오류 복구를 위한 CRC 코드 커버링 패턴의 탐색 방법

Search Methods for Covering Patterns of CRC Codes for Error Recovery

성 원 진
(Wonjin Sung)

Abstract : Error detection and correction using CRC and the general class of cyclic codes is an important part of designing reliable data transmission schemes. The decoding method for cyclic codes using covering patterns is easily-implementable, and its complexity depends on the number of covering patterns employed. Determination of the minimal set of covering patterns for a given code is an open problem. In this paper, an efficient search method for constructing minimal sets of covering patterns is proposed and compared with several existing search methods. The result is applicable to various codes of practical interest.

Keywords : cyclic codes, covering pattern, error recovery, forward error correction

I. Introduction

CRC(Cyclic Redundancy Check) codes play an important role in reliable transmission of data information in communication networks. By calculating check-sum at the destination node, errors incurred during the transmission are easily detected and erroneous data packets are requested for re- transmission. CRC codes have been extensively studied in coding community under the name shortened cyclic codes. Many standard CRC codes have been proposed and used for different level of protection, such as CRC-8, CRC-12, CRC-CCITT, CRC-32[1], where the numeric parts represent the number of parity (or redundancy) bits.

Performance of error detection using CRC codes has been investigated for a number of different applications, including recent reports on ATM networks[2], interactive video streaming[3]. and traffic load balancing[4]. Novel approaches for utilizing unique properties of CRC have been also proposed[5][6]. Software and hardware implementation with minimal computational complexity is an important issue, and [7] and [8] report efficient algorithms. As is well-known, CRC codes and the general class of cyclic codes is capable of not only detecting errors, but also correcting errors, and increasingly many applications and protocols support error correction via decoding cyclic codes.

This paper relates to the problem of devising an efficient and simple decoder for (n, k) cyclic block codes. In particular, we focus on error-trapping decoding and its generalized form using covering patterns, which are also called covering polynomials [9]. The algorithm using covering patterns is a simple and flexible method of decoding cyclic codes, and its complexity depends on the number of covering patterns employed. The determination of a minimal set of covering patterns for a given cyclic code, which is equivalent to designing minimal complexity decoder, is an open problem [10]. The minimal sets are known for limited cases, and covering patterns for decoders are often determined in an ad-hoc fashion or via extensive computer search.

Here we compare several search procedures for finding the covering patterns and propose an efficient search algorithm which determines a minimal set of covering patterns. In Section 2, algorithm description of the decoding cyclic codes using covering patterns is given, and search methods for finding minimal sets of covering patterns are described and compared in Section 3. Conclusions are given in Section 4.

II. Algorithm Description

The error-trapping-based decoding algorithms all rely for their effectiveness on the re-encoding principle: if we know the correct values of any set of k linearly independent positions, we can re-encode the codeword from them (and by comparison with the received word, we can determine the error pattern, if desired). Thus instead of having to determine the complete set of errors, it is sufficient to determine the errors in a given set of k linearly independent bits.

The various error-trapping-based algorithms differ in the ways they attempt to find the true values of the symbols in a set of k linearly independent positions. In information set decoding, we take a large enough number of information sets to ensure that at least one is error-free. It has been shown that this algorithm is the best of the family in an asymptotic sense: for virtually all long codes, this is essentially optimum. Unfortunately, in this case it is extremely difficult to determine a minimal set for a specific code. As a result, optimum information sets are known for very few codes, notably the Golay codes, and the general problem seems unlikely to be solved. Note also that each re-encoding operation can take k^2 operations, which for medium length codes can easily nullify the asymptotic advantage the algorithm

접수일자 : 2001. 12. 15., 수정완료 : 2002. 1. 15.
성원진 : 서강대학교 전자공학과(wsung@sogang.ac.kr)
※ 본 연구는 한국과학재단 목적기초연구 R01-2001-00542 지원으로 수행되었음.

has over other members of the family.

Permutation decoding addresses the problem of ensuring that the k-sets are linearly independent by using the automorphism group of the code. One known information set can then be permuted onto others. We are still left with the problem of finding minimal sets, and we lose the property that any code is completely decodable by the method.

