Synchronization using Insertion/Deletion Correcting Permutation Codes

Ling Cheng, Theo G. Swart and Hendrik C. Ferreira

Department of Electrical and Electronic Engineering Science
University of Johannesburg, P.O. Box 524,
Auckland Park, 2006, South Africa
E-mail: lcheng@postgrad.uj.ac.za, tgswart@postgrad.uj.ac.za, hcferreira@uj.ac.za

***Abstract*— In this paper, we present a fast synchronization coding scheme, which uses single insertion/deletion error correcting permutation codes. A possible application to M-ary FSK for the CENELEC A band power-line communications (PLC) is considered. Compared to conventional timing recovery schemes, no redundancies for preamble sequences and no processing delays from decision devices are needed.**

I. Introduction

Synchronization is an important issue for the design of a reliable communication system. Most of the investigations on error correcting codes only consider additive errors. Nevertheless, when messages are transmitted through an asynchronous channel, the message received may have a different size compared to that of the message sent. Thus, insertion and deletion errors are defined for a channel having synchronization problems. An insertion is the transform whereby one symbol is added at an unknown index in the message during transmission, which results in the increase of the message size by 1. A deletion is the transform whereby one symbol is dropped off the message during transmission, which results in the decrease of the message size by 1.

Taking a pulse amplitude modulation (PAM) system into consideration, the signal arriving at the receiver can be presented as

$$r(t) = \sum_i a_i h(t - iT - \tau_i) + n(t), \qquad (1)$$

where $a_i \in \{+1, -1\}$ is the i-th bit and $h(\cdot)$ is the pulse shape. In (1), τ_i is the uncertainty in the timing at the i-th bit, and $n(\cdot)$ is the additive noise. Clearly, if the estimation of the timing at the receiver is not precise, insertion or deletion errors are introduced. It is important to mention that, usually before the received sequence is resynchronized, the message is corrupted by the error propagation due to insertion/deletion errors and is totally useless to the receiver.

Most of the conventional communications systems are designed to work under very low signal-to-noise-ratios (SNR) and at high transmission rates, which demand more reliable timing recovery techniques to cooperate. In practice, nearly all existing synchronization schemes are based on phase-locked loops (PLL), which requires processing delay and depends on the reliability of the decision devices. Clearly, without efficient coding techniques, it is almost impossible for the traditional synchronization schemes based on PLL to work under very low SNR. It prompts more state-of-the-art synchronization schemes, e.g. turbo-like approaches and iterative timing-recovery schemes [1], to be investigated.

Permutation codes combined with M-ary FSK modulation are considered to combat additive noise, impulse noise and permanent frequency disturbances. In [2] and [3] the authors show that they are good candidates for narrow-band PLC. Some performance simulations for these codes can be found in [4]. In this paper, the insertion/deletion error correcting capability of the permutation codes is investigated. A real-time synchronization scheme is designed, based on the permutation codes, which can correct single insertion/deletion errors in each code word. This limits the error propagation and reduces the delays of the resynchronization process.

The paper is organized as follows: A brief introduction to the permutation codes and Tenengolts' non-binary single insertion/deletion error correcting codes are presented in Section II. The construction of single insertion/deletion permutation codes is introduced in Section III. An approach to alleviate the known-boundary limitation of this coding scheme is presented in Section IV. We present a modified dynamic information inference (decoding) algorithm to correct insertion/deletion/substitution errors by using the new type of single insertion/deletion permutation codes in Section V. We also provide computer simulation results to demonstrate the performance. We conclude the paper in Section VI.

It is worth mentioning that M-ary FSK modulation is preferred for narrow-band PLC in the CENELEC A band in our case, however the results will also be applicable to other M-ary schemes such as PAM. Much research has been done in the field of broadband PLC, but little documentation can be found in the low frequency range (below 100 kHz). In this range communication is considered with a low rate that can provide very high accuracy, for applications such as automatic meter reading and demand side management.

II. Preliminaries

A. Permutation Trellis Codes and M-ary FSK

The definition for a permutation code is as follows:

978-1-4244-1976-0/08/$25.00 ©2008 IEEE.

