IEEE Africon 2011 - The Falls Resort and Conference Centre, Livingstone, Zambia, 13 - 15 September 2011

Correcting Adjacent Errors Using Permutation Code Trees

R. Heymann, T.G. Swart and H.C. Ferreira
Department of Electrical and Electronic Engineering Science
University of Johannesburg
P.O. Box 524
Aucklandpark, 2006
South Africa
E-mail: rheymann@uj.ac.za, tgswart@uj.ac.za, hcferreira@uj.ac.za

***Abstract*—Permutation codes are M-ary codes which can be used, in combination with M-ary FSK, to correct errors in Power Line Communications (PLC). It has been shown in [1] that permutation code trees can be used to correct a single substitution or synchronization error per codeword, without the use of markers. In this paper, we show that, due to the structure of the permutation code tree, adjacent errors can also be corrected if the codebook is adapted.**

I. Introduction

It has been shown in [1] that permutation code trees can be used to correct a single substitution or synchronization error per codeword, without the use of markers. The codewords used are permutation codes, which can be defined as follows:

Definition 1: A permutation code C consists of $|C|$ codewords of length M, where every codeword contains M different integers $1, 2, \ldots, M$ as symbols.

Permutation codes can be used, combined with M-ary FSK modulation, for error correction in Power Line Communications (PLC) [2] where every symbol in the codeword is mapped to a particular frequency. Distance preserving techniques for mapping binary codewords to permutation codewords were investigated in [3]. Mappings can be made to preserve the distance but also to increase or decrease the distance from the binary codewords to the permutation codewords. A permutation trellis is then used to decode the permutation codewords and correct errors caused by impulsive noise, background noise and permanent frequency disturbers.

Synchronization errors, modelled as insertion(s) or deletion(s) of symbols, were investigated in [4]. The error correction capability of the proposed scheme is one synchronization error per codeword, which was improved on in [5].

The problem of adjacent deletion errors is motivated in [6] and correction schemes are proposed. Burst of errors, in particular adjacent errors, occur more frequently than substitution errors or single synchronization errors in, for example, internet applications.

In this paper, it will be shown that the permutation code tree in [1] can correct more that one error per codeword, if the errors are adjacent. The codebooks from [1] should also be adjusted. It is important to note that this scheme does not make use of markers to aid the resynchronization process.

The paper is organised as follows: The basic structure of the permutation code tree used in [1], is presented in Section II so that the paper can be read independently. The error correction procedure is given in Section III and it is shown how the structure described in Section II enables the correction of adjacent errors. The permutation codebooks need to be adjusted to support the correction of adjacent errors. The codebooks are given in Section IV. Computer simulations are used to show that the system is capable of correcting adjacent errors. The results of the simulations are given in Section V, followed by final conclusions in Section VI.

II. Permutation Code Trees

Permutation codes can be represented with a tree structure, called a permutation code tree. The permutation code tree is a form of a decision tree and has a very specific structure in order for it to be used in error correction.

The tree has a root node that does not contain an element. All the other nodes in the tree each contains an element that corresponds to a symbol in a codeword. A path from the root node to an external or leaf node represents a complete codeword in the codebook. The root node has a number of children nodes, n_r, which forms the first level of the tree, and is bounded as follows:

$$1 \leq n_r \leq M. \tag{1}$$

Every node on the first level is also the root node of a subtree. The number of children nodes that a node on the first level can have is bounded as:

$$1 \leq n_s \leq M - 1. \tag{2}$$

All the nodes from the second level onwards may only have one child node.

A node cannot have the same symbol as any of its ancestors or descendants, i.e. a symbol can only appear once in any path through the tree, due to the definition of a permutation code.

A symbol can also only occur once on every level in a specific subtree.

978-1-61284-993-5/11/$26.00 ©2011 IEEE

III. Error Correction Algorithm

The algorithm will first be explained by looking at an example where only one substitution error occurs.

