1, y_1), ..., \mathrm{wt}(x_N, y_N)$ and parameters k and D.
OUTPUT: A polynomial $Q(x, y)$ such that

1. For any point (x_i, y_i), Q can be represented as a sum

$$Q = \sum_{t=0}^{\mathrm{wt}(x_i,y_i)} g_{t,i}(x, y)(x - x_i)^t (y - y_i)^{\mathrm{wt}(x_i,y_i)-t} \tag{3}$$

2. $\deg_x(Q) + k \deg_y(Q) \le D$

In the case of continuous field, the first condition states that $Q(x, y)$ has a singularity of order $\mathrm{wt}(x_i, y_i)$ in (x_i, y_i) or equivalently, the first $\mathrm{wt}(x_i, y_i)$ derivatives of Q equal to 0 in the point (x_i, y_i).

The original polynomial algorithm of Guruswami and Sudan deals with the weighted curve fitting by solving the corresponding homogeneous linear system. Following the publication of [Sud97] and [GS99], several papers have been published that present fast implementations of that list decoding algorithm. In one of these papers Feng [F99] (followed by the journal publication of Feng and Giraud [FG04]) suggested a faster interpolation procedure for solving Problem 3 in time $(w\frac{n}{k})^{O(1)} n \log^3 n$. We construct an alternative fast algorithm that uses different ideas, in particular we explicitly describe the Groebner basis for the ideal of all curves satisfying (3). Our upper bound is $\log n$ factor better than that of Feng and Giraud (see Section 4 for the brief comparison of these approaches). The techniques we use can be considered as a natural far-going generalization of the rational interpolation problem, which lies in the base of original Welch-Berlekamp decoding algorithm [WB86] (in particular the rational interpolation is a partial case of Problem 1). Our result is described by

Theorem 1.2 *There exists an algorithm that solves Problem 3 in time $(w\frac{n}{k})^{O(1)} M(n) \log n$, where w is the maximal overall weight assigned to a single line $x = x_i$ for all choices of i.*

The following corollary follows the ideas of Roth and Ruckenstein [RR98], [F99] and [FG04]. We discuss it in the appendix.

Corollary 1.3 *There exists an algorithm that solves Problem 2 in time $(w\frac{n}{k})^{O(1)} M(n) \log n$, where w is the maximal overall weight assigned to a single line $x = x_i$ for all choices of i, provided $W > \sqrt{k \sum_{i=1}^{N} \mathrm{wt}^2(x_i, y_i)}$.*

The paper is organized in the following way. We prove Theorem 1.1 in Section 2, whereas Section 3 is dedicated to the proof of Theorem 1.2. Finally, in the appendix we overview the further steps in the weighted polynomial reconstruction after the curve with required properties was found (how to infer Corollary 1.3 from Theorem 1.2).

2. Solving linear Diophantine equations

In this section we design a fast algorithm solving Problem 1. First we reformulate the problem in terms of the shortest vector for the appropriate lattice. Here we exploit the classical lattice techniques for solving linear Diophantine equations and the problems of simultaneous approximation that goes back to the works of Dirichlet and Minkowski of the 19th century.

Next, as in the original Knuth – Schönhage algorithm we design a basic (slow) routine finding the shortest vector in Section 2.1 (in fact in Knuth – Schönhage procedure such routine is the Euclid algorithm), and finally in Section 2.2 we optimize it to achieve the bounds of Theorem 1.1.

We will need the following basic facts about lattices over commutative rings. For a good survey on the topic we advise the reader to see [Lov86].

Definition 2.1 Let R be an arbitrary commutative ring with 1. The *lattice* $\mathcal{L}_B$ generated by basis $B = \{\beta_1, ..., \beta_\ell\}$, $\beta_i \in R^m$ is the set of all linear combinations

$$\mathcal{L}_B = \{f_1\beta_1 + f_2\beta_2 + ... + f_\ell\beta_\ell : \ f_i \in R\}.$$

It is convenient to represent a basis as $m \times \ell$-matrix $B = \{\beta_i\}$, that contain the vectors β_i as columns, then the lattice is described as $\mathcal{L}_B = \{B \cdot y : \ y \in R^\ell\}$.

Definition 2.2 Consider the space $R^{\ell\times\ell}$ of matrices with coefficients in R. A matrix M is *unimodular* ($M \in U_\ell(R)$) iff M can be represented as a product of elementary transvections $I + fe_{ij}$, where I is identity and e_{ij} contains a single 1 in the position (i, j) and zeros in the rest.

Two bases B_1 and B_2 are *equivalent* iff $B_1 = B_2 \cdot U$, for some $U \in U_\ell(R)$. In other words, they are equivalent if B_2 can be transformed to B_1 by elementary operations on the columns.

Since $U_\ell(R)$ is a group, the equivalence relation is well-defined, equivalent bases generate the same lattice. Let us now confine ourselves to the ring of polynomials $R = \mathbf{F}[x]$. We generalize the degree of a polynomial for a vector of polynomials and a set of vectors.

Definition 2.3 For a vector $\beta \in \mathbf{F}[x]^m$ denote its jth coordinate by $\beta(j)$. Let

$$\deg \beta = \max_{j=[1..m]} \deg \beta(j).$$

For a basis $B = \{\beta_1, \beta_2, ..., \beta_\ell\}$ let $\deg B = \sum_{i=1}^{\ell} \deg \beta_i$.

Definition 2.4 An element $v \in \mathcal{L}$ is the *shortest vector* iff it is non-zero and for any other non-zero $v' \in \mathcal{L}$, $\deg v \leq \deg v'$.

Let us now look at Problem 1 in the light of this definition. It is easy to see that $d_1, ..., d_m$ in (1) are the degrees of an arbitrary vector in the lattice generated by P_{ij} as a basis. Thus Problem 1 can be reformulated in the following equivalent form:

Problem 4 Given a basis $B \in (\mathbf{F}[x]^{m\times\ell})$ find any shortest vector in $\mathcal{L}_B$.

