Discrete Applied Mathematics 157 (2009) 812–818

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Improvements on the Johnson bound for Reed–Solomon codes

V.N. Muralidhara [*], Sandeep Sen

Department of Computer Science and Engineering, Indian Institute of Technology, New Delhi - 110016, India

ARTICLE INFO

Article history:
Received 10 October 2006
Received in revised form 16 April 2008
Accepted 9 June 2008
Available online 8 August 2008

Keywords:
Reed Solomom codes
List decodability

ABSTRACT

For Reed–Solomon codes with block length n and dimension k, the Johnson theorem states that for a Hamming ball of radius smaller than $n - \sqrt{nk}$, there can be at most $O(n^2)$ codewords. It was not known whether for larger radius, the number of codewords is polynomial. The best known list decoding algorithm for Reed–Solomon codes due to Guruswami and Sudan [Venkatesan Guruswami, Madhu Sudan, Improved decoding of Reed–Solomon and algebraic-geometry codes, IEEE Transactions on Information Theory 45 (6) (1999) 1757–1767] is also known to work in polynomial time only within this radius. In this paper we prove that when $k < \alpha n$ for any constant $0 < \alpha < 1$, we can overcome the barrier of the Johnson bound for list decoding of Reed–Solomon codes (even if the field size is exponential). More specifically in such a case, we prove that for Hamming ball of radius $n - \sqrt{nk} + c$ (for any $c > 0$) there can be at most $O(n^{c\frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2}+c+2})$ number of codewords. For any constant c, we describe a polynomial time algorithm for enumerating all of them, thereby also improving on the Guruswami–Sudan algorithm. Although the improvement is modest, this provides evidence for the first time that the $n - \sqrt{nk}$ bound is not sacrosanct for such a high rate.

We apply our method to obtain sharper bounds on a list recovery problem introduced by Guruswami and Rudra [Venkatesan Guruswami, Atri Rudra, Limits to list decoding Reed–Solomon codes, IEEE Transactions on Information Theory 52 (8) (2006) 3642–3649] where they establish super-polynomial lower bounds on the output size when the list size exceeds $\lceil \frac{n}{k} \rceil$. We show that even for larger list sizes the problem can be solved in polynomial time for certain values of k.

1. Introduction

An error-correcting code C over a finite alphabet Σ is an injective map $e : \Sigma^k \rightarrow \Sigma^n$. To transmit a message m of k symbols over a noisy channel, we send $e(m)$ of n symbols. The Hamming distance between two sequence of letters of the same length is the number of positions where two sequences differ. A good error-correcting code should have a large minimum distance d, which is defined to be the minimum Hamming distance between any two codewords in $e(\Sigma^k)$. A received message, possibly corrupted, with no more than $(d - 1)/2$ errors, corresponds to a unique codeword. Thus the original message can be recovered despite errors occur during the communication. We would like to design efficient encoding and decoding algorithms that can correct as many errors as possible.

Reed–Solomon codes are used in a wide variety of commercial applications, most prominently in CDs and DVDs, in data transmission technologies such as DSL and WiMAX, and broadcast systems such as DVB and ATTIC. Informally, Reed–Solomon code is an error-correcting code that works by oversampling a polynomial constructed from the massage. The polynomial is evaluated at several points, and these values are sent or recorded. Sampling the polynomial more often

[*] Corresponding author.
E-mail addresses: murali@cse.iitd.ernet.in (V.N. Muralidhara), ssen@cse.iitd.ernet.in (S. Sen).

0166-218X/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2008.06.014

than is necessary makes the polynomial overdetermined. As long as it receives a *certain minimum* number of the points correctly, the receiver can recover the original polynomial even in the presence of some erroneous points.

Formally, the Reed–Solomon code $[n, k]$ (denoted by $RS[n, k]$) over a finite field $\mathbb{F}_q$ (q is number of elements in the field) is the map from $\mathbb{F}_q^k \mapsto \mathbb{F}_q^n$ given by $e(a_0, a_1, \dots, a_{k-1}) = (P(x_1), P(x_2), \dots, P(x_n))$ where $P(X) = \sum_{i=0}^{k-1} a_i X^i$. Here $x_1, x_2, \dots, x_n$ are distinct points in $\mathbb{F}_q$. Since any two different polynomials with degree $k - 1$ agree at at most $k - 1$ points, the minimum distance of the Reed–Solomon code is $n - k + 1$. A Hamming ball with radius less than half of the minimum distance contains at most one codeword. Finding the codeword is called unambiguous decoding and it can be efficiently solved by a simple algorithm [2].

