CORE
Metadata, citation and similar papers at core.ac.uk
Provided by University of Johannesburg Institutional Repository

Decoding Distance-preserving Permutation Codes for Power-line Communications

Theo G. Swart and Hendrik C. Ferreira
Department of Electrical and Electronic Engineering Science,
University of Johannesburg, PO Box 524,
Auckland Park, 2006, South Africa
Email: ts@ing.rau.ac.za, hcferreira@uj.ac.za

***Abstract*— A new decoding method is presented for permutation codes obtained from distance-preserving mapping algorithms, used in conjunction with M-ary FSK for use on power-line channels. The new approach makes it possible for the permutation code to be used as an inner code with any other error correction code used as an outer code. The memory and number of computations necessary for this method is lower than when using a minimum distance decoding method.**

I. Introduction

Renewed interest in permutation codes was inspired by Vinck [1] who suggested using these codes for power-line communications, see also [2]. Frequencies in an M-ary FSK system are used in certain time slots to represent the permutation symbols, providing time- and frequency-diversity to overcome background noise, impulse noise and permanent frequency disturbances that are common on power-lines. This approach is used to keep the demodulator/decoder as simple as possible to keep costs and complexity down.

The construction of long permutation block codes is a difficult mathematical problem and a general decoding algorithm is not known for this application. Therefore, permutation trellis codes were introduced by Ferreira and Vinck [3] and Ferreira *et al.* [4], making use of a distance-preserving mapping (DPM) to map the binary output symbols of a convolutional code to permutation symbols.

The main advantage of using permutation trellis codes is that an alternative decoding algorithm is not needed as the well-known Viterbi algorithm is used. Also, the added error correcting capabilities of the convolutional code in addition to that of the permutation code results in good performance on very bad channels [4], [5]. Since this performance can be obtained with relatively short permutation codes, there was no need to go to longer codes when using trellis codes. However, these codes have an overall low rate, and to use higher rate convolutional codes forces one to use longer permutation codes, increasing the complexity of the trellis and decoding.

In recent times research has focused on the distance-preserving mappings themselves, with several new algorithms being proposed by Chang *et al.* [6], Lee [7]–[9] and Chang [10]. Swart *et al.* [11] considered the error correcting capabilities of these mappings and showed that an upper bound exists on the sum of the Hamming distance in such mappings. Subsequently, Swart and Ferreira [12] proposed a new multilevel algorithm, resulting in mappings that attain this upper bound for certain cases, and improves over previous mappings in all other cases. Swart *et al.* [13] showed how graphs could be used to analyze and construct permutation distance-preserving algorithms.

We propose a new decoding algorithm for permutation codes that are obtained from distance-preserving mapping algorithms and are used in conjunction with an M-ary FSK system. Although performance is sub-optimum, compared to similar permutation trellis codes, this approach is much simpler and results in a demodulator/decoder that would be cheaper and less complex.

Section II and III cover the relevant previous work in more detail, as foundation for our new work, and Section IV presents a brief motivation for using this new method. In Section V we present and illustrate our new decoding algorithm. Memory and computation comparisons as well as performance results are presented in Section VI and the conclusion is in Section VII.

II. Distance-preserving mappings

Let a binary code, $\mathcal{C}_b$, consist of $|\mathcal{C}_b|$ sequences of length n, where every sequence contains 0s and 1s as symbols. Similarly, let a permutation code, $\mathcal{C}_p$, consist of $|\mathcal{C}_p|$ sequences of length M, where every sequence contains the M different integers $1, 2, \ldots, M$ as symbols. The symmetric permutation group, S_M, consists of the sequences obtained by permuting the symbols $1, 2, \ldots, M$ in all the possible ways, with $|S_M| = M!$.

Mappings are considered where $\mathcal{C}_b$ consists of all the possible binary sequences with $|\mathcal{C}_b| = 2^n$, and $\mathcal{C}_p \subseteq S_M$ with $|\mathcal{C}_p| = |\mathcal{C}_b|$. In addition, the distances between sequences for one set are preserved amongst the sequences of the other set.

