Analysis of Permutation Distance-Preserving Mappings using Graphs

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
Email: ts@ing.rau.ac.za, hcferreira@uj.ac.za

***Abstract*— A new way of analyzing permutation distance-preserving mappings is presented by making use of a graph representation. The properties necessary to make such graphs distance-preserving and how this relates to the total sum of distances that exist for such mappings, are investigated. This new knowledge is used to analyze previous constructions, as well as showing the existence or non-existence of simple algorithms for mappings attaining the upper bound on the sum of distances. Finally, two applications for such graphs are considered.**

1. Introduction

Mappings from binary sequences of length n to permutation sequences of length M are considered where the Hamming distance between sequences in one set is preserved between the respective sequences of the other set. Such mappings are referred to as distance-preserving mappings (DPMs).

Vinck [1] suggested using permutation codes for power-line communications and subsequently Ferreira and Vinck [2] used distance-preserving mappings and permutation codes to create permutation trellis codes. This provided the inspiration for more research on permutation mappings [3]–[9].

In [2] and [3] a prefix construction was suggested to create mappings, where an $M+1$-length mapping is created using an M-length mapping. It was only explicitly shown to work for $M \leq 8$, but Chang *et al.* [4] generalized this to an algorithm for any M, followed by mapping algorithms from Lee [5] and Chang [6].

Lee [7], [8] presented mapping algorithms where a graph representation was used to illustrate how the new algorithms work. These graphs led to the work we are presenting here.

Swart and Ferreira [9] presented an upper bound on the sum of the Hamming distances in a mapping, also showing that none of the previous mappings attain this upper bound, except for some trivial cases. They further proposed a multilevel construction for mappings, showing that the sum of the Hamming distances in almost all cases are larger than that of previous mappings, as well as attaining the upper bound for certain values of M. Although mappings can be obtained for $n \neq M$, see [9], we will restrict this work to the case where $n = M$.

In Section 2 we introduce definitions and notations to be used, as well as examples of mappings and their algorithms. The graph representation of permutations and DPMs is shown in Section 3, with previous constructions being used to further illustrate its use. Section 4 investigates the properties of such mapping graphs while Section 5 provides two applications for such graphs.

2. Preliminaries

We begin with a brief overview of related definitions and give a description of DPMs.

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

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

Let $|\mathbf{E}|$ be the sum of all the distances in $\mathbf{E}$, with

$$|\mathbf{E}| = \sum_{i=1}^{|\mathcal{C}_b|} \sum_{j=1}^{|\mathcal{C}_b|} e_{ij}.$$

A DPM is created if $e_{ij} \geq d_{ij}$, $\forall i \neq j$, with equality achieved at least once.

Example 1 A possible mapping of $n = 2 \rightarrow M = 3$ 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 $i \neq j$ and $|\mathbf{E}| = 28$. Note that this is only used as a simple example, as the first $n = M$ example is for $M = 4$. □

A binary sequence, $x_1 x_2 \ldots x_M$, is used as input to an algorithm, which then outputs the permutation sequence, $y_1 y_2 \ldots y_M$. This algorithm generally takes the following form

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

where $\texttt{swap}(a, b)$ 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. We call an algorithm of this form a *simple algorithm* (as used in [8]).

The ones in the binary input sequence thus determine which swaps occur to generate the permutation sequence. This can be represented with graphs, as we will do in the next section. More complex algorithms, such as the construction in [5] and Construction 2 from [6] cannot be represented in this manner.

3. Graph Representation of Permutation DPMs

All the M symbol positions are represented by placing them on a circle. Transpositions of symbols are then represented by a connecting line between the two symbols' positions to be transposed. In Fig. 1(a) the graphs for $4 \leq M \leq 9$ are shown. Note that for simplicity in the graphs we only use the position index, such as 3 instead of y_3.

Similar to choosing a subset of S_M to construct a DPM, a subset of the connecting lines in the graph is used to construct a DPM. Therefore, DPM graphs will be subgraphs of those in Fig. 1(a).

