The decoding Library for List Decoding

Guillaume Quintin

▶ **To cite this version:**

Guillaume Quintin. The decoding Library for List Decoding. International Symposium on Symbolic and Algebraic Computation, Jul 2012, Grenoble, France. pp.168-170, 10.1145/2429135.2429174 . hal-00700397v2

HAL Id: hal-00700397
https://inria.hal.science/hal-00700397v2

Submitted on 6 Jun 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

The DECODING Library for List Decoding

Guillaume Quintin
Laboratoire d'informatique (LIX)
École polytechnique
91128 Palaiseau Cedex
France
quintin@lix.polytechnique.fr

1 Introduction and motivation

Reed-Solomon (RS) codes form an important and well-studied family of codes. They were first proposed in 1960 by Reed and Solomon in their original paper [RS60]. They are widely used in practice [WB99]. RS codes can be efficiently unique decoded [Gao02] and [Jus76]. Sudan's 1997 breakthrough on list decoding of RS codes [Sud97], further improved by Guruswami and Sudan in [GS98], showed that RS codes are list decodable up to the Johnson bound in polynomial time. DECODING is a C library whose main goal is to implement as efficiently as possible the Guruswami-Sudan algorithm. It is written in C89 and is stand-alone.

1.1 The Guruswami-Sudan algorithm

The DECODING library is devoted to algorithms concerning the Guruswami-Sudan list decoding scheme and does not limit itself to finite fields as it is often the case in coding theory. Let us fix a finite ring with identity A not necessarily commutative. The Guruswami-Sudan algorithm has two main steps. The first one (interpolation step) consists in finding a "curve" of equation $Q(X, Y) = 0$ in A^2 which passes through given points with certain multiplicities. The second step (root-finding) finds the roots of $Q(X, Y)$ seen in $(A[X])[Y]$.

The interpolation step dominates the cost of the whole Guruswami-Sudan algorithm. Many methods have been proposed but without any available implementations or comparisons to other ones. A few timings of the Guruswami-Sudan algorithm can be found, but again without the corresponding implementation.

2 The implementation

To the knowledge of the author no implementation of the Guruswami-Sudan algorithm has been proposed. The only available implementation is constituted by a set of C++ functions, not directly accessible, inside PERCY++ [Gol07] whose purpose is not error correction and which does not use fast algorithms for dense bivariate polynomials.

2.1 The algorithms provided by decoding

The implemented algorithm for interpolation is a variant of the Koetter algorithm [McE03] in the `include/decoding/algos/koetter.c` file. It uses polynomial arithmetic with fast bivariate shifting (computation of $Q(X + x_0, Y + y_0)$ where $(x_0, y_0) \in A^2$) in `include/decoding/algos/dbpol_shift_fast.c` and fast univariate shifting (computation of $f(X + x_0)$ where $f \in A[X]$ and $x_0 \in A$) in `include/decoding/algos/upol_shift_fast.c`. Specific variants of these algorithms for commutative rings of characteristic 2 are also present in the same files.

The second step (root-finding) implemented in the library is a variant of the Roth and Ruckenstein algorithm [RR98] and the naive algorithm of [BLQ11]. It is in `include/decoding/algos/dbpol_Xroots.c`.

2.2 The design of decoding

The DECODING library is designed to be easy to use in a C or C++ program. One of its particularities is to use the C preprocessor to generate algorithms for a ring (generally a finite field) which must be provided by the end-user. Therefore efficient libraries like MPFQ [GT06] can be used by the end-user. For the sake of completeness, some finite fields are provided by default.

Error correcting code are often regarded over finite fields, in particular $\mathbb{F}_2$, together with the classical Hamming distance. But other distances, like the Lee distance, are better suited for some applications. Usually the Lee distance is needed for codes over Galois rings. Error correcting codes over the ring of matrices over a finite field or a finite commutative ring are also considered for example in [OSB12]. Therefore DECODING proposes generic algorithms whenever possible. This flexibility is needed when studying codes over Galois rings for example where the end-user needs to manipulate codes over a Galois ring and its residue field at the same time.

Although DECODING proposes certain fast bivariate polynomial algorithms, it is not its goal to propose fast algorithms for univariate and bivariate polynomial multiplication. In fact, DECODING is designed to be used in conjunction with other efficient libraries like GMP [Gra91], NTL [Sho90] or FLINT [Har10]. For the sake of completeness DECODING provides these algorithms in their "schoolbook" form but it is recommended, for efficiency, to use external libraries.