Let $\mathbf{x}_i$ be the i-th binary sequence in $\mathcal{C}_b$. The Hamming distance $d_H(\mathbf{x}_i, \mathbf{x}_j)$ is defined as the number of positions in which the two sequences $\mathbf{x}_i$ and $\mathbf{x}_j$ differ. Construct a distance matrix $\mathbf{D}$ whose entries are the distances between binary sequences in $\mathcal{C}_b$, where

$$\mathbf{D} = [d_{ij}] \quad \text{with} \quad d_{ij} = d_H(\mathbf{x}_i, \mathbf{x}_j). \tag{1}$$

Similarly, let $\mathbf{y}_i$ be the i-th permutation sequence in $\mathcal{C}_p$. The Hamming distance $d_H(\mathbf{y}_i, \mathbf{y}_j)$ is defined as the number of positions in which the two sequences $\mathbf{y}_i$ and $\mathbf{y}_j$ differ.

Construct a distance matrix $\mathbf{E}$ whose entries are the distances between permutation sequences in $\mathcal{C}_p$, where

$$\mathbf{E} = [e_{ij}] \quad \text{with} \quad e_{ij} = d_H(\mathbf{y}_i, \mathbf{y}_j). \qquad (2)$$

A DPM is created if $e_{ij} \geq d_{ij} + \delta$, $\forall i \neq j$, with equality achieved at least once. Depending on the value of δ, three different types of DPMs can be obtained. Distance-conserving mappings (DCMs) are obtained when $\delta = 0$, distance-increasing mapping (DIMs) when $\delta > 0$ and distance-reducing mappings (DRMs) when $\delta < 0$. The term distance-preserving mappings is thus used to describe all three types of mappings. See [4] for more detail.

Example 1 A possible DIM with $n = 2$ and $M = 3$ is

$$\{00, 01, 10, 11\} \rightarrow \{123, 132, 213, 231\}.$$

Using (1) and (2), for this mapping we obtain

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 0 & 2 & 1 \\ 1 & 2 & 0 & 1 \\ 2 & 1 & 1 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{E} = \begin{bmatrix} 0 & 2 & 2 & 3 \\ 2 & 0 & 3 & 2 \\ 2 & 3 & 0 & 2 \\ 3 & 2 & 2 & 0 \end{bmatrix}.$$

In this case all entries had an increase in distance, i.e. $e_{ij} \geq d_{ij} + 1$, for $i \neq j$. □

Note that from here forth we drop the subscript denoting the position of the sequence in the code. A binary sequence, $\mathbf{x} = x_1 x_2 \dots x_M$, is used as input to an algorithm, which then outputs the permutation sequence, $\mathbf{y} = y_1 y_2 \dots y_M$. This algorithm generally takes the following form

```
Input: $(x_1, x_2, \dots, x_M)$
Output: $(y_1, y_2, \dots, y_M)$
begin
    $(y_1, y_2, \dots, y_M) \leftarrow (1, 2, \dots, M)$
    for $i$ from 1 to $M$
        if $x_i = 1$ then swap$(y_{f(i)}, y_{g(i)})$
end,
```

where swap(a, b) denotes the transposition of symbols in positions a and b, and the functions $f(i)$ and $g(i)$ determine the positions of the symbols to be swapped.

In [13] it is shown how these algorithms can be represented by graphs. All the M symbol positions, y_i, are represented by placing them on a graph. Transpositions of symbols are then represented by a connecting line, x_i, between the two symbols' positions to be transposed. When $x_i = 1$, the symbols in the positions connected to the corresponding line in the graph is transposed, and this is done in the order $i = 1, 2, \dots, M$. When $x_i = 0$, the symbols are left unchanged. Initially the symbols are placed in the positions with the corresponding index, i.e. $y_i = i$, $1 \leq i \leq M$.

Example 2 The binary sequence $x_1 x_2 x_3 x_4$ is mapped to a permutation sequence $y_1 y_2 y_3 y_4$, according to the following algorithm:

```
Mapping algorithm for $M = 4$
Input: $(x_1, x_2, x_3, x_4)$
Output: $(y_1, y_2, y_3, y_4)$
begin
    $(y_1, y_2, y_3, y_4) \leftarrow (1, 2, 3, 4)$
    if $x_1 = 1$ then swap$(y_1, y_2)$
    if $x_2 = 1$ then swap$(y_3, y_4)$
    if $x_3 = 1$ then swap$(y_1, y_3)$
    if $x_4 = 1$ then swap$(y_2, y_4)$
end.
```