Definition 1: A permutation code $\mathcal{C}$ consists of $|\mathcal{C}|$ code words of length M, where every code word contains the M different integers $1, 2, \ldots, M$ as symbols.

In the M-ary FSK system every symbol corresponds uniquely to a frequency from an M-FSK modulator and the M-ary symbols are then transmitted in time as the corresponding frequencies. A more detailed explanation of the system and how different types of noise on the power-line affects it, can be found in [2] and [3]. In this paper we will focus on insertion/deletion errors.

Since decoding of permutation codes can be difficult in this scenario, an approach is used whereby the convolutional code's error correcting capabilities are mapped to the permutation codes. In [3] it is described how permutation trellis codes can be created by using distance-preserving mappings. Briefly, the outputs of a binary convolutional encoder are mapped to the code words from a permutation code. A mapping consists of choosing an ordered subset of 2^n M-tuples, from the full set of permutation M-tuples, to map to the corresponding convolutional base code's n-tuples. The subset is chosen such that the Hamming distance between any two permutation M-tuples is at least as large as the distance between the corresponding convolutional code's output n-tuples which are mapped to them. This results in a permutation trellis that can be decoded using the Viterbi algorithm.

Using some of the properties that we will discuss shortly, permutation trellis codes can be designed that have insertion/deletion correcting capabilities as well.

B. Tenengolts' non-binary single insertion/deletion correcting codes

In [5], Tenengolts presented a class of non-binary single insertion or deletion error correcting codes, which we will in short refer to as Tenengolts codes. In the construction of the Tenengolts code, the relation rule,

$$\alpha_i = \begin{cases} 1, & \text{if } x_{i+1} \geq x_i, \\ 0, & \text{if } x_{i+1} < x_i. \end{cases} \tag{2}$$

is first applied to convert the non-binary codeword to a binary sequence, which has the same length as that of the non-binary code word.

Provided that the corresponding binary sequence satisfies the selection (partition) criterion of the binary one insertion/deletion error correcting code studied by Levenshtein [6], it can correct one insertion or deletion error. The first-order moment function used to construct the single insertion/deletion error correcting can be presented as follows

$$\sigma = \sum_{i=1}^{n} i\alpha_i \equiv \gamma \pmod{m}, \tag{3}$$

where γ and m are fixed nonnegative integers. If $m \geq n+1$, this code is a single insertion or deletion error correcting code. Note that the first-order moment function was first investigated by Varshamov and Tenengolts in [7] for correcting asymmetric errors. Later, Levenshtein [8] found that the Varshamov-Tenengolts codes could be used to correct single insertion/deletion error. For a non-binary one insertion or deletion error correcting code word $\mathbf{x} = x_1 x_2 \ldots x_n$, where $x_i \in \{0, 1, \ldots, q-1\}$ and q is the alphabet size, the second selection criterion is stipulated for each codeword as

$$\sum_{i=1}^{n} x_i \equiv \beta \pmod{q}. \tag{4}$$

When a symbol is deleted or inserted, a bit in the corresponding binary sequence is also deleted or inserted. Based on the first criterion, the insertion/deletion error is corrected in this binary sequence. By using an inverse process of the relation rule in (2), the position of the symbol deleted or inserted in the non-binary codeword is located. Then the value of the non-binary symbol inserted or deleted is retrieved by using the second selection criterion.

III. Single Insertion/Deletion Error Correcting Permutation Codes

It follows naturally that a permutation code satisfies the second criterion of the Tenengolts code.

Theorem 1: For a permutation code word $\mathbf{x} = x_1 x_2 \ldots x_n$, we have

$$\sum_{i=1}^{n} x_i = \frac{M(M+1)}{2}, \tag{5}$$

where $x_i \in \{1, 2, \ldots, M\}$.

When applying the first selection criterion of the Tenengolts code (3), we can obtain the permutation code that can correct a single insertion/deletion error. Let $|P_\gamma|$ denote the cardinality of a permutation code P, which satisfies (3) for γ and $m = n+1$. By observation, we find

$$|P_0| = |P_1| = \ldots = |P_\gamma| = \ldots = |P_{M-1}| = (M-1)!. \tag{6}$$

This result has been found by Levenshtein [9], however he did not consider it as a subset of the Tenengolts codes.

