Using Graphs for the Analysis and Construction of Permutation Distance-Preserving Mappings

Theo G. Swart, Hendrik C. Ferreira, *Member, IEEE* and Khmaies Ouahada

***Abstract*— A new way of looking at permutation distance-preserving mappings is presented by making use of a graph representation. The properties necessary to make such a graph distance-preserving, are also investigated. Further, this new knowledge is used to analyze previous constructions, as well as to construct a new general mapping algorithm for a previous multilevel construction.**

***Index Terms*— Code constructions, distance-preserving mappings (DPMs), graphs, permutation coding**

I. 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).

Since Vinck [1] suggested using permutation codes for power-line communications and Ferreira and Vinck [2] used distance-preserving mappings and permutation codes to create permutation trellis codes, much research has been done on permutation mappings [3]–[8].

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

Swart, de Beer and Ferreira [7] 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. Swart and Ferreira [8] 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.

Permutation DPMs are used for two main applications: to create lower bounds for permutation arrays or to create error correcting codes such as permutation trellis codes. Any DPM that satisfies the distance-preserving property is sufficient for new lower bounds on permutation arrays. However, since several different mappings can be created for the same parameters, it stands to reason that some might be better than others when error correcting are considered. For this reason, the upper bound on the distance sum is used as a measure for correcting capabilities, with mappings attaining or getting close to the bound performing better than those that do not [7]. Although it is not an absolute measure, it is a good starting point to look for permutation DPMs with good error correcting capabilities.

Lee [9], [10] presented further 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. Although mappings can be obtained for $n \neq M$, see [8], we will restrict this work to the case where $n = M$.

In Section II 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 III, with previous constructions being used to further illustrate its use. Section IV investigates the properties of such mapping graphs, and this is used to analyze different known mappings. In Section V the graphs are represented as trellises, leading to a general algorithm for the multilevel construction of [8]. Section VI concludes with some final remarks.

II. 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 of length n, with $|\mathcal{C}_b| = 2^n$, and $\mathcal{C}_p$ consists of some subset of 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

This paper was presented in part at the International Symposium on Communication Theory and its Applications, Ambleside, England, July 2007.

T. G. Swart, H. C. Ferreira and K. Ouahada are with the Department of Electric and Electronic Engineering Science, University of Johannesburg, Auckland Park, 2006, South Africa. (e-mail: tgswart@postgrad.uj.ac.za, hcferreira@uj.ac.za, ktouahada@postgrad.uj.ac.za).

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 $e_{ij} \geq d_{ij}$ for all $i \neq j$, 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 $\texttt{swap}(y_a, y_b)$ denotes the swapping 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 [10]).

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.

III. Graph Representation of Permutation DPMs

All the M permutation positions and symbols are represented by placing them on a circle, with symbol i in position y_i, $1 \leq i \leq M$. Transpositions of symbols are then represented by a connecting line between the two positions of the symbols to be transposed, as in Fig. 1. For simplicity we omit the position labels, y_i, from the graphs to follow. In Fig. 2 the graphs for $4 \leq M \leq 9$ are shown.

Using combinations of these transpositions one would be able to generate all the possible permutation sequences from the symmetric group, S_M. 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. 2.

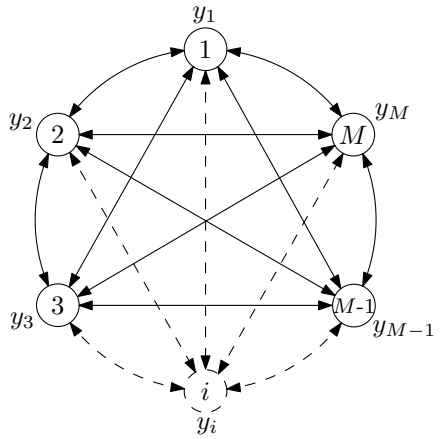

Fig. 1. General graph representation for a permutation distance-preserving mapping