There is a subtle point that in (1) we require the vector in the lattice have degree strictly greater than 0, whilst in Problem 4 any non-zero solution qualifies. This difficulty can be ruled out by multiplying every P_{ij} by x.

2.1. Basic algorithm computing the shortest vector

Recall, that for a polynomial $f = a_1x^d + a_2x^{d-1} + ... + a_{d+1}$, its *leading term* is the monomial of the highest degree: $LT(f) = a_1x^d$. In the next definition we introduce a leading coordinate of a vector, and then define an important notion of the reduced basis [1].

[1]After the publication of the preliminary version of this paper it was pointed out by Peter Buergisser that this notion (as well as Proposition 2.3) was known earlier in the context of factoring polynomials over finite fields (see [Gat84, Len85] and the references cited therein).

```
Algorithm R(B)
Input: basis B ∈ F[x]^{m×ℓ}
Output: matrix R(B) ∈ U_ℓ(F[x]) s.t. either B · R(B) is reduced
     or deg(B · R(B)) < deg B.
begin
      let n = deg B; U = I
      while deg(B) = n do
      begin
          if B is reduced then return U
(1)       else find i_1 ≠ i_2 s.t. LC(β_{i_1}) = LC(β_{i_2})
          let j = LC(β_{i_1})
          if deg β_{i_1} < deg β_{i_2} then switch i_1 and i_2
          let f = LT(β_{i_1}[j])/LT(β_{i_2}[j])
          let T = I − f e_{i_2 i_1}
          let U = U · T
          let B = B · T
      end
      return U
end
```

Figure 1: The elementary reduction operator

Definition 2.5 For a vector β define its *leading coordinate* as

$$LC(\beta) = \max\{j : \deg \beta(j) = \deg \beta\}$$

(the biggest index of polynomial of the maximal degree in β).

The basis $B = \{\beta_i\}$ is *reduced* iff for any pair of distinct non-zero vectors $\beta_1, \beta_2 \in B$ holds $LC(\beta_1) \neq LC(\beta_2)$.

If a basis is not reduced then one can try to reduce it by elementary operations on its columns. It can be done efficiently as we show in the following

Proposition 2.1 *For any basis there exists an equivalent reduced one.*

Proof. First we construct an algorithm, that either decreases the degree of a basis at least by one or outputs the reduced basis. Consider the algorithm depicted in Figure 1.

Given a basis B it computes a unimodular matrix $R(B)$ s.t either $\deg(B \cdot R(B)) < \deg B$ or $B \cdot R(B)$ is reduced. Informally, it does the following: if the basis is not reduced then one can find two columns that have polynomials of the highest degree in the same position j (line 1 of the algorithm). Let $LT(\beta_{i_1}(j)) = a_1 x^{d_1}$, $LT(\beta_{i_2}(j)) = a_2 x^{d_2}$, assume w.l.o.g. that $d_1 \geq d_2$. We subtract the i_2nd column multiplied by $(a_1/a_2)x^{d_1 - d_2}$ from the i_1st column to decrease the degree of $\beta_{i_1}(j)$. We store this elementary column operation in U and continue the process until we either decrease the degree of B or completely reduce the basis. Note, that at each step either the degree of the whole β_{i_1} decreases or the leading coordinate $LC(\beta_{i_1})$ becomes strictly less, thus at some moment the process will be finished. It will be useful later to have the following estimate on the complexity of computing R:

Algorithm $R(B,t)$
Input: basis $B \in (\mathbf{F}[x])^{m\times\ell}$
Output: matrix $R(B,t) \in U_\ell(\mathbf{F}[x])$ s.t. either $B \cdot R(B,t)$ is reduced or $\deg(B \cdot R(B,t)) \le \deg B - t$.

```
begin
    let n = deg B; U = I
    while deg(B · U) > n − t do
    begin
        if B · U is reduced then return U
        else let U = U · R(B · U)
    end
    return U
end
```

Figure 2: The iterative reduction operator

Lemma 2.2 *$R(B)$ does at most $O(m\ell(m+\ell))$ operations of polynomial addition and multiplication of the polynomial by a constant.*

Proof. At each step the algorithm decreases $LC(\beta_{i_1})$ for some i_1. Clearly the maximal possible number of steps is ℓm. Finally to multiply the matrices U and B by the transvection $I - fe_{i_1 i_2}$ one can subtract the column i_2 multiplied by f from the column i_1 (this step takes $m+\ell$ operations).∎

Given an elementary reduction R one can define another algorithm depicted in Figure 2 that reduces the degree of the basis at least by t (or outputs the reduced basis). At each step it checks if the basis is reduced, and if not then it applies the operator R to decrease the degree further, until the degree decreases at least by t.

Finally, given a basis B it is sufficient to compute the matrix $R(B, \deg B)$ to get the reduced equivalent basis $B \cdot R(B, \deg B)$. Proposition 2.1 is proved.∎

Proposition 2.3 *Assume that B is a reduced basis. Let β be any non-zero vector of the minimal degree in B. Then β is the shortest vector in $\mathcal{L}_B$.*

Proof. For the sake of contradiction assume that there exist $f_1, ..., f_k$ s.t.

$$\deg\left(\sum_{i=1}^{\ell} f_i\beta_i\right) < \deg\beta. \tag{4}$$

Notice, that $LC(f_i\beta_i) = LC(\beta_i)$ for any non-zero f_i. Let us consider the vector $f_i\beta_i$ in the sum (4) of the maximal degree, if there are several such vectors choose one with the maximal leading coordinate (it can be done in the unique way because B is reduced). Denote $j = LC(f_i\beta_i)$. We claim that the leading term of $f_i\beta_i(j)$ does not cancel in the sum (4). Indeed, $LT(f_i\beta_i(j))$ can be canceled only by another vector $f_{i'}\beta_{i'}$ of the maximal degree. But this can not happen as we have chosen the vector of the maximal degree with the maximal leading coordinate. The proposition follows.∎

One can observe that algorithm $R(B,t)$ already solves Problem 4. However we are interested in faster solutions, for that we apply a trick similar to that of the original Knuth-Schönhage algorithm.