1.1. List decoding

If we gradually increase the radius, there may be two or more codewords lying in some Hamming balls. Can we efficiently enumerate all the codewords in any Hamming ball of certain radius? This is the so called list decoding problem and was introduced independently by Elias [3] and Wozencraft [18].

In other words, list decoding is a relaxation of the usual decoding procedure, wherein the decoder is allowed to output a list of possible original messages. This relaxation is required to get around *half the distance barrier* for unique decoding. Apart from being able to correct more errors (than the case of unique decoding), algorithms for list decoding of various codes have played a central role in a variety of results in complexity theory. Some important applications are construction of hardcore predicates from one-way permutations, amplifying hardness of boolean functions and construction of extractors [17].

The list decoding problem of Reed–Solomon codes can also be reformulated as the problem of curve fitting or polynomial reconstruction. In this problem, we are given n distinct points $(x_1, x_2, \dots, x_n)$ in $\mathbb{F}_q$ and another n points (which need not be distinct) $(y_1, y_2, \dots, y_n)$ in $\mathbb{F}_q$. The goal is to find polynomials $P(x)$ of degree at most $k - 1$ that agree at at least t points (i.e. $P(x_i) = y_i$) for at least t points. It is easy to see the connection between the problem of list decoding Reed–Solomon code and the polynomial reconstruction problem. Finding all polynomials that agree at t points is equivalent to enumerating all the codewords in any Hamming ball of radius $n - t$. In the rest of the paper, t will be used to denote the number of agreements.

We have the following two fundamental questions related to list decoding Reed–Solomon codes.

1. (Combinatorial) What is the maximum value of t for which the number of codewords in a Hamming ball of radius $n - t$ is polynomially bounded?
2. (Algorithmic) What is the maximum value of t for which we can enumerate all of codewords in any Hamming ball of radius $n - t$ in polynomial time?

We note that the bound for algorithmic version of the problem is greater than or equal to the combinatorial version of the problem.

Since for $t \leq k$ it is possible that there are exponentially many solutions (there will be exactly $\binom{n}{t}$ solutions), we will focus on $t > k$ in the remaining paper.

1.2. Previous related results

The Johnson bound for Reed–Solomon codes [6,4] states that there can be at most $O(n^2)$ codewords in any Hamming ball of radius smaller than $n - \sqrt{nk}$. It is not known whether the number of codewords in a Hamming ball of radius greater than $n - \sqrt{nk}$ is polynomially bounded. The *tightness* of the Johnson bound will imply that the previous property doesn't hold.

In [9], the authors study the following question: Given n pairs of distinct points $((x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)) \in \mathbb{F}_q^2$, find polynomials of degree at most $k - 1$ that agree at at least t points. Note that their construction uses $(x_1, x_2, \dots, x_n)$ that are not distinct, unlike Reed–Solomon codes. They show that there exists a received codeword for which there are *super-polynomial* ($\omega(n^c)$ for any constant c) number of polynomials of degree at most $k - 1$ for $t < \sqrt{(1 - \epsilon)nk}$. Extending this result to distinct $x_i's$ would imply that Johnson bound is (nearly) tight for Reed–Solomon codes. In the same paper, the authors construct Reed–Solomon codes (distinct x_i's) with $k = n^\delta$ for some $\delta > 0$, and exhibit an explicit received codeword r with a super-polynomial number of codewords that agree with it on $(2 - \epsilon)k$ points, for any $\epsilon > 0$. This is improved in a recent work by Ben-Sasson, Kopparty and Radhakrishnan [7], who show that for any $\delta \in (0, 1)$, there exist codes $RS[n, k = n^\delta]$ which have super-polynomial codewords with agreement $n^{\sqrt{\delta}}$. (Note that $2n^\delta < n^{\sqrt{\delta}}$ for small enough δ.) This still leaves a fairly large gap between the bounds for distinct and non-distinct x_i's and the leaves open the question of the tightness of the Johnson bound for Reed–Solomon codes.