A very simple mechanism using the C preprocessor allows one to override the default generic algorithms proposed by DECODING. For example see at the `include/decoding/rings/GF5.c` file which implements the finite field $\mathbb{F}_5$. It shows how to replace the univariate polynomial root finding over $\mathbb{F}_5$. All C macros that control this mechanism are in the `include/decoding/ring_reset.h` file.

3 Presentation

The DECODING library is the first library which proposes a flexible and efficient way to implement algorithms related to the Guruswami-Sudan decoding scheme. It can be used with efficient external libraries to obtain more efficient implementations of Guruswami-Sudan related algorithms.

I will first present quickly the history of the Guruswami-Sudan algorithm and show that it needs dense bivariate polynomials only available, not necessarily directly, in computer algebra systems such as MAGMA [BCP97] or MATHEMAGIX [H$^+$02]. As error correcting codes are often used over binary fields, dedicated fast algorithms must be used. The bivariate polynomials appearing in the list decoding algorithms can have large degrees even when RS codes with small parameters are considered. Hence fast algorithms are needed.

I will then present the flexibility of DECODING, needed to obtain efficient algorithms over several finite rings and fields.

- It is easy to replace a key algorithm, such as univariate polynomial multiplication or univariate polynomial root-finding, by a very efficient one provided by an external library such as FLINT or NTL for example.

- It is easy to choose a finite ring or a finite field, or even to use different rings at the same time in order to implement algorithms related to RS codes over Galois rings. The DECODING library is not restricted to mathematical object whose binary representation holds in a single machine word. It requires no supplementary efforts to use, for example, multiple precision integers from GMP or large binary fields from MPFQ.

Finally, I will present the provided algorithms concerning dense bivariate polynomials and their applications to list decoding.

References

[BCP97] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[BLQ11] J. Berthomieu, G. Lecerf, and G. Quintin. Polynomial root finding over local rings and application to error correcting codes. `http://hal.inria.fr/hal-00642075`, 2011.

[Gao02] S. Gao. A New Algorithm for Decoding Reed-Solomon Codes. In *Communications, Information and Network Security, V. Bhargava, H.V. Poor, V. Tarokh, and S. Yoon*, pages 55–68. Kluwer, 2002.

[Gol07] I. Goldberg. Percy++. Software available from `http://percy.sourceforge.net/`, 2007.

[Gra91] T. Granlund. The GNU Multiple Precision Arithmetic Library, 1991. `http://gmplib.org/`.

[GS98] V. Guruswami and M. Sudan. Improved Decoding of Reed-Solomon and Algebraic-Geometric Codes. *IEEE Trans. Inform. Theory*, 45:1757–1767, 1998.

[GT06] P. Gaudry and E. Thomé. MPFQ : Fast Finite fields, 2006. `http://mpfq.gforge.inria.fr/`.

[H+02] J. van der Hoeven et al. Mathemagix. Software available from `http://www.mathemagix.org`, 2002.

[Har10] W. Hart. Fast Library for Number Theory: An Introduction. In Komei Fukuda, Joris Hoeven, Michael Joswig, and Nobuki Takayama, editors, *Mathematical Software ICMS 2010*, volume 6327 of *Lecture Notes in Computer Science*, pages 88–91. Springer Berlin / Heidelberg, 2010. `http://www.flintlib.org/`.

[Jus76] J. Justesen. On the complexity of decoding Reed-Solomon codes (Corresp.). *IEEE Trans. Inform. Theory*, 22(2):237–238, March 1976.

[McE03] R. J. McEliece. The Guruswami-Sudan Decoding Algorithm for Reed-Solomon Codes, 2003.

[OSB12] F. Oggier, P. Sole, and J.-C. Belfiore. Codes Over Matrix Rings for Space-Time Coded Modulations. *IEEE Trans. Inform. Theory*, 58(2):734–746, February 2012.

[RR98] R. M. Roth and G. Ruckenstein. Efficient decoding of Reed-Solomon codes beyond half the minimum distance. In *IEEE Trans. Inform. Theory*, page 56, 1998.

[RS60] I. S. Reed and G. Solomon. Polynomial Codes Over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.

[Sho90] V. Shoup. NTL: A Library for doing Number Theory, 1990. `http://www.shoup.net/ntl/index.html`.

[Sud97] M. Sudan. Decoding Reed-Solomon codes beyond the error-correction diameter. In *the 35th Annual Allerton Conference on Communication, Control and Computing*, pages 215–224, 1997.

[WB99] S.B. Wicker and V.K. Bhargava. *Reed-Solomon Codes and Their Applications*. John Wiley & Sons, 1999.