VI. CONCLUDING REMARKS

In this correspondence, we have shown that a $T(2, 7, v)$-code exists if and only if $v \equiv 1$ or 7 (mod 21) with one exception and 68 possible exceptions of $v$, which are presented in Theorem 43. A large number of explicit constructions for $T^*(2, 7, v)$-codes with $v < 2350$ are developed in Section IV. Based on these constructions, a number of infinite series of $T^*(2, 7, v)$-codes and an asymptotic existence result are obtained in Lemmas 44-45 and Theorem 59 respectively. For a complete solution to the existence problem on the codes, we have to handle all outstanding values of $v$ below 2350. This is apparently a rather difficult task. It seems to us that new techniques, direct and recursive, are needed. We will continue our research and report our result in future.

ACKNOWLEDGMENT

The authors wish to thank the reviewers for their comments and suggestions that much improved this correspondence.

REFERENCES


A Generalized Upper Bound and a Multilevel Construction for Distance-Preserving Mappings

Theo G. Swart and Hendrik C. Ferreira, Member, IEEE

Abstract—A new general upper bound is derived on the sum of the Hamming distances between sequences when mapping from one set of sequences to another. It is shown that a similar upper bound for mappings from binary sequences to permutation sequences is a special case of this upper bound and this is used to evaluate known mappings. Also, new distance-preserving mappings (DPMs) from binary sequences to permutation sequences are presented, based on a multilevel construction. In addition to explicit distance-preserving mappings, distance-increasing, and distance-reducing mappings are also presented. Several of the new DPMs attain the upper bound.

Index Terms—Code constructions, distance bounds, distance-preserving mappings (DPMs), Hamming distance, permutation coding.

I. INTRODUCTION

In this correspondence, we consider distance-preserving mappings (DPMs) from binary sequences to permutation sequences where the Hamming distance is preserved, at the same time attempting to optimize the total distance in the mapping. The motivation being that mappings with optimal total distance perform better as error correcting codes than mappings that are not optimal.

Vinck [1] renewed interest in permutation codes by proposing it as an error correcting code for power-line communications. With permutation arrays and permutation block codes having been studied for some time [2]–[4], Ferreira and Vinck [5] combined the idea of DPMs ([6], [7]) with permutation codes, creating mappings from binary sequences of length $n$ to permutation sequences of length $M$, to construct permutation trellis codes. In this case, it was shown how new mappings could be constructed by making use of a prefix method, however, it was only explicitly presented for $M \leq 8$. Swart, de Beer and Ferreira [8] and

Manuscript received August 16, 2005; revised February 3, 2006. This work was supported in part by the National Research Foundation under Grant Number 2053408. The material in this correspondence was presented in part at the IEEE International Symposium on Information Theory, Adelaide, Australia, September 2005, and in part at the IEEE International Symposium on Information Theory, Seattle, WA, July 2006. The authors are with the Department of Electrical and Electronic Engineering Science, University of Johannesburg, Auckland Park, 2006, South Africa (e-mail: ts@ing.rau.ac.za; hcf@ing.rau.ac.za).

Communicated by R. J. McEliece, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2006.878175
Ferreira et al. [9] further investigated the performance of these mappings in power-line communications. Chang et al. [10] extended the work on constructing mappings further by presenting several constructions for creating DPMs for $M$ in general. More recent constructions have been proposed by Lee [11] and Chang [12]. Another motivation for considering DPMs is to establish lower bounds on permutation arrays [10].

Swart, de Beer and Ferreira [13] also presented an upper bound on the sum of distances when mapping from binary sequences to permutation sequences. In this paper we generalize this upper bound to apply to any mapping and show that the previous upper bound is a special case of the generalized upper bound.