Algorithmically there was little progress on this problem for radius slightly larger than half of the minimum distance, until Sudan [5] that was subsequently improved by Guruswami and Sudan [10]. The latter paper solves the list decoding problem for radius as large as $n - \sqrt{nk}$ ($t > \sqrt{nk}$). Note that this matches the Johnson bound and in the remaining paper, we will refer to this as the **GS** algorithm. This surprising coincidence has led many researchers to conjecture that the Johnson bound is tight. In particular, there may be super-polynomially many codewords in some Hamming ball of radius exceeding $n - \sqrt{nk}$.

In [1], the authors have proved that the list decoding cannot be done for $t < g(n, k, q)$, otherwise the discrete logarithm over $F_{q^{g(n,k,q)-k}}$ is easy, where $g(n, k, q)$ is the smallest positive integer g^* such that $\binom{n}{g^*}/q^{g*-k} < 1$ and they have proved that such a g^* lies in $[k, \sqrt{nk}]$. While this gives strong evidence that it may be hard to list decode for $t < g^*$, the possible gap between the values of g^* and $\sqrt{nk}$ leaves open the question of the tightness of the Johnson bound (algorithmic version). We note that their reduction is a probabilistic reduction, which means that an algorithm for the list decoding Reed–Solomon codes with agreement smaller than g^* would lead to a randomized algorithm for a discrete logarithm over finite fields.

The Maximum-Likelihood Decoding Problem of Reed–Solomon codes is: Given a integer $w > 0$ and a target vector $y \in F_q^n$, is there a codeword c such that the Hamming distance between c and y is at most w? In [12] Guruswami and Vardy show the existence of Reed–Solomon codes for which the Maximum-Likelihood Decoding Problem with $w = k + 1$ is NP-complete. The specific class of Reed–Solomon codes that they construct have an exponential field size.

In a restricted scenario where errors occur in a *synchronized* fashion over $M - 1$ consecutive codewords (errors occur in same positions), Parvaresh and Vardy [15] show that one can achieve a much larger list decoding radius, viz., $n(1-(\frac{kM}{n})^{\frac{M-1}{M}})$. In [16], the authors show that by departing from the conventional Reed–Solomon codes, one can obtain $n(1 - (\frac{kM}{n})^{\frac{M-1}{M}})$ error correction bounds in the worst case. Building on this, Guruswami and Rudra [13] have constructed a code that they call folded Reed–Solomon codes, that achieves (nearly) optimal $(n(1 - \epsilon) - k)$ list decoding radius.

1.3. Our results

We prove that when $k < \alpha n$ for any constant $0 < \alpha < 1$ then the Johnson bound is not tight for Reed–Solomon codes. More specifically, for any $c > 0$, we prove that a Hamming ball of radius $n - \sqrt{nk} + c$ (note that this exceeds the Johnson bound), there can be at most $O(n^{c\frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2}+c+2})$ number of codewords. Our proof is constructive leading to a polynomial time algorithm to enumerate all of them. The reader may note that there is a trivial algorithm for increasing the Guruswami–Sudan radius by any additive constant c. That is, we can guess the correct values for $\binom{n}{c}$ positions before we apply the Guruswami–Sudan algorithm. This will work in polynomial time *if the field size is polynomial*. Our result only uses the fact that the field operations take polynomial time (and therefore can handle exponential size fields).

Therefore under a very mild assumption we are able to improve the Johnson bound as well as improve the algorithmic bound of **GS** [10]. There could still be a large gap between g^* and $\sqrt{nk} - c$ for any constant $c > 0$.

Using the above method we obtain sharper bounds for the list recovery problem (introduced by Guruswami and Indyk [8] and defined formally in Section 2). Guruswami and Rudra [9] establish super-polynomial lower bounds on the output size when the list size exceeds $\lceil \frac{n}{k} \rceil$. We show that even for larger list sizes the problem can be solved in polynomial time for certain values of k.

2. Breaking through the barrier of $n - \sqrt{nk}$ for Reed–Solomon code

Let $\mathbb{F}_q$ be a field with q elements and t be an integer such that $\sqrt{nk} - 1 < t \le \sqrt{nk}$. We recall that the list decoding problem of Reed–Solomon codes can be reformulated into the problem of curve fitting or polynomial reconstruction on *distinct* points. In this problem, we are given n distinct pair of points $((x_1, y_1), (x_2, y_2), \dots, (x_n, y_n))$ in $\mathbb{F}_q^2$. The goal is to find polynomials $P(X)$ of degree at most $k - 1$ such that $P(x_i) = y_i$ for at least t points. The **GS** [10] solves the more general problem for $t < \sqrt{nk}$.