To illustrate this, we make use of the Construction 2 algorithm presented in [4], with the position function a constant 1. In Fig. 1(b) we see the graph representation of this algorithm for $4 \leq M \leq 9$. A binary sequence of $x_1 x_2 \ldots x_M$ is used as input. When $x_i = 1$, the symbols connected to the corresponding line in the graph are transposed, and this is done in the order $i = 1, 2, \ldots, M$. When $x_i = 0$, the symbols are left unchanged. The section determining swaps in the algorithm would be

```
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)
for i from 5 to M
   if x_i = 1 then swap(y_1, y_i).
```

This construction was based on the idea of the prefix construction [2] where an $M + 1$-length mapping is created from an M-length mapping. This can clearly be seen in the figure, where each successive graph makes use of the previous one, with the swap for 1 and $M + 1$ when $x_{M+1} = 1$ being added to the M-length graph.

The graphs for Constructions A–C of Lee [8] are shown in Fig. 1(c). Construction A generates mappings for M even, Construction B generates mappings for $M = 4z + 1$ and Construction C generates mappings for $M = 4z - 1$, with z some integer. We see in Fig. 1(c) that all the graphs for M even, have the same structure and the graphs for $M = 5$ and $M = 9$ have the same structure. Note that in the case of $M = 4z + 1$, x_j and x'_j labels are present. Both are used when $x_j = 1$, however, the swap for x_j is done first, followed by the swap for x'_j. For $M = 4z + 1$ it is a bit more complex: for $M = 7$ the x'_6 swap comes directly after the x_4 swap. The rest of the swaps then follow in the normal order. (Note that the input bit subscripts are relabeled, compared to the original algorithm, so they follow in the same order that the swaps must be done. This in no way affects the distance between sequences.)

The mappings found by using the multilevel construction [9] are illustrated in Fig. 1(d). A greater number of transpositions is used in this case, as opposed to those in Fig. 1(b) and (c), with certain input bits assigned to multiple swaps. This results in these mappings' $|\mathbf{E}|$-values being greater [9].

4. DPM Graph Properties

A useful property to analyze DPM graphs, is the *symbol path*. This is the possible path that a symbol will follow to appear in different positions. The following example shows the steps to obtain the symbols paths.

Example 2 Consider the path that symbol 1 can follow in the $M = 8$ multilevel mapping from Fig. 1(d). The possible paths, followed according to the input bits, are shown in Fig. 2. At first, symbol 1 only appears in position 1. If $x_1 = 1$, then $\texttt{swap}(y_1, y_2)(y_5, y_6)$ is used, and it is possible for symbol 1 to also appear in position 2. Since it does not appear in either position 5 or 6, these have no bearing on the path. For $x_2 = 1$, $\texttt{swap}(y_3, y_4)(y_7, y_8)$ is used, but symbol 1 does not

