Fast decoding of binary quadratic residue codes

Yannick Saouter

Fast decoding of binary quadratic residue codes

Y. Saouter

In a recent article, Y. Li *et al* proposed an algorithm for the decoding of binary quadratic residue codes with tiny memory requirements. In this letter, this algorithm is modified in order to dramatically improve the decoding speed. The case of the $(89, 45, 17)$ binary quadratic residue code is used to illustrate the new algorithm.

Decoding binary quadratic residue codes: Binary quadratic residue codes are (n, k, d) cyclic binary codes where n is a prime number congruent to ± 1 modulo 8, $k = \lceil n/2 \rceil$ and $d \geq \sqrt{n}$. A detailed exposure of their construction and properties can be found in [2, chap. 16]. These codes have thus an encoding rate close $1/2$ and a large minimum distance, making their decoding especially difficult. There is quite a large bibliography focusing on this topic. Proposals can be split in two categories. The first category of decoding algorithms involves algebraic decoding. The syndromes of codes are computed and the associated locator polynomial is determined. The situation is complicated by the fact that, for quadratic residue codes and contrary to the case of BCH codes, all the required syndromes cannot be computed directly. Then sophisticated nonlinear techniques are involved in obtaining the missing syndromes. These algorithms are generally difficult to build and have a high computational cost. This type of algorithm has been pioneered in reference [3] to the $(23, 12, 7)$ Golay code. In what concerns our following illustrating example, say the binary quadratic residue code $(89, 45, 17)$, reference [4] describes the first complete algebraic decoding procedure. The second category of decoding algorithms uses the value of parity check vectors to make a one-to-one correspondence with error patterns. For instance, by direct lookup table, it is possible to build a table with 2^{n-k} entries corresponding to all the possible parity check vectors and containing pointers to all error patterns. This technique permits fast decoding and may be considered for short codes. If we set $t = \lfloor (d-1)/2 \rfloor$, it could be observed that the number of distinct error patterns is $N = \sum_{i=0}^{t} \binom{n}{i}$. This number is generally much smaller than 2^{n-k}. Using a sorted table and binary search, the memory requirements can then be reduced. The price to pay is the decoding process involves then $\lceil \log_2(N) \rceil$ accesses to the table. However the memory requirements are still prohibitive for large codes. Using the pigeonhole principle and on-the-fly generations of error events, the paper [1] proposes an algorithm with tiny memory requirements. The drawback is that an eventually numerous number of error patterns have to be tested, which in turn penalizes decoding speed.

Speed, memory and hostile environments: Memory limitations at the present time are far above than in the early times of digital electronic. Nowadays, digital chips dedicated to server applications have multimegabyte on-chip caches. This limit can be still overcome by external DRAM memories which can offer up to hundreds of gigabytes. Modern microcontrollers also generally possess DRAM controllers. For FPGA implementations, most of the vendors provide synthesizable IPs for this type of controller. Non-volatile memories such as mask ROM or Nand Flash RAM follow the same evolution and are now available in comparable size. Some emerging technologies like magnetoresistive RAMs (MRAMs) are even expected to push limit memory further in the near future. These technologies are even used in difficult environments like outer space. For instance, the New Horizons data probe launched in 2006 is equipped with a Moongoose-V 32-bit microprocessor and a data storage of 16 GB provided by two SSD recorders [5]. However, these devices have the additional constraint to be radiation hardened. Although such devices are now easily available, for microprocessors, one consequence of radiation hardening is to lower clock frequency for safety of execution. For instance, while the Moongoose-V microprocessor implements the R3000 instruction set, the clock speed in New Horizons is lowered to 12 MHz compared to the 33 MHz maximal frequency of the original R3000 microprocessor. These remarks justify the fact that in hostile environments, speed constraints are much more restrictive than memory constraints. Therefore, if a substantial decrease of execution time is expected, algorithms involving large lookup tables could be preferable to algorithms with optimized memory cost.