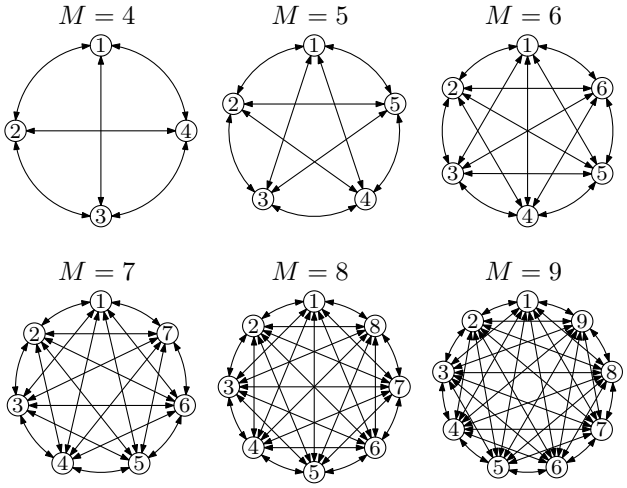

Fig. 2. Graph representation of all possible transpositions between symbols

To illustrate this, we make use of the Construction 2 algorithm presented in [4], with the position function a constant 1. In Fig. 3 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 is transposed, and this is done in the order $i = 1, 2, \ldots, M$. When $x_i = 0$, the symbols are left unchanged.

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

The mappings found by using the multilevel construction [8] are illustrated in Fig. 4. A greater number of transpositions is used in this case, as opposed to those in Fig. 3, and this results in the sum of Hamming distances, $|\mathbf{E}|$, being greater, as was shown in [8].

Each input bit in Fig. 3 is only assigned a single transposition, while some of the input bits in Fig. 4 are assigned more than one transposition. It is important to note that

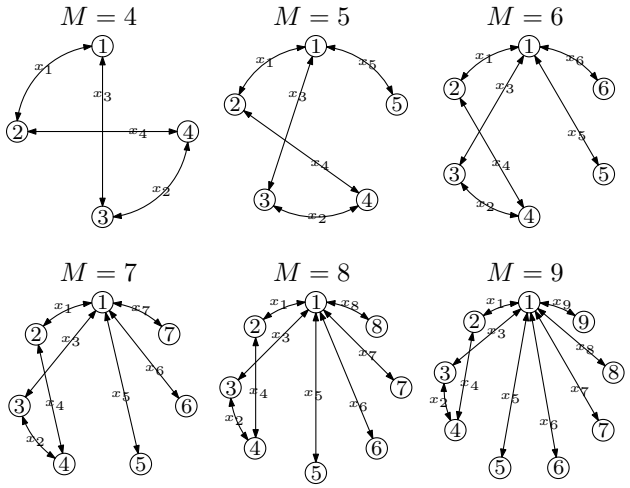

Fig. 3. Graph representation of DPMs from Construction 2 [4]

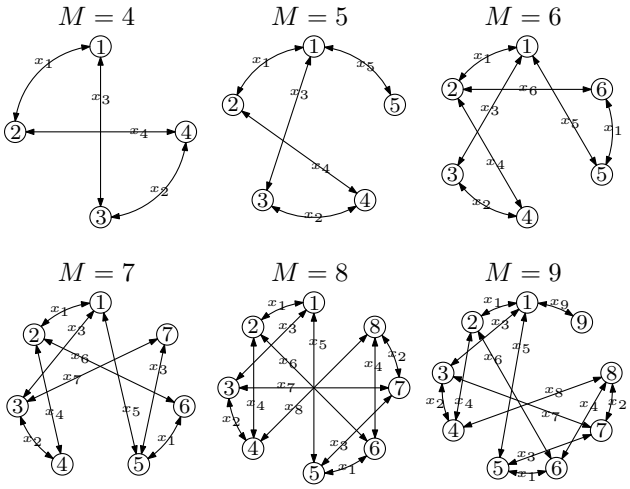

Fig. 4. Graph representation of DPMs from multilevel construction [8]

the transpositions assigned with the same input bit are all independent, therefore it does not matter in which order these transpositions are done. This is as a result of the multilevel construction where transpositions are obtained from different levels.

IV. 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 followed to obtain the symbols paths.

Example 2: Consider the path that symbol 1 can follow in the $M = 8$ multilevel mapping from Fig. 4. The possible paths, followed according to the input bits, are shown in Fig. 5. 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 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.