The idea for our new construction in this paper came from the work by Wadayama and Vinck [14]. They made use of balanced constant weight codes with known minimum distance to construct permutation block codes using a multilevel construction, similar to the idea that we will employ here. However, their construction was limited to the case where $n = 2^m$, with $m$ some positive integer. One could view this as a mapping from balanced constant weight codes to permutation codes. The new construction presented here can be used for any $n$ and we map all possible binary sequences to permutation sequences.

In Section II we provide definitions and notations to be used as well as defining three types of DPMs. In Section III we prove that an upper bound exists on the sum of Hamming distances in any mapping, also looking at special cases of this upper bound. In Section IV we present a multilevel construction for DPMs from binary sequences to permutation sequences. Section V investigates the distance optimality of the new DPMs, as well as comparing it to those of previous mappings. We conclude with some final remarks in Section VI.

II. DISTANCE-PRESERVING MAPPINGS (DPMs)

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

1. Definition 1: A binary code $C_b$ consists of $|C_b|$ sequences of length $n$, where every sequence contains 0s and 1s as symbols.

2. Definition 2: A permutation code $C_p$ consists of $|C_p|$ sequences of length $M$, where every sequence contains the $M$ different integers $1, 2, \ldots, M$ as symbols.

3. Definition 3: The symmetric 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 $C_b$ consists of all the possible binary sequences of length $n$, with $|C_b| = 2^n$, and $C_p$ consists of some subset of $S_M$, with $|C_p| = |C_b|$. In addition, the distances between sequences for one set are preserved amongst the sequences of the other set.

For binary sequences, let $x_i$ be the $i$th binary sequence in $C_b$, and let $x_i = (x_i^{(1)}, x_i^{(2)}, \ldots, x_i^{(n)})$. The Hamming distance $d_{H}(x_i, x_j)$ is defined as the number of positions in which the two sequences $x_i$ and $x_j$ differ. Construct a distance matrix $D$ whose entries are the distances between binary sequences in $C_b$, with

$$D = [d_{ij}] \text{ with } d_{ij} = d_{H}(x_i, x_j). \quad (1)$$

Similarly for permutation sequences, let $y_i$ be the $i$th permutation sequence in $C_p$, and let $y_i = (y_i^{(0)}, y_i^{(0)}, \ldots, y_i^{(M-1)})$. The Hamming distance $d_{H}(y_i, y_j)$ is defined as the number of positions in which the two sequences $y_i$ and $y_j$ differ. Construct a distance matrix $E$ whose entries are the distances between permutation sequences in $C_p$, with

$$E = [e_{ij}] \text{ with } e_{ij} = d_{H}(y_i, y_j). \quad (2)$$

Example 1: The following is a possible mapping of $n = 2 \rightarrow M = 3$ (for subsequent mappings the binary sequences will be omitted, which will follow the usual lexicography)

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

Using (1) and (2), we have for the above mapping

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

In this case all entries had an increase in distance (except the main diagonal where there is always zero distance).

Three different types of DPMs can be obtained, depending on how the Hamming distance is preserved. Previously this was defined in terms of trellis codes [9], but since only the mappings themselves are considered, we will redefine it purely in terms of distances between the binary and permutation sequences.

- **Distance-conserving mapping (DCM):** Guarantees conservation of the binary sequences’ Hamming distance, such that $e_{ij} \geq d_{ij}$, for all $i \neq j$, with equality achieved at least once.

- **Distance-increasing mapping (DIM):** Guarantees that the permutation sequences’ distance will always have some increase above the binary sequences’ distance, such that $e_{ij} \geq d_{ij} + \delta, \delta \in \{1, 2, \ldots\}$ for all $i \neq j$, with equality achieved at least once.

- **Distance-reducing mapping (DRM):** The permutation sequences’ distance has a distance loss which is guaranteed not to be more than a fixed amount compared to the binary sequences’ distance, such that $e_{ij} \geq d_{ij} + \delta$, $\delta \in \{-1, -2, \ldots\}$ for all $i \neq j$, with equality achieved at least once.

