Words of Minimal Weight and Weight Distribution in Binary Goppa Codes

Matthieu Finiasz

ISIT 2003 – Yokohama

Binary Goppa Codes

Introduction

- Used for cryptography (**McEliece** cryptosystem)
  - Indistinguishable from a random linear code (1)
  - Efficient decoding algorithm (2)
- Their weight distribution is "close" to the binomial distribution (required for (1))
  - **F. Levy-dit-Vehel** and **S. Litsyn** gave a bound for this "closeness" in 1997
  - very good for medium weights, but of no use concerning small weight words
- No precise theoretical bounds being known, I tried to obtain experimental results. Thanks to (2) it was possible to:
  - implement an algorithm to find words of minimal weight
  - run it to obtain statistical results
  - extend it to words of small weight in general

Finding words of minimal weight

Algorithm 1: Decoding

Let Γ be a binary Goppa code of length $n = 2^m$, dimension k and minimal distance $2t + 1$. We have $n - k = mt$.

The decoding algorithm can decode up to t errors

- for any given word it can determine if there exists a code word at distance t or less

We try to decode words of weight $t + 1$

- if the decoding fails we try with another word
- if it succeeds we have obtained a codeword of minimal weight

If we denote by $\mathcal{N}_{2t+1}$ the number of codewords of weight $2t + 1$ the average number of decoding attempts for a successful one is:

$$A_1 = \frac{\binom{n}{t+1}}{\mathcal{N}_{2t+1} \times \binom{2t+1}{t+1}}$$

INRIA

Finding words of minimal weight
Algorithm 2: Locator Polynomial

We note g the Goppa polynomial of the code and for any word c we note $\mathcal{L}_c$ the locator polynomial of c, that is, the polynomial of roots the non-zero positions of c.

- given a word c, g^2 divides $\mathcal{L}'_c$ if and only if c is in the code

For a word of minimal weight $\mathcal{L}_c$ is monic of degree $2t+1$. As g is also monic and of degree t we have exactly: $g^2 = \mathcal{L}'_c$

- we know $\mathcal{L}'_c$ so we know half the coefficients of $\mathcal{L}_c$
- we can try random values for the other half. Each time $\mathcal{L}_c$ is split we have a word of minimal weight

This time the average number of attempts is:

$$A_2 = \frac{n^{(t+1)}}{\mathcal{N}_{2t+1}}$$

INRIA

We can compare A_1 and A_2:

$$\frac{A_1}{A_2} = \frac{\binom{n}{t+1}}{\binom{2t+1}{t+1} n^{(t+1)}} \approx \frac{t!}{(2t+1)!}$$

$$A_1 \approx \frac{t!}{(2t+1)!} A_2$$

- the first algorithm is asymptotically faster

Decoding is not much slower than testing if a polynomial is split

- A_1 will be faster, even for small values of t

Theory. . .
What we should expect

In [CFS01] the case of decoding a random syndrome in a Goppa code is studied.

- the ratio of decodable random syndromes is approximately $\dfrac{1}{t!}$
- this is true for a random syndrome
  - is it still true for syndromes of words of weight $t+1$?

If this ratio is respected we would have $A_1 = \dfrac{1}{t!}$ and so:

$$\mathcal{N}_{2t+1} \approx \frac{n^{t+1}}{(2t+1)!} \approx \binom{n}{2t+1} \times \frac{1}{2^{mt}}$$

This is exactly the binomial distribution.

INRIA

Known Values

- Goppa codes correcting 3 errors of length $\leqslant 512$ have been classified
  - for each class the exact number of minimal weight word is known

n	exact number	expected number
16	~ 4	2.8
32	128	103
64	$\sim 2\ 640$	$2\ 370$
128	$47\ 616$	$45\ 073$
256	$\sim 806\ 000$	$784\ 509$
512	$13\ 264\ 896$	$13\ 084\ 604$

- expected number corresponds to the binomial distribution value
  - the error decreases exponentially with n: 30%, 20%, 10.3%, 5.3%, 2.7%, $1.36\%\ldots$

INRIA

Experimental Results

To see what happens with greater lengths we used the following technique

- for a given set of parameters n and t
  - generate 20 different random Goppa codes
  - for each code find 50 words of minimal weight (using Algorithm 1)
  - compute Σ the average value of A_1
  - compute σ the standard deviation between the different codes

- if we had a binomial distribution we would get
  - $\Sigma \approx t!$
  - $\sigma \approx \dfrac{t!}{\sqrt{50}}$

We have to perform $1000 \times t!$ decodings for each set of parameters so the computation takes quite a long time.

INRIA

Here are the results which were obtained:

n \ t	5		6		7		8		9	
	Σ	σ	Σ	σ	Σ	σ	Σ	σ	Σ	σ
512	146	21	866	129	5 903	882	45 491	5 128	–	–
1 024	138	30	755	100	5 308	755	44 172	5 387	425 400	52 409
2 048	125	16	721	73	4 892	673	44 827	5 094	367 767	48 077
4 096	119	15	769	144	4 773	962	38 685	6 250	368 646	48 756
8 192	120	17	750	112	5 235	790	41 036	5 041	383 443	56 764
16 384	123	14	732	91	5 470	846	39 351	6 242	374 139	59 313
32 768	120	18	662	99	5 193	933	42 309	8 629	357 590	39 353
65 536	116	16	693	81	5 372	914	39 643	5 719	360 973	41 858
Theory	120	17	720	102	5 040	713	40 320	5 702	362 880	51 319

- Σ denotes the average number of attempts
- σ denotes the standard deviation between the averages obtained with the different Goppa codes

Weight Distribution

Extending to other small weight words

It is possible to run the same experiment for words of larger weight:

- take a word of weight $t+2$ and decode it
  - either you obtain a word of weight $2t+1 \longrightarrow$ the probability is known
  - or you obtain a word of weight $2t+2 \longrightarrow$ make some statistics
- if the ratio of decodable words is $\frac{1}{t!}$ then $\mathcal{N}_{2t+2}$ still corresponds to the binomial distribution

Statistics tend to show that this ratio is respected when decoding words of any weight (greater than $t+1$)

- Binary Goppa codes follow the binomial distribution for any small weight

Conclusion

- We are able to find words of minimal weight in binary Goppa codes correcting few errors

- For all the tested parameters the weight distribution is close to the binomial distribution

- This is true in average but also for any particular code

  - We have exactly what we could have expected!

- What will happen when t is greater?

- Is it possible to use the algorithm for other purposes?

- Can syndromes of words of weight $t+1$ be considered as random syndromes?

INRIA