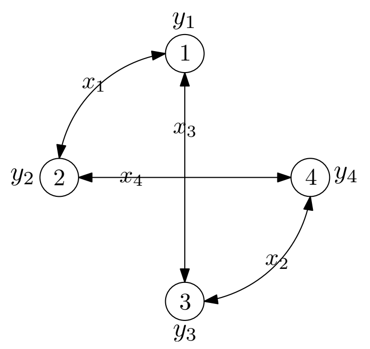

Fig. 1. DPM graph for $M = 4$

The graph in Fig. 1 is used to graphically represent this algorithm, where the following binary to permutation mapping is obtained:

$$\begin{Bmatrix} 0000, 0001, 0010, 0011 \\ 0100, 0101, 0110, 0111 \\ 1000, 1001, 1010, 1011 \\ 1100, 1101, 1110, 1111 \end{Bmatrix} \rightarrow \begin{Bmatrix} 1234, 1432, 3214, 3412 \\ 1243, 1342, 4213, 4312 \\ 2134, 2431, 3124, 3421 \\ 2143, 2341, 4123, 4321 \end{Bmatrix}.$$

□

All DPM algorithms can be represented with such a graph. In Section V we will show how these graphs can be used for decoding.

III. M-ARY FSK FOR POWER-LINE COMMUNICATION

Every permutation symbol in $\mathbf{y}$ corresponds uniquely to a frequency from an M-FSK modulator. The M-ary symbols are transmitted in time as the corresponding frequencies, thus the transmitted signal has a constant envelope.

The demodulator consists of a modified envelope detector for each frequency, that outputs a one if the signal envelope is above a certain threshold and outputs a zero otherwise. Thus for each symbol transmitted, M outputs are obtained from the demodulator. These result in an $M \times M$ binary matrix that is used for decoding, where the rows represent the frequencies used and the columns represent the position or time in the sequences.

A PLC channel may have an unpredictable and widely varying mixture of noise components, including additive background noise, impulse noise, and permanent frequency disturbers [14]. These three types of noise affect the received matrix in different ways, as will be illustrated in the following example.

Example 3 The $M = 4$ permutation code word 2341 is sent. If received correctly, the output of the demodulator would be

$$\begin{array}{ccccc} f_1 & 0 & 0 & 0 & 1 \\ f_2 & 1 & 0 & 0 & 0 \\ f_3 & 0 & 1 & 0 & 0 \\ f_4 & 0 & 0 & 1 & 0 \\ & t_1 & t_2 & t_3 & t_4 \end{array},$$

where f_i represents the output for the detector at frequency i and t_j represents the time interval j in which it occurs, for $1 \leq i, j \leq 4$.

Channel noise causes errors in the received matrix, which can be represented by the following matrices.

- *Background noise*: a one becomes a zero, or vice versa.

$$\begin{array}{cccc} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & \underline{1} \\ 0 & \underline{0} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}$$

- *Impulse noise*: a complete column is received as ones.

$$\begin{array}{cccc} 0 & 0 & \underline{1} & 1 \\ 1 & 0 & \underline{1} & 0 \\ 0 & 1 & \underline{1} & 0 \\ 0 & 0 & \underline{1} & 0 \end{array}$$

- *Permanent frequency disturbance*: a complete row is received as ones.

$$\begin{array}{cccc} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \underline{1} & \underline{1} & \underline{1} & \underline{1} \end{array}$$

□

IV. Motivation

Although permutation trellis codes provide very good performance, the trellis for high rate codes can become very complex. When using high-rate punctured convolutional codes the simplicity of decoding is lost since several time intervals of the trellis must be combined into a single time interval.

As example, choose an $R = 1/2$ convolutional code of which one bit is punctured in every second interval to obtain an $R = 2/3$ punctured convolutional code. Furthermore, choose $M = 6$ for the permutation code, then four time intervals of the punctured convolutional code must be used to achieve $n = 6$, which can be mapped to $M = 6$. However, the four time intervals must be combined into a single time interval, thereby creating an $R = 4/6$ convolutional code, since the $M = 6$ permutation code cannot be broken down into four time intervals again. The $R = 4/6$ trellis of the combined time intervals is much more complex than the equivalent $R = 2/3$ trellis of the punctured convolutional code.