As known, Tenengolts codes can correct single insertion/deletion under the assumption that the boundaries of the code word are known. This assumption makes the implementation impractical. In the next section, we will introduce an approach to alleviate this limitation.

IV. Alleviation of Known-Boundary Limitation

Consider two consecutive permutation code words without knowing the boundaries. If both of the code words are not affected, and due to one deletion already taking place before these two code words, both code words shift forward by one symbol. Clearly, if framing continues at the original indices, we can immediately detect that one symbol is repeated, except in the case where the first symbols of both code words are identical. Thus, we can conclude the alleviation of known-boundary limitation as follows.

Theorem 2: A single deletion can be detected when two consecutive code words after the deletion are error-free and the first symbols of both code words are not identical.

Moreover, we also can conclude the case for insertion errors.

Theorem 3: A single insertion can be detected when the next code word after this corrupted code word is error-free and the last symbol of the error-free code word is not identical to that of the corrupted code word.

Thus, the consequential error propagation after deletion (insertion) errors can be prevented by using the permutation code, if this permutation code complies to the constraint that the first (last) symbols of two adjacent code words are different. The constraint can be illustrated by a Markov chain as shown in Fig. 1.

Consider an M-ary codeword. According to Theorem 2 and Theorem 3, as a valid codeword, the heading (tailing) symbols of two consecutive code words are not identical. Thus we can construct an M-state Markov chain. A 4-state Markov chain is shown in Fig. 1, which can be presented by a transition probability matrix as follows

$$\mathbf{Q} = \begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \end{bmatrix}$$

Assume the transition probability from one state to the other is identical. The transition probability matrix has $M-1$ nonzero elements on each of its M rows. Each element has value $\frac{1}{M-1}$. Therefore, the entropy of this Markov information source is

$$H\{X\} = \sum_{k=1}^{M} \pi_k H_k. \tag{7}$$

Since

$$\sum_{k=1}^{M} \pi_k = 1, \tag{8}$$

without losing generality, we can write

$$\pi_k = \frac{1}{M}, k \in \{1, 2, \ldots, M\}, \tag{9}$$

and

$$H_k = -\sum_{i=1}^{M} p_i \log_2 p_i = \log_2 (M-1). \tag{10}$$

Combining (7), (9) and (10), we get

$$H\{X\} = \log_2 (M-1). \tag{11}$$

This indicates the rate of code is upper bounded by

$$r \leq \log_M (M-1). \tag{12}$$

We can furthermore obtain

Theorem 4:

$$\lim_{M \to \infty} r = 1. \tag{13}$$

Therefore, we can conclude that the known-boundary limitation can be alleviated, and that, as M increases, the redundancies introduced can be negligible.

V. Simulations and Results

In this section the insertion/deletion scheme using the permutation codes is demonstrated by computer simulations.

A. Insertion/deletion/substitution channel model

The Davey-MacKay channel model [10] can be illustrated by Fig. 2. At interval t_i, the sent symbol has a probability of p_d to be deleted. This symbol cannot reach the t_{i+1} interval. At interval t_i, there is also a probability of p_i that a random symbol is inserted, and a probability of p_t that the symbol is transmitted. However, after the symbol is transmitted, the probability of error-free transmission is $1 - p_s$. We have

$$p_i + p_d + p_t = 1, \tag{14}$$

and thus the probability of error-free transmission from the t_i interval to t_{i+1} is $p_t(1 - p_s)$.

B. Modified decoding algorithm for channels with three types of errors

The lattice diagram shown in Fig. 3 is a convenient way to illustrate a stochastic insertion/deletion/substitution channel. In such a channel, messages can be corrupted by three types of errors. To deal with this channel, we need to use the dynamic algorithm to differentiate these three types of errors. The computation complexity of this algorithm is $O(N^2)$, where N is the length of the message.

In Fig. 3, according to the Davey-MacKay channel model, the branches having arrows towards the top right corner represent possible deletions, and the branches having arrows towards the bottom right corner represent possible insertions. Meanwhile, the horizontal branches represent a transmitted symbol (or block). In [11], the authors presented a method to correct garbled words based on the Levenshtein metric. The algorithm is designed based on the lattice graph. Inspired by this work, we found a modified algorithm to correct the insertion/deletion/substitution errors for permutation code words.