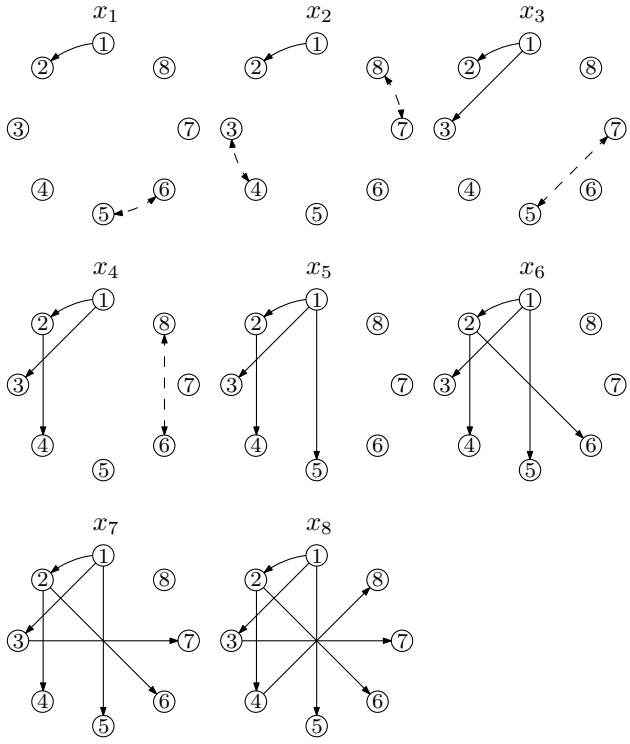

Fig. 5. Symbol 1 path for $M = 8$ multilevel mapping [8]

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. 3 are shown in Fig. 6. Similarly, the symbol paths for the $M = 8$ mapping from Fig. 4 are shown in Fig. 7.

In [8], it was established in a proof that the sum of the Hamming distances contributed by symbols in position k, can be calculated as

$$|\mathbf{E}^{(k)}| = 2^{2n} - (m_{k,1}^2 + m_{k,2}^2 + \ldots + m_{k,M}^2), \qquad (3)$$

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

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

The upper bound on $|\mathbf{E}|$ was calculated as

$$|\mathbf{E}_{\max}| = M[2^{2n} - (2\alpha\beta + \beta + \alpha^2 M)], \qquad (4)$$

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. 6 do not connect to all the possible positions, and therefore the distribution of symbols is not optimal. In contrast, the symbol paths in Fig. 7

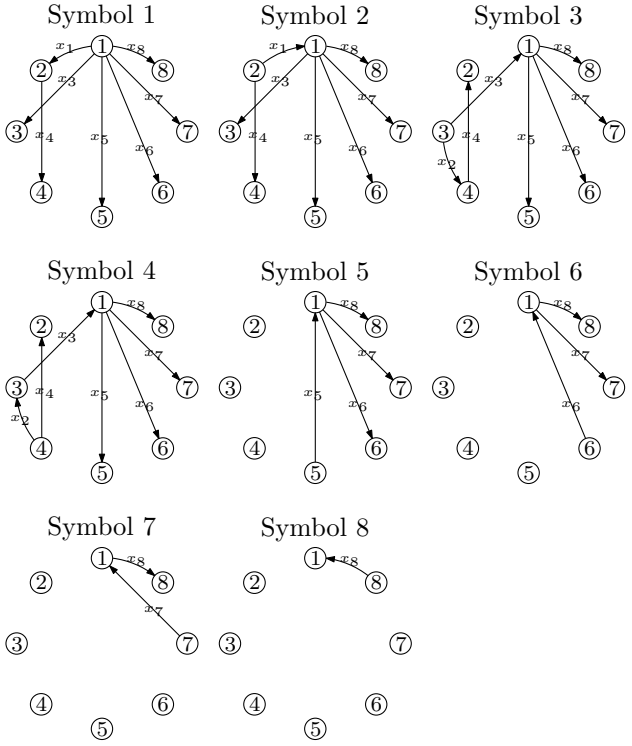

Fig. 6. Symbol paths for $M = 8$ Construction 2 mapping [4]