By returning to a block decoder for the permutation code, codes with an overall high rate will be possible with lower complexity and the two codes are independent from each other. In fact, the permutation code is used as an inner code while any other error correcting code, such as convolutional codes, can be used as an outer code.

In addition, our new decoding algorithm uses less memory and fewer computations than traditional minimum distance decoding [2]. In this case the decoder compares the received matrix with all the possible codewords that could have been sent and the one with the minimum distance as chosen as output.

V. Decoding algorithm

We illustrate the decoding with the following examples, thereafter we formalize the decoding in an algorithm.

Example 4 We return to the algorithm of Example 2. Since a symbol in the graph is only swapped when the input bit on the corresponding branch is equal to 1, one is able to deduce from the positions which symbols are received in, which input bits would have produced such a sequence. As example, should symbol 1 be received in position 1, then from the graph it is obvious that $x_1 = 0$ and $x_3 = 0$, nothing can be deduced for the other input bits. Thus, receiving symbol 1 in position 1 tells us that the sequence could have been $0{\times}0{\times}$, where $\times$ denotes a position where the input bit is unknown.

In a similar manner, any symbol in any position is associated with a partial input sequence. Let the partial input sequence for symbol s in position p be denoted by $\hat{x}_{sp}$. Then for this algorithm, we have

Symbol 1	Symbol 2	Symbol 3	Symbol 4
$\hat{x}_{11} = 0{\times}0{\times}$	$\hat{x}_{21} = 1{\times}0{\times}$	$\hat{x}_{31} = {\times}01{\times}$	$\hat{x}_{41} = {\times}11{\times}$
$\hat{x}_{12} = 1{\times}{\times}0$	$\hat{x}_{22} = 0{\times}{\times}0$	$\hat{x}_{32} = {\times}1{\times}1$	$\hat{x}_{42} = {\times}0{\times}1$
$\hat{x}_{13} = 0{\times}1{\times}$	$\hat{x}_{23} = 1{\times}1{\times}$	$\hat{x}_{33} = {\times}00{\times}$	$\hat{x}_{43} = {\times}10{\times}$
$\hat{x}_{14} = 1{\times}{\times}1$	$\hat{x}_{24} = 0{\times}{\times}1$	$\hat{x}_{34} = {\times}1{\times}0$	$\hat{x}_{44} = {\times}0{\times}0$

Each received symbol (correct or in error) contributes to determining the input sequence by way of each partial input sequence. For instance, should we receive 3124 as in the following matrix

$$\begin{array}{cccc} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array},$$

then the partial input sequences for $\hat{x}_{31}$, $\hat{x}_{12}$, $\hat{x}_{23}$ and $\hat{x}_{44}$ would be ${\times}01{\times}$, $1{\times}{\times}0$, $1{\times}1{\times}$ and ${\times}0{\times}0$ respectively. By using majority logic on the partial sequences we can calculate what the binary sequence was.

Let p_i be the estimate of input bit x_i, $1 \leq i \leq 4$, equal to zero initially. Let the contribution for a 1 be $+1$ and for a 0 be -1, then if $p_i > 0$ then $x_i = 1$, if $p_i < 0$ then $x_i = 0$ and if $p_i = 0$ then $x_i = \varepsilon$, where ε represents an erasure. Using the partial sequences for the symbols received, we obtain $(p_1, p_2, p_3, p_4) = (+2, -2, +2, -2)$, resulting in a binary input sequence of 1010. □

When errors occur in the matrix, it will contribute to errors in the decoding, as incorrect partial sequences will be considered. Also, in the majority logic it is possible for a tie to occur, in which case a random bit must be chosen. In the case where an outer code is used in addition to the permutation code, it is beneficial to let a tie result in an erasure. We will call this method *partial permutation decoding* (PPD).

Furthermore, since an error-free matrix have a single one in each row and column, we conclude that when this property is violated, the symbols involved are less reliable. In such a case, a lower weight is assigned in the majority logic to such symbols. We will call this method *weighted partial permutation decoding* (WPPD).

Example 5 We again use the partial input sequences from the previous example for the algorithm of Example 2. Let the received matrix from the demodulator be

$$\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{array}.$$

The partial sequences are $\hat{x}_{11}$, $\hat{x}_{31}$, $\hat{x}_{12}$, $\hat{x}_{23}$, $\hat{x}_{43}$ and $\hat{x}_{44}$ which correspond to $0{\times}0{\times}$, ${\times}01{\times}$, $1{\times}{\times}0$, $1{\times}1{\times}$, ${\times}10{\times}$ and ${\times}0{\times}0$. Using PPD this would be decoded as $10\varepsilon0$.