2.2. Improved algorithm computing the shortest vector

One of the main concepts of Knuth-Schönhage GCD algorithm is the following definition (we allowed ourselves a small modification to adjust it for our purposes):

Definition 2.6 For a polynomial P and an integer t, the *accuracy t approximation* of P is given by the formula

$$P|_t = \pi\left(Px^{t-\deg(P)-1}\right)x^{\deg(P)+1-t},$$

where $\pi : \mathbf{F}[x, x^{-1}] \to \mathbf{F}[x]$ is a linear map defined by $x^k \mapsto x^k$ for $k \geq 0$ and $x^k \mapsto 0$ for $k < 0$. Two polynomials P_1 and P_2 are *equivalent up to accuracy t* ($P_1 \overset{t}{\sim} P_2$) iff $(P_1)|_t = (P_2)|_t$.

Informally, the accuracy t approximation of a polynomial is the sum of t highest terms:

$$\left(a_1x^d + a_2x^{d-1} + ... + a_{d+1}\right)|_t = a_1x^d + ... + a_tx^{d-t+1}$$

We generalize this definition for vectors of polynomials:

Definition 2.7 For $\beta \in \mathbf{F}^m$ let

$$\beta|_t = \pi\left(\beta \cdot x^{t-\deg(\beta)-1}\right)x^{\deg\beta+1-t},$$

where vector β is multiplied by x^d and rounded up by $\pi(\cdot)$ independently for each coordinate. For $B \in \mathbf{F}^{m\times\ell}$ let

$$B|_t = \{(\beta_i)|_t\}$$

The accuracy relation is extended in the straightforward way:

$$\beta_1 \overset{t}{\sim} \beta_2 \iff (\beta_1)|_t = (\beta_2)|_t$$

$$B_1 \overset{t}{\sim} B_2 \iff (B_1)|_t = (B_2)|_t$$

Lemma 2.4

1. *If $\beta \overset{t}{\sim} \beta'$ then $x^l\beta \overset{t}{\sim} x^l\beta'$.*
2. *If $\beta_1 \overset{t}{\sim} {\beta_1}'$, $\beta_2 \overset{t}{\sim} {\beta_2}'$ and $\deg(\beta_1) = \deg(\beta_2) = d$ then*

$$\beta_1 + \beta_2 \overset{\hat{t}}{\sim} {\beta_1}' + {\beta_2}',$$

where $\hat{t} = t - (d - \deg(\beta_1 + \beta_2))$.

Proof. The first claim is checked independently for each coordinate in the straightforward way.

Let us prove the second claim, we can assume w.l.o.g. that $\hat{t} \geq 1$ (otherwise it is trivial). In this case we can write

$$\pi\left((\beta_1 + \beta_2) \cdot x^{\hat{t}-\deg(\beta_1+\beta_2)-1}\right) = \pi\left((\beta_1 + \beta_2)x^{t-d-1}\right) = \pi\left((\beta_1' + \beta_2')x^{t-d-1}\right),$$

which implies that the degrees of $\beta_1 + \beta_2$ and $\beta_1' + \beta_2'$ coincide and $\beta_1 + \beta_2 \overset{\hat{t}}{\sim} {\beta_1}' + {\beta_2}'$.∎

Definition 2.8 The *length* of a polynomial f is the minimal t s.t. $f(x) = a_1 x^n + a_2 x^{n-1} + ... + a_t^{n-t+1}$.

Proposition 2.5 *For any field* $\mathbf{F}$ *two polynomials of length* t *can be multiplied in* $M(t)$ *arithmetic operations.*

Theorem 2.1 *There is an algorithm that finds the shortest vector in a lattice* $\mathcal{L}_B$ *in time* $O((\ell + m)^4 M(n) \log n)$ *where* n *is the maximal degree of polynomials in* B*.*

Proof.

First we need to establish several crucial properties of the operator $R(B,t)$.

Lemma 2.6 *For any* $t_1 < t$

$$R(B,t) = R\left[B, t_1\right] \cdot R\left[B \cdot R(B,t_1), t - (\deg B - \deg(B \cdot R(B,t_1)))\right]$$

Proof. Recall the construction of $R(B,t)$. The lemma states that in order to reduce the degree of a basis by t, the algorithm first reduces it by t_1, computes the basis $\hat{B} = B \cdot R(B,t_1)$ and then reduces it further by $t - (\deg B - \deg \hat{B})$. Finally it takes the composition of these reductions.$\blacksquare$

Lemma 2.7

$$R(B,t) = R(B|_t, t) \tag{5}$$

Proof. Lemma 2.7 Let $B_1 = B$, $B_2 = B|_t$. Let us analyze the performance of $R(\cdot,t)$ on B_1 and B_2, for that we run two copies of the algorithm in parallel. At the start, $B_1 \overset{t}{\sim} B_2$. By our definition of $R(B)$, the procedure $R(\cdot,t)$ will make the same elementary transformations on B_1 and B_2 as long as $B_1 \overset{1}{\sim} B_2$.

Assume that at some step of the algorithm $B_1 \overset{\hat{t}}{\sim} B_2$. By Lemma 2.4 whenever R makes an elementary transformation $\beta_1 := \beta_1 + f\beta_2$ that does not change the degree of β_1, the accuracy does not change either: $B_1 \overset{\hat{t}}{\sim} B_2$. If the degree of B_1 decreases by d then the accuracy will decrease by the same d. It is left to notice that the algorithm $R(\cdot,t)$ stops after the degree is reduced by t.$\blacksquare$

Lemma 2.8 *Any monomial in* $R(B,t)_{ij}$ *has the form* $ax^{d_{ij}+e}$*, where* $d_{ij} = \deg \beta_i - \deg \beta_j$ *and* $|e| \leq t-1$*.*

Proof. Lemma 2.8 We prove it by induction. The base follows readily from our definition of $R(B)$ (notice that $R(B)$ stops its computation immediately after the degree of any vector β_j has decreased). Let us check the induction step. By Lemma 2.6 (for $t_1 = 1$)

$$R(B,t) = R(B) \cdot R\left[B \cdot R(B), t - (\deg B - \deg(B \cdot R(B)))\right]$$

Consider the matrix $\hat{B} = B \cdot R(B)$, assume that $\deg \hat{B} = \deg B - d$. Clearly the degree of all columns in $\hat{B}$ coincides with those of B except one vector $\hat{\beta}_i$, whose degree is $\deg \beta_i - d$, hence for all i,j the difference $|\hat{d}_{ij} - d_{ij}| \leq d$. By the induction hypothesis any monomial in $R(\hat{B}, t-d)_{ij}$ has the form $ax^{\hat{d}_{ij}+e}$, $|e| < t-d-1$, while any monomial in $R(B)_{ij}$ has the form $bx^{d_{ij}}$.