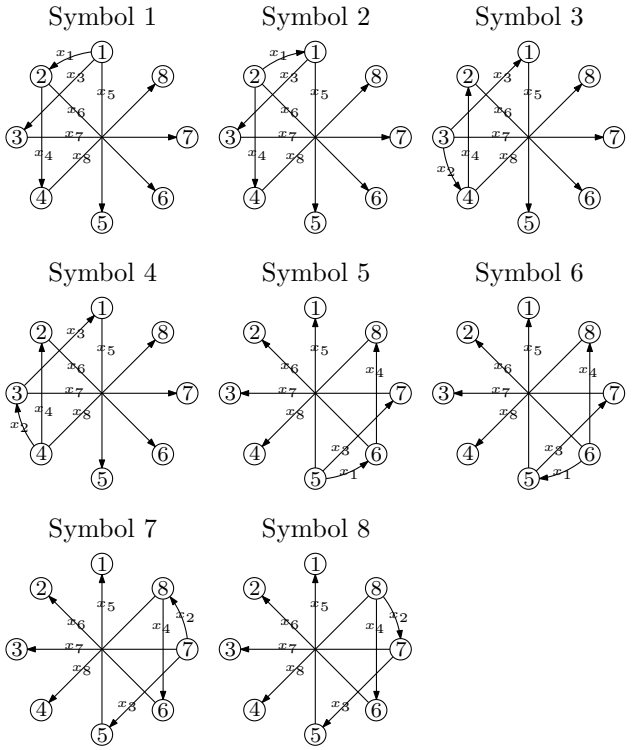

Fig. 7. Symbol paths for $M = 8$ multilevel mapping [8]

connect to all the possible positions, resulting in a distribution of symbols that lets this mapping attain the upper bound.

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

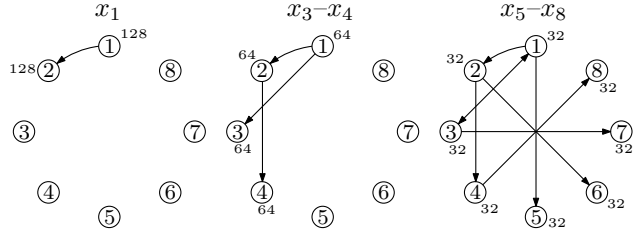

Fig. 8. $M = 8$ symbol 1 path with number of appearances

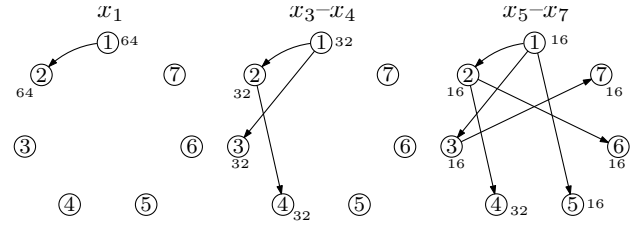

Fig. 9. $M = 7$ symbol 1 path with number of appearances

example illustrates this.

Example 3: In Fig. 8, 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. Since $\alpha = 32$ and $\beta = 0$ in (4), these are the exact values required to attain the upper bound.

Fig. 9 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.

In general, substituting the symbol quantities into (3), it is possible to calculate $|\mathbf{E}^{(k)}|$, $1 \leq k \leq M$, without having to calculate the distances between all sequences.

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}, $0 \leq j \leq n$.

If $M = 2^l$, with l some integer, then the upper bound in (4) 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 other M value, $M \neq 2^l$, 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, it is impossible to attain the upper bound. ∎

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.

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 [9] 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. 10(a) and one that is a DPM in Fig. 11(a). Their symbol 1 paths are shown in Fig. 10(b) and Fig. 11(b) respectively.

In Fig. 10(b) it can be seen that the symbol 1 path merges with itself in positions 1 and 6. 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. 11(b) shows no merging and therefore symbol 1 cannot appear in the same position, following two different paths, for two different input sequences.

