

Constructions of Almost Block-Decodable Runlength-Limited Codes

Kees A. Schouhamer Immink, *Fellow, IEEE*

Abstract—The correspondence describes a new technique for constructing fixed-length (d, k) runlength-limited block codes. The new codes are very close to block-decodable codes, as decoding of the retrieved sequence can be accomplished by observing (part of) the received codeword plus a very small part (usually only a single bit) of the previous codeword. The basic idea of the new construction is to uniquely represent each source word by a (d, k) sequence with specific predefined properties, and to construct a bridge of β , $1 \leq \beta \leq d$, merging bits between every pair of adjacent words. An essential element of the new coding principle is look ahead. The merging bits are governed by the state of the encoder (the history), the present source word to be translated, and by the upcoming source word. The new constructions have the virtue that only one look-up table is required for encoding and decoding.

Index Terms—RLL code, look-ahead encoding, error propagation, recording code.

I. INTRODUCTION

Coding methods based on (d, k) -constrained sequences have been predominant in magnetic and optical disks or tapes. A binary sequence is said to be (d, k) -constrained if the number of “zeros” between any pair of consecutive “ones” is at least d and at most k , $k > d$. Properties and applications of (d, k) -constrained sequences, or runlength-limited (RLL) sequences as they are often called, are surveyed in [1].

Codes are used for translating source data into the constrained sequence. In state-of-the-art recorders, the source data are partitioned into words of length m , and under the coding rules, these m -tuples are translated into n -tuples, called *codewords*. Popular (d, k) codes incorporated in disk file systems are the $(2, 7)$ and $(1, 7)$ codes of rate $m/n = 1/2$ and $2/3$, respectively [2]. The codes can be designed using the bounded-delay method [3], [4] or alternatively by the ACH, or sliding-block code, algorithm [5]. The length of the decoding window is an important design parameter as it affects both the amount of error propagation and decoding hardware.

An alternative to the sliding-block coding scheme was proposed by Tang and Bahl [6], Beenker and Immink [7], Weber and Abdel-Ghaffar [8], Immink [9], and recently by Gu and Fuja [10]. There, the authors use (d, k) codes compiled from codewords of fixed length that can be decoded without the knowledge of preceding or succeeding codewords. Codes with this property, that is, codes that can be decoded by observing single codewords will be called *block(-decodable) codes* (it should be appreciated that the encoding operation is allowed to be state-dependent). Evidently, block-decodable codes offer an advantageous solution relative to sliding-block codes since they make it easier to preserve a particular mapping between the source and the code symbols, and, obviously, error propagation is localized to one decoded m -tuple. Block-decodable codes are highly suitable in conjunction with Reed–Solomon error-control codes. In the preferred embodiment of the coding system, the codewords have a 1-1 correspondence with the elements of the finite

field $GF(2^m)$, thus enabling the construction of a Reed–Solomon code directly over the (d, k) -constrained codewords.

The new technique presented in this correspondence is broadly similar to Beenker and Immink’s “Constructions 1 and 2” [7]. An essential, and very useful, property of these constructions, which they have in common with the new constructions is that source words have a one-to-one relationship with finite-length (dk) sequences that have additional constraints on the number of leading and trailing zeros. After translating the source words into (dk) sequences, these sequences are multiplexed with d bits, called *merging bits*, which are needed to preserve the predefined runlength constraints. The above constructions have the engineering virtues that a) the translation of source words into codewords can be accomplished with a single look-up table, b) the cascading of codewords can be done with a simple merging rule, c) their structure permits enumerative coding techniques to simplify encoding and decoding hardware, which is a particularly desirable feature when the codeword length is relatively large, and d) they are simple to understand. In the new constructions, on the other hand, less than d merging bits permit the cascading of the (dk) sequences. This has a direct bearing on the rate of the code. The merging operation affects both the merging bits and a few bits of the adjacent (dk) sequences. An essential feature of the new codes is look-ahead, that is, during the encoding process, the codeword is not only a function of the history, but is a function of the upcoming data to be encoded as well. In contrast with Construction 1 and 2, the new codes are almost block-decodable as decoding of the sequence can be accomplished by observing (part of) the received codeword plus a small part of the previous codeword. In two new constructions, called Construction 3 and 4, only one bit of the previous codeword has to be used during decoding, while in Construction 5, three bits of the previous codeword are used. As a result, minor error propagation may occur during decoding.