Codebook $Q(6)$ is a permutation codebook, with $M = 6$, that works effectively with the permutation code tree method of error correction. More information on the codebooks that can be used with this error correction method will be given in the next section.

$$Q(6) = \left\{ \begin{array}{cccc} 123456 & 132564 & 145623 & 213654 \\ 241536 & 264351 & 362415 & 354126 \\ 316524 & 463152 & 435216 & 452631 \\ 543162 & 536421 & 624513 & 615432 \end{array} \right\} \quad (3)$$

The following figure represents the partial permutation code tree based on the codebook.

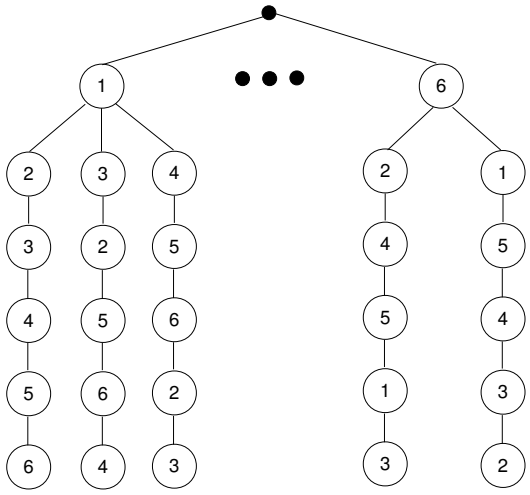

Fig. 1. Partial permutation tree

It will be assumed for now that only substitution errors can occur, in order to simplify the explanation. The first step of the algorithm is to extract the codeword that needs to be decoded from the received sequence. Let the transmitted codeword be $C_T = \{1, 2, 3, 4, 5, 6\}$ and the received codeword be $C_R = \{1, 2, 3, 3, 5, 6\}$.

The path related to the codeword C_R, will now be followed through the code tree. The algorithm starts at the root node and proceeds to the node on the first level containing a 1. From there it continues to the node containing a 2 and then the one containing a 3. According to the received codeword, the next symbol should also be a 3, but the only possible symbol in the code tree is a 4. The algorithm will thus stop at node 3. It is noted that the third level of the permutation code tree has been reached.

The error correction algorithm is a forward-backward algorithm and the same procedure is thus followed from the leaf nodes. All the leaf nodes containing symbol 6 are considered a starting point to follow the reversed path through the tree.

Fig. 2 shows the permutation code tree with the first subtree as well as all the paths through the tree ending with a symbol 6. It is clear that the path corresponding to word $C = \{1, 2, 3, 4, 5, 6\}$ has the most visited nodes and is thus assumed to be the transmitted codeword.

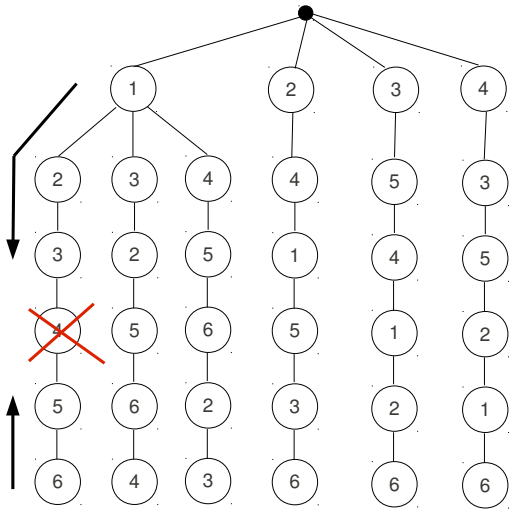

Fig. 2. Partial permutation tree showing one error

The decoding can also be done with a forward and a reversed permutation code tree. Every path through both the trees will correspond to a specific codeword. This method is faster since you only have one starting point in the backwards algorithm but you need more overhead to link each codeword to a path in the tree.

A similar method using matrices is given in [1], which can be implemented on devices with memory constraints.

If a channel is prone to substitution, deletion and insertion errors, the following procedure will be followed:

- Assume that a substitution error has occurred and extract the codeword $C_{R1} = \{x_1, x_2, \ldots, x_M\}$ from the received sequence. Follow the procedure explained above and note the number of nodes visited.
- Assume that a deletion error has occurred and extract the codeword $C_{R2} = \{x_1, x_2, \ldots, x_{M-1}\}$ from the received sequence. The same procedure can be followed as explained above and the number of nodes visited is noted again.
- Assume that an insertion error occurs and follow the same steps as for the other two errors. The only difference is that the extracted codeword will be $C_{R3} = \{x_1, x_2, \ldots, x_{M+1}\}$.
- Compare the number of nodes visited under each assumption. The assumption with the highest number of nodes visited will be assumed to be the correct assumption and thus the correct decoded word. A tie can be randomly resolved or, if one has a prior knowledge of the noise in the channel, the error type with the highest probability will be assumed to be the correct assumption.