For WPPD we count the number of ones in the same row and column as the symbol being considered. Thus, when considering $\hat{x}_{11}$, there are 2 ones in the first row and 2 ones in the first column. We subtract this from the maximum possible value for rows and columns combined, which is 8. The weight associated with $\hat{x}_{11}$ and $0{\times}0{\times}$ is therefore 4. If the partial sequence shows a zero then the weight is subtracted from the estimate, if it is a one then we add the weight to the estimate. After considering the partial input for first symbol we obtain $(p_1, p_2, p_3, p_4) = (-4, 0, -4, 0)$.

Following the same procedure, the following estimates are obtained after considering the partial input for each received symbol:

	p_1	p_2	p_3	p_4
$\hat{x}_{11} = 0{\times}0{\times}$	-4	0	-4	0
$\hat{x}_{31} = {\times}01{\times}$	-4	-5	$+1$	0
$\hat{x}_{12} = 1{\times}{\times}0$	$+1$	-5	$+1$	-5
$\hat{x}_{23} = 1{\times}1{\times}$	$+6$	-5	$+6$	-5
$\hat{x}_{43} = {\times}10{\times}$	$+6$	-1	$+2$	-5
$\hat{x}_{44} = {\times}0{\times}0$	$+6$	-6	$+2$	-10

Thus, the input sequence in this case was 1010. □

Note that a symbol positioned where impulse noise and a permanent frequency disturbance meet, as in

$$\begin{array}{cccc} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & \underline{1} & 1 & 1 \\ 0 & 1 & 0 & 1 \end{array}$$

would have zero weight associated with it, since there are 4 ones in the third row and 4 ones in the second column. This is the case for symbol 3 in position 2.

We now formalize these methods in the following algorithms.

Let $\mathbf{b}$ denote the binary received matrix from the demodulator and $\hat{\mathbf{x}}$ denote the partial input sequences for symbols received in certain positions. Let the k-th symbol in the partial input sequence $\hat{x}_{sp}$ be denoted by $\hat{x}_{sp}^{(k)}$.

```
PPD algorithm
Input: $\mathbf{b}, \hat{\mathbf{x}}$
Output: $(x_1, x_2, \ldots, x_n)$
begin
  $(p_1, p_2, \ldots, p_M) \leftarrow (0, 0, \ldots, 0)$
  for $i$ from 1 to $M$
    for $j$ from 1 to $M$
      if $b_{ij} = 1$ then
        for $k$ from 1 to $n$
          if $\hat{x}_{ij}^{(k)} = 1$ then $p_k \leftarrow p_k + 1$
          elseif $\hat{x}_{ij}^{(k)} = 0$ then $p_k \leftarrow p_k - 1$
  for $k$ from 1 to $n$
    if $p_k > 0$ then $x_k = 1$
    elseif $p_k < 0$ then $x_k = 0$
    else $x_k = \varepsilon$
end.
```

Additionally, for WPPD the number of symbols in each row and column is needed. Let r_i denote the number of ones in row i and let c_j denote the number of ones in column j.

```
WPPD algorithm
Input: $\mathbf{b}, \hat{\mathbf{x}}$
Output: $(x_1, x_2, \ldots, x_n)$
begin
  $(p_1, p_2, \ldots, p_M) \leftarrow (0, 0, \ldots, 0)$
  for $i$ from 1 to $M$
    $r_i \leftarrow b_{1i} + b_{2i} + \cdots + b_{Mi}$
    $c_i \leftarrow b_{i1} + b_{i2} + \cdots + b_{iM}$
  for $i$ from 1 to $M$
    for $j$ from 1 to $M$
      if $b_{ij} = 1$ then
        for $k$ from 1 to $n$
          if $\hat{x}_{ij}^{(k)} = 1$ then
            $p_k \leftarrow p_k + (2M - r_i - c_j)$
          elseif $\hat{x}_{ij}^{(k)} = 0$ then
            $p_k \leftarrow p_k - (2M - r_i - c_j)$
  for $k$ from 1 to $n$
    if $p_k > 0$ then $x_k = 1$
    elseif $p_k < 0$ then $x_k = 0$
    else $x_k = \varepsilon$
end.
```

VI. Comparison and Performance

Here we consider the memory and computation requirements of MDD, PPD and WPPD. For the memory requirements we do not consider memory that is necessary to do the computations. Only the information that is needed prior to the computations starting.