We start with a brief outline of code constructions of fixed-length codes based on $(dklr)$ sequences, followed, in Section III, by a description of the new coding technique. Results of the application of the new algorithm are given in Section IV.

II. CODES BASED ON $(dklr)$ SEQUENCES

The following systematic construction techniques, taken from Beenker and Immink [7], employing sequences with special properties, called $(dklr)$ sequences, have the advantage that only one look-up table is needed for the generation of the $(dklr)$ sequences plus some logic for determining the merging bits used to cascade the $(dklr)$ sequences. A $(dklr)$ sequence obeys the following properties:

- 1) d constraint—between consecutive “ones” are at least d “zeros.”
- 2) k constraint—between consecutive “ones” are at most k “zeros.”
- 3) l constraint—the number of consecutive leading “zeros” of the sequence, that is, the number of “zeros” preceding the first “ones” is at most l .
- 4) r constraint—the number of consecutive trailing “zeros” of the sequence, that is, the number of “zeros” succeeding the last “ones” is at most r .

A (dk) sequence satisfies the d and k conditions only.

Finite-length sequences satisfying the (dk) or $(dklr)$ constraints cannot be cascaded without violating the (dk) constraints. Inserting a number β of “merging” bits between adjoining sequences makes it possible to preserve the d and k constraints. Tang and Bahl [6] presented a coding scheme where (dk) sequences of a given (finite) length are multiplexed with $\beta = d + 2$, $d > 0$, merging bits. The

Manuscript received October 5, 1993; revised June 3, 1994. This paper was presented in part at the International Symposium on Information Theory, Trondheim, Norway, 1994.

The author is with Philips Research Laboratories, 5656 AA Eindhoven, The Netherlands.

IEEE Log Number 9407258.

TABLE I
MERGING RULE OF $(dklr)$ SEQUENCES

s, t	Merging bits
$s + t + d \leq k$	0^d
$s + t + d > k$	
if $s < d$	$0^{d-s}10^{s-1}$
if $s \geq d$	10^{d-1}

additional constraints on the number of leading and trailing "zeros" of $(dklr)$ sequences allow the design of more efficient block codes than provided by the technique of Tang and Bahl. Only $\beta = d$ merging bits are required for $(dklr)$ sequences, provided that l and r are suitably chosen. We shall briefly describe two constructions of fixed-length codes [7] that utilize $(dklr)$ sequences.

Construction 1: Choose the parameters d, k, r , and l such that $r + l \leq k - d$, and let the number of merging bits be $\beta = d$. Every message is associated with a $(dklr)$ sequence of length $n - d$. As a direct result of the judicious choice of l and r , the $(dklr)$ sequences can be freely cascaded without violating the specified d and k constraints provided the β merging bits are all set of "zero." For symmetry reasons we may expect that the number of $(dklr)$ sequences is maximized if the "zeros" are equally distributed between the beginning and the end of the words, that is, the "best" value for l and r are $l = \lfloor (k - \beta)/2 \rfloor$ and $r = k - \beta - l$, where $\lfloor x \rfloor$ means the largest integer in a nonnegative number x . ■

The rate-ineffectiveness of the d merging bits, which are all set to "zero," can be improved by the following construction.