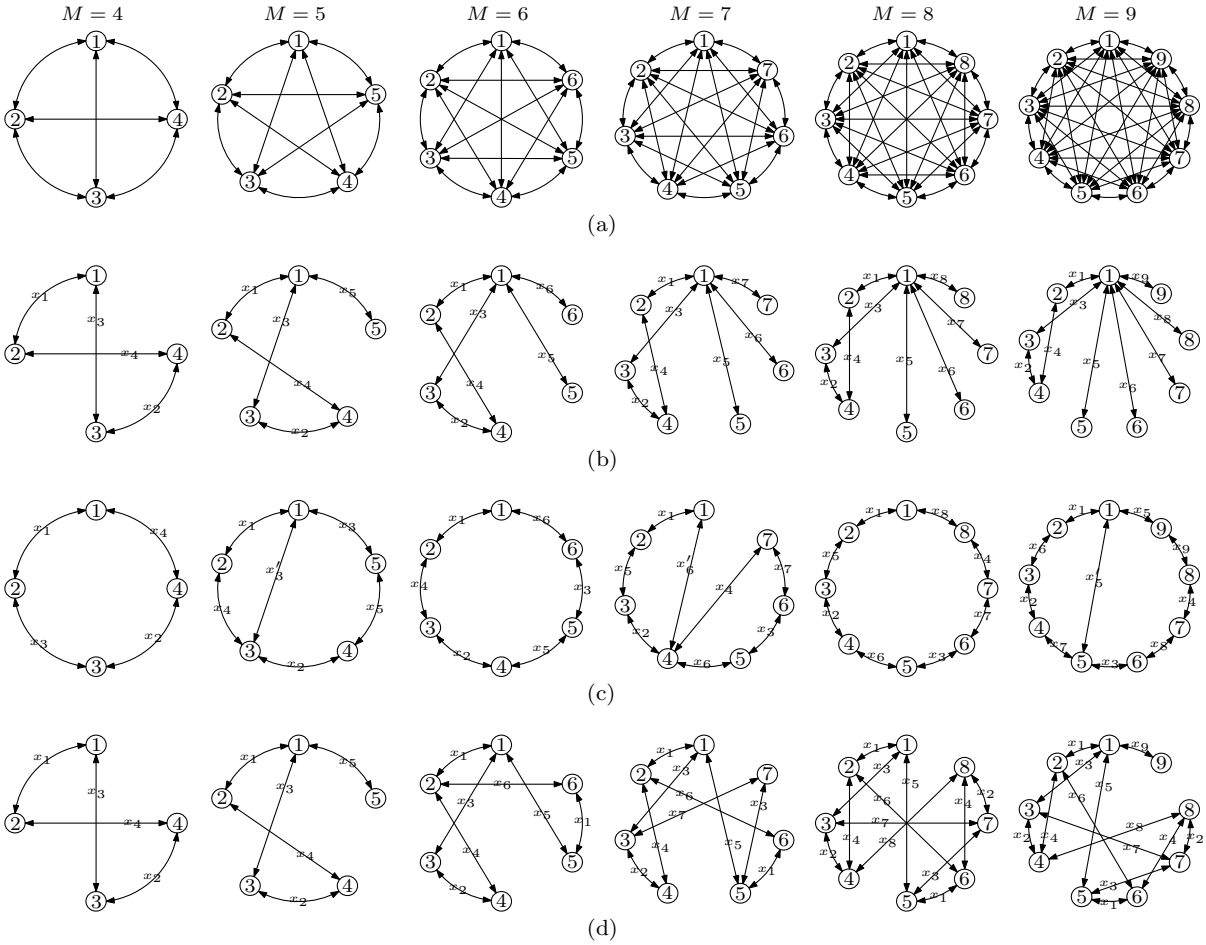

Fig. 1. Graph representation of (a) all possible transpositions between symbols, (b) DPMs from Construction 2 [4], (c) DPMs from Constructions A–C [8] and (d) DPMs from multilevel construction [9]

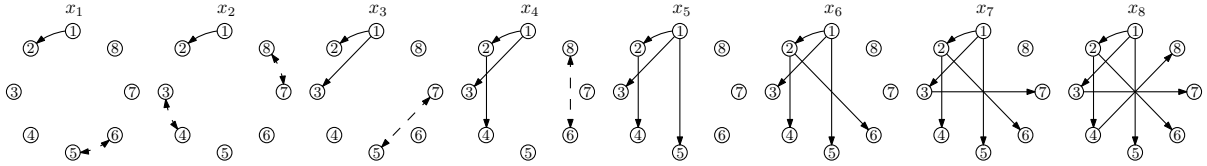

Fig. 2. Steps to determine symbol 1 path for $M = 8$ mapping from multilevel construction [9]

appear in any of these positions. This procedure is followed for every input bit, up to $x_8 = 1$, where $\texttt{swap}(y_4, y_8)$ is used, making it possible for symbol 1 to appear in all the positions. □

By keeping track of which positions a symbol can appear in during an algorithm, one can determine the symbol paths for all of the symbols. The symbol paths for the $M = 8$ mapping from Fig. 1(b) are shown in Fig. 3(a), those for the $M = 8$ mapping from Fig. 1(c) are shown in Fig. 3(b) and those for the $M = 8$ mapping from Fig. 1(d) are shown in Fig. 3(c).