MDD requires the decoder to have all the possible codewords to compare with. Each codeword is a binary matrix of size $M \times M$ and if we map from all binary sequences of length n then there are 2^n possible codewords. Hence the memory requirements for MDD is $o(2^{2^n M^2})$.

For PPD and WPPD the decoder requires only the partial input sequences associated with each possible symbol in the matrix. There are $M \times M$ possible symbols that can received and each partial input sequence is of length n. (Remember that the partial input sequences are ternary sequences as these also contain the $\times$ symbol.) Hence the memory requirements for PPD and WPPD is $o(3^{nM^2})$. This is significantly less than that needed for MDD, especially for large n.

Next, we consider the number of times that a comparison is performed, as in `if...then`, as well as the number of times that a calculation, such as a sum, must be performed for each decoding type. These values are only approximates as it will vary depending on the errors that occur in the received matrix. The computations for MDD is $o(2^n M^2 + 2^n)$. The computations for PPD is $o(M^2 + nM + n)$ and the computations for WPPD is $o(M^2 + nM + n + 2M)$. For MDD the number of computations grows exponentially as n increases, while for PPD and WPPD it only grows linearly.

We use the same simple error model that was used previously [5] to evaluate different mappings. Errors are generated in the received matrix according to certain error parameters. For background noise each symbol in the received matrix has a probability, p_b, of being in error, i.e. a zero is changed to a one, or vice versa. (The error parameters were assumed to be equal for all frequency sub-bands.) For impulse noise each column in the received matrix has a probability, p_i, of resulting in an impulse noise, where the entire column's symbols are set to ones. Length restrictions compel us to limit our results to the following. In Figs. 2–7 we compare the performance for MDD, PPD and WPPD when background noise and impulse noise are present, using DCMs for $M = 4$, $M = 5$ and $M = 6$ from [12].

Fig. 2 shows the performance for an $M = 4$ DCM in the presence of background noise. The error rate for PPD and WPPD is better than that of MDD. While the error rates for PPD and WPPD are the same, the erasure rate for WPPD is lower than for PPD. Fig. 3 shows the performance for the same mapping with impulse noise. Again, the error rates for PPD and WPPD coincide and are better than that for MDD. However, in this case the erasure rate for WPPD is significantly lower than for PPD. More importantly, the erasure rate for WPPD is almost the same as the error rate for MDD. When combined with an outer code, WPPD will perform much better than MDD.

Figs. 4 and 5 show the performance for an $M = 5$ DCM in the presence of background and impulse noise respectively. For background noise similar performance patterns can be seen as for the $M = 4$ case. An unexpected result appears in the impulse noise case. While the erasure rates are as expected, the error rates for PPD and WPPD is worse than the erasure rates. In this case the problem lies with the mapping: whenever an impulse noise appears in the fifth time slot, the result will always be an error. A different mapping algorithm could possibly solve this problem.

Figs. 6 and 7 show the performance for an $M = 6$ DCM in the presence of background and impulse noise respectively. Again, in the case of background noise similar performance is observed than for the $M = 4$ and $M = 5$ mappings. For impulse noise the erasure rate for PPD and WPPD coincide, but the error rates differ substantially. PPD shows a similar trend to the $M = 5$ case where the error rate is worse than the erasure rate, while the error rate for WPPD shows a huge improvement overall.

VII. Conclusion

We presented a new decoding algorithm for permutation mappings, derived from mapping algorithms, which can be used independently from an outer code. The memory and computation requirements are also much lower than the previously used decoding method.

A possible improvement to this algorithm could be to vary the weighting according to the channel parameters. Further improvements and refinements might be possible as this algorithm was designed with the emphasis on simplicity. Some of the results also showed that certain mappings are not suited to the new decoding algorithm. The design of new mappings that can make full use of the new decoding algorithm thus presents a new challenge. Also of interest would be further performance results with an outer decoder to correct erasures.