Let us sketch the idea of our construction before we provide the formal details. Given n distinct points $(x_1, x_2, \dots, x_n)$ in $\mathbb{F}_q$ and another n points $(y_1, y_2, \dots, y_n)$ in $\mathbb{F}_q$, we try to solve the polynomial reconstruction problem with input $x_2, x_3, \dots, x_n$, $\frac{(y_2-y_1)}{(x_2-x_1)}, \frac{(y_3-y_1)}{(x_3-x_1)}, \dots, \frac{(y_n-y_1)}{(x_n-x_1)}$. That is, we wish to find all polynomials $P_1(X)$ of degree at most $k-2$ such that $P_1(x_i) = \frac{(y_i-y_1)}{(x_i-x_1)}$ for least $t-1$ (of $n-1$) points. Similarly we can define polynomials $P_i(X)$'s of degree at most $k-2$ such that $P_i(x_j) = \frac{(y_j-y_i)}{(x_j-x_i)}$ for least $t - 1$ (of $n - 1$) points other than x_i. In other words, given an (n, k, t) instance of the problem we reduce it to an $(n - 1, k - 1, t - 1)$ instance of the problem. This we call as one step in the reduction process.

We note that the distinctness of x_i's is not crucial for our reduction step. Suppose x_1 appears l_1 times, $(x_1, y_1), \dots (x_1, y_{l_1})$. Then, when we try polynomials that pass through (x_1, y_1), the points $(x_1, y_2), \dots, (x_1, y_{l_1})$ are already eliminated. So, we have a problem with $l_1 > 0$ fewer points, degree $= k-1$ and still needing agreement $t-1$. In other words, given an (n, k, t) instance of the problem we reduce it to an $(n-l_1, k-1, t-1)$ instance of the problem. In the rest of this section l_i's denote the number of times x_i's appear in the input.

In next lemma, we prove that number of polynomials $P_i(X)$ of degree at most $k - 2$ such that $P_i(x_j) = \frac{(y_j-y_i)}{(x_j-x_i)}$ for least $t - 1$ (of $n - 1$) points other than x_i is same as the number of polynomials $P(X)$ of degree at most $k - 1$ such that $P(x_j) = y_j$ for least t of $n1$ points. In other words, our reduction preserves the list size.

Lemma 2.1. *There is a polynomial $P(X)$ of degree at most $k - 1$ such that $P(x_i) = y_i$ for at least $t - 1$ points and $P(x_1) = y_1$ iff there is a polynomial $P_1(X)$ of degree at most $k - 2$ such that $P_1(x_i) = \frac{(y_i-y_1)}{(x_i-x_1)}$ for at least $t - 1$ points.*

Proof. Let $P(X)$ be a polynomial of degree at most $k-1$ such that $P(x_i) = y_i$, $i > 1$, for at least $t-1$ points and $P(x_1) = y_1$. Let $P_1(X)$ be the polynomial obtained by dividing $P(X)$ by $(X-x_i)$. Since $P(x_1) = y_1$ it follows that $P(X) = P_1(X)(X-x_1)+y_1$.

Clearly $P_1(X)$ is a polynomial of degree at most $k-2$ and for any i if $P(x_i) = y_i$ then for such an i we note that $P_1(x_i) = \frac{(y_i-y_1)}{(x_i-x_1)}$. By the choice of $P(X)$, $P(x_i) = y_i$ for at least $t-1$ points. This implies that $P_1(x_i) = \frac{(y_i-y_1)}{(x_i-x_1)}$ for at least $t-1$ points.

The converse follows similarly by multiplying $P_1(X)$ by $(X - x_1)$ and adding y_1. ■

Let $Poly = \{P(X) \in \mathbb{F}_q[X] \mid \deg(P(X)) \le k-1, P(x_i) = y_i$ for at least t points$\}$. That is, $Poly$ is the set of polynomials of degree at most $k-1$ which agrees at at least t of the n points. Let $Poly(i) = \{P_i(X) \in \mathbb{F}_q[X] \mid \deg(P_i(X)) \le k-2, P_i(x_j) = \frac{y_j-y_i}{x_j-x_i}, i \neq j$ for at least $t-1$ points$\}$. The following lemma proves that $Poly = \cup_{i=1}^{n}\{P_i(X)(X-x_i) + y_i \mid P_i(X) \in Poly(i)\}$.