Thus, in general, $\delta$ defines the type of DPM, with $\delta = 0$ indicating a DCM, $\delta > 0$ indicating a DIM and $\delta < 0$ indicating a DRM. We now introduce the notation $M(n, M, \delta)$ to indicate DPMs from $n$-binary sequences to $M$-permutation sequences with $\delta$ indicating the lower bound on the distance change and the mapping type. The mapping in Example 1 would thus be an $M(2, 3, 1)$ mapping.

Most of the previously published constructions have been used to create DCMs. It is interesting to note that although the mappings in [12] are called increasing mappings, these are still conserving mappings according to the definitions presented here. Equal length sequences were used, i.e., $n = M$, and since $\max \{d_{ij}(x_i, x_j)\} = n$ and $\max \{d_{H}(y_i, y_j)\} = M$ we have that $\delta = 0$. These mappings did indeed show an increase except where it was impossible to do so: on the skew diagonal in the matrices where the maximum distances are located.

If one is able to construct a DIM, then it implies that a DCM and a DRM are also possible for the same parameters. In general, $\delta \leq M-n$, but since a mapping with a higher $\delta$ is better than a mapping with a lower $\delta$, it is only necessary to consider mappings with $\delta = M-n$. A consequence of this is that DIMs can only exist for $n < M$ and that for $n > M$ a DRM will always be obtained.

Using the multilevel construction that we will present, one would be able to create mappings of all three types. To our knowledge this is the first general construction that is able to achieve this.

III. UPPER BOUND ON THE SUM OF DISTANCES IN MAPPINGS

Since several different mappings can be found to satisfy the criterion of distance-preserving, a means to test the optimality of a given mapping is needed. As example, in [11] and [12] the distance expansion distribution was used to compare different mappings, though this can be tedious as $M \times M$ matrices have to be compared. Here we will use
the sum of the distances between sequences and show that there is an upper bound on this value.

Define the distance that a symbol in position \( k \) contributes toward the sum over the entries of \( E \) as
\[
d_{ij}^{(k)} (y_i, y_j) = \begin{cases} 
1, & \text{if } y_i^{(k)} \neq y_j^{(k)} \\
0, & \text{if } y_i^{(k)} = y_j^{(k)}. 
\end{cases}
\]

Let the matrix obtained from the distance for symbols in position \( k \), represented by \( E^{(k)} \), be constructed as
\[
E^{(k)} = \left[ e_{ij}^{(k)} \right], \quad \text{with } e_{ij}^{(k)} = d_{ij}^{(k)} (y_i, y_j).
\]

Let \( |E| \) and \( |E^{(k)}| \) be the sum of all the distances in the matrices \( E \) and \( E^{(k)} \), respectively, with
\[
|E| = \sum_{k=1}^{M} |E^{(k)}| \quad \text{and} \quad |E^{(k)}| = \sum_{i=1}^{m} \sum_{j=1}^{m} e_{ij}^{(k)}.
\]

(From the text it should be clear when \( | \cdot | \) denotes cardinality, as in \(|C_i|\), and when it denotes the sum of distances in the matrix, as in \(|E|\). Therefore it will not be specified each time.)

To determine an upper bound on the distance in a mapping, it is necessary to know what the maximum possible distance sum achievable is and since
\[
|E| = \sum_{k=1}^{M} |E^{(k)}|
\]

\( |E| \) will be maximized if all \( |E^{(k)}|, 1 \leq k \leq M, \) are maximized.