Algorithm $\hat{R}(B,t)$
Input: basis $B \in (\mathbf{F}[x])^{m\times\ell}$
Output: matrix $\hat{R}(B,t) \in U_\ell(\mathbf{F}[x])$ s.t. either $B \cdot \hat{R}(B,t)$ is reduced or $\deg(B \cdot \hat{R}(B,t)) \le \deg B - t$.
begin
  let $B = B|_t$
  if $t = 1$ **then return** $R(B)$
  let $U_1 = \hat{R}(B, \lfloor t/2 \rfloor)$
  let $B_1 = B \cdot U_1$
  return $U_1 \cdot \hat{R}(B_1, t - [\deg B - \deg B_1])$
end

Figure 3: The improved iterative reduction operator

Finally, any monomial in the product $\left(R(B) \cdot R(\hat{B}, t-d)\right)_{ij}$ comes from the polynomial

$$R(B)_{ik} R(\hat{B}, t-d)_{kj}$$

for some k and hence can be written as the sum of terms

$$bx^{d_{ik}} \cdot ax^{\hat{d}_{kj}+e} = bax^{d_{ik}} \cdot x^{d_{kj}} \cdot x^{\hat{d}_{kj}-d_{kj}+e} = bax^{d_{ij}+e'},$$

where $|e'| = |d_{kj} - \hat{d}_{kj} + e| \le d + (t-d-1) = t-1.$ ■

Now we are ready to construct a faster algorithm to compute $R(B,t)$, depicted in Figure. 3. The behavior of $\hat{R}(B,t)$ is described by the following two lemmas.

Lemma 2.9 (correctness)

$$\hat{R}(B,t) = R(B,t)$$

Proof. We prove it by induction. If $t = 1$ then the equality is obvious. If $t > 1$ then

$$\hat{R}(B,t) = \hat{R}\left[B|_t, \lfloor t/2 \rfloor\right] \cdot \hat{R}\left[B|_t \cdot \hat{R}(B|_t, \lfloor t/2 \rfloor), t - (\deg B|_t - \deg(B|_t \cdot \hat{R}(B|_t, \lfloor t/2 \rfloor)))\right] =$$

$$= \text{(induction hypothesis)} =$$

$$= R\left[B|_t, \lfloor t/2 \rfloor\right] \cdot R\left[B|_t \cdot R(B|_t, \lfloor t/2 \rfloor), t - (\deg B|_t - \deg(B|_t \cdot R(B|_t, \lfloor t/2 \rfloor)))\right] =$$

$$= \text{(Lemma 2.6)} = R(B|_t, t) = \text{(Lemma 2.7)} = R(B,t).$$

■

Lemma 2.10 *The running time of $\hat{R}(B,t)$ is bounded by $O((\ell+m)^3 M(t) \log t)$.*

Proof. The proof is based on Lemma 2.8. In fact the only reason why $\hat{R}(B,t)$ is more effective than $R(B,t)$ is that it deals with polynomials of smaller length.

Let us first estimate the number of calls to $R(B)$. Each time we apply this operator the degree is decreased at least by one, thus there can be at most t calls that take time $O(\ell m(\ell+m)t)$. Indeed, by Lemma 2.2, the cost of each call is $O(\ell m(\ell+m))$ additions and multiplications of polynomials of length 1 (e.g. single monomials).

Let $f(t)$ be the running time of $\hat{R}(\cdot,t)$ without taking into account the time for $R(B)$. Then

$$f(t) \leq O(\ell^2(\ell+m)M(t)) + 2f(t/2),$$

where the first term is the price of multiplying two matrices that contain polynomials of length $2t$. As a corollary, $f(t) = O(\ell^2(\ell+m)M(t)\log t)$, and the lemma follows.■

Now we finish the proof of Theorem 2.1. At the beginning, the basis B has degree at most ℓn. It takes $O((\ell+m)^4 M(n)\log n)$ operations to compute the matrix $R(B,\ell n)$. It is left to apply it to B to get the reduced equivalent basis $B\cdot R(B,\ell n)$. The theorem is proved.■

Since Problems 1 and 4 are equivalent, this implies Theorem 1.1.

3. Interpolation and soft decoding of Reed-Solomon codes

3.1. Warm-up: interpolation of singletons

Before considering the general interpolation problem we show how the techniques developed in the previous section help to solve the interpolation task that arises in the original algorithm of Sudan [Sud97]: all weights are equal one and all x_i are distinct.

Assume that $x_1,...,x_n \in \mathbf{F}$ are different points and the values $y_1,...,y_n \in \mathbf{F}$ are given. The goal is to find a polynomial $Q(x,y)$ which evaluates to zero on $(x_1,y_1),...,(x_n,y_n)$ s.t. $k\deg_y(Q)+\deg_x(Q) \leq D$, where k,D are given parameters.

Lemma 3.1 *Consider the polynomials*

$$Q_0(x) = \prod_{i=1}^{n}(x-x_i), \quad Q_1(x) = \sum_{i=1}^{n} y_i \prod_{j\neq i}\frac{(x-x_j)}{(x_i-x_j)},$$

Any polynomial $Q(x,y)$ s.t. $\forall i\in[n]$ $Q(x_i,y_i)=0$ can be represented in the form $(y-Q_1(x))g(x,y)+Q_0(x)g_0(x)$, where $g_1(x,y), g_0(x)\in \mathbf{F}[x,y]$.