Difference syndrome algorithm: This section briefly describes the decoding procedure of reference paper [1]. Let then $\mathbf{H} = (\mathbf{I_{n-k}}|\mathbf{P})$ be a parity check matrix in systematic form of the binary quadratic residue code (n, k, d), where $\mathbf{P}$ is a $(n-k) \times k$ binary matrix. Let $\mathbf{x}$ be a size n column vector such that $\mathbf{x} = \mathbf{c} + \mathbf{e}$ where $\mathbf{c}$ represents a codeword of the code and $\mathbf{e}$ represents a perturbation error event whose Hamming weight is at most t. Then a complete decoding procedure, given $\mathbf{x}$, can recover $\mathbf{c}$. The parity check vector of $\mathbf{x}$ is the size $(n-k)$ row vector $\mathbf{V} = \mathbf{H}.\mathbf{x} = \mathbf{H}.\mathbf{e}$. First, if we suppose that the Hamming weight of $\mathbf{V}$ is such that $w_H(\mathbf{V}) \leq t$, since $d > 2t$, all errors have occurred in the $n-k$ places of x. Moreover, we have $\mathbf{r} = \mathbf{x} + \mathbf{W}$, where $\mathbf{W}$ is the size n column vector obtained by adding k null entries at the bottom of $\mathbf{V}^t$. Now suppose that $\mathbf{e}'$ is a n row vector with possible non-null entries only in the last k coordinates. Let $t' = w_H(\mathbf{e}') \leq t$ and $\mathbf{V}' = \mathbf{H}.(\mathbf{x} + \mathbf{e}') = \mathbf{H}.\mathbf{e} + \mathbf{H}.\mathbf{e}'$. Again, if we suppose that $w_H(\mathbf{V}') \leq (t - t')$, we can conclude that all errors occurred in the $n-k$ places of $\mathbf{x}$ and that we have $\mathbf{r} = \mathbf{x} + \mathbf{e}' + \mathbf{W}'$ where $\mathbf{W}'$ is the size n column vector obtained by adding k null entries at the bottom of $\mathbf{V}'^t$. Since the code is cyclic, these facts still hold for any vector obtained from r by cyclic shifts. We define then $\mathbf{r}^{\leftarrow} = \mathbf{r} << (n-k)$ as the size n column vector obtained by shifting $\mathbf{r}$ cyclically of $n-k$ places to the top. Let $t_2 = \lfloor t/2 \rfloor$. If we suppose that the error perturbation vector has at most t_2 non-null entries in its k coordinates, then by computing Hamming weights of vectors $\mathbf{V}'$ for every vector $\mathbf{e}'$ out of the $\sum_{i=0}^{t_2} \binom{k}{i}$ possible ones, it is then possible to decode the vector $\mathbf{r}$. If this procedure does not succeed, the same can be done for the vector $\mathbf{r}^{\leftarrow}$. If both procedures fail, then necessarily $\mathbf{e}$ and $\mathbf{e}^{\leftarrow}$ have more than t_2 non-null entries in their k coordinates. We have then $w_H(\mathbf{e}) \leq 2(t_2 + 1) - e_n$. However, since $w_H(\mathbf{e}) \leq t$, then necessarily t must be odd and $e_n = 1$. Therefore, in this case, it is possible to decode the vector $\mathbf{r}$ by enumerating the $\sum_{i=0}^{t_2} \binom{k-1}{i}$ vectors $\mathbf{e}'$ having Hamming weights equal to 0 (resp. t_2, 1) on the first $n-k$ coordinates (resp. the following $n-k$ coordinates, the last coordinate). All syndromes which require to be computed can be generated in sequence. As a consequence, the difference syndrome algorithm can be implemented with only parity check matrix $\mathbf{H}$ in constant memory.