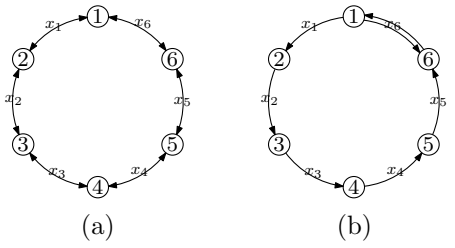

Fig. 10. $M = 6$ non-DPM with (a) graph representation and (b) symbol 1 path

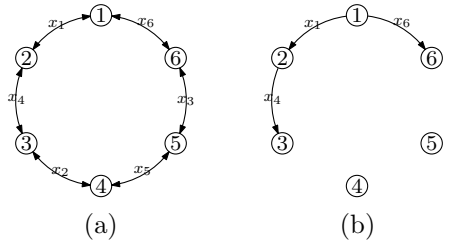

Fig. 11. $M = 6$ DPM with (a) graph representation and (b) symbol 1 path

Proposition 2: The symbol paths of a permutation DPM graph constructed by a simple algorithm can never merge at another position if there is no common input bit between the two path segments.

Proof: Since two segments cannot have an input bit in common, each input bit can be assigned to one swap only. Consider any arbitrary symbol g path, which merges with itself at another position, and break this path into two segments. Let $\mathbf{x} = (x_1, x_2, \ldots, x_M)$ be the input bits for the first segment and let $\mathbf{x}' = (x'_1, x'_2, \ldots, x'_M)$ be the input bits for the second segment. Let $\{a, b, c, \ldots\}$ be the set of indices for the input bits that effect the first segment with $1 \leq a < b < c < \cdots \leq M$, $\{a, b, c, \ldots\} \in \mathbb{N}$ and let $\{a', b', c', \ldots\}$ be the set of indices for the input bits that effect the second segment with $1 \leq a' < b' < c' < \cdots \leq M$, $\{a', b', c', \ldots\} \in \mathbb{N}$. With no common input bit between the two segments, we have that $\{a, b, c, \ldots\} \cap \{a', b', c', \ldots\} = \emptyset$.

For $a < a'$, the input bit in position a will play a role in both segments. For the first segment, $x_a = 1$ for the symbol to move along the segment. For the second segment, $x'_a = 0$ as the first segment will not be followed (the second segment only begins when $x'_{a'} = 1$). Certain input bits play no role in these two segments and can thus take on any binary bit, denoted by $\times$, then

$$\begin{aligned} d_H(\mathbf{x}, \mathbf{x}') &= d_H(x_1, x'_1) + d_H(x_2, x'_2) + \cdots \\ &\quad + d_H(x_a, x'_a) + \cdots + d_H(x_{a'}, x'_{a'}) + \cdots \\ &\quad + d_H(x_b, x'_b) + \cdots + d_H(x_{b'}, x'_{b'}) + \cdots \\ &\quad + d_H(x_M, x'_M) \\ &= d_H(\times, \times) + d_H(\times, \times) + \cdots \\ &\quad + d_H(1, 0) + \cdots + d_H(\times, 1) + \cdots \\ &\quad + d_H(1, \times) + \cdots + d_H(\times, 1) + \cdots \\ &\quad + d_H(\times, \times) \end{aligned}$$

Now, two input sequences can always be chosen such that the two segments are followed and $d_H(\mathbf{x}, \mathbf{x}') = M$. However, symbol g appears in the same position but following different segments, hence $d_H(\mathbf{y}, \mathbf{y}') \leq M - 1$. Since $d_H(\mathbf{y}, \mathbf{y}') < d_H(\mathbf{x}, \mathbf{x}')$, this path cannot form part of a DPM algorithm. ∎

This proposition is applicable to many of the mappings used to construct permutation arrays, where it is sufficient for each input bit to be assigned to one swap only, hence two segments cannot have an input bit in common.

When there is a common input bit between the two segments that merge, it could be possible to obtain a DPM, however none has been found thus far. The reason for this is that the two input sequences for the two segments have one bit that is the same, since the input bit is present in both segments. The corresponding permutation sequences also have one symbol that is the same, since the symbols appear in the same position when the two segments merge. Therefore it could be possible to have binary and permutation sequences that satisfy the distance-preserving criterion.

Another interesting observation from Fig. 10 and Fig. 11 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 of the multilevel construction, where every mapping uses the same subgraph, but the input bits can be assigned differently to obtain numerous different mappings.