In [9], the upper bound on $|\mathbf{E}|$ was calculated as

$$|\mathbf{E}_{\max}| = M[2^{2n} - (2\alpha\beta + \beta + \alpha^2 M)], \qquad (3)$$

with $\alpha = \lfloor 2^n/M \rfloor$ and $\beta \equiv 2^n \pmod M$. Briefly, for a mapping to attain the upper bound, β of the symbols must appear $\alpha+1$ times in a position and the remaining $M-\beta$ symbols must appear α times in the same position. Following the same reasoning, a certain symbol must appear $\alpha+1$ times in β of the positions and α times in the remaining $M-\beta$ positions.

For a mapping to attain the upper bound (or get close to it), the symbol paths should distribute the symbols as much as possible. Clearly, the symbol paths in Fig. 3(a) and (b) do not connect to all the possible positions, and therefore the distribution of symbols is not optimal. In contrast, the symbol paths in Fig. 3(c) connect to all the possible positions, resulting in a distribution of symbols that lets this mapping attain the upper bound.

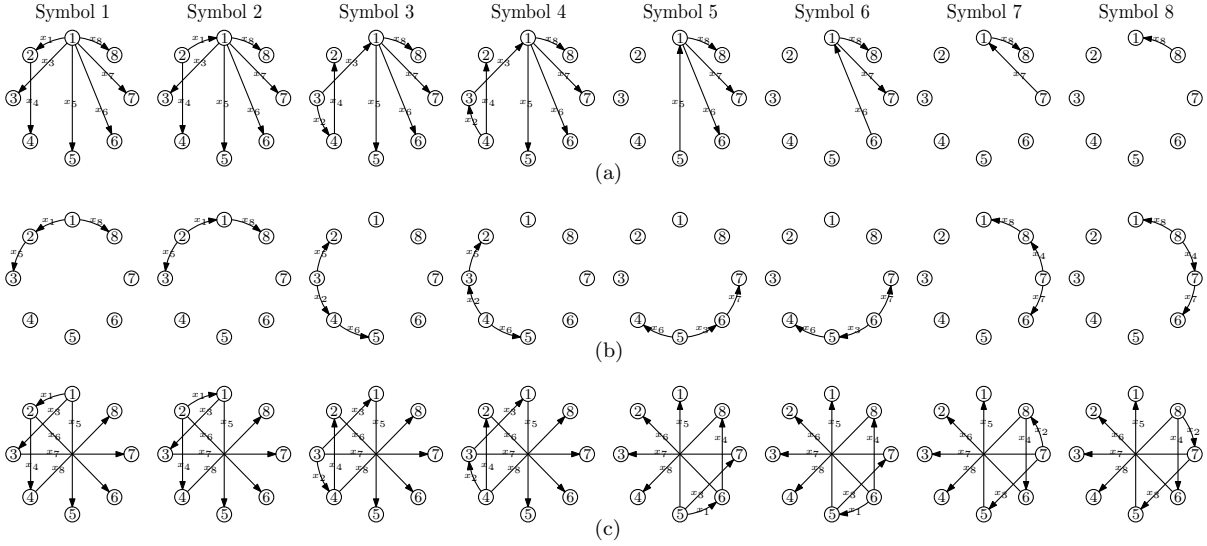

Fig. 3. Symbol paths for $M = 8$ mappings from (a) Construction 2 [4], (b) Constructions A–C [8] and (c) multilevel construction [9]

With the graphs, it can be shown that using a simple algorithm, the upper bound will be reached for $M = 2^l$ and that for other M values, it would be impossible. The following example illustrates this.

Example 3 In Fig. 4, the symbol 1 path is shown for an $M = 8$ multilevel mapping, with the number of times that symbol 1 appears in each position at every stage. At the start, all 256 sequences have symbol 1 in position 1. Since 128 binary sequences have $x_1 = 0$ and 128 have $x_1 = 1$, 128 permutation sequences will have symbol 1 in position 1 and 128 will have it in position 2 after $\texttt{swap}(y_1, y_2)$. Of the 128 binary sequences with $x_1 = 0$, 64 have $x_3 = 0$ and 64 have $x_3 = 1$ and similarly for those with $x_1 = 1$ and $x_4 = 0$ or $x_4 = 1$. Therefore, after $\texttt{swap}(y_1, y_3)$ and $\texttt{swap}(y_2, y_4)$, symbol 1 will appear 64 times in positions 1, 2, 3 and 4. After all the swaps are done, symbol 1 appears 32 times in all the positions. For this case these are the exact values required to attain the upper bound, with $\alpha = 32$ and $\beta = 0$ in (3).