978-1-61284-993-5/11/$26.00 ©2011 IEEE

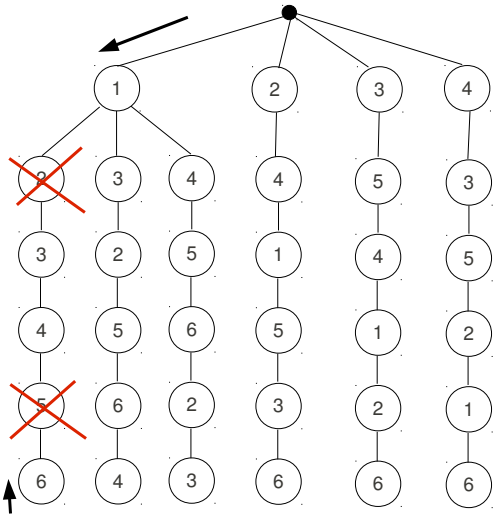

Fig. 3. Partial permutation tree showing two errors

Because this method is a forward-backward algorithm, it does not work if multiple, non-adjacent errors occur in one codeword. Figure 3 shows the scenario where two errors occur which are not adjacent.

Because the algorithm immediately stops if it cannot continue to the next level in the tree, the symbols

between the two errors will not be considered and the information cannot be used in the decoding process. There is thus insufficient information to decode the received codeword correctly. But if the errors are adjacent, as shown in Figure 4, then the algorithm will be able to correct it.

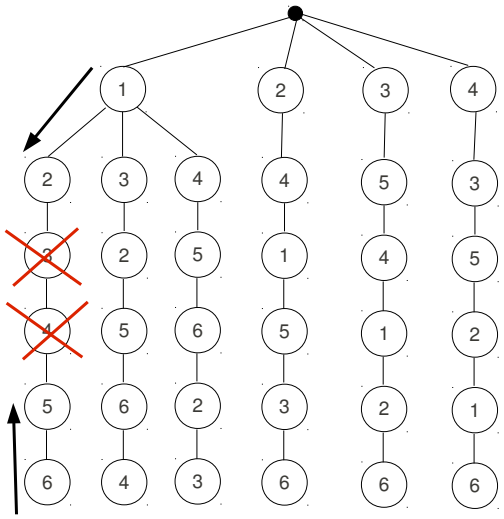

Fig. 4. Partial permutation tree showing two adjacent errors

IV. Codebooks

The codebooks in [1] were constructed using computer searches. These codebooks will only be adapted in order to facilitate the correction of adjacent errors.

Table I gives the codebooks for $M = 6$ and $M = 7$ used in [1] and Table II gives the adapted codebooks for these two M values for the adjacent error correction scheme, which is a subset of the codebooks in Table I.

TABLE I
CODEBOOKS FOR SINGLE ERROR CORRECTION

$$Q(6) = \left\{ \begin{array}{cccc} 123456 & 132564 & 145623 & 213654 \\ 241536 & 264351 & 362415 & 354126 \\ 316524 & 463152 & 435216 & 452631 \\ 543162 & 536421 & 624513 & 615432 \end{array} \right\}$$

$$Q(7) = \left\{ \begin{array}{cccc} 1362574 & 1427365 & 1536427 & 1674253 \\ 1745632 & 2463751 & 2537614 & 2714536 \\ 2156347 & 3427516 & 3514762 & 3751624 \\ 3176245 & 3265471 & 4527136 & 4613572 \\ 4251367 & 4375621 & 4736215 & 5624713 \\ 5163427 & 5217364 & 5341276 & 6735142 \\ 6142531 & 6274351 & 6451273 & 6523714 \\ 7132546 & 7216354 & 7364125 & 7425631 \end{array} \right\}$$

A computer search is used to identify the codewords in Table I which are not suitable when adjacent errors can occur. If two codewords can be confused with one another after adjacent errors occur, one of the codewords is removed from the codebook.