In the covering pattern method, we assume we have a cyclic code, and take n information sets, consisting of all sets of k(cyclically) consecutive positions. We restrict the guesses of errors to have the same pattern for each information set. These guesses can be represented as polynomials, therefore also called covering polynomials. To implement this, we would have a number of independent sub-processors, each devoted to a single covering pattern. Starting from the syndrome sequence, each sub-processor adds the codeword whose information positions correspond to its covering pattern. This gives another coset element. The likelihood of this coset element is computed for each subprocessor estimate using some metric computer, and the best estimate is stored. Each subprocessor subtracts out its added codeword, shifts the syndrome sequence left one position, and repeats the above operations. After n such steps, the candidate coset leader that gave the best metric is taken as the correct coset leader.

The procedure has several advantages. We are not limited by the bounded distance of the code, and can achieve complete hard decision decoding. The optimal error pattern will not be immediately recognizable, but can be declared optimal when the algorithm terminates. More importantly, we are not constrained to use hard decision decoding. The metric calculator mentioned in the algorithm description can be adapted to perform soft-decision decoding.

III. Search Methods

One approach to obtain the solution to the problem of finding minimal sets of covering patterns is using search procedures. For a given problem, we have a set of candidate covering patterns and a set of "interval patterns" to be covered. Interval patterns refer to vectors representing relative intervals between successive error locations in a transmitted code block. The most straightforward way of searching is to consider all combinations of the increasing number of candidate covering patterns, until a combination is found to cover all interval patterns. Due to its large computational effort, this brute force search can only be performed for the problems of small dimensions (short codes correcting small number of errors). Combinatorial approaches can be found in the literature, and a systematic formulation of these is the *prime implicant table problem*[11].

1. Prime Implicant Tables

A prime implicant table is a matrix with an arbitrary number of rows and columns and its elements are either marked or unmarked. For a given table, the prime implicant table problem is to find a smallest set of rows such that every column of the table has a marked entry in at least one of the rows in the set. By letting the rows represent covering patterns and the columns represent interval patterns, we formulate the covering pattern problem into the prime implicant table problem. A table element is marked if the covering pattern of the element row covers the interval pattern of the element column. Approaches to solve the prime implicant table problem have been suggested, and for general solutions, search algorithms including enumeration procedures of some sort must be used.

One way to simplify the prime implicant table is to use *row dominance* and *column dominance*. Row i dominates row j if row i has marked elements in every column in which row j has marked elements. Similarly, column i is said to dominate column j if column i has marked elements in every row in which column j has marked elements. Any dominating row or column can be removed from the table without affecting the solution, and when no rows or columns can be removed by dominance, the table is called *cyclic*. The prime implicant table problems can be simplified by taking advantage of row and column dominance, and these properties can also be applied to determine the covering patterns.

2. Linear Programming and Branching Method

One of the most commonly used techniques to solve the prime implicant table problem having a cyclic form is to apply integer linear programming algorithms. The (primal) linear programming method with integer constraint is equivalent to the covering pattern problem when rows are taken as covering patterns and columns are taken as interval patterns:

$$\text{minimize } \sum_{i=1}^{I} x_i \text{ subject to } \sum_{i=1}^{I} a_{ij}\, x_i \geq 1$$

for $j=1,\ldots,J$, where I is the number of rows, J is the number of columns, $a_{ij}=1$ if i-th row covers the j-th column and 0 otherwise, and $x_i=0$ or 1 for $i=1,\ldots,I$

The dual problem is interpreted as finding a largest set of interval patterns such that at most one covering pattern covers each interval pattern in the set. It is well known that the global solutions of the primal and the dual linear programming without the integer restrictions are identical. Furthermore, the solution is upper bounded by any feasible solutions of the primal problem, and lower bounded by any feasible solutions of the dual problem. It follows that the global solution of the unrestricted problem, and any feasible solutions of the restricted dual problem give lower bounds to the solution. The linear programming without integer restrictions can be computed by efficient algorithms such as the simplex algorithm, and the solutions provide a tighter lower bound than the dual integer solutions.

An upper bound can be obtained by a *greedy algorithm*. The algorithm begins with an empty set of covering patterns and all interval patterns to be covered. At each stage of the greedy algorithm, the covering pattern that covers the most number of the

remaining interval patterns is chosen and added to the set, and the interval patterns covered by the chosen covering pattern are deleted. The procedure continues until all interval patterns are deleted, and the resulting set of covering patterns is claimed as a covering set for the problem.