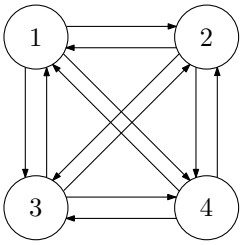

Fig. 1. $M = 4$-state Markov chain

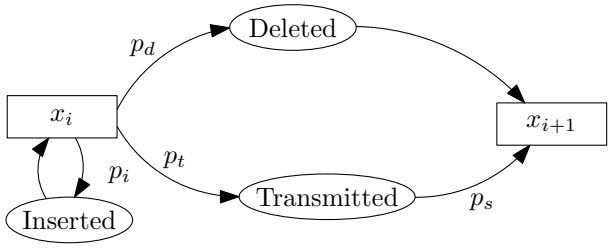

Fig. 2. Davey-MacKay insertion/deletion/substitution channel model

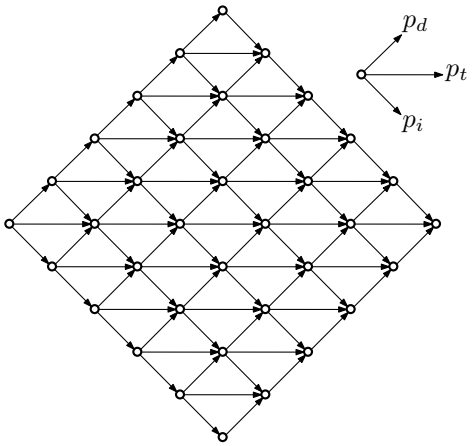

Fig. 3. Lattice diagram for the insertion/deletion/substitution channels

Before presenting the modified algorithm, we first define a function $u(\cdot)$ to count the number of unique symbols in a sequence and a function $d(\cdot)$ to measure the minimum distance for permutation code words. Let $\mathbf{x} = x_1 x_2 \dots x_M$ be a permutation code word and let $\mathbf{x}' = x'_1 x'_2 \dots x'_s$ be a corrupted sequence of $\mathbf{x}$ as a result of substitution, deletion and/or insertion errors. Define $u(\mathbf{x}')$ as the number of unique symbols in $\mathbf{x}'$. The minimum distance function $d(\cdot)$ is furthermore defined as

$$d(\mathbf{x}') = \begin{cases} s - u(\mathbf{x}') & \text{for } s \geq M, \\ M - u(\mathbf{x}') & \text{for } s < M. \end{cases} \tag{15}$$

The modified decoding algorithm can be illustrated by an example.

Example 1: Consider the message

$$\mathbf{x} = 2341\ \ 4312\ \ 3421\ \ 1243\ \ 2431$$

is sent. As a result of the deletion of symbol 1 at the seventh index and the substitution $3 \rightarrow 1$ at the ninth index, the message

$$\mathbf{x}' = 2341432142112432431$$

is received.

In the modified decoding algorithm, as shown in Fig 4, the lattice is truncated according to the number of code words sent. Each vertex in the graph represents a possible starting index of a new block (code word). When comparing Fig. 3 to Fig. 4, notice that the modified algorithm employs additional vertices only on the horizontal branches. This is due to the fact that we use block comparisons instead of symbol comparisons. According to the dynamic indices of the blocks, the received sequence is framed and allocated to the branches. Then the minimum distances $d(\cdot)$ of branches are computed. At each vertex, which has more than one branch flowing in, we set the minimum value according to the competition results of the survivor branches. The details can be referred to the add-compare-select procedure of the Viterbi algorithm [12]. The final step of this algorithm is to select the vertex, which has the minimum accumulated distance value and to trace back the

Sent: 2341 4312 3421 1243 2431
Received: 2341432142112432431
Synchronized: 2341 4321 421 1243 2431