For example, consider the codewords $\{1, 2, 3, 4, 5, 6\}$ and $\{1, 4, 5, 6, 2, 3\}$. If an adjacent deletion occurs and the symbols $\{2, 3\}$ are deleted from either one of the codewords, it will not be possible for the decoding algorithm to determine which one of the codewords were transmitted. To avoid confusion, one of these words are removed from the codebook.

The codebooks in Table I and II were generated to show the performance of the decoding algorithms. Codebooks for higher M values have not been investigated yet.

TABLE II
CODEBOOKS FOR ADJACENT ERROR CORRECTION

$$Q(6) = \left\{ \begin{array}{cccc} 123456 & 132564 & 213654 & 362415 \\ 316524 & 452631 & 536421 & 615432 \end{array} \right\}$$

$$Q(7) = \left\{ \begin{array}{cccc} 1362574 & 1674253 & 2463751 & 2156347 \\ 3751624 & 3176245 & 4613572 & 4375621 \\ 4736215 & 5163427 & 5217364 & 6735142 \\ 6274351 & 6523714 & 7216354 & 7425631 \end{array} \right\}$$

Using computer searches to determine the codebooks with this trial-and-error method is time consuming and ineffective. A method to generate these codebooks more efficiently should be investigated. Table III also shows the cardinalities and code rates for the different codebooks. If the codebooks can be generated more effectively, higher cardinalities can be obtained and thus better code rates. The code rate is calculated by using the following equation:

$$R = \frac{\log_2 |C|}{M}. \tag{4}$$

978-1-61284-993-5/11/$26.00 ©2011 IEEE

TABLE III
CODEBOOK CARDINALITIES AND RATES

	Single errors		Adjacent errors	
M	$\|C\|$	R	$\|C\|$	R
6	16	0.67	8	0.5
7	32	0.71	16	0.57

V. SIMULATIONS AND RESULTS

The results are obtained by computer simulations. The channel is simulated to produce adjacent errors in order to determine if the decoding scheme can correct these errors. A permutation code tree as discussed previously is used to detect and correct errors introduced by the channel. The simulation is repeated a number of times to ensure a good statistical average. The word error rate (WER) is calculated to measure the success of the decoding scheme and is displayed in the figures below.

Figure 5 shows the resulting WER if adjacent deletions of burst length 2 occur, using the codebooks of Table II. Only adjacent errors occur in this simulation. The codebook with $M = 7$ performs better than the $M = 6$ codebook. If an adjacent deletion occurs in a codeword with $M = 6$ symbols, then that leaves only 4 intact symbols to decode the received word. However, with $M = 7$, the number of correct symbols after the adjacent deletions is still high enough to make a decision on what the transmitted codeword is.

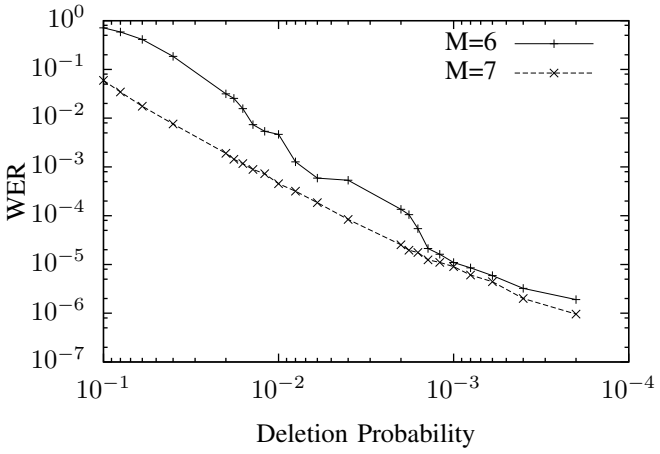

Fig. 5. Decoding performance with adjacent deletions

In Figure 6, adjacent errors as well as single errors are simulated and the $M = 7$ codebook is used. This shows that the decoding scheme can still correct single errors even though it has been extended to correct adjacent errors.