Construction 2: Choose the parameters d, k , and n such that $k \geq 2d$ and $d \geq 1$. Let $r = l = k - d$ and $\beta = d$. Then the $(dklr)$ sequences can be cascaded without violating the specified (d, k) constraints if the β merging bits are governed by the following rules, which can easily be implemented. Let an $(n - d)$ -sequence end with a run of s "zeros" ($s \leq r$) while the next sequence starts with t ($t \leq l$) leading "zeros."

Table I shows the merging rule for the $\beta = d$ merging bits. Note that the shorthand notation 0^p denotes a string of p "zeros," if $p \leq 0$ the string is empty. ■

Codes constructed with Constructions 1 and 2 are block-decodable as the original $(dklr)$ sequences remain unchanged during the merging process. Decoding is done by observing the $n - \beta$ bits of the codewords that have a one-to-one relationship with the source words. The β merging bits do not contain relevant information for the decoder, and are therefore skipped. Construction 2 has a high rate compared with other systematic techniques. The next construction, due to Franaszek [11] and Gu and Fuja [10], is optimal in the sense that it maximizes the code size.

Construction by Franaszek and Gu and Fuja: The (d, k) constraints can be modeled by a finite-state machine of $k + 1$ states [11]. The crucial problem for the creation of fixed-length codes is to find a subset of the channel states, referred to as *principal states*, in one of which the channel must be at the beginning of a codeword. From any principal state, there must emanate a sufficient number of sequences that terminate at the same or other principal states. The existence of a set of principal states is a necessary and sufficient condition for the existence of a code with the specified rate and codeword length. Franaszek [11] developed a recursive elimination technique for determining the existence of a set of principal states through operations on the connection matrix. Gu and Fuja presented an explicit solution to the above recursive elimination method. They demonstrated (by construction) that the size of the largest block-decodable (d, k) code equals the number of (d, k) sequences of length n with at least d leading "zeros" and at most $k - 1$ trailing "zeros." In the technique suggested by Gu and Fuja [10], every

message is associated with a $(d, k, k - d, k - 1)$ sequence as opposed to Construction 2, where $(d, k, k - d, k - d)$ sequences are used. The number of look-up tables required for encoding and decoding is at most three. Their approach demonstrated that Construction 1 is optimal if $d = 0$, and furthermore that Construction 2 is optimal if $d = 1$, but is, however, suboptimal for $d > 1$. ■

III. DESCRIPTION OF THE NEW CONSTRUCTIONS

We will now discuss three general code constructions of increasing complexity. The constructions are based on the representation of source words by $(dklr)$ sequences of length $n - \beta$, which are cascaded using $\beta = d - 1, d > 1$, or $\beta = d - 2, d > 4$, merging bits.

Construction 3: Choose the parameters $d, d \geq 2, k$ and n , and let the number of merging bits be $\beta = d - 1$. We have to choose k so that $k - \lfloor d/2 \rfloor > d$. The parameters r and l are specified by $l + r = k - \beta$. For reasons of symmetry, the set of permitted words is maximized if $r = \lfloor (k - \beta)/2 \rfloor$ and $l = (k - \beta - r)$. Then the $(d, k - \lfloor d/2 \rfloor, l, r)$ sequences of length $n - \beta$, excluding the words $0^{n-\beta}$ and $0^{n-\beta-1}1$, can be cascaded with a simple rule. Note that these two words are excluded to insure that the juxtaposition of more than two codewords does not produce a potential problem. This condition is merely imposed for convenience as codes can indeed be built with the excluded words [12]–[14].

The merging operation runs as follows. Set the β merging bits to "zero." If both the last bit of the current word and the first bit of the upcoming word are equal to "one," i.e., the d constraint would be violated after a concatenation, then take the following action:

- set both these bits equal to "zero"
- set the merging bit at merging bit position $a = \lfloor d/2 \rfloor$ equal to "one."