The exact solution of a covering pattern problem is the solution of the corresponding primal integer programming problem. The *branch* and *bound algorithm* is a useful approach for solving discrete optimization and integer programming problems in general. It searches for an optimal feasible solution by doing only a partial enumeration. A brief description of the algorithm is as follows: At each stage of the algorithm, the set of feasible solutions is partitioned into many simpler subsets, and a lower bounding strategy using Lagrangian relaxation method is applied to determine which of the subsets is most promising for further search. Some of the subsets are discarded from the search if the lower bound for those subsets is greater than the *incumbent*, which is the best solution known at that stage. The selected subset is searched for to find the best feasible solution in the set. If such a solution is found, we either update the incumbent or discard the set. If such a solution is not found, another partition is made and the procedure continues. After all undiscarded subsets have been searched, the algorithm terminates and the incumbent is guaranteed to be the optimal solution.

The (21,15) code with s = 3, where s is the number of errors to be corrected, is an example that the dual integer solution can be the same as the unrestricted global solution. To give an idea of the computational complexity of the search algorithm, we mention that the (31,21) code with s=3 has 231 variables (covering patterns) with 109 constraint equations (coverage of each interval pattern by the covering patterns), and the (31,16) code with s=4 has 136 variables with 651 constraints. The branch and bound algorithm works reasonably well for short codes, but the complexity grows rapidly for larger values of n, k, and s. It is also observed that the lower bounding strategy using the Lagrangian relaxation method becomes less effective when the number of interval patterns becomes larger, meaning a larger number of candidate covering pattern sets are searched for. Computation results for some example codes are shown in Table 1. Simplex algorithm results provide tight lower bounds for the cases of (21,11), (21,14), and (31,21) codes, and confirm the greedy algorithm result is optimal for the case of (31,21) code.

Table 1. The number of covering patterns.

(n, k)	s	Dual	Simplex	Optimal	Greedy
(21,11)	4	3	5.25	6	7
(21,14)	3	4	4.31	5	6
(21,15)	3	6	6.00	7	8
(31,16)	4	5	6.96	9	10
(31,21)	3	5	5.25	6	6

Greedy algorithm produces covering pattern sets that are suboptimal in the given examples.

3. A Reduced-Complexity Search Algorithm

Covering sets for an (n, k) code with a specified value of s can include only covering patterns of a single bit location, termed single patterns hereafter, if the condition $k/n < 2/s$ is satisfied. Determination of covering sets for the case of $R < 2/s$ is an important special class of the covering pattern problem. Many cyclic codes of short to medium length satisfy such a rate condition when they are applied to correct up to the bounded distance or more.

The proposed search algorithm finds a minimal set of single patterns for given n, k, and s that satisfy the condition $k/n < 2/s$. The algorithm searches for a small subset of covering pattern combinations rather than performing a brute force search. The basic idea of the algorithm is to build up sufficient sets of covering patterns from the middle of the information positions outwards. Thus a new covering pattern is always added to the left of the current leftmost covering pattern or to the right of the rightmost. This reduces a significant amount of the required number of computations; however, it can readily be verified that in this kind of procedure, a bad choice of a new covering pattern can irreparably damage the sufficiency of the resulting covering pattern set. Thus the procedure is designed to ensure that no error patterns will be missed in the building of the covering pattern set. The procedure results in a number of sufficient sets, and the minimal such set is then guaranteed to be globally optimum. The proposed search algorithm is summarized as follows. (Also see Fig. 1.)

1) We are given an (n, k) code to correct s errors, where $k/n<2/s$. For each $i=k+1-[n/s], \ldots, [n/s]$, let $j=1$, $m_1=i$, and $M_1=\{x^{m_1}\}$, where $[x]$ denotes the ceiling function of x.

2) Determine h_j for the given monomial set M_j. If $h_j \leq k$, the search procedure continues in Step 3. Otherwise, go to Step 4.

3) Increase j by one, and let $m_j=h_{j-1}$ if j is odd, $m_j=$

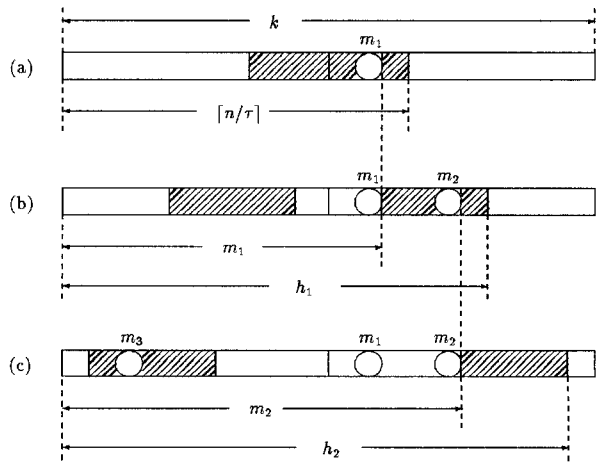

Fig. 1. Search locations for covering patterns.