Proof. Consider any such polynomial Q and divide it on $y-Q_1(x)$: $Q(x,y) = (y-Q_1(x))g(x,y)+f(x)$. Since for all $i\in[n]$ $y_i - Q_1(x_i)$ equals 0, $f(x_i)=0$ for every i. This means that f is divisible on $Q_0(x)$.■

This lemma gives a way to find the solution Q. One may seek for Q in the form

$$Q = Q_0(x)\hat{g}(x) + (y-Q_1(x))(\sum_{t=0}^{n/k} g_t(x)y^t).$$

Then the condition $k\deg_y(Q)+\deg_x(Q)\leq D$ corresponds to the following optimization system:

$$\begin{gathered}
\deg(Q_0\hat{g} - Q_1 g_0) = d_0 \\
\deg(g_0 - Q_1 g_1) = d_1 \\
\ldots \\
\deg(g_{n/k} - Q_1 g_{n/k-1}) = d_{n/k} \\
\\
\max(d_0, d_1+k, d_2+2k, ..., d_{n/k}+(n/k)k) \leq D \\
\max(d_0, d_1+k, d_2+2k, ..., d_{n/k}+(n/k)k) \neq 0
\end{gathered}$$

The ith row in this system corresponds to the x-degree of monomials with y-degree equal i. The system is reducible to (1) by multiplying the ith line by $x^{k(i-1)}$. Thus by Theorem 1.1 it may be solved in time $(n/k)^{O(1)} M(n) \log n$. It is left to check that one may generate $Q_0(x)$ and $Q_1(x)$ fast.

Proposition 3.2 *For every (not necessarily different) $x_1, ..., x_n$ the polynomial $Q_0(x)$ is computable in time $O(n \log n)$. If $x_1, ... x_n$ are pairwise different then the polynomial $Q_1(x)$ is computable in time $O(M(n) \log n)$.*

The proof of the first part of Proposition 3.2 may be found in [BCS96], page 54, Exercise 2.8 (because the coefficients of $Q_0(x)$ are exactly the elementary symmetric polynomials). The second part of Proposition 3.2 is described in [BCS96], pages 75–79.

3.2. Introducing weights

In this section we prove Theorem 1.2. Let us briefly recall some basic notions in commutative algebra. We advise to read [Sud98] for more complete information on the subject.

We will work in the ring $\mathbf{F}[x, y]$ of two-variable polynomials over $\mathbf{F}$. Polynomials consist of *terms*, a term (or monomial) is an expression $ax^{d_x}y^{d_y}$, $a \in \mathbf{F}$. Denote the set of all terms by T_{xy}. In order to study ideals of the ring of multivariate polynomials, it is convenient to define an ordering on the set of all terms. A total ordering $\preceq$ on T_{xy} is called *admissible* iff

1. $t_1 \preceq t_2 \;\Rightarrow\; \forall t \in T_{xy}\; tt_1 \preceq tt_2$.
2. $\forall t \in T_{xy}\; 1 \preceq t$.

In this Section we consider the following family of admissible orderings. For k a natural parameter let $(1, k)$-*weighted degree* of term $t = ax^{d_x}y^{d_y}$ be $\deg_{(1,k)}(t) = d_x + kd_y$. For any k fix an admissible ordering $\preceq_k$, s.t.

$$\deg_{(1,k)}(t_1) < \deg_{(1,k)}(t_2) \;\Rightarrow\; t_1 \prec_k t_2$$

(if both terms have the same $(1, k)$-weighted degree then we compare them in lexicographical order). One can also define the ordering $\preceq_\infty$ that corresponds to infinite k:

$$x^{d_x^1}y^{d_y^1} \prec_\infty x^{d_x^2}y^{d_y^2} \iff \begin{cases} d_y^1 < d_y^2 \text{ or} \\ d_y^1 = d_y^2 \text{ and } d_x^1 < d_x^2 \end{cases}$$

For a fixed ordering $\preceq$, the *leading term* $LT(P)$ of a polynomial P is the maximal term according to $\preceq$. For an ideal I, $LT(I)$ is the ideal generated by the leading terms of I:

$$LT(I) = Span(LT(P)|\; P \in I).$$

A set of polynomials $Q_1, ..., Q_t$ is called the *Groebner basis* for I iff each $Q_i \in I$ and

$$Span(LT(Q_1), ..., LT(Q_t)) = LT(I).$$

The following simple statement is a useful description of the ideal in terms of its Groebner basis:

Statement 3.3 *If $Q_1, ..., Q_t$ is a Groebner basis for the ideal I then every polynomial $Q \in I$ can be written as*

$$Q = \sum g_i Q_i$$

so that $LT(Q) = LT(g_iQ_i)$ for some $i \in [1..t]$.

Finally, we say that Q is the *minimal polynomial* for an ideal I, iff $Q \in I$, and $LT(Q) = \min\{LT(Q')|Q' \in I,\ Q' \neq 0\}$.

Let us now turn to the proof of Theorem 1.2. Assume that the points $(x_i, y_i) \in \mathbf{F}^2$ with weights $\mathrm{wt}(x_i, y_i) \in \mathbf{N}$ are given. We are interested in the following ideal in $\mathbf{F}[x, y]$ which essentially corresponds to the set of all algebraic curves that go through points (x_i, y_i) at least $\mathrm{wt}(x_i, y_i)$ times:

$$I = \bigcap_{i=1}^{N} I_{\mathrm{wt}(x_i,y_i)}(x_i, y_i), \tag{6}$$

where

$$I_w(x_i, y_i) = Span\left((x - x_i)^d \cdot (y - y_i)^{w-d} | d \in [0..w]\right). \tag{7}$$

We can now reformulate the weighted curve fitting problem in terms of the minimal polynomial in I:

Problem 5 Given a tuple of N points $(x_i, y_i) \in \mathbf{F}^2$ with weights, and an integer k find a minimal polynomial in I according to the ordering $\preceq_k$.

The rest of the section will be dedicated to solving this problem. Our first goal is to describe a Groebner basis for the ideal I according to $\preceq_\infty$. Given this basis it will be easy to modify it to find the minimal polynomial for $\preceq_k$.