The merging bit at position a is termed the *pivot bit*, and the operation described above is called a *pivoting operation*. The pivoting operation provides that the two "ones" that are too close, will be replaced by a single "one" at merging bit position a . As a result of the pivoting operation, the leftmost "one" of the upcoming word is "shifted" $\lfloor d/2 \rfloor$ positions, which is the reason for restricting the codebook to $(dklr)$ sequences with a maximum runlength equal to $k - \lfloor d/2 \rfloor$. Generally speaking, this constraint is too tight, as it is in force for the entire word. It is easily seen that, for d even, it is sufficient to exclude from the set of $(dklr)$ sequences those sequences that start with the string $10^p, p > k - d/2$, or end with $0^p 1, p > k - d/2$. For odd d , it is sufficient to bar those sequences that start with $10^p, p > k - (d - 1)/2$, or end with $0^p 1, p > k - (d + 1)/2$. The reason for restricting the code set to $(d, k - \lfloor d/2 \rfloor, l, r)$ sequences is that this set has the desirable property that it consists of $(dklr)$ sequences only, which permit enumerative encoding and decoding. Of course, if the usage of enumerative coding is not contemplated, then in certain instances a more efficient code can be designed by the less restrictive conditions.

In contrast with both Constructions 1 and 2, where during the decoding operation, all merging bits are skipped by the decoder, decoding is done here by observing $n - \beta$ bits of the received word plus the pivot bits preceding and succeeding the actual codeword. That is, a window of $n - d + 3$ bits is required for the decoding function. A logic array, which embodies the inverse of the encoding function, can readily be used for translating the $n - d + 3$ bits into the recovered m -bit source symbols. If enumerative coding is employed, then decoding is done in two steps. With a small logic array the $n - d + 3$ bits can be uniquely reconstituted into the corresponding $(n - \beta)$ -bit $(dklr)$ sequence, which, in turn, can be translated, using enumerative coding, into the recovered m -bit source tuples. It should be appreciated that merging bits, except the pivot bit, can be skipped.

TABLE II
MERGING RULE OF $(dklr)$ SEQUENCES

s, t	Merging bits
$s = t = 0$	$0^{a-1}10^{d-a-1}$
$1 \leq s + t \leq k - d + 1$	0^{d-1}
$s + t > k - d + 1$	
if $s < d \cap d - s \neq a - 1$	$0^{d-s}10^{s-2}$
if $s < d \cap d - s = a - 1$	0^s10^{d-a-2}
if $s \geq d > 3$	10^{d-2}
if $s \geq d = 3$	01

As the code format involves the translation of source words into $(dklr)$ sequences, the new codes permit enumerative coding techniques to simplify encoding and decoding hardware. Note that the removal of the two words 0^{n-d+1} and $0^{n-d}1$ does not destroy the simple lexicographical ordering of the $(dklr)$ sequences. Enumerative encoding and decoding is particularly attractive when the codeword length is relatively large. A description of enumerative coding of $(dklr)$ sequences can be found in [7].

It is immediate from the construction above that the code size of d -constrained sequences, $d \geq 2$, k infinite, equals $N_d(n-d+1)$, $n \geq 2d$, where $N_d(n)$ denotes the number of d -constrained sequences of length n . In this respect it is of interest to note that the code size of block-decodable d -constrained codes equals $N_d(n-d)$, which shows a clear indication of the improvement. For large n we have $N_d(n) \approx a\lambda^n$, where a is a constant independent of n , and λ is the largest real root of

$$\lambda^{d+1} = 1 - \lambda^d. \quad (1)$$

It is clear from the above that the gain in code size is approximately λ . For $d = 2$, for example, we have $\lambda \approx 1.47$, and we conclude that the code size increases by almost 50%.

In the previous constructions, one of the β merging bits is used as a pivot bit, and the remaining $\beta - 1$ merging bits are set of to "zero." In the next construction, called Construction 4, for $d \geq 2$, these "unused" merging bits can be exploited, in line with Construction 2, for constraining the maximum runlength. The rate effectiveness of Construction 3 can thus be improved with slightly more complex merging rule.