V. Mapping Algorithm for Multilevel Construction

As an alternative, a trellis representation of the graphs can be used. The different positions are shown as states, with the input bits determining where the symbols will go to next. Whenever an input bit is zero, the symbol in a specific position stays in that position, and this is represented by the horizontal branches. If an input bit is one, then it is possible for a symbol to move to another position, and this is represented by the diagonal branches. If no diagonal branch is present, then the symbol stays in that position, irrespective of the input bit value. This is illustrated in Fig. 12 for $M = 4$.

This led to the general algorithm for mappings attaining the upper bound when $M = 2^l$, derived from the multilevel construction of [8]. The trellis diagrams for the $M = 4$, $M = 8$ and $M = 16$ multilevel mappings are shown in Fig. 13. By looking at the trellises, one can see that the $M = 8$ trellis is constructed by making use of two $M = 4$ trellises for the x_1 to x_4 input bits. Branches are then added for the x_5 to x_8 input bits. Similarly, the $M = 16$ trellis is constructed by making use of two $M = 8$ trellises for the x_1 to x_8 input bits, while branches are added for the remaining x_9 to x_{16} input bits. Larger mappings can then be constructed following this same process.

The general algorithm in this case is

```
Input: (x_1, x_2, ..., x_M)
Output: (y_1, y_2, ..., y_M)
begin
  (y_1, y_2, ..., y_M) ← (1, 2, ..., M)
  if x_1 = 1 then
    for i from 1 to M/4
      swap(y_{4i-3}, y_{4i-2})
  if x_2 = 1 then
    for i from 1 to M/4
      swap(y_{4i-1}, y_{4i})
  for i from 1 to L − 1
    for j from 1 to 2^i
      if x_{j+2^i} = 1 then
        for k from 1 to M/2^{i+1}
          p = j + 2^{i+1}(k − 1)
          swap(y_p, y_{p+2^i})
end.
```

Since the algorithm uses the multilevel construction from [8], the resulting mapping will be a DPM and we will not prove it again.

Using this algorithm, we can also obtain a general algorithm to construct multilevel mappings for any M value. As example, to obtain an $M = 7$ mapping, an $M = 8$ mapping can be used, but with symbol 8 and position 8 removed. Therefore, position 8 and any edges connecting to this position must be removed. Similarly, position 8 and any branches in the trellis going to position 8 must be deleted. This is similar to the idea of prefixing, where an $M + 1$ mapping is obtained from an M mapping, except that in this case we are obtaining $M - 1$ mappings from M mappings. This can be verified by looking at Fig. 4 and deleting position M to obtain the $M-1$ mapping.

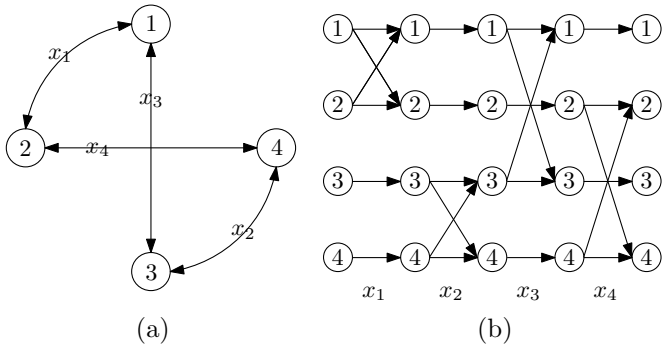

Fig. 12. $M = 4$ mapping with (a) graph representation and (b) trellis representation

Proposition 3: By deleting position $M + 1$ and any connecting lines in a multilevel DPM graph, the equivalent M multilevel DPM graph is obtained.

Proof: The multilevel construction for $M+1$ makes use of the identity permutation sequence, $012 \ldots M$, written in a multilevel representation [8], with each symbol's equivalent binary representation written in columns. Binary ones in the columns represent possible swaps that can be used. Therefore, each column represents a symbol position in the DPM graph and the ones in a column represent the connecting lines in the DPM graph for that position.