Definition 3.1 For S a multi-set with $|S| = w$ we say that the enumeration $L = \{s_1, ..., s_w\}$ of its elements is *balanced* iff it was generated by the following algorithm

- Draw the most frequent element in S and remove it from S
- Repeat w times.

For a balanced list $L = \{s_1, ..., s_w\}$ let the *k-shift* of L be $L^{(k)} = \{s_{k+1}, s_{k+2}, ..., s_w\}$. Denote by $d(L)$ the maximal number of equal elements in the list L (which coincides with the multiplicity of s_1).

Definition 3.2 Let n be the number of different values for x_i, and w_i be the sum of all weights over the line $x = x_i$. Assume w.l.o.g. that the coordinates $x_1, ..., x_n$ are different. Enumerate all points in every vertical line in the list $L_i = ((x_i, y_i^1), (x_i, y_i^2), ..., (x_i, y_i^{w_i}))$ so that

- Each point (x_j, y_j) is represented exactly $\mathrm{wt}(x_j, y_j)$ times in the list L_i s.t. $x_i = x_j$, $i \leq n$.
- The enumeration is balanced.

Let $w = \max_{i=1}^{n} w_i$ be the maximal weight among all vertical lines.

Theorem 3.1 *Assume that $L_i = \{(x_i, y_i^j)\}$ is a balanced enumeration of the points, let $f_1(x), ..., f_w(x)$ be a family of univariate polynomials such that $\forall j \in [1..w]$ f_j is an interpolant given by $f_j(x_i) = y_i^j$ for all $i \in [1..n]$ (see Figure 4). Let*

$$Q_t(x, y) = \prod_{j=1}^{t}(y - f_j(x)) \prod_{i=1}^{n}(x - x_i)^{d_i(t)},$$

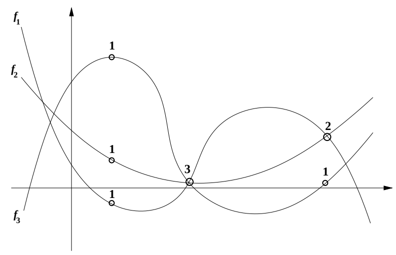

Figure 4: Construction of $f_1,\ f_2,\ f_3$ for $w = 3$

where $d_i(t) = d(L_i^{(t)})$ (in particular, $Q_0(x, y)$ is a univariate polynomial that depends on x). Then the family

$$\mathcal{G}(I) = \{Q_0(x, y), Q_1(x, y), ..., Q_w(x, y)\} \tag{8}$$

is the Groebner basis for the ideal I according to $\preceq_\infty$.

Proof. First we have to check that the polynomials $Q_0, ..., Q_w$ indeed belong to I. For that, fix some $t \in [0..w]$ and consider a point $(x_i, y_i^j) \in L_i$. Assume that d is the multiplicity of (x_i, y_i^j) in the shifted list $L_i^{(t)}$. The following is obvious in the case of continuous field (and can be checked in the straightforward way for an arbitrary field): the product $\prod_{j=1}^{t}(y - f_j(x))$ has a singularity of the order $\geq \mathrm{wt}(x_i, y_i^j) - d \geq \mathrm{wt}(x_i, y_i^j) - d_i(t)$ in the point (x_i, y_i^j), thus $Q_t \in I_{\mathrm{wt}(x_i,y_i)}(x_i, y_i)$.

It is left to show that the leading terms of $Q_0, ..., Q_w$ generate the ideal $LT(I)$. For that, fix a polynomial $Q(x, y) \in I$ with $\deg_y(LT(Q)) = t$. If $t \geq w$ then $LT(Q) \in Span(LT(Q_w))$ (because $d_i(w) = 0$ for any i), thus it is sufficient to consider the case $t < w$.

Lemma 3.4 *For $i \in [1..n]$ let $I(L_i)$ be the ideal (6) spanned on the points of L_i (e.g. on the points of the vertical line $x = x_i$). Then any polynomial $Q \in I(L_i)$ divides upon $(x - x_i)^{d_i(t)}$.*

Proof.

We use induction according to Definition 3.1. The base ($t = 0$) follows from Bezout theorem. To see the induction step, let us consider the point (x_i, y_i^1) (first in the list and of the maximal weight). Since $Q \in I_{\mathrm{wt}(x_i,y_i^1)}(x_i, y_i^1)$ we can write it as

$$Q = (y - y_i^1)Q'(x, y) + (x - x_i)^{\mathrm{wt}(y_i^1)}Q_0(x, y).$$

The second term in this sum belongs to $I(L_i)$ and satisfies the condition of the claim, thus we can subtract it from Q.

Let us decrease the weight of y_i^1 by 1 and consider the resulting ideal $I' = I(L_i^{(1)})$. Notice that $\deg_y(Q') = t - 1$ and this polynomial belongs to I'. Clearly,

$$d_1'(t-1) = d\left(\left[L_i^{(1)}\right]^{(t-1)}\right) = d(L_i^{(t)}) = d_1(t),$$

and we can apply our induction hypothesis to show that Q' divides on $(x - x_i)^{d_1'(t-1)}$. The lemma follows.■

Now we can easily finish the proof of the theorem. For any i, Q divides on $(x - x_i)^{d_i(t)}$, thus $LT(Q) \in Span(LT(Q_t))$. The theorem follows.■

Corollary 3.2 *There is an algorithm that solves Problems 3 and 5 in time* $\left(w \cdot \frac{n}{k}\right)^{O(1)} M(n) \log n$*, where* w *is the maximal overall weight assigned to a single line and* n *is the number of distinct lines.*

Proof.