**Example 2:** Consider the \( \mathcal{M}(2, 4, 1) \) mapping \{1234, 1342, 1423, 2341\}. This mapping produces the following matrices:

\[
E = \begin{bmatrix}
0 & 3 & 3 & 4 \\
3 & 0 & 3 & 2 \\
3 & 3 & 0 & 4 \\
4 & 2 & 4 & 0
\end{bmatrix}
\]

\[
E^{(1)} = \begin{bmatrix}
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 \\
1 & 1 & 1 & 0
\end{bmatrix}
\]

\[
E^{(2)} = \begin{bmatrix}
0 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 \\
1 & 0 & 1 & 0
\end{bmatrix}
\]

\[
E^{(3)} = \begin{bmatrix}
0 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0
\end{bmatrix}
\]

with \(|E| = 38, |E^{(1)}| = 6, |E^{(2)}| = |E^{(3)}| = 10, \) and \(|E^{(4)}| = 12.\) Clearly the symbols 1, 1, 1, and 2 in position 1 do not build the same distance as the symbols 4, 2, 3, and 1 in position 4.

Next, consider the \( \mathcal{M}(2, 4, 2) \) mapping \{1234, 2341, 3412, 4123\} which produces

\[
E = \begin{bmatrix}
0 & 4 & 4 & 4 \\
4 & 0 & 4 & 4 \\
4 & 4 & 0 & 4 \\
4 & 4 & 4 & 0
\end{bmatrix}
\]

\[
E^{(1)} = E^{(2)} = E^{(3)} = E^{(4)} = \begin{bmatrix}
0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 1 & 0
\end{bmatrix}
\]

with \(|E| = 48, |E^{(1)}| = |E^{(2)}| = |E^{(3)}| = |E^{(4)}| = 12.\) In all positions the maximum distance is built because all the symbols differ from each other, hence \(|E|\) is also maximized.

The upper bound on the sum of the distances for mappings from all binary sequences to permutation sequences, derived in [13], depended on the following:

1) the cardinality of the set that is being mapped from, which is \( 2^n \) if all binary sequences of length \( n \) are considered;
2) the number of symbols in the set that is being mapped to, which is \( M \) for the permutation sequences;
3) the length of the sequences that is being mapped to, which is also \( M \) for the permutation sequences.

To generalize, consider mapping from some arbitrary set \( C \) to another arbitrary set \( C' \). The above variables will then, respectively, be

1) \(|C|\), the cardinality of \( C \);
2) \( q \), the number of symbols used in \( C' \);
3) \( l \), the length of sequences in \( C' \).

**Proposition 1:** An upper bound on the distance that symbols in the \( k \)th position can contribute in a mapping is
\[
|E_{\text{max}}^{(k)}| = |C|^2 - (2\alpha \beta + \beta + \alpha^2 q),
\]

where \( \alpha = \lfloor |C|/q \rfloor \) and \( \beta \equiv |C| \mod q \), with \( \lfloor \cdot \rfloor \) the floor function, producing the integer part after division and \( \mod \) producing the remainder after division.

**Proof:** Let \( m_i \) denote the number of times that symbol \( i \) appears in position \( k \) in all of the \(|C|\) sequences, with
\[
|C| = m_1 + m_2 + \cdots + m_q.
\]

The contribution from a single \( i \) symbol to \( |E^{(k)}| \) is \( |C| - m_i \), while the contribution from all the \( i \) symbols to \( |E^{(k)}| \) is \( m_i (|C| - m_i) \). Therefore,
\[
|E^{(k)}| = m_1 (|C| - m_1) + m_2 (|C| - m_2) + \cdots + m_q (|C| - m_q)
\]

\[
= |C|^2 - (m_1^2 + m_2^2 + \cdots + m_q^2) + \sum_{i=1}^{q} m_i (|C| - m_i).
\]

Using a Lagrange multiplier and (3) as the constraint, it can be shown that to maximize \( |E^{(k)}| \) we require
\[
m_1 = m_2 = \cdots = m_q.
\]

Substituting this into (3) results in
\[
|C| = q m_q \Rightarrow m_q = \frac{|C|}{q}, 1 \leq i \leq q.
\]

However, if \(|C| \) is not divisible by \( q \), then the \( m_q \) will not be integers. In this case, let \( \alpha = \lfloor |C|/q \rfloor \) and assume that an arbitrary \( j \) of the \( m_q = \alpha + 1 \) and that \((q-j)\) of the \( m_j = \alpha \). Using inequalities it can be proved that this assumption maximizes \( |E^{(k)}| \).