Fig. 5 shows the symbol 1 path for an $M = 7$ multilevel mapping. The number of times that symbol 1 will appear in a position is determined as previously. However, with $\alpha = 18$ and $\beta = 2$, the upper bound can only be achieved if symbol 1 appears 19 times in two of the positions and 18 times in the others and this is clearly not the case in this mapping. This mapping cannot attain the upper bound on the sum of the Hamming distances.□

Proposition 1 *Using a simple algorithm, DPMs can be obtained which reaches the upper bound on the distance sum when $M = 2^l$, with l some positive integer. For other values of M, it is impossible to reach the upper bound using a simple algorithm.* □

PROOF Since $x_i = 0$, $1 \leq i \leq M$, for half of the binary sequences and $x_i = 1$, $1 \leq i \leq M$, for the other half, the number of permutation symbols that will appear in a certain position, using a simple algorithm, will always be a power of two, such as 2^{n-j}, $0 \leq j \leq n$.

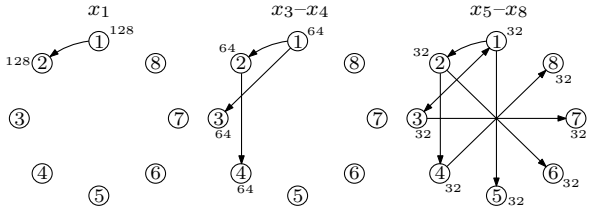

Fig. 4. $M = 8$ symbol 1 path with number of appearances

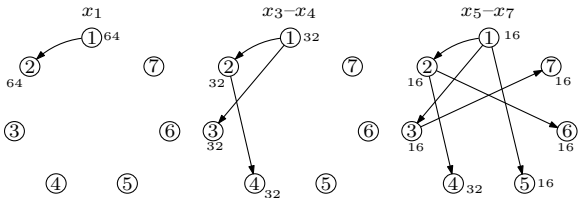

Fig. 5. $M = 7$ symbol 1 path with number of appearances

If $M = 2^l$, with l some integer, then the upper bound in (3) simplifies with $\alpha = 2^{n-l}$ and $\beta = 0$. This means that all the symbols must appear 2^{n-l} times in all the positions. This is possible for a simple algorithm, since the symbols can appear 2^{n-j} times in any position.

For any $M \neq 2^l$, the upper bound can only be attained if β symbols appear $\alpha + 1$ times in certain positions and $M - \beta$ symbols appear α times in the remaining positions. Since $\alpha \neq 2^{n-j}$ in this case, the upper bound cannot be attained. ■

We can therefore conclude that a more complex mapping algorithm would be necessary to produce mappings that attain the upper bound for all values of M.

In [9] it was established in the proof of (3) that the sum of the Hamming distances contributed by symbols in position k, can be calculated as

$$|\mathbf{E}^{(k)}| = 2^{2n} - (m_{k,1}^2 + m_{k,2}^2 + \ldots + m_{k,M}^2), \quad (4)$$

where $m_{k,i}$ denotes the number of times that symbol i appears in position k. Hence, the total sum of Hamming distances in the mapping is

$$|\mathbf{E}| = |\mathbf{E}^{(1)}| + |\mathbf{E}^{(2)}| + \cdots + |\mathbf{E}^{(M)}|.$$

Substituting the symbol quantities (as in Example 3) into (4), it is possible to calculate $|\mathbf{E}^{(k)}|$, $1 \leq k \leq M$. This method can thus be used to calculate $|\mathbf{E}|$, an alternative to calculating all the distance entries in $\mathbf{E}$ and summing it.

Looking at the symbol paths of the graphs presented so far, one notices that the paths never merge at another position. This is an important property for a DPM to build distance. Briefly, if a symbol appears in the same position, but following different paths, it means that the symbol does not build distance in that position, although the input bits build distance since it is two different paths. This is illustrated in the following example. (These two mappings were used in [7] as examples of two cyclic mappings, where one is a valid DPM and the other is not.)