Lemma 3.5 *For every* $t \leq w$ Q_t *may be generated in time* $O(w^4 M(n) \log n)$*.*

Proof. By Proposition 3.2 the product $\prod_{i=1}^{n}(x-x_i)^{d_i(t)}$ may be computed in time $O(wn\log(wn))$, each interpolant $f_j(x)$ is computable in time $M(n)\log n$, finally it takes at most t^3 of univariate polynomial multiplications to compute $\prod_{j=1}^{t}(y - f_j(x))$ each multiplication takes $M(wn)$ steps.■

Consider the minimal $Q \in I$ according to $\preceq_k$ with $\deg_y(Q) = \ell$. We can assume w.l.o.g. that $\ell \leq \frac{wn}{k}$ (because the ideal contains a univariate polynomial of x-degree wn). By Statement 3.3, we can write

$$Q = \sum_{t \leq w} g_t(x,y) \cdot Q_t(x,y) \tag{9}$$

so that the highest term of the right part doesnot cancel out, in particular for every g_t, $\deg_y(g_t) \leq \ell$. Thus one can represent g_t as the sum of $\ell + 1$ univariate polynomials, $g_t = \sum_{j=0}^{\ell} g_t^j(x) y^j$. Let us consider each $g_t^j(x)$ as an unknown variable and write the optimization problem $\deg_x(Q) + k\deg_y(Q) \to min$ in the form:

$$\begin{gathered}\forall r \in [0..\ell + w] \;\; \deg\left(x^{kr}\pi_{y_r}(\textstyle\sum g_t Q_t)\right) = d_r \\ \\ \max_{r=[0..\ell+w]}(d_r) \to min \\ \max_{r=[0..\ell+w]}(d_r) \neq 0\end{gathered} \tag{10}$$

where $\pi_{y_r}(\sum a_j(x)y^j) = a_r(x)$ is the projection on y^r. Clearly, $\pi_{y_r}(\sum g_t Q_t)$ is a linear form over $(w{+}1)(\ell{+}1)$ unknown variables g_t^j that can be computed from the input in time $(w\ell)^{O(1)} M(n)\log n$. The degree d_r in (10) is exactly the sum $k\deg_y + \deg x$ of the resulting polynomial. We have reduced the problem to the system (1), which can be solved by the algorithm of Section 2. This solves Problem 3. In the analogous way one can find the polynomial Q which is optimal for (10) subject to constraint $\deg_y(Q) \leq d_y$ for $d_y = 0, 1, ..., \ell$ and choose the minimal polynomial according to $\preceq_k$ to solve Problem 5.■

Theorem 1.2 is proved.

4. Comparison with the algorithm of Feng and Giraud

In this section we briefly compare our algorithm with that of Feng and Giraud [FG04] which works in time $O(n\log^3 n)$. Their algorithm uses a different approach of interpolation which looks in the algebraic language as follows. For the pairs $(x_1, y_i), ..., (x_n, y_n)$ denote by $I^{(i)}$ the ideal of all curves that go through the points $(x_1, y_1), ..., (x_i, y_i)$. Then we get a chain of ideals $I^{(0)} \supset I^{(1)} \supset ... \supset I^{(n)}$. For every i [FG04] iteratively construct a basis $\vec{Q}^{(i)}$ for $I^{(i)}$ using previously constructed bases. In order to perform the computation fast the divide-n-conquer paradigm is recursively used two times, hence another $\log n$ factor in the running time.

5. Acknowledgments

We are grateful to Madhu Sudan, Piotr Indyk and Venkatesan Guruswami for helpful discussions. Also, the author would like to thank Peter Buergisser for pointing out the references [Gat84, Len85].

References

[AHU74] Aho, Hopcroft, and Ullman. The Design and Analysis of Computer Algorithms. *Addison-Wesley Series in Computer Science and Information Processing, pages 302-308,* 1974.

[Ber67] E. R. Berlekamp. Factoring Polynomials over Finite Fields. *Bell System Technical Journal, 46:1853-1859,* 1967.

[BCS96] P. Bürgisser, M. Clausen and M. Shokrollahi. Algebraic Complexity Theory. *Springer Verlag Heidelberg, Number 315 of Grundlehren der mathematischen Wissenschaften*, 1996.

[F99] G. L. Feng. Two Fast Algorithms in the Sudan Decoding Procedure. In *Proceedings of 37-th Annual Allerton Conference on Communication, Control and Computing*, pp. 545-554, 1999.

[FG04] G. L. Feng and X. Giraud. Fast Algorithms in Sudan Decoding Procedure for Reed-Solomon Codes. To appear in *IEEE Transactions on Information Theory*, 2004.

[Gat84] J. von zur Gathen. Hensel and Newton methods in valuation rings. *Mathematics of Computation*, 42:637–661, 1984.

[Gur01] V. Guruswami. List decoding of error-correcting codes. *Ph.D.-thesis, MIT,* 2001.

[GI01] V. Guruswami and P. Indyk. Expander-based constructions of efficiently decodable codes. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, 2001.

[GS99] V. Guruswami and M. Sudan. Improved decoding of Reed-Solomon and algebraic-geometric codes. *IEEE Transactions on Information Theory, 45:1757-1767,* 1999.

[KV00] R. Koetter, A. Vardy. Algebraic soft-decoding of Reed-Solomon codes, *IEEE Transactions on Information Theory, submitted,* 2000.

[Len85] A.K. Lenstra. Factoring multivariate polynomials over finite fields. *Journal of Computer and System Sciences*, 30: 235–248, 1985.

[Lov86] L. Lovász. An Algorithmic Theory of Numbers, Graphs and Convexity. *Conference Board of the Mathematical Sciences, Regional Conference Series,* 1986.

[RR98] R. Roth and G. Ruckenstein. Efficient decoding of Reed-Solomon codes beyond half the minimum distance. *IEEE Transactions on Information Theory, 46:246-257,* 2000.

[Sud97] M. Sudan. Decoding of Reed-Soloman codes beyond the error-correction bound. *Journal of Complexity, 13(1):180-193,* 1997.

[Sud98] M. Sudan. Algebra and Computation. *Course notes available at* http://theory.lcs.mit.edu/~madhu, 1998.

[WB86] L.R. Welch and E.R. Berlekamp. Error correction of algebraic block codes. *US Patent Number 4,633,470,* 1986.