Construction 4: Choose the parameters $d, d \geq 3, k$, and $n, k > 2d$, and let the number of merging bits be $\beta = d - 1$. We have to choose k so that $k - \lfloor d/2 \rfloor > d$. The parameters r and l are given by $r = k - d - 1$ and $l = k - d$.

The merging bits are a function of the number of "zeros" preceding and succeeding them. In order to define this function, let s be the number of trailing "zeros" of the current $(dklr)$ sequence, and let t be the number of leading "zeros" of the upcoming $(dklr)$ sequence. Then $(d, k - \lfloor d/2 \rfloor, l, r)$ sequences of length $n - \beta$, excluding the all-0's word, can be cascaded by choosing the merging bits as listed in Table II. It should be noted that the position a of the pivot bit is given by $a = \lfloor d/2 \rfloor$. If the last bit of the current word and the first bit of the upcoming word are both equal to "one," i.e., $s = t = 0$ then a pivot operation is performed, i.e., both the last bit of the current actual codeword are set to "zero."

Essentially, all $\beta = d - 1$ merging bits are set to "zero," unless the d - or k -constraint would be violated. In the former case, the pivot bit at merging bit position a is set to "one," and in the latter case a properly chosen merging bit, not being the pivot bit at position a , is set to "one." Table II displays a fixed rule, but it should be understood that there is a certain degree of freedom to position the "one." The degree of freedom offered can, if required, be utilized for specific purposes, such as synchronization or the minimization of the signal power at given frequencies.

Decoding is done, as described in Construction 3, observing $n - \beta$ bits of the received word plus the pivot bits preceding and succeeding them. All merging bits except the pivot bit can be skipped. ■

In the preceding construction, one of the $\beta = d - 1$ merging bits is used as pivot bit, and we exploited, for $d > 2$, the surplus of merging bits for constraining the maximum runlength. The next construction, which is valid for $d > 4$, employs only $\beta = d - 2$ merging bits; three of them are pivot bits.

Construction 5: Choose the parameters $d, d \geq 5, k$, and n , and let the number of merging bits be $\beta = d - 2$. We have to choose k so that $k - \lfloor d/2 \rfloor > d$. The parameters r and l are given by $r = \lfloor (k - \beta)/2 \rfloor$ and $l = (k - r - \beta)$. Then the $(d, k - \lfloor d/2 \rfloor, l, r)$ sequences of length $n - \beta$, excluding the words $0^{n-\beta}, 0^{n-\beta-1}1$, and $0^{n-\beta-2}10$, can be cascaded with a simple rule. In this construction, three of the β merging bits act as pivot bits. The positions of the pivot bits are denoted by a_1, a_2 and a_3 . The basic idea is that the pivot bits can be set to "one" if the d -constraint would be violated during the cascading operation. The pivot bits are chosen, for reasons of symmetry, as close to the middle of the β merging bits as possible. Let s be the number of trailing "zeros" of the current $(dklr)$ sequence, and let t be the number of leading "zeros" of the upcoming $(dklr)$ sequence. Then, as can easily be verified, there are three combinations where during a cascading operation the minimum runlength could be violated, namely, $(s = t = 0), (s = 1, t = 0)$, and $(s = 0, t = 1)$. If the minimum runlength is in danger of becoming too short, a violation of the d -constraint can be circumvented by executing a pivoting operation involving one of the three pivot bits. If, for example, $s = t = 0$, then pivoting is performed with the pivot bit at position a_1 (by setting it to "one" and by setting the "ones" at the beginning and end of the adjoining $(dklr)$ sequences to "zero"). If $(s = 1, t = 0)$, the pivoting operation is carried out with the pivot bit at position a_2 , etc. The possible execution of one of the three distinct pivoting operations can be uniquely decoded by observing the pivot bits, and an action, at the receiver's end, undertaken to restore the original $(dklr)$ sequences is easily performed. Thereafter, decoding is as usual. It should be appreciated that all merging bits, except the three pivot bits, can be skipped. ■

It is not difficult to see that the above construction can be generalized. For $d \geq 8$, it is possible to use a construction with five pivot bits and in total $d - 3$ merging bits. Extended study of these generalizations has not been conducted as the values of the minimum runlength d seem impractically large to justify the effort.