Example 4 Consider two $M = 6$ mappings, one that is not a DPM in Fig. 6(a) and one that is a DPM in Fig. 6(c). Their symbol 1 paths are shown in Fig. 6(b) and (d) respectively.

In Fig. 6(b) it can be seen that the symbol 1 path merges with itself in position 1. In this mapping, input sequences of 000000 and 111111 result in permutation sequences of 123456 and 134562 respectively. Thus, a distance of 6 between the binary sequences only maps to a distance of 5 between the permutation sequences. This is a consequence of symbol 1 appearing in the same position, but following different paths, or equivalently, having different input bits.

In contrast, the symbol 1 path in Fig. 6(d) shows no merging and therefore symbol 1 cannot appear in the same position, following two different paths, for two different input sequences. □

Proposition 2 *The symbol paths of a permutation DPM graph can never merge at another position.* □

PROOF Consider any subgraph of a mapping graph, where an arbitrary symbol path, say the symbol a path, merges with itself at another position. First consider the shortest merging path with only three positions, say a, b and c, as well as the input bits $x_i x_j x_k$ that affect the path of symbol a. If $x_i = 1$, then use $\texttt{swap}(y_a, y_b)$, if $x_j = 1$, then use $\texttt{swap}(y_b, y_c)$ and if $x_k = 1$, then use $\texttt{swap}(y_a, y_c)$. For input 000, the output is abc and for input 111, the output is acb, resulting in a distance of 3 between binary sequences and a distance of 2 between permutation sequences. Therefore, this subgraph could not form part of a DPM graph.

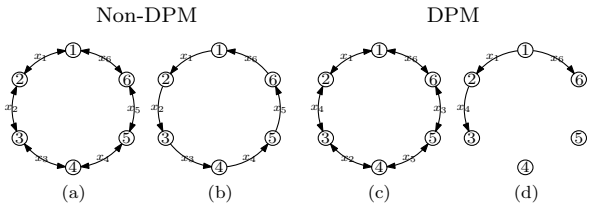

Fig. 6. $M = 6$ (a) non-DPM graph representation, (b) non-DPM symbol 1 path, (c) DPM graph representation and (d) DPM symbol 1 path

For each path segment (or swap) added to the path, another input bit is necessary. If θ swaps are necessary to return symbol a to its original position, θ input bits, each equal to 1, is necessary to effect that path. Therefore, a distance of θ between binary sequences will only have a corresponding maximum distance of $\theta - 1$ between permutation sequences, and can thus not form part of a DPM graph. ■

Another interesting observation from Fig. 6 is that the same subgraph can result in mappings that is distance-preserving and that is not distance-preserving, depending on where the input bits are assigned. This also formed the basis for the multilevel construction of [9], where every mapping uses the same subgraph, but the input bits can be assigned differently to obtain numerous different mappings.

5. Other Applications

In addition to using this new knowledge to analyze existing mappings, it can also be used for other applications. We will briefly discuss two such applications. The details are not covered as research is ongoing on both applications.

5.1. New decoding algorithm

In [1]–[3] it was discussed how permutation codes can be used in conjunction with M-ary FSK to provide time and frequency diversity for very noisy power-line communication channels. In essence, each permutation symbol is represented by one of the M frequencies in one of M time slots. At the receiver, a threshold detector for each frequency outputs a one when the signal is over the threshold, and a zero otherwise. Thus a binary $M \times M$ matrix is received with a single one in each row and each column. Noise on the channel will then cause errors in this matrix.

In [2] and [3] convolutional codes are combined with DPMs to form permutation trellis codes. A disadvantage is that the decoder becomes non-standard and complex when high rate convolutional codes are used. Also, should another convolutional code be used, even by

simple puncturing, the entire decoder and DPM must be changed.

Using this new algorithm, the convolutional code and DPM are kept separate, thus forming an inner and outer code. Standard convolutional decoders can then be used and the rates can easily be changed using puncturing.