Fast search procedure: The proposed technique for decoding was initially proposed in [6] for quite a similar problem. Let i be an integer less than t_2. All vectors $\mathbf{e}'$ of Hamming weight i and first $n-k$ null coordinates are generated and size $n-k$ corresponding syndrome vectors $\mathbf{H}.\mathbf{e}'$ are generated and stored in an array S_i. The index values of the i non-null coordinates of $\mathbf{e}'$ are stored in another array I_i. These arrays have $\binom{k}{i}$ entries. The elementary task of the decoder of the previous paragraph is given $\mathbf{r}$, to find, if it exists, an error configuration $\mathbf{e}'$, such that $w_H(\mathbf{H}.(\mathbf{r} + \mathbf{e}')) \leq (t - i)$. We suppose that each entry of S_i is cut in $t - i + 1$ disjoint slices. If a matching configuration $\mathbf{e}'$ exists, then the two parity vectors $\mathbf{H}.\mathbf{r}$ and $\mathbf{H}.\mathbf{e}'$ necessarily coincide in at least one slice. We set $l = \lfloor (n-k)/(t-i+1) \rfloor$ and for j between 1 and $t - i + 1$, an array L_{ij} is generated. Array L_{ij} contains the $\binom{k}{i}$ addresses of configurations I_i, sorted in increasing order according to the values of corresponding syndrome vectors along the j-th block. At this point, it is possible to find candidate $\mathbf{e}'$ vectors by dichotomy. However, in a sake of speed, direct access has been preferred. To any array L_{ij}, a new array A_{ij} with 2^l entries is associated. The address s of A_{ij} contains then the least value k, such that the configuration pointed at by $L_{ij}(k)$ has a value greater or equal to s in its j-th slice. As a consequence, by construction, given any s value, for all k with $A_{ij}(s) \leq k < A_{ij}(s+1)$, the configuration pointed at by address $L_{ij}(k)$ has a j-th slice equal to s. Note that this set can empty. This is the case if s does not appear as possible value for any j-th. In this case, we have effectively $A_{ij}(s+1) = A_{ij}(s)$. The entire decoding procedure is then as follows. First compute $\mathbf{V}_1 = \mathbf{H}.\mathbf{r}$ and $\mathbf{V}_2 = \mathbf{H}.\mathbf{r}^{\leftarrow}$. If $w_H(\mathbf{V}_1) \leq T$ or $w_H(\mathbf{V}_2) \leq T$, then the decoding is immediate using the first criterion of difference syndrome algorithm. If not, for i ranging from 1 to t_2, compute $l = \lfloor (n-k)/(t-i+1) \rfloor$ and cut $\mathbf{V}_1$ and $\mathbf{V}_2$ in $t - i + 1$ slices of l bits wide. If s_j is the j-th slice, for all k such that $A_{ij}(s_j) \leq k < A_{ij}(s_j + 1)$, compute the parity check vector with address $L_{ij}(k)$ in the array I_i. Compute the Hamming weight of the sum of this vector with $\mathbf{V}_1$ (or $\mathbf{V}_2$ according to the case). If this weight is less or equal than $t - i$, then decode according to the difference syndrome algorithm. If t is odd and the previous procedure fails, compute $\mathbf{V}_3 = \mathbf{H}.\mathbf{r} + \mathbf{H}.e_n$ and perform the procedure for $i = t_2$ with a maximum admissible weight equal to $t - t_2 - 1$.

Nb. err.	ALG [4]	DS [1]	FS
1	2.3	1.1	1.1
2	180	2.4	1.2
3	2300	4.1	1.6
4	4900	24	2.7
5	8200	48	5.6
6	12000	270	14
7	22000	540	31
8	34000	2400	82

Table 1: Average decoding time for decoding algorithms (in μs)