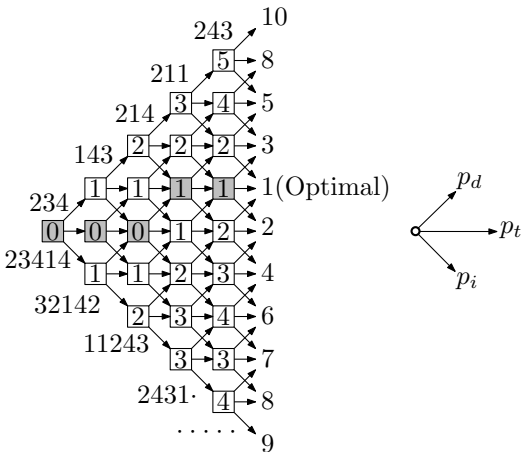

Fig. 4. Truncated lattice diagram for the insertion/deletion/substitution error correction

resynchronized sequence. As shown in Fig 4, the algorithm results in an optimal distance value 1, and the grayed vertices indicate the trace-back path.

The resynchronized sequence is

$$2341\ \ 4321\ \ 421\ \ \ 1243\ \ 2431.$$

It is found that the code word 421 has one deletion, which can be corrected by the decoding algorithm for the Tenengolts code. After completing the process, the received sequence is re-synchronized with two substitution errors remaining. Clearly, it can be solved by concatenation with an outer burst error correcting code.

We provide a brief description of the algorithm as follows:

Algorithm 1: Let $v_{t,j}$ denote a vertex in the lattice graph, where $v_{t,j} := (l_{t,j}, e_{t,j})$. Let $l_{t,j}$ denote the index of the vertex in the $t-1$'th interval which gives rise to the least accumulated errors $e_{t,j}$ at vertex $v_{t,j}$. We assume that the sequence $\mathbf{x}$ is received within T intervals. Clearly, within T intervals, the number of the symbols sent is TM. Note that, according to the structural property of the lattice graph, for any $v_{t,j}$, we have $0 \leq j \leq 2t$.

Initialization:

$$\begin{aligned} v_{0,0} &= (0, 0) \\ v_{1,0} &= (0, d(x_0 x_1 \dots x_{M-2})) \\ v_{1,1} &= (0, d(x_0 x_1 \dots x_{M-1})) \\ v_{1,2} &= (0, d(x_0 x_1 \dots x_M)) \end{aligned}$$

We define functions $\alpha(\cdot)$, $\beta(\cdot)$ and $\gamma(\cdot)$ as follows:

$$\begin{aligned} \alpha(t,j) &= e_{t-1,j-1} + d(x_{(t-1)(M-1)+j-1} \dots x_{t(M-1)+j-1}) \\ \beta(t,j) &= e_{t-1,j} + d(x_{(t-1)(M-1)+j} \dots x_{t(M-1)+j-1}) \\ \gamma(t,j) &= e_{t-1,j-2} + d(x_{(t-1)(M-1)+j-2} \dots x_{t(M-1)+j-1}) \end{aligned} \tag{16}$$

We furthermore define the functions $\Lambda(\cdot)$ and $\Delta(\cdot)$ as follows:

$$\Lambda(t,j) = \min(\alpha(t,j), \beta(t,j), \gamma(t,j)). \tag{17}$$

$$\Delta(t,j) = \begin{cases} -1 & \text{if } \Lambda(t,j) = \alpha(t,j), \\ 0 & \text{if } \Lambda(t,j) = \beta(t,j), \\ -2 & \text{if } \Lambda(t,j) = \gamma(t,j). \end{cases} \tag{18}$$

Note that if more than one condition in (18) is satisfied, we can choose one of them randomly.

Iteration: Repeat the following steps for $t = 2$ to T:

1) $l_{t,0} = 0$;
2) $e_{t,0} = e_{t-1,0} + d(x_{(t-1)(M-1)} \cdots x_{t(M-1)-1})$;
3) $l_{t,1} = 1 + \Delta(t,1)$, where $\Lambda(t,1) = \min(\alpha(t,1), \beta(t,1))$;
4) $e_{t,1} = \Lambda(t,1)$;
5) $l_{t,2t-1} = 2t - 1 + \Delta(t, 2t-1)$, where $\Lambda(t, 2t-1) = \min(\alpha(t, 2t-1), \gamma(t, 2t-1))$;
6) $e_{t,2t-1} = \Lambda(t, 2t-1)$;
7) $l_{t,2t} = 2(t-1)$;
8) $e_{t,2t} = e_{t-1,2(t-1)} + d(x_{(t-1)(M+1)} \cdots x_{t(M+1)-1})$;
9) $l_{t,j} = j + \Delta(t,j)$, where $2 < j < 2t-1$;
10) $e_{t,j} = \Lambda(t,j)$, where $2 < j < 2t-1$;

Trace-back: At the interval T, we find $e_{T,j_{\min}} = \min(e_{T,0}, e_{T,1}, \ldots, e_{T,2T})$. According to $l_{T,j_{\min}}$ of the survivor vertex $v_{T,j_{\min}}$, trace-back through the lattice and obtain the shortest path.