IV. RESULTS

The algorithm outlined in the previous section has been implemented and successfully applied to find new almost block-decodable codes. Our effort has been focused on the finding of good candidate codes that are byte-oriented. Selected results of our investigations are listed in Table III. The parameter $C(d, k)$ is the capacity of Shannon's discrete noiseless runlength-limited channel. The code efficiency η is defined as the quotient of the information rate $R = m/n$ and the capacity $C(d, k)$ of the noiseless runlength-limited channel, that is

$$\eta = R/C(d, k).$$

The capacity $C(d, k)$ is equal to $\log_2(\lambda)$, where λ is the largest root of

$$\lambda^{k+1} = 1 + \lambda + \dots + \lambda^{k-d}. \quad (2)$$

It can be seen that the efficiency of the codes is better than 90%. Of particular interest in this respect is the (2, 16), rate 8/15 code achieving an efficiency of 97%. In order to compare the results with other construction methods, we have listed in Table III are block-decodable, while the new codes listed in Table IV parameters of

TABLE III
CODES BASED ON NEW CONSTRUCTIONS

d	k	n	R	$C(d, k)$	$\eta = R/C(d, k)$	Construction
2	16	15	8/15	0.551	0.97	3
2	9	16	8/16	0.537	0.93	3
3	11	19	8/19	0.464	0.91	4
4	14	22	8/22	0.397	0.92	4
5	20	24	8/24	0.359	0.93	5

TABLE IV
BLOCK-DECODABLE CODES BASED ON CONSTRUCTION 2

d	k	n	R	C	$\eta = R/C$
2	10	16	8/16	0.542	0.92
3	10	20	8/20	0.446	0.90
4	12	23	8/23	0.389	0.90

block-decodable codes which were designed by applying Construction 2. The codes, taken from [7], are also byte-oriented. Both the codeword length n and maximum runlength k were selected in such a way that the information rate R was maximized. As remarked earlier, Construction 2 is not optimal for $d > 1$, and an attempt was made to improve the efficiency by applying Gu and Fuja's method. It revealed that, for the selected values given in Table IV, only the $d = 3$ block-decodable code can be improved with respect to its rate efficiency. The optimal method enables the design of a (3, 9), rate 8/20 code. A comparison of Tables III and IV shows that the rate efficiency of the codes constructed with the new methods is a few percent better than that of codes built with Construction 2. Of course, we should keep in mind that the codes listed in Table IV are block-decoded, while the new codes listed in Table III may suffer from a slight error propagation.

Of practical interest is the byte-oriented (2, 9), rate 8/16, code found by application of Construction 3. This code has a slightly smaller maximum runlength than the rate (2, 10), rate 8/16, block-decodable code that can be established with Construction 2. It is the difficult task of the system architect to weigh the smaller maximum runlength and increased risk of error propagation of the former code against the larger maximum runlength and absence of error propagation of the latter code. The answer to this question depends on a number of factors peculiar to a particular usage which are beyond the scope of this correspondence.

V. CONCLUSIONS

We have developed new constructions for systematically designing (d, k) codes. Possible advantages of the new codes are that a) the translation of source words into codewords can be accomplished with a single look-up table, b) the cascading of the codewords can be done with a simple merging rule, and c) their structure permits enumerative coding techniques to simplify encoding and decoding hardware, which is a particularly desirable feature when the codeword length is relatively large. Various new look-ahead codes with relatively short block lengths have been unveiled that are simpler to implement than conventional (d, k) block codes currently being used. The new codes are very "close" to block-decodable codes, as decoding can be accomplished by observing (part of) the received codeword plus a small part of the previous codeword. As a result, minor error propagation may occur during decoding. Of specific interest are the byte-oriented (2, 9), rate 8/16, code and the (2, 16), rate 8/15 code both found by Construction 3.