Lemma 2.2. *By solving all the $(n-l_i, k-1, t-1)$ instances for computing $Poly(i)$ $1 \le i \le n$, we can solve the (n, k, t) instance for computing Poly.*

Proof. Any polynomial in the set $Poly$ must have $P(x_i) = y_i$ for some i and hence can be recovered from $Poly(i)$ by using the reduction described above. ■

We can solve the $(n-l_i, k-1, t-1)$ instance of the problem using the **GS** [10] algorithm provided $t-1 > \sqrt{(n-l_i)(k-1)}$. If $t-1 < \sqrt{(n-l_i)(k-1)}$ we apply the reduction again until we can solve the problem by using the **GS** algorithm. That is, we apply the reduction s times, such that $t-s > \sqrt{(n-s)(k-s)}$. Now by solving n^s number of $(n-s, k-s, t-s)$ instances of the problem using the **GS** algorithm we can solve the (n, k, t) instance of the problem. In the following theorem we give an upper bound on the number of reduction steps required to achieve such an s.

Theorem 2.3. *If $k < \alpha n$ for some $0 < \alpha < 1$ and $\sqrt{nk} - 1 < t \le \sqrt{nk}$ then the number of polynomials of degree at most $k-1$ agreeing at at least t points is at most $O(n^{\frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2}+3})$ and we can enumerate all of them in $O(n^{\frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2}+1+10})$ time.*

Proof. If $k < \alpha n$ then we apply the reduction $s = \frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2} + 1$ times:

$$
\begin{aligned}
& s > \frac{2\sqrt{\alpha} - 1/n}{(1-\sqrt{\alpha})^2} > \frac{2\sqrt{nk} - 1}{n + k - 2\sqrt{nk}}. \\
& \Longrightarrow \sqrt{nk} < \frac{n+k}{2} - \frac{\sqrt{nk}}{s} + \frac{1}{2s}. \\
& \Longrightarrow t < \frac{n+k}{2} - \frac{\sqrt{nk}}{s} + \frac{1}{2s} \quad \text{as } t \le \sqrt{nk} \\
& \Longrightarrow -2st > 2\sqrt{nk} - 1 - s(n+k) \\
& \Longrightarrow -2st > nk - t^2 - s(n+k) \quad \text{as } nk - t^2 < 2\sqrt{nk} - 1 (\text{as } t > \sqrt{nk} - 1) \\
& \Longrightarrow (t-s)^2 > nk - s(n+k) + s^2 \\
& \Longrightarrow (t-s) > \sqrt{(n-s)(k-s)}.
\end{aligned}
$$

Hence we can solve $(n-s, k-s, t-s)$ instances of the problem by using the **GS** algorithm. So we can solve the polynomial reconstruction problem for $\sqrt{nk} - 1 < t \le \sqrt{nk}$ by applying the **GS** algorithm $O(n^{\frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2}+1})$ times. Since the **GS** algorithm takes $O(n^{10})$ time [14,11] and returns a list of size $O(n^2)$, the theorem follows. Note that for $k \le \alpha n$, the same value of s suffices. ■

We recall [7], in which the authors prove that for any $\delta \in (0, 1)$, there exist codes $RS[n, k = n^{\delta}]$ which have super-polynomial number of codewords with agreement $n^{\sqrt{\delta}}$. We now prove that this cannot happen for codewords with agreement $\sqrt{nk} - c < t \le \sqrt{nk} - c + 1$.

Theorem 2.4. *If $k < \alpha n$ for some $0 < \alpha < 1$ and $\sqrt{nk} - c < t \le \sqrt{nk} - c + 1$ for any $c > 0$, then the number of polynomials of degree at most $k-1$ agreeing at at least t points is at most $O(n^{c\frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2}+c+2})$ and we can enumerate all of them in $O(n^{c\frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2}+c+10})$ time. In particular, for any constant c, the running time is polynomial.*

Proof. We prove this by induction on c. Theorem 2.3 proves the result for the base case when $c = 1$. Now inductively we assume that we can list decode for $\sqrt{nk} - c < t \le \sqrt{nk} - c + 1$ and prove for $\sqrt{nk} - c - 1 < t \le \sqrt{nk} - c$. We apply the same technique that we used in proving Theorem 2.3. We apply reduction s times such the we can solve the problem inductively. That is we can solve by induction provided $(t-s) > \sqrt{(n-s)(k-s)} - c$.