A. Polynomial reconstruction problem

In this section we overview the further steps to reconstruct all "close" polynomials after the weighted curve fitting was done. For the sake of completeness we provide a sketch of the fast factorization algorithm described in [RR98], [F99], [FG04].

Theorem A.1 (Guruswami, Sudan [GS99]) *Assume that Q is a solution of the weighted curve fitting with $D > \sqrt{k \sum w_i^2}$. Then for any polynomial p s.t. $\sum_{i:p(x_i)=y_i} w_i \geq D$ holds $Q(x, f(x)) \equiv 0$.*

Thus, in order to construct the required list of polynomials it is sufficient to find all roots $y = f(x)$ of the equation $Q(x, y) = 0$.

Theorem A.2 *There exists an algorithm that given a bivariate polynomial $Q(x, y)$ finds all solutions $y = f(x)$ s.t. $Q(x, f(x)) = 0$ in time $\ell^{O(1)} n \log n$, where $n = \deg_x(Q)$ and $\ell = \deg_y(Q(0, y))$.*

Proof.
The following lemma is a corollary of Roth and Ruckenstein analysis in [RR98].

Lemma A.1 *Consider the equation*

$$Q(x, y) = 0 \mod x^t, \tag{11}$$

where $y = f(x)$ is an unknown polynomial. Let $\ell = \deg_y(Q(0, y))$. Then there are at most ℓ non-intersecting sets

$$G_i = \{f_i(x) + gx^{d_i} | g \in \mathbf{F}[x]\},$$

s.t.

$$\{y | Q(x, y) = 0 \mod x^t\} = G_1 \sqcup G_2 \sqcup ... \sqcup G_k.$$

Proof. By induction on t. The base $t = 0$ is trivial. Consider the case $t > 0$. Let $\mathcal{R} = \{y_1, ..., y_k\}$, be the superset of all roots of the univariate polynomial $Q(0, y) = 0$ (assume w.l.o.g. that it has exactly ℓ roots). Clearly, any solution of (11) has the form $y_i + x\hat{y}$, where $y_i \in \mathcal{R}$ and $\hat{y} \in \mathbf{F}[x]$. So consider any y_i of multiplicity m_i and write

$$Q(x, y_i + x\hat{y}) = 0 \mod x^t \iff \frac{Q(x, y_i + x\hat{y})}{x^{r_i}} = 0 \mod x^{t-r_i},$$

where $r_i \geq 1$ is the maximal coefficient s.t. $x^{r_i} | Q(x, y_i + x\hat{y})$. Denote $Q_i(x, \hat{y}) = Q(x, y_i + x\hat{y})/x^{r_i}$. It is a crucial observation, that $\deg_{\hat{y}}(Q_i(0, \hat{y})) \leq m_i$. Too see this, write

$$Q(x, y) = (y - y_i)^{m_i} P_1(y) + xP_2(x, y),$$

so that $P_1(y_i) \neq 0$ and plug in $y = y_i + x\hat{y}$ to get

$$Q(x, \hat{y}) = Q(x, y_i + x\hat{y}) = x^{m_i} \hat{y}^{m_i} P_1(y_i + x\hat{y}) + xP_2(x, y_i + x\hat{y})$$

Any term of $\hat{y}$-degree greater than m_i has also x-degree greater than m_i in this polynomial. Thus if $\deg_{\hat{y}} Q_i(0, \hat{y}) > m_i$ then $r_i > m_i$. This is however impossible, since $Q(x, \hat{y})$ contains a term

$$P_1(y_i) x^{m_i} \hat{y}^{m_i}$$

that comes from the product $x^{m_i} \hat{y}^{m_i} P_1(y_i + x\hat{y})$ and doesn't cancel with any terms in $xP_2(x, y_i + x\hat{y})$, because the latter only contains terms with $\deg_{\hat{y}} < \deg_x$.

Algorithm $Roots(Q,t)$
Input: polynomial $Q(x,y)$, integer t
Output: The list of all solutions for $Q(x,y)=0 \mod x^t$
represented in the form $G_1 \sqcup G_2 \sqcup ... \sqcup G_\ell$ s.t. $G_i = \{y_i + gx^{d_i} | g \in \mathbf{F}[x]\}$
begin
  let $Q = Q \mod x^t$
  if $t=1$ **then return** all solutions of $Q(0,y)$ using any factorization algorithm
   for univariate polynomials
  else let $t_1 = \lfloor t/2 \rfloor$
  let $G_1, ..., G_\ell = Roots(Q, t_1)$
  for all $i = 1..\ell$ **do**
  begin
   let $r_i = \max\{r_i \text{ s.t. } x^{r_i} | Q(x, y_i + x^{d_i}\hat{y})\}$
   let $Q_i = Q(x, y_i + x^{d_i}\hat{y})/x^{r_i}$
   let $G_1^i, ..., G_\ell^i = Roots(Q_i, t - r_i)$
  end
  return all non-empty lists G_i^j
end

Figure 5: Factorization of $Q(x,y)$

We have proved that $\deg_{\hat{y}} Q_i(0,\hat{y}) \leq m_i$ and thus we can use the induction hypothesis and construct at most m_i sets G_j^i that cover all solutions of (11) of the form $y = y_i + x\hat{y}$. After we combine these lists for all $y_i \in \mathcal{R}$ we get at most ℓ solution series. The lemma follows.■

Now it is simple to finish the proof of Theorem A.2. Consider an algorithm, depicted in Figure 5. It uses a standard divide-n-conquer techniques to find all roots of the equation (11). For $t=1$ it invokes any factoring procedure to find all roots of the univariate polynomial $Q(0,y)$ in polynomial time (for example, [Ber67]). Let us estimate the time complexity of the algorithm, denote by $f(t,\ell)$ the time complexity of solving (11) provided $\ell = \deg_y(Q(0,y))$. We have a recursive bound:

$$f(t,\ell) \leq \ell^{O(1)} t + f(t/2,\ell) + \max_{a_1+a_2+...+a_\ell=\ell} \sum_{i=1}^{\ell} f(t/2, a_i),$$

which gives the desired bound $f(t,\ell) = \ell^{O(1)} t \log t$.■