Similarly, the multilevel construction for M makes use of the identity permutation sequence $012 \ldots M - 1$, which is obtained by deleting M from the $M + 1$ identity sequence, thus deleting the $M + 1$-th column from the multilevel representation, as well as any ones that were present in this column. Equivalently, position M in the DPM graph is deleted, as well as any connecting lines to this position. ■

Any random positions can be deleted to obtain different mappings, however a DPM cannot always be guaranteed. As example, consider deleting positions 4 and 8 from the $M = 8$ trellis in Fig. 13, to obtain an $M = 6$ mapping. After deleting any connecting lines to these two positions, one will observe that input bits x_2, x_4 and x_8 have no swaps left. Two of these can be deleted since only six input bits are needed, leaving one input bit that is not building distance. In this case a DPM is not obtained. Hence, if random positions are to be deleted, careful consideration must be given to ensure that a valid DPM results.

Returning to an algorithm for the general case, it is necessary to modify the previous algorithm slightly, considering that certain positions are removed. The general algorithm to construct permutation DPMs from the multilevel construction,

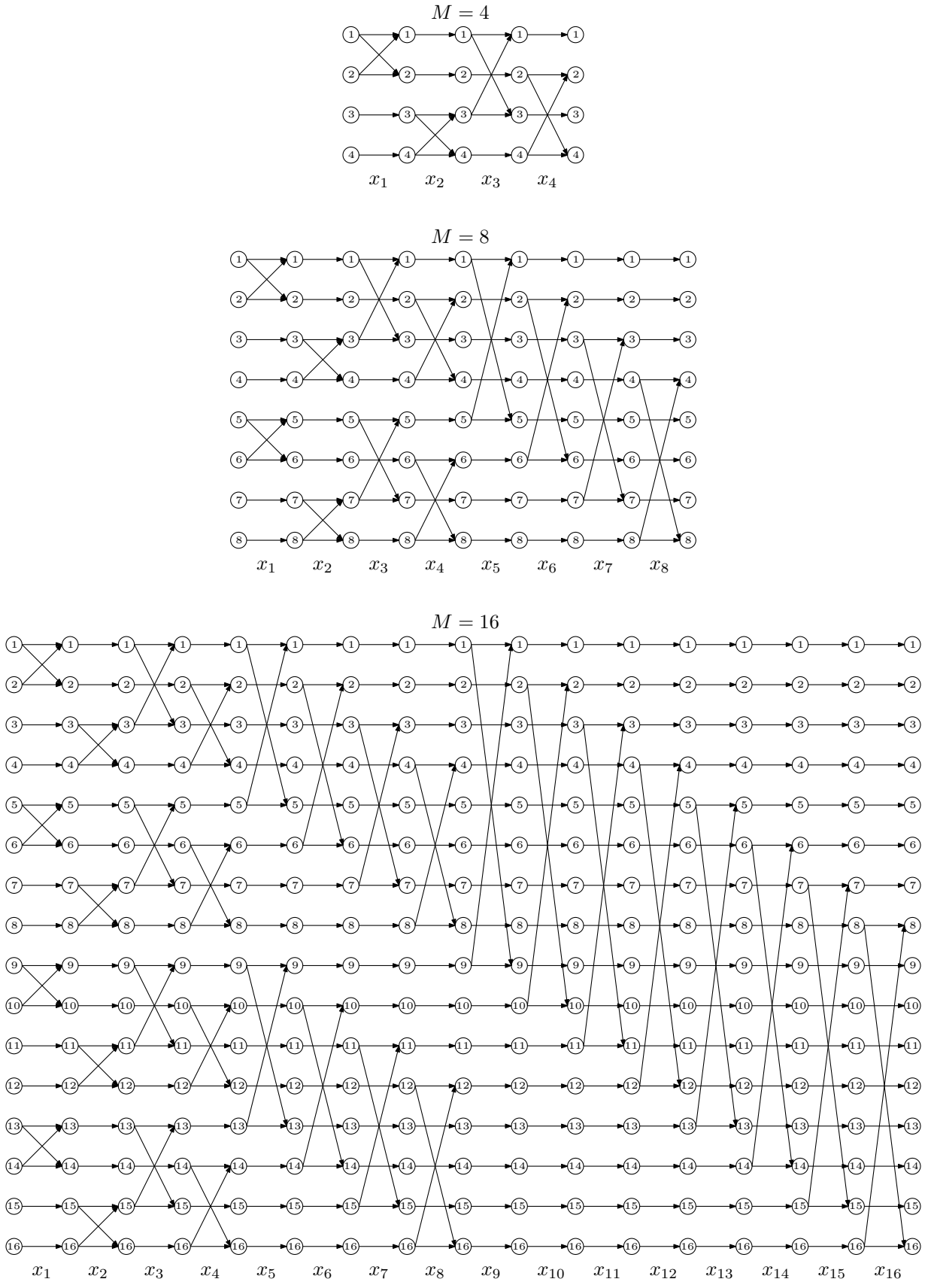

Fig. 13. Trellis representation for multilevel mappings attaining the upper bound

is

```
Input: (x_1, x_2, ..., x_M)
Output: (y_1, y_2, ..., y_M)
begin
  (y_1, y_2, ..., y_M) ← (1, 2, ..., M)
  if x_1 = 1 then
    for i from 1 to ⌊(M + 2)/4⌋
      swap(y_{4i-3}, y_{4i-2})
  if x_2 = 1 then
    for i from 1 to ⌊M/4⌋
      swap(y_{4i-1}, y_{4i})
  for i from 1 to L − 1
    for j from 1 to 2^i
      if j ≤ M − 2^i then
        if x_{j+2^i} = 1 then
          for k from 1 to ⌊(M − j + 2^i)/2^{i+1}⌋
            p = j + 2^{i+1}(k − 1)
            swap(y_p, y_{p+2^i})
end.
```

This algorithm then produces mappings with the same $|\mathbf{E}|$ values as those listed in [8] for the multilevel mappings. In almost all cases these values are larger than those for previous mappings.

VI. Conclusion

We have shown how the use of graphs can give new insight into the analysis and construction of permutation DPMs. 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. These graphs can also be used in the decoding of permutation codes obtained from mapping algorithms [11].

Although the multilevel construction of [8] was flexible and could produce numerous different mappings for the same M value, some empirical work or computer searches were still necessary to obtain the mappings. Using the trellis representation of the graphs, we were able to construct a general algorithm for this construction, even though only a subset of all the possible mappings are obtained for a certain M.