Equation (3) now becomes
\[
|C| = j (\alpha + 1) + (q - j) \alpha = \alpha q + j.
\]

We know that \(|C| = \alpha q + \beta\), thus \( j = \beta \). Substituting into (4), we prove the proposition with
\[
|E_{\text{max}}^{(k)}| = |C|^2 - (\beta (\alpha + 1)^2 + (q - \beta) \alpha^2)
\]

\[
= |C|^2 - (2\alpha \beta + \beta + \alpha^2 q).
\]
Proposition 2: An upper bound on the sum of the distances for any mapping is

$$|E_{\text{max}}| = f'(|C|)² - (2\alpha \beta + \beta + \alpha \beta² q).$$

(5)

Proof: This follows directly from Proposition 1. If, in a sequence of length \( l \), each symbol in the \( k \)th position contributes its maximum distance, then

$$|E_{\text{max}}| = \left| E^{(1)}_{\text{max}} \right| + \left| E^{(2)}_{\text{max}} \right| + \ldots + \left| E^{(l)}_{\text{max}} \right| = f'(|C|)² - (2\alpha \beta + \beta + \alpha \beta² q).$$

The following are special cases of these upper bounds.

1) \( q \)-Ary \((n, k)\) Codes: The \( q \)-ary \((n, k)\) codes where \( k \) input bits are encoded to \( n \) output bits can be seen as a \( k \rightarrow n \) mapping. For this case we have \(|C| = q^k\) and \( l = n \). The upper bound in (5) then simplifies to

$$|E_{\text{max}}| = (q - 1)q^{2k-1},$$

(6)

with \( \alpha = q^{k-1} \) and \( \beta = 0 \).

Example 3: Summing all the distances between the codewords of a binary \((7, 4)\) Hamming code, \(|E| = 896\) is obtained and using the upper bound in (6), \(|E_{\text{max}}| = 896\) is obtained. Similarly, summing all the distances between the codewords of a binary \((23, 12)\) Golay code, \(|E| = 192937984\) and \(|E_{\text{max}}| = 192937984\) are obtained. For a ternary \((11, 6)\) Golay code, \(|E| = 3897234\) and \(|E_{\text{max}}| = 3897234\) are obtained.

2) \( M(n, M, \delta)\) Permutation Codes: This is the upper bound derived in [13] for mappings from binary to permutation sequences. For this case we have \(|C| = 2^\alpha • q = M\), and \( l = M\), and (5) becomes

$$|E_{\text{max}}| = M²^{2n} - (2\beta \alpha \beta + \alpha \beta² M)$$

(7)

with \( \alpha = \left[ 2^n / M \right]\) and \( \beta = 2^{n \mod M} \).

Example 4: An \( M = 5 \) mapping [13] was found using an exhaustive search, with

$$\begin{bmatrix} 12534, 21435, 13254, 24153 \\ 21534, 23145, 25134, 21415 \\ 15243, 51423, 25134, 53241 \\ 43125, 21543, 31524, 35142 \\ 14235, 12453, 34251, 54132 \\ 42513, 32415, 34512, 43152 \\ 54321, 52431, 45231, 35421 \\ 52314, 45312, 43521, 53412 \end{bmatrix}$$

Summing all the distances between the sequences we obtain \(|E| = 4090\). Using the upper bound in (7), we also obtain \(|E_{\text{max}}| = 4090\).

A mapping’s distance optimality can be used as a measure of how close the distance sum of a mapping is to that of the upper bound.

Definition 4: The distance optimality of a mapping is given by

$$\eta = \frac{|E|}{|E_{\text{max}}|}$$

(8)

where \(|E|\) is the sum of the Hamming distances for the mapping and \(|E_{\text{max}}|\) is the upper bound on the sum of the Hamming distances for the mapping.