If $k < \alpha n$ then we apply the reduction $s = \frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2} + 1$ times:

$$
\begin{aligned}
& s > \frac{2\sqrt{\alpha} - 1/n}{(1-\sqrt{\alpha})^2} > \frac{2\sqrt{nk} - 1}{n + k - 2\sqrt{nk}}. \\
& \Longrightarrow \sqrt{nk} < \frac{n+k}{2} - \frac{\sqrt{nk}}{s} + \frac{1}{2s}. \\
& \Longrightarrow t + c < \frac{n+k}{2} - \frac{\sqrt{nk}}{s} + \frac{1}{2s} \quad \text{as } t + c < \sqrt{nk} \\
& \Longrightarrow -2s(t+c) > 2\sqrt{nk} - 1 - s(n+k) \\
& \Longrightarrow -2s(t+c) > nk - (t+c)^2 - s(n+k) \quad \text{as } nk - (t+c)^2 < 2\sqrt{nk} - 1 (\text{as } t + c > \sqrt{nk} - 1) \\
& \Longrightarrow (t + c - s)^2 > nk - s(n+k) + s^2 \\
& \Longrightarrow (t - s) > \sqrt{(n-s)(k-s)} - c.
\end{aligned}
$$

Now the $(n-s, k-s, t-s)$ instance of the problem can be solved inductively. We note that we make $n^{\frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2}+1}$ calls to the inductive step. Hence the total number of polynomials of degree at most $k-1$ agreeing at at least t points is at most $O(n^{c\frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2}+c+2})$ and we can enumerate all them in $O(n^{c\frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2}+c+10})$ time. Clearly, the same s suffices for $k < \alpha n$. Note that $\frac{k-1}{n-1} < \frac{k}{n} < \alpha$; hence $s = \frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2} + 1$ is the upper bound for the number of reductions to be applied in the induction step. ■

Corollary 2.5. *Let $k = n^{\delta}$ for any $\delta > 0$ and $n^{1-\delta} > 16$. If $\sqrt{nk} - 1 < t \leq \sqrt{nk}$ then the number of polynomials of degree at most $k-1$ agreeing at at least t points is at most $O(n^3)$ and we can enumerate all them in $O(n^{11})$ time.*

Proof. We want to prove that $s = 1$ in Theorem 2.3 will work in this case.

$$
\begin{aligned}
& n^{1-\delta} > 16 \Longrightarrow 4n^{\frac{1+\delta}{2}} < n < n + n^{\delta} + 1 \\
& \Longrightarrow 4\sqrt{nk} < n + k + 1 \\
& \Longrightarrow \sqrt{nk} < \frac{n+k+1}{2} - \sqrt{nk} \\
& \Longrightarrow t < \sqrt{nk} < \frac{n+k+1}{2} - \sqrt{nk} \\
& \Longrightarrow t - 1 > \sqrt{(n-1)(k-1)} \quad \text{(This step follows by using } s = 1\text{in the proof of Theorem 2.3.).}
\end{aligned}
$$

By applying the reduction n times we can solve this problem using the **GS** algorithm. Hence then number of polynomials of degree at most $k-1$ agreeing at at least t points is at most $O(n^3)$ and we can enumerate all of them in $O(n^{11})$ time. ■

Corollary 2.6. *Let $k = n^{\delta}$ for any $\delta > 0$ and $n^{1-\delta} > 16$. If $\sqrt{nk} - c < t \leq \sqrt{nk} - c + 1$ then the number of polynomials of degree at most $k-1$ agreeing at at least t points is at most $O(n^{c+2})$ and we can enumerate all of them in $O(n^{c+10})$ time.*

The proof follows from induction (the same as in Theorem 2.4).

3. List recovery

We recall the list recovery problem as introduced by Guruswami and Indyk [8]: We say that a code C is α, ℓ, L $(\alpha \leq 1)$ recoverable if for any given a set $S_i \subseteq F_q$ of possible symbols, $1 \leq i \leq n$, where each S_i has at most ℓ elements and number of codewords $\langle c_1, c_2, \ldots, c_n \rangle$, such that $c_i \in S_i$ for at least αn symbols, is at most L.

Now we consider the special case when $\alpha = 1$, i.e. we are given a set $S_i \in F_q$ (of size ℓ) of possible symbols, $1 \leq i \leq n$, and the goal is to output all codewords $\langle c_1, c_2, \ldots, c_n \rangle$ such that $c_i \in S_i$, $\forall i$. The **GS** [10] algorithm of list decoding Reed–Solomon codes can be used to solve the list recovery problem as long as the input list size ℓ is less than $\lceil \frac{n}{k} \rceil$.