REFERENCES

- [1] K. A. S. Immink, *Coding Techniques for Digital Recorders*. Englewood Cliffs, NJ: Prentice-Hall International (UK) Ltd., 1991.
- [2] T. D. Howell, "Statistical properties of selected recording codes," *IBM J. Res. Develop.*, vol. 33, no. 1, pp. 60-73, Jan. 1989.
- [3] P. A. Franaszek, "On future-dependent block coding for input-restricted channels," *IBM J. Res. Develop.*, vol. 23, pp. 75-81, 1979.
- [4] —, "Construction of bounded delay codes for discrete noiseless channels," *IBM J. Res. Develop.*, vol. 26, no. 4, pp. 506-514, July 1982.
- [5] R. L. Adler, D. Coppersmith, and M. Hassner, "Algorithms for sliding block codes. An application of symbolic dynamics to information theory," *IEEE Trans. Inform. Theory*, vol. IT-29, no. 1, pp. 5-22, Jan. 1983.
- [6] D. T. Tang and L. R. Bahl, "Block codes for a class of constrained noiseless channels," *Inform. Contr.*, vol. 17, pp. 436-461, 1970.
- [7] G. F. M. Beenker and K. A. S. Immink, "A generalized method for encoding and decoding runlength-limited binary sequences," *IEEE Trans. Inform. Theory*, vol. IT-29, no. 5, pp. 751-754, Sept. 1983.
- [8] J. H. Weber and K. A. S. Abdel-Ghaffar, "Cascading runlength-limited sequences," *IEEE Trans. Inform. Theory*, vol. 39, no. 6, pp. 1976-1984, Nov. 1993.
- [9] K. A. S. Immink, "Block-decodable runlength-limited codes via look-ahead technique," *Philips J. Res.*, vol. 46, no. 6, pp. 293-310, 1991.
- [10] J. Gu and T. Fuja, "A new approach to constructing optimal block codes for runlength-limited channels," *IEEE Trans. Inform. Theory*, vol. 40, no. 3, pp. 774-785, May 1994.
- [11] P. A. Franaszek, "Sequence-state encoding for digital transmission," *Bell Syst. Tech. J.*, vol. 47, pp. 143-157, Jan. 1968.
- [12] G. V. Jacoby, "A new look-ahead code for increasing data density," *IEEE Trans. Magn.*, vol. MAG-13, no. 5, pp. 1202-1204, Sept. 1977. See also GB Patent 1590404, June 1981.
- [13] S. Tanaka, "Method and apparatus for encoding binary signals," EP Patent 0 178 813, Oct. 1985.
- [14] S. B. McClelland, "Compatible digital magnetic recording system," U.S. Patent 4 261 019, Apr. 1981.

Minimum Average Cost Testing for Partially Ordered Components

Marc J. Lipman and Julia Abrahams

Abstract—The problem of designing a sequence of optimal binary tests for the identification of a single faulty component is addressed. For components in linear order this is equivalent to the classical alphabetic coding problem solved by Hu and Tucker. For partially ordered components the problem is solved by reduction to a minimization over a set of alphabetic problems.

Index Terms—Variable-length source codes, partial orders, Hu-Tucker algorithm, combinatorial search.

I. INTRODUCTION

The problem of searching for a defect in a pipeline, for instance, is equivalent to the problem of constructing optimal binary variable-length source codes subject to a linear order restriction and has been well studied [1]. The pipeline test problem consists of a finite number

Manuscript received February 18, 1994; revised May 23, 1994. This paper was presented at the International Symposium on Information Theory, San Antonio, TX, 1993.

The authors are with the Mathematical, Computer, and Information Sciences Division, Office of Naval Research, Arlington, VA 22217-5660 USA.

IEEE Log Number 9406734.