A mapping with \( \eta = 1 \) has thus attained the upper bound.

It needs to be noted that a mapping attaining the upper bound does not guarantee that the mapping is distance-preserving (or even a good error correcting code). For instance, if in a DPM the permutation sequences themselves are permuted in relation to the binary sequences, the same sum of distances will be retained, but the relation to the binary sequences is lost. This could result in a mapping that attains the upper bound but that is not distance-preserving.

IV. MULTILEVEL CONSTRUCTION

The idea of the multilevel construction is to create a DPM from binary sequences to multilevel binary sequences, which can then be transformed into permutation sequences.

In the construction the multilevel representation is used, which allows for several possible multilevel sequences on each level, where \( L = \left[ \log_2 M \right] \) is the number of levels. We call the set of possible sequences the multilevel permutations and use \( P_k \) to denote this set for level \( k \). To construct a DPM we choose a subset of the multilevel permutations for each level and assign input bits to each, where \( n_k \) is the number of input bits assigned to level \( k \). We call these subsets the multilevel components and use \( C_k \) to denote the subset for level \( k \). Thus, \( C_k \subseteq P_k \) with \(|C_k| = 2^n k \) for \( 1 \leq k \leq L \) and \( n = n_1 + n_2 + \ldots + n_L \).

The construction is illustrated in Fig. 1(b).

We define the multilevel identity matrix, \( B \), as the matrix obtained when each integer in the permutation identity sequence, \( 012 \ldots M - 1 \), is converted to binary and the resulting sequence is placed in the corresponding column in \( B \), with the least significant bit in the first row. As example, for \( M = 4 \) and 0123 we have

$$B = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

The multilevel permutations for level \( k \) are obtained by swapping (or transposing) the columns in \( B \), given the following rule: column \( i \) can only be swapped with column \( j \) if \( b_{i,k} \neq b_{j,k} \) and \( b_{i,m} = b_{j,m} \) for \( (k + 1) \leq m \leq L \). (Note that in the multilevel construction symbols \( 0, 1, \ldots, M - 1 \) will be used instead of \( 1, 2, \ldots, M \).)

First, an example will be used to show how all the permutations can be generated using the multilevel representation. The shorthand \( \text{swap}(i, j) \) will be used to denote the transposition of columns \( i \) and \( j \) or of symbols in positions \( i \) and \( j \).

Example 5: Let \( M = 3 \) and write the identity permutation, 012, as a multilevel identity matrix, such that

$$012 \rightarrow \begin{bmatrix} 010 \\ 001 \end{bmatrix}.$$
In this case \( \text{swap}(1, 2) \) is used as a 0 and a 1 appear in these columns on the first row, with both having a 0 below them. The set of possible sequences after transpositions for the first level is \( \{010, 100\} \), and this results in

\[
\begin{bmatrix}
010 \\
001
\end{bmatrix} \rightarrow 012, \quad \begin{bmatrix}
100 \\
001
\end{bmatrix} \rightarrow 102
\]

where the binary to decimal conversion is used on each column to obtain the permutation symbols.

Since there is no level below the second level, any transposition can take place. Therefore \( \text{swap}(1, 3) \) as well as \( \text{swap}(2, 3) \) can be used. The set of possible sequences after transpositions for the second level is \( \{001, 010, 100\} \). Thus resulting in

\[
\begin{bmatrix}
010 \\
001
\end{bmatrix} \rightarrow 012, \quad \begin{bmatrix}
100 \\
001
\end{bmatrix} \rightarrow 102
\]

\[
\begin{bmatrix}
010 \\
001
\end{bmatrix} \rightarrow 021, \quad \begin{bmatrix}
100 \\
010
\end{bmatrix} \rightarrow 120
\]

\[
\begin{bmatrix}
010 \\
100
\end{bmatrix} \rightarrow 210, \quad \begin{bmatrix}
001 \\
100
\end{bmatrix} \rightarrow 201
\]

where all the permutation sequences from \( S_3 \) have now been generated.