Note that if the boundaries of the whole sequence $\mathbf{x}$ are known, we can furthermore treat the vertices in the T'th interval before trace-back as follows:

Assume the number of the received sequence $\mathbf{x}$ is N.

For $j = 0$ to $2T$, $e_{T,j} = e_{T,j} + N - T(M-1) - j$.

C. Performance

For only one type of error, e.g. insertion or deletion errors, the computation complexity is reduced to $O(N)$. The re-synchronization process can be real-time. In our simulations, we use two comparisons to demonstrate the performance of the permutation codes with insertion/deletion error correction. The first comparison is according to the synchronization-loss rate of the system. This factor is defined as the rate of the number of the synchronized packets over the number of all the packets sent. In our simulations, for channel with one type of error, each packet has 6000 symbols. Delay is another important factor of this system to be evaluated. It is defined as the number of symbols between the index where the insertion/deletion error is detected and the index where the insertion/deletion error exactly appears within the sequence. Clearly, the delays are required to be small. The performance of this system in terms of the synchronization-loss rate and delays are illustrated in Fig. 5 and Fig. 6.

If the channel contains more than one type of error, the modified dynamic algorithm is an option to deal with it. The delay, in this case, is not negligible. Considering the complexity and the delay of this algorithm, we only demonstrate the performance when the packet size is small (600 symbols). As shown in Fig. 7, the synchronization-loss rate reaches close to 10^{-5} when insertion/deletion/substitution error rate is 5×10^{-3} for $M = 6$. The new algorithm has space for further optimization. For example, it is not necessary to consider paths far away from the current optimal path, thereby reducing the $O(N^2)$ complexity.

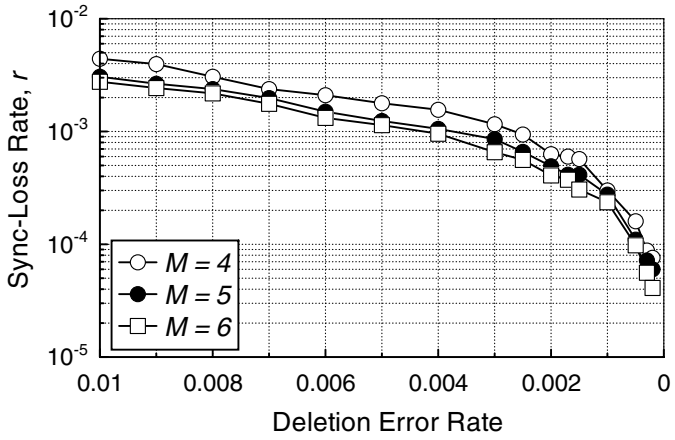

Fig. 5. Synchronization-loss rate performance as a function of the deletion error rate