References

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

[2] A. J. H. Vinck, J. Häring and T. Wadayama, "Coded M-FSK for power line communications," in *Proc. Int. Symp. on Inform. Theory*, Sorrento, Italy, June 25–30, 2000, p. 137.

[3] H. C. Ferreira and A. J. H. Vinck, "Interference cancellation with permutation trellis codes," in *Proc. IEEE Veh. Technol. Conf. Fall 2000*, Boston, MA, Sep. 2000, pp. 2401–2407.

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

[5] 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, BC, Canada, Apr. 6-8, 2005, pp. 317-321.

[6] J.-C. Chang, R.-J. Chen, T. Kløve and S.-C. Tsai, "Distance-preserving mappings from binary vectors to permutations," *IEEE Trans. Inf. Theory*, vol. 49, no. 4, pp. 1054–1059, Apr. 2003.

[7] K. Lee, "New distance-preserving mappings of odd length," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2539–2543, Oct. 2004.

[8] K. Lee, "Cyclic constructions of distance-preserving maps," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4392–4396, Dec. 2005.

[9] K. Lee, "Distance-increasing mappings of all lengths by simple mapping algorithms," *IEEE Trans. Inf. Theory*, vol. 52, no. 7, pp. 3344–3348, Jul. 2006.

[10] J.-C. Chang, "Distance-increasing mappings from binary vectors to permutations," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 359–363, Jan. 2005.

[11] T. G. Swart, I. de Beer and H. C. Ferreira, "On the optimality of permutation mappings," in *Proc. Int. Symp. Inf. Theory*, Adelaide, Australia, Sept. 4–9, 2005, pp. 1068–1072.

[12] T. G. Swart and H. C. Ferreira, "A generalized upper bound and a multilevel construction for distance-preserving mappings," *IEEE Trans. Inf. Theory*, vol. 52, no. 8, pp. 3685–3695, Aug. 2006.

[13] T. G. Swart, H. C. Ferreira and K. Ouahada, "Using graphs for the analysis and construction of permutation distance-preserving mappings", *IEEE Trans. Inf. Theory*, submitted for publication.

[14] H. C. Ferreira, H. M. Grove, O. Hooijen and A. J. H. Vinck, "Power line communication," in *Wiley Encyclopedia of Electrical and Electronics Engineering*, J. G. Webster, Ed. New York: Wiley, 1999, vol. 16, pp. 706–716.