For \( M = 3 \) the multilevel permutations are

\[
P_1 = \{010, 100\}
\]
\[
P_2 = \{001, 010, 100\}.
\]

The number of possibilities for the first level is \( |P_1| = 2 \) and for the second level it is \( |P_2| = 3 \), thereby giving the total number of possibilities as 6, which is in agreement with \( |S_3| = 6 \).

To be able to choose a subset of the multilevel permutations to form a DPM, it must be shown that the multilevel permutations generate all the possible permutations from \( S_{3M} \). We illustrated this in Example 5 for \( M = 3 \) and will now prove it for \( M \), in general.

**Proposition 3:** If the multilevel permutations, \( P_k, 1 \leq k \leq L \) are used for the multilevel construction, then for \( M \) it generates all the permutation sequences from \( S_{3M} \), with \( |S_{3M}| = M! \).

**Proof:** Let \( \phi_{3M} \) denote the total number of transpositions possible for \( M, i.e., \phi_{3M} = |P_1||P_2|\cdots|P_L| \). Let \( M = M_1 + M_2 \), with \( M_1 = 2^{M_1-1} \) and \( M_2 = M - 2^{M_2-1} \). For \( M_1 = 1 \) we have \( \phi_1 = 1! = 1 \) since the only possibility is \( \{0\} \) and for \( M = 2 \) we have \( \phi_2 = 2! = 2 \) since the possibilities are \( \{01, 10\} \). For \( M = 3 \) a combination of \( \phi_1 \) and \( \phi_2 \) is obtained, combined with the possibilities for the last level

\[
\phi_3 = \phi_2 \times \phi_1 \times C(3, 1)
\]

Therefore, \( \phi_3 = \phi_2 \times \phi_1 \times C(3, 1) \). Next, \( M = 4 \) is dependent on \( \phi_2 \) and the possibilities for the last level

\[
\phi_4 = \phi_2 \times \phi_2 \times C(4, 2)
\]

Therefore, \( \phi_4 = \phi_2 \times \phi_2 \times C(4, 2) \). For \( M = 5 \) we have

\[
\phi_5 = \phi_4 \times \phi_1 \times C(5, 1)
\]
\[
\phi_6 = \phi_4 \times \phi_2 \times C(6, 2)
\]
\[
\phi_7 = \phi_4 \times \phi_3 \times C(7, 3)
\]
\[
\phi_8 = \phi_4 \times \phi_4 \times C(8, 4)
\]
\[
\phi_9 = \phi_4 \times \phi_5 \times C(9, 1)
\]
\[
\phi_{10} = \phi_4 \times \phi_2 \times C(10, 2)
\]
\[
\phi_{11} = \phi_4 \times \phi_3 \times C(11, 3)
\]

\[
\vdots
\]
\[
\phi_{3M} = \phi_{3M_1} \times \phi_{3M_2} \times \gamma(M, M_2)
\]

We know that \( \phi_k = k! \) for \( k \leq 3 \) and using induction it can be proved for the general case that

\[
\phi_{3M} = \phi_{2L-1} \times \phi_{3M-2L-1} \times \gamma(M, M_2)
\]

\[
\gamma(M, M_2)
\]

Since all the columns we started with were unique and all the transpositions were unique, for \( M \) we have that \( M! \) unique permutations are constructed.

The following two remarks should be kept in mind when creating DPMs.