In [9], Guruswami and Rudra demonstrate that this latter performance is the best possible with surprising accuracy. Specifically, they have shown that when $l = \lceil \frac{n}{k} \rceil$, there are instances of the problem for which the list of output polynomials will be super-polynomially large in n. More specifically they proved the following result.

For every prime power $q \geq 3$ and integer $m > 1$ and for the choice $n = q^m$ and $k = \frac{q^m - 1}{q-1}$ (note that $\lceil \frac{n}{k} \rceil = q$), there are Reed–Solomon codes of dimension $k+1$ and block length n, such that list recovering with $\ell = \lceil \frac{n}{k} \rceil$ results in super-polynomial size output.

Note that in the above result $k \gg \ell$- below we show that when k and ℓ have comparable values, one can list recover in polynomial time even when $\ell > \lceil \frac{n}{k} \rceil$.

Theorem 3.1. *Let $l = \frac{n}{k} + \Delta$ where $\Delta > 0$. The list recovery problem with input lists of size ℓ can be solved in $O(n^{\gamma+c})$ time for $k^2 \leq \frac{nc}{\Delta}$ for any $c > 0$ where γ is a constant.*

Proof. We apply our reduction method to the list recovery problem. After fixing s points, we obtain a smaller problem with $n - s$ sets each with ℓ elements. So, if $\ell < \frac{n-s}{k-s}$, that is, $s > \frac{k\ell - n}{\ell - 1}$, then we can list the polynomials using the **GS** algorithm.

When $\ell = \frac{n}{k} + \Delta$ the list recovering problem with input lists of size ℓ can be solved in polynomial time provided

$$s > \frac{k\ell - n}{\ell - 1} = \frac{k\left(\frac{n}{k} + \Delta\right) - n}{\frac{n}{k} + \Delta - 1} = \frac{k^2\Delta}{n + k(\Delta - 1)}.$$

But we have $k^2 \leq \frac{nc}{\Delta}$, which implies $c \geq \frac{k^2\Delta}{n} > \frac{k^2\Delta}{n+k(\Delta-1)}$. Now we can choose s such that $c \leq s < c + 1$ which implies $s \geq \frac{k^2\Delta}{n} > \frac{k^2\Delta}{n+k(\Delta-1)}$.

Since we have to solve $\binom{n}{s}$ instances of the problem, and each instance is solved by applying the **GS** algorithm which takes $O((nl)^{10})$ time, since $\binom{n}{s<n^s<n^{c+1}}$ and $l < n$ the result follows. ■

Using the above result we can improve the bounds for a special class (n, ℓ) polynomial reconstruction problem where each (of n) x_i's repeats exactly ℓ times.

Corollary 3.2. *If $k^2 \leq \frac{nc}{\Delta}$ for any $c > 0$, $\Delta > 0$ then the (n, ℓ) polynomial reconstruction problem can be solved for agreement parameter t, $t > \sqrt{nk} - \frac{k\Delta}{2}$, in $O(n^{\gamma+c})$ time where γ is a constant.*

Proof. Using the notation of Guruswami and Rudra in [9] (page 6), and using $\ell = \frac{n}{k} + \Delta$, $\sqrt{nk(\frac{n}{k} + \Delta)} - \frac{k\Delta}{2} = \sqrt{n^2 + nk\Delta} - \frac{k\Delta}{2} < n + \frac{k\Delta}{2} - \frac{k\Delta}{2} = n = t$. The result follows from Theorem 3.1. ■

4. Concluding remarks

In this paper, we have shown that for $k \leq \alpha n$ for any constant $0 < \alpha < 1$, for Reed–Solomon codes a Hamming ball of radius $n - \sqrt{nk} + c$, (for any $c > 0$) contains at most $O(n^{c\frac{2\sqrt{\alpha}}{(1-\sqrt{\alpha})^2}+c+2})$ number of codewords. We also describe a polynomial time algorithm for enumerating all of them, thereby also improving on the Guruswami–Sudan algorithm. We note that the number of codewords grows very rapidly with α and in particular for $\alpha = 1 - o(1)$, our construction fails to provide a polynomial bound.

The most obvious question is that of whether one can obtain more significant improvement of the Johnson bound. For example, can we do it when $c = n^\delta$ or $c = \log n$ or for any non-constant c?