The DPM graph is similar to a state machine in that the input bits determine in what state the machine will be, in our case it determines in which positions the permutation symbols will appear. Each received one in the $M \times M$ matrix represents a permutation symbol in a certain position, whether received correctly or not. This can be used in conjunction with the graph to determine an estimate of the input bits for each received symbol. The estimates for all symbols are then used to determine what the input bits were.

5.2. New constructions

While the multilevel mappings from [9] attain higher distance sums than previous mappings, a general mapping algorithm was not known. However, using the graphs, in particular a trellis-like representation of the graphs, a general algorithm can be obtained for the case where $M = 2^l$, with l some integer.

In Fig. 7 the trellis representation of the $M = 4$ and $M = 8$ graphs from Fig. 1(d) is shown. An input bit is assigned to each interval. (This representation also makes it clear in what order the input bits are considered.) As before, if the input bit is one, then the symbols are swapped as shown by the diagonal branches, otherwise the symbols stay in the same positions.

The general algorithm is based on the fact that the trellis for $M = 2^l$ is derived from the trellis for $M' = 2^{l-1}$. In Fig. 7 it can be seen that the $M = 4$ trellis is used twice in the $M = 8$ trellis: for symbols 1 to 4 with input bits x_1 to x_4 and again for symbols 5 to 8 with input bits x_1 to x_4. In addition, $\texttt{swap}(i-4, i)$ is added for input bit x_i, $5 \leq i \leq 8$. Similarly, the $M = 16$ trellis is derived by using two $M = 8$ trellises: for symbols 1 to 8 with input bits x_1 to x_8 and again for symbols 9 to 16 with input bits x_1 to x_8. In addition, $\texttt{swap}(i-8, i)$ is added for input bit x_i, $9 \leq i \leq 16$.

In general, the $M = 2^l$ trellis is constructed by using the $M' = 2^{l-1}$ trellis for symbols 1 to M' with input bits x_1 to $x_{M'}$ and again for symbols $M'+1$ to M with input bits x_1 to $x_{M'}$. To this $\texttt{swap}(i-M', i)$ for input bit x_i, $M'+1 \leq i \leq M$ are added. The explicit algorithm will not be presented here because of length restrictions.

6. Conclusion

We have shown how the use of graphs can give new insight into the analysis of permutation DPMs and serve as a compact representation of mapping algorithms. The graphs can visually aid one in determining the positions of symbols at certain stages in a mapping, as well as showing why some mappings cannot be distance-preserving.

It also provides an alternative method to calculate the sum of distances for permutation mappings, moreover showing that for certain M it is impossible for simple algorithms to generate mappings that can attain the upper bound on the distance sum.

Finally, the mapping graphs can be used to decode mappings obtained from existing mapping algorithms, as well as to construct new mapping algorithms.

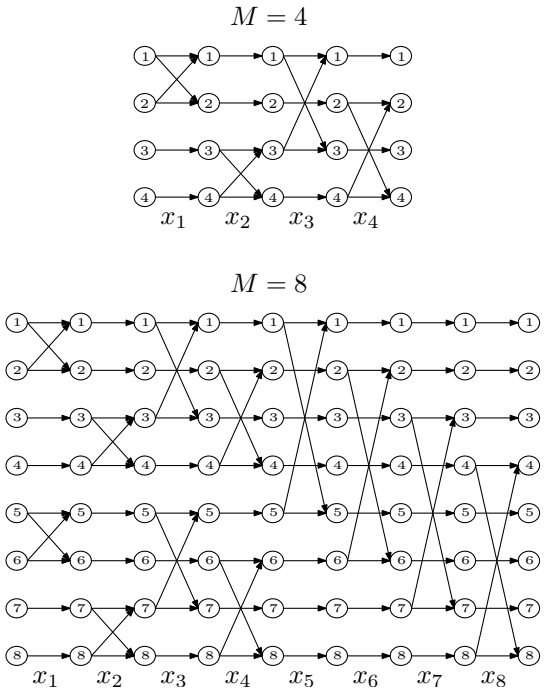

Fig. 7. $M = 4$ and $M = 8$ trellis representation of multilevel DPMs

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] 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.

[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] 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.

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

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

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

[8] 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.

[9] 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.