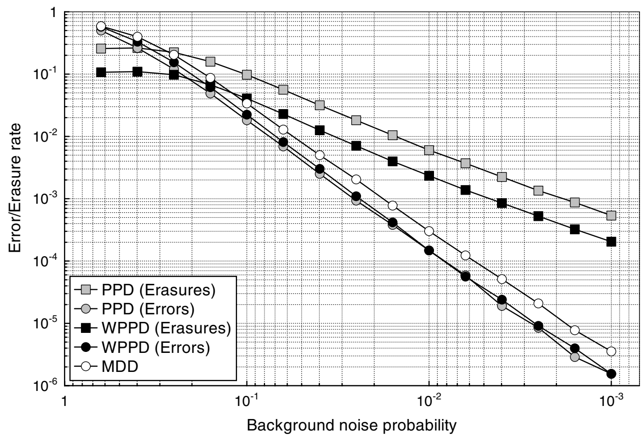

Fig. 2. Performance for $M = 4$ with background noise

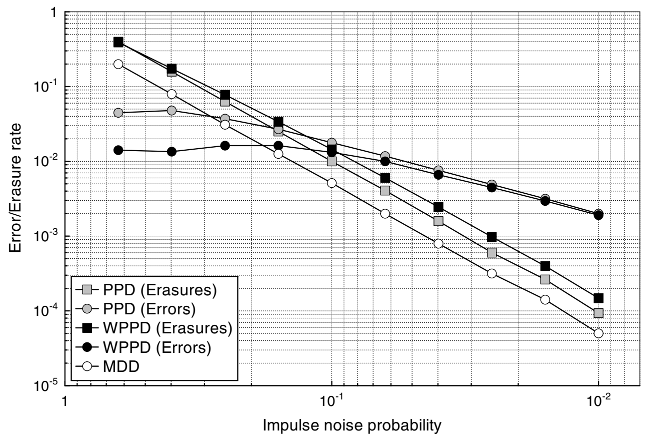

Fig. 5. Performance for $M = 5$ with impulse noise

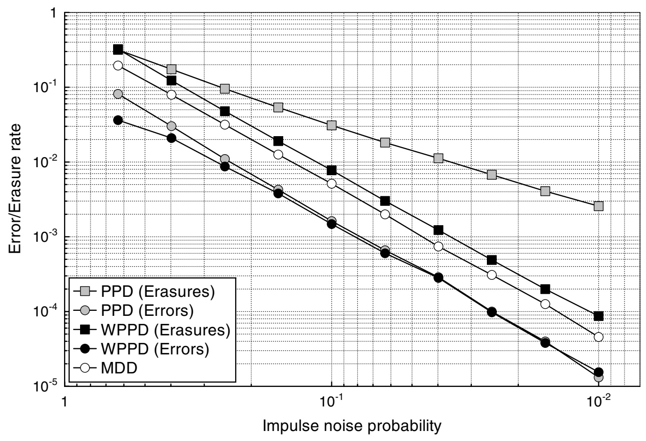

Fig. 3. Performance for $M = 4$ with impulse noise

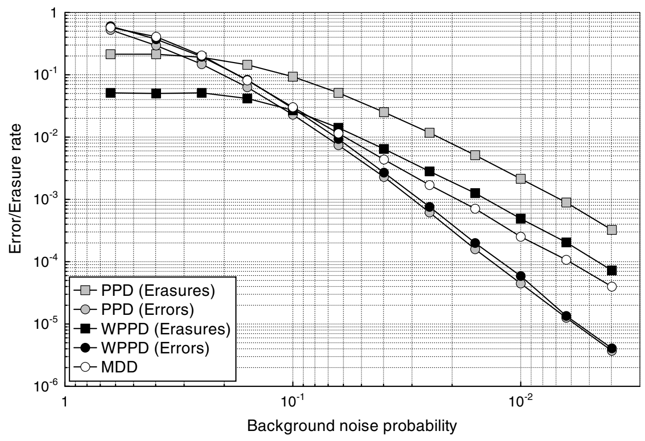

Fig. 6. Performance for $M = 6$ with background noise

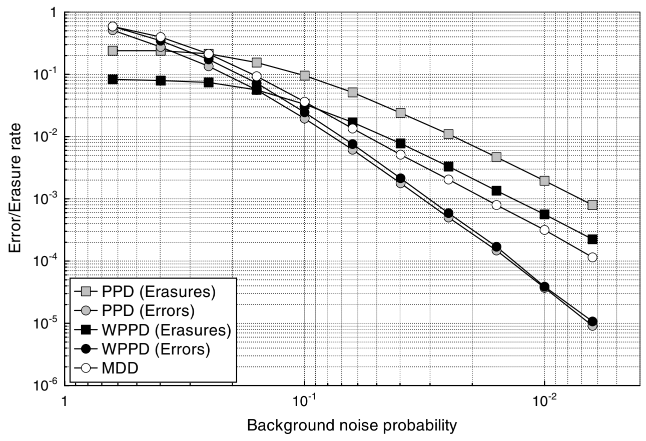

Fig. 4. Performance for $M = 5$ with background noise

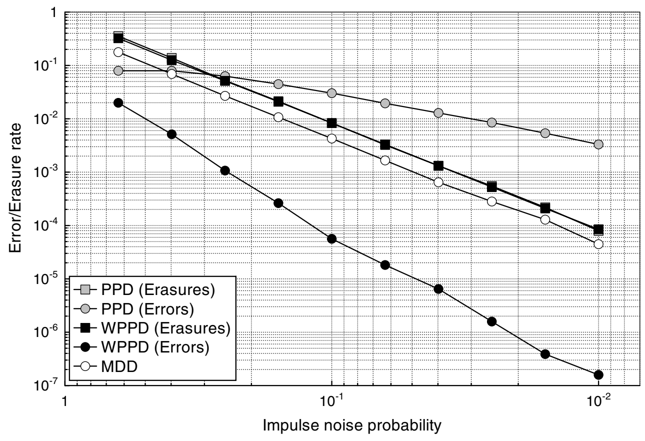

Fig. 7. Performance for $M = 6$ with impulse noise

Copyright Information

© 2007 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.