The deletion probability in Figure 6 is still the probability that a deletion error might occur. Approximately a half of these errors are forced to be adjacent, i.e. a burst of two deletions.

When a combination of single and adjacent errors occur, a misclassification error can occur. In other words, a single deletion can be mistakenly classified as an adjacent error and vice versa and thus result in a higher WER. The decoding scheme can recover from this mistake in following codewords.

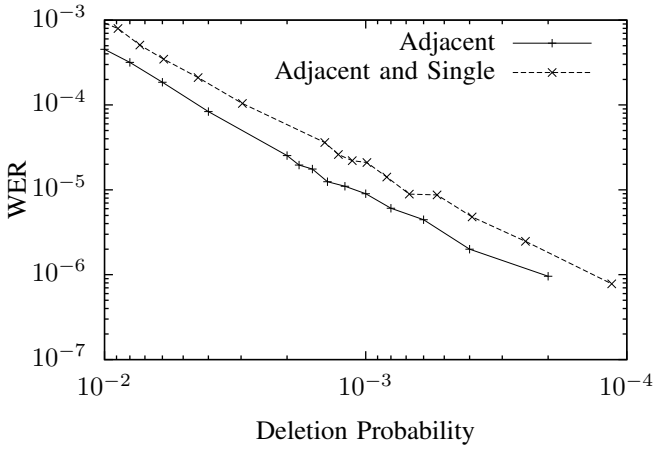

Fig. 6. Decoding performance with adjacent and single deletions

Figure 7 shows the results if two adjacent substitution errors occur.

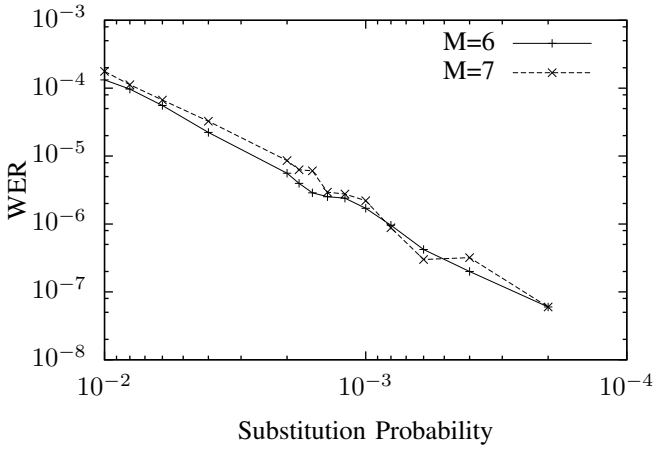

Fig. 7. Decoding performance with adjacent substitution errors

VI. CONCLUSION

The permutation code tree proposed in [1] can be used to detect and correct synchronization and substitution errors, as long as the errors are adjacent. The permutation code tree can also correct a combination of single and adjacent errors. The codebooks in [1] need to be adjusted to accommodate adjacent errors. More than one error per codeword can thus be corrected, as long as the errors are adjacent.

REFERENCES

[1] R. Heymann and H. C. Ferreira, "Using Tree Structures to Resynchronize Permutation Codes", in *Proc. Int. Symp. on Powerline Commun. and its Applications*, Rio de Janeiro, Brazil, March 28-31, 2010.

[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, pp. 1782-1789, Nov. 2005.

[4] L. Cheng, T. G. Swart and H. C. Ferreira, "Synchronization using Insertion/Deletion Correcting Permutation Codes," in *Proc. Int. Symp. on Powerline Commun. and its Applications*, Jeju Island, Korea, Apr. 2-4, 2008, pp. 135-140.

[5] L. Cheng, T. G. Swart and H. C. Ferreira, "Re-Synchronization of Permutation Codes with Viterbi-Like Decoding," in *Proc. Int. Symp. on Powerline Commun. and its Applications*, Dresden, Germany, 29 March - 1 April, 2009, pp. 36-40.

[6] L. Cheng, "Coding Techniques for Insertion/Deletion Error Correction", Doctoral Dissertation, University of Johannesburg, March 2011.

978-1-61284-993-5/11/$26.00 ©2011 IEEE