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] 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.

[8] 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.

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

[10] 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.

[11] T. G. Swart and H. C. Ferreira, "Decoding distance-preserving permutation codes for power-line communications," accepted for *IEEE AFRICON 2007*.

Theo G. Swart received the B.Eng. and M.Eng. degrees in electric and electronic engineering from the Rand Afrikaans University, South Africa, in 1999 and 2001, respectively, and the D.Eng. degree from the University of Johannesburg, South Africa. He is currently doing post-doctoral research in the Telecommunications Research Group at the University of Johannesburg. His research interests include digital communications, power-line communications and error correction coding.

Hendrik C. Ferreira was born and educated in South Africa where he received the D.Sc. (Eng.) degree from the University of Pretoria in 1980.

From 1980 to 1981, he was a post doctoral researcher at the Linkabit Corporation in San Diego, CA, USA. In 1983, he joined the Rand Afrikaans University (now the University of Johannesburg), Johannesburg, South Africa where he was promoted to professor in 1989. He has served two terms as Chairman of the Department of Electrical and Electronic Engineering, from 1994 to 1999.

His research interests are in Digital Communications and Information Theory, especially Coding Techniques. From 1989 until 1993, he held a "Presidential Award for Young Investigators", a prestigious research grant from the South African Foundation for Research Development.

Since 1984, he has been a visiting researcher at various institutions in the USA and Europe. He is a past chairman of the Communications and Signal Processing Chapter of the IEEE South Africa section and since 1997 he has been Editor-in-Chief of the Transactions of the South African Institute of Electrical Engineers. He has served as chairman of several conferences, including the international 1999 IEEE Information Theory Workshop in the Kruger National Park, South Africa.

Khmaies Ouahada received the B.Eng. degree in electric and electronic engineering from the University of Khartoum, Sudan, in 1995 and his M.Eng. degree in electric and electronic engineering from the Rand Afrikaans University, South Africa, in 2002. He is currently busy with his D.Eng. degree at the University of Johannesburg, South Africa. His research interests include digital communications, error correction coding and spectral shaping techniques.