Application to the $(89, 45, 17)$ *binary quadratic residue code:* This procedure has been applied successfully to each binary quadratic residue code of length less than 100. In this paragraph, our attention will be focused on the $(89, 45, 17)$ binary quadratic residue code. A full description of this code can be found in [4]. This code is able to correct up to $t = 8$ errors and is unique with this property amongst binary quadratic residue codes of length less than 100. This code is also the only one with $t_2 = 4$. Thus configurations up to weight 4 have to be stored which is clearly a disadvantage for the difference syndrome algorithm as well as for the presented algorithm. On the contrary, the working Galois field of this code is $GF(2^{11})$, like the Golay code. This field is much smaller than that of other binary quadratic residue codes. This particularity was used in [4] to design an efficient algebraic decoder which is faster than algebraic decoders of other binary quadratic residue code of comparable length. The three decoding algorithms have been implemented and have been tested for decoding speed on a recent Intel-i7 computer. Each algorithm is able to correct error patterns up to half the minimum distance. Therefore decoding performance is identical for all the three algorithms and is depicted in [4, p. 5011]. Average decoding time has been computed according to error weights. Up to 5 errors, the global set of configurations was tested. For 6 error patterns and more, configurations were generated in a pseudo-random way. In any cases, several millions of configurations were decoded in order to obtain representative timing performance. Table 1 summarizes the average decoding time obtained. It can be seen that the fast search algorithm provides a speedup of up to 29 with respect to the difference syndrome algorithm, and up to 414 for the algebraic decoder. The size of the require arrays is now estimated. Although these arrays could be packed, we will suppose that they are in fact byte-aligned. This avoids the need for extraction procedures. Arrays S_i are not stored. Syndromes are in fact regenerated from data of I_i arrays. These latter arrays require $\binom{k}{i} \times i$ bytes for storage. For each possible value i, exactly $l = \lfloor (n-k)/(t-i+1) \rfloor$ arrays L_{ij} are required. Each of these arrays requires $\binom{k}{i} \times K$ bytes for storage with $K = \lceil \log_{256} \binom{k}{i} \rceil$. Finally, l arrays A_{ij} of size $(2^l) \times K$ bytes are also required. Using these formulas for our working case gives a global memory cost of 3070368 bytes (see table 2). At this point, following a remark of [6], an additional speedup can be obtained. In the case $i = 4$, the syndrome is cut in 5 slices, so that syndromes should coincide in at least 9 consecutive bits. If we suppose that bit expansions are distributed at random, syndromes have on average 1 chance out of 2^9 to match. Therefore, for each L_{ij} table, we have on average $148995/2^9$ candidates to test. Suppose now that syndromes are cut in 3 slices of 14 bits. In this case, syndromes should coincide on at least 13 bits of the 14 bits wide section. Therefore, in order to enumerate all possible candidates, given the syndrome of the received vector, we have to enumerate all possible candidates matching with one of the 15 sections differing at most with one bit with the section of the received syndrome. For each L_{ij} table, the average number of candidates decrease then to $148995 \times 15/2^{14}$. Moreover, since the corresponding l value decreases from 5 to 3, the number of tables L_{ij} decreases in turn giving an additional speed improvement. The average decoding time of 8 error patterns decrease then to 78 μs. Morever, in table 2, for $i = 3$, we have then $L_{3j} = 1340955$ and $A_{3j} = 147456$. The memory requirement of the entire algorithm decreases then to 2316174 bytes. The final speedups of this solution are then equal to 435 with respect to the algebraic decoding and 30 with respect to the difference syndrome algorithm.

Conclusion: In this paper, the author described an algorithm to perform decoding of cyclic error correcting codes of near-half rate $1/2$ applied to binary quadratic code. This algorithm exhibits a large speedup in regard of classical algorithms. The price to be paid is the implementation of lookup tables whose size is less than 4 MB and can easily be implemented with current technologies. With some slight modifications, it can also be applied to other codes with large automorphism groups like quasi-cyclic codes.

i	I_i	L_{ij}	A_{ij}
1	45	360	256
2	1980	13860	896
3	42570	170280	1536
4	595980	2234925	7680

Table 2: Memory requirements for fast search algorithm (in bytes)

Y. Saouter (*Lab-STICC, Brest, France*)
E-mail: Yannick.Saouter@imt-atlantique.fr

References

1 Y. Li, Y. Duan, H.-C. Chang, H. Liu, and T.-K. Truong.
Using the difference of syndromes to decode quadratic residue codes.
IEEE Trans. Information Theory, 64(7):5179–5190, 2018.
DOI: 10.1109/TIT.2018.2830327.

2 F.J. MacWilliams and N.J.A. Sloane.
The Theory of Error-Correcting Codes.
North-Holland, 1977.

3 M. Elia.
Algebraic decoding of the $(23, 12, 7)$ Golay code.
IEEE Transactions on Information Theory, 33(1):150–151, 1987.
DOI: 10.1109/TIT.1987.1057262.

4 T.-K. Truong, P.-Y. Shih, W.-K. Su, C.-D. Lee, and Y. Chang.
Algebraic decoding of the $(89, 45, 17)$ quadratic residue code.
IEEE Transactions on Information Theory, 54(11):5005–5011, 2008.
DOI: 10.1109/TIT.2008.929956.

5 New Horizons. The first mission to Pluto and the Kuiper Belt: Exploring frontier worlds.
Launch Press Kit.
`https://www.nasa.gov/pdf/139889main_PressKit12_05.pdf`.

6 J.F. Voloch.
Computing the minimal distance of cyclic codes.
Computational and Applied Mathematics, (24):393–398, 2005.
`https://web.ma.utexas.edu/users/voloch/Preprints/quad2.pdf`.