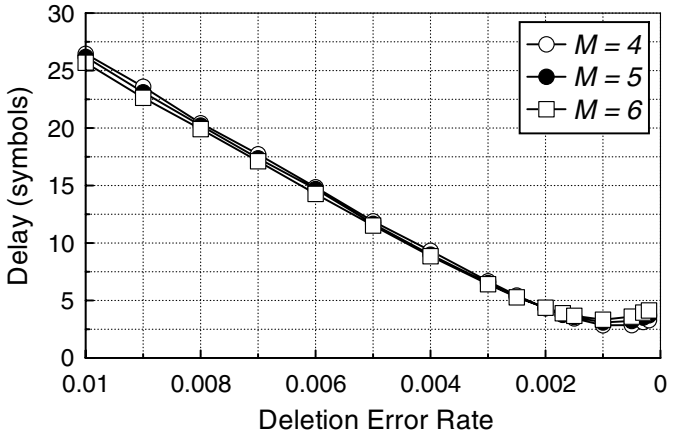

Fig. 6. Delay performance as a function of the deletion error rate

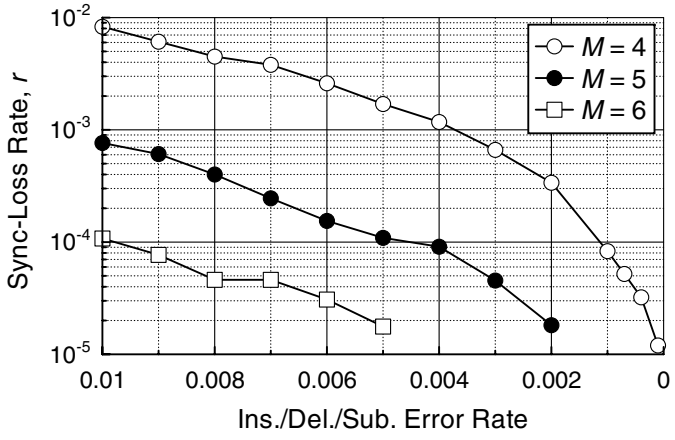

Fig. 7. Synchronization-loss rate performance as a function of the insertion/deletion/substitution error rate

VI. Conclusions

In this paper, a new type of single insertion/deletion error correcting permutation code is investigated. An approach to alleviate the limitation of the coding scheme is presented. We also prove the redundancy introduced by this approach is negligible. Furthermore, we develop a new algorithm to correct insertion/deletion/substitution errors at the same time. Computer simulation results are provided. In terms of the delays of the re-synchronization process, the redundancies, the insertion/deletion/substitution error correcting capabilities and the reliabilities of the system, this coding system is superior to conventional timing recovery schemes.

References

[1] J. R. Barry, A. Kavcic, S. W. McLaughlin, A. Nayak, and W. Zeng, "Iterative timing recovery," *IEEE Signal Processing Mag.*, pp. 89–102, Jan. 2004.

[2] A. J. H. Vinck, "Coded modulation for powerline communications," *Proc. Int. J. Elec. Commun.*, vol. 54, no. 1, pp. 45–49, 2000.

[3] H. C. Ferreira, A. J. H. Vinck, T. G. Swart and I. de Beer, "Permutation trellis codes," *IEEE Trans. Commun.*, vol. 53, no. 11, p. 1782–1789, Nov. 2005.

[4] T. G. Swart, I. de Beer, H. C. Ferreira, and A. J. H. Vinck, "Simulation results for permutation trellis codes using M-ary FSK," in *Proc. Int. Symp. on Power Line Commun. and its Applications*, Vancouver, Canada, Apr. 6–8, 2005, pp. 317–321.

[5] G. M. Tenengolts, "Nonbinary codes, correcting single deletion or insertion," *IEEE Trans. Inform. Theory*, vol. 30, no. 5, pp. 766–769, Sept. 1984.

[6] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," *Soviet Physics-Doklady*, vol. 10, no. 8, pp. 707–710, Feb. 1966.

[7] R. P. Varshamov and G. M. Tenengolts, "Correction code for single asymmetric errors," *Automat. Telemekh.*, vol. 26, pp. 288–292, 1965.

[8] V. I. Levenshtein, "Binary codes capable of correcting spurious insertions and deletions of ones," *Problemy Peredachi Informatsii*, vol. 1, no. 1, pp. 12–25, 1965.

[9] V. I. Levenshtein, "On perfect codes in deletion and insertion metric," *Discrete Mathematics and its Applications*, vol. 2, no. 3, pp. 241–258, 1992.

[10] M. C. Davey and D. J. C. MacKay, "Reliable communication over channels with insertions, deletions and substitutions," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 687–698, Feb. 2001.

[11] T. Okuda, E. Tanaka, and T. Kasai, "A method for the correction of garbled words based on the Levenshtein metric," *IEEE Trans. Comput.*, vol. c-25, no. 2, pp. 172–176, Feb. 1976.

[12] S. Lin and D. J. Costello, Jr., *Error control coding*, 2nd ed. Upper Saddle River, New Jersey: Pearson Prentice Hall, 2004.