$k+1-h_{j-1}$ if j is even. $M_j = M_{j-1} \cup \{m^{m_i}\}$. Repeat Step 2.

4. The set $M_j = \{x^{m_1}, x^{m_2}, \ldots, x^{m_i}\}$ is a minimal covering set of single patterns for the code.

The proposed algorithm is much more efficient than the brute-force search in that it only searches for the single pattern sets that are sufficient to cover all interval patterns. To find an optimal covering pattern set of size c for an (n, k) code, a brute-force search should at least consider $\binom{k}{0}+\binom{k}{1}+\ldots+\binom{k}{c-1}$ sets of single patterns, and furthermore would need to check whether each of them is sufficient to cover all interval patterns, which itself requires considerable computations. The search algorithm considers $L_1 \times L_2 \times \ldots \times L_{c-1}$ different combinations of covering patterns where L_i is the length of the region for the i -th pattern.

IV. Conclusions

The minimal set of covering patterns for given cyclic codes can be obtained by using search algorithms. We observed that the covering pattern problem can be formulated as a particular case of the prime implicant table problem. The general prime implicant table problem belongs to the NP-complete class, which suggests the difficulty of obtaining general solutions for the covering pattern problem. The branch and bound algorithm is an effective method to find optimal covering sets for any given problem, but its applicability is limited to relatively short codes. The simplex algorithm has been used to give a numerical lower bound to the optimal size.

The search algorithm proposed in Section III. 3 is applicable for arbitrary code parameters n, k, and s satisfying $k/n < 2/s$, and its complexity is no greater than $\left(\frac{k}{c-1}\right)^{c-1}$ where c is the minimal number of covering patterns required. The algorithm is implemented by a simple recursive structure to determine a minimal covering set for a given code.

성 원 진

1990년 서울대 전자공학과 졸업. 1992년 미국 미시건대 대학원 전기공학과(EECS) 석사. 1995년 동대학원 박사. 1996년 1월~2000년 8월 미국 Hughes Network Systems사 책임연구원. 2000년 9월~현재 서강대학교 전자공학과 조교수. 관심분야는 디지털 전송기술, 오류 제어.

참고문헌

[1] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Upper Saddle River, NJ: Prentice-Hall, 1995.

[2] F. Braun and M. Waldvogel, "Fast incremental CRC updates for IP over ATM networks," in *Proc. IEEE Workshop on High Performance Switching and Routing*, pp. 48-52, 2001.

[3] J. Cai and C. W. Chen, "Video streaming: An FEC-based novel approach," in *Proc. Canadian Conf. Elec. and Comp. Eng.*, vol. 1, pp. 613-618, 2001.

[4] Z. Cao, Z. Wang, and E. Zegura, "Performance of hasing-based schemes for internet load balancing," in *Proc. 2000 IEEE INFOCOM*, vol. 1, pp. 332-341, 2001.

[5] R. Anand, K. Ramchandran, and I. V. Kozintsev, "Continuous error detection for reliable communications," *IEEE Trans. Commun.*, vol. 49, no. 9, pp. 1540-1549, Sept. 2001.

[6] A. J. McAuley, "Weighted sum codes for error detection and their comparison with existing codes," *IEEE/ACM Trans. Networking*, vol. 2, no. 1, pp. 16-22, Feb. 1994.

[7] D. C. Feldmeier, "Fast software implementation of error detection codes," *IEEE/ACM Trans. Networking*, vol. 3, no. 6, pp. 640-651, Dec. 1995.

[8] R. Nair, G. Ryan, and F. Farzaneh, "A symbol based algorithm for hardware implementation of cyclic redundancy check," in *Proc. 1997 VHDL Int. Users' Forum*, pp. 82-87.

[9] T. Kasami, "A decoding procedure for multiple- error-correcting cyclic codes," *IEEE Trans. Inform. Theory*, vol. 10, pp. 134-138, 1964.

[10] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*, North-Holland, New York, 1977.

[11] I. B. Pyne and E. J. McCluskey, Jr., "The reduction of redundancy in solving prime implicant tables," *IEEE Trans. Electronic Computers*, vol. 11, pp. 473-482, 1962.