Even if this was possible we may not be able to match it algorithmically, because of the reduction to a discrete logarithm problem [1]. The recent results in [9,7] show the existence of super-polynomial size lists for some received message but their construction works when $n = q$.

Another significant direction to explore is relaxing the Reed–Solomon decodability for *any* evaluation point $(x_1, x_2, \ldots, x_n)\mathbb{F}_q^n$. Even if we can find one evaluation point for which the decoding radius is significantly larger, it will suffice for our purpose. In this context, we may want to analyse the situation for a random evaluation tuple.

Acknowledgments

We would like to thank the anonymous reviewer of a previous version of our paper for pointing out the connection between list recovery and our reduction. We thank Madhu Sudan who pointed out that if the field size is polynomial, constant improvement in listing radius for any code can be achieved by guessing the correct values for $\binom{n}{c}$ positions.

Part of the second author's work was done when the second author was in the Department of CSE, IIT Kharagpur, India.

References

[1] Qi Cheng, Daqing Wan, On the list and bounded distance decodability of Reed–Solomon codes (extended abstract), in: FOCS, 2004, pp. 335–341.
[2] L. Welch, E. Berlekamp, Error correction of algebraic block codes. US Patent Number 4633470, 1986. A description also appears in Peter Gemmell and Madhu Sudan, Highly Resilient Correctors for Polynomials Information Processing Letters 43 (4) (1992) 169–174.
[3] Peter Elias, List decoding for noisy channels, in: 1957-IRE WESCON Convention Record, 1957, pp. 94–104.
[4] O. Goldreich, R. Rubinfeld, M. Sudan, Learning polynomials with queries: The highly noisy case, SIAM Journal on Discrete Mathematics (2000).
[5] Madhu Sudan, Decoding Reed–Solomon codes beyond the error-correction bound, Journal of Complexity 13 (1) (1997) 180–193.
[6] Madhu Sudan, Lecture Notes on Algorithmic Introduction to Coding Theory, 2002 (Chapter 12).
[7] Eli Ben-Sasson, Swastik Kopparty, Jaikumar Radhakrishnan, Subspace polynomials and list decoding of Reed–Solomon codes, in: FOCS, 2006, pp. 207–216.
[8] Venkatesan Guruswami, Piotr Indyk, Expander-based constructions of efficiently decodable codes, in: FOCS, 2001, pp. 658–667.
[9] Venkatesan Guruswami, Atri Rudra, Limits to list decoding Reed–Solomon codes, IEEE Transactions on Information Theory 52 (8) (2006) 3642–3649.

[10] Venkatesan Guruswami, Madhu Sudan, Improved decoding of Reed–Solomon and algebraic-geometry codes, IEEE Transactions on Information Theory 45 (6) (1999) 1757–1767.

[11] Venkatesan Guruswami, Madhu Sudan, Reflections on improved decoding of Reed–Solomon and algebraic-geometric codes, IEEE Information Theory Newsletter (2002).

[12] Venkatesan Guruswami, Alexander Vardy, Maximum-likelihood decoding of Reed–Solomon codes is NP-hard, IEEE Transactions on Information Theory 51 (7) (2005) 2249–2256.

[13] Venkatesan Guruswami, Atri Rudra, Explicit capacity-achieving list-decodable codes, in: STOC, 2006, pp. 1–10.

[14] Venkatesan Guruswami, List Decoding of Error-Correcting Codes, in: Lecture Notes in Computer Science, vol. 3282, Springer-Verlag, ISBN: 3-540-24051-9, 2005, p. 350.

[15] F. Parvaresh, A. Vardy, Multivariate interpolation decoding beyond Guruswami–Sudan radius, in: 42nd Annual Allerton Conference on Communication Control and Computing, Urbana, IL, October 2004.

[16] F. Parvaresh, A. Vardy, Correcting errors beyond the Guruswami–Sudan radius in polynomial time, in: FOCS, 2005, pp. 285–294.

[17] Madhu Sudan, List decoding: Algorithms and applications, in: J. van Leeuwen, O. Watanabe, M. Hagiya, P.D. Mosses, T. Ito (Eds.), Lecture Notes in Computer Science, vol. 1872, Springer, August 2000, pp. 25–41.

[18] John M. Wozencraft, List decoding, Quarterly Progress Report, Research Laboratory of Electronics, MIT, 1958, 48 pp. 90–95.