**Remark 1:** The transpositions throughout a construction should be done consistently, e.g., if we have \( \{0011, 1001, 0110, 1100\} \), then

\[
\begin{align*}
0011 & \quad \{ \text{swap}(1, 3) \rightarrow 1001 \\
0110 & \quad \{ \text{swap}(2, 4) \rightarrow 0110 \\
1100 & \quad \{ \text{swap}(1, 3), (2, 4) \rightarrow 1100 
\end{align*}
\]

or if we have \( \{0011, 1010, 0101, 1100\} \), then

\[
\begin{align*}
0011 & \quad \{ \text{swap}(1, 4) \rightarrow 1010 \\
0110 & \quad \{ \text{swap}(2, 3) \rightarrow 0101 \\
1100 & \quad \{ \text{swap}(1, 4), (2, 3) \rightarrow 1100 
\end{align*}
\]

As will be shown in Example 7, each single input bit that is 1 (e.g., 1000, 0100, 0010, 0001), is assigned a transposition or a set of transpositions. If more than one bit is 1 (e.g., 1100 = 1000 + 0100), then a combination of the transpositions assigned to the respective single bits are used. Hence, in the example \( \text{swap}(1, 3) \) and \( \text{swap}(2, 4) \) can not be used with \( \text{swap}(1, 4)(2, 3) \). In general, the following transpositions will be used for each level:

Level 1: \( \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \cdot \cdot \cdot \)

Level 2: \( \{1, 3, 2, 4, 5, 7, 6, 8, 9, 11\} \cdot \cdot \cdot \)

Level 3: \( \{1, 5, 2, 6, 3, 7, 4, 8, 9, 13\} \cdot \cdot \cdot \)

Level 4: \( \{1, 9, 2, 10, 3, 11, 4, 12, 5, 13\} \cdot \cdot \cdot \)

\[\vdots\]

It is not necessary to use these transpositions, but it does have a nice structure that simplifies finding mappings from the binary sequences to the multilevel components.
Remark 2: As the size of $M$ increases, extra consideration should be given for mapping input bits to multilevel components. As example, for $M = 11$ we have

$$B = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$  

First, consider only the first eight columns where there are twelve 1s which we can assign transpositions (or equivalently input bits), as shown in Remark 1. A possible assignment of input bits is $n_1 = 4$, $n_2 = 4$, $n_3 = 2$ and $n_4 = 1$. If the first three levels’ transpositions are all limited to the first eight columns, it would mean that a maximum distance of 10 between input bits will only result in a maximum distance of 8 between permutation sequences. Therefore, either the input bits can be assigned in such a way that the first three levels will only have eight input bits, or it must be ensured that the first three levels’ transpositions are not limited to the first eight columns. If the mapping is written in the form presented in Example 7, this becomes easy to check.

Let $x$ be the binary sequence of length $n$ and break it up into subsequences of input bits for each level, with the $i$th input subsequence of length $n_i$ on level $k$ denoted by $x_{i,k}$. Assign $c_{i,k}$ to denote the $i$th multilevel component for level $k$, such that $C_k = \{c_{1,k}, c_{2,k}, \ldots, c_{n_k,k}\}$. Thus, we have a mapping of $x_{i,k} \rightarrow c_{i,k}$ on level $k$.

**Proposition 4:** For $1 \leq k \leq L$, choose a subset of the multilevel permutations, $P_k$, to form the multilevel components, $C_k$, such that $C_k \subseteq P_k$, and assign input bits of length $n_k$ to each possibility in $C_k$, such that $|C_k| = 2^{n_k}$ with $n_1 + n_2 + \ldots + n_L = n$. A DPM from binary sequences to permutation sequences is obtained if

1) $d_{ij}(c_{i,k}, c_{j,k}) \geq 2^{d_{ij}(x_{i,k}, x_{j,k})}$, for DCMs and DRMs;
2) $d_{ij}(c_{i,k}, c_{j,k}) \geq 2^{d_{ij}(x_{i,k}, x_{j,k})} + \delta - 1$, for DPMs with $\delta \geq 1$; for all $i \neq j$.

**Proof:** We will first prove 1) for the case of mapping any binary sequence to the multilevel components on a certain level and then prove that if 1) and 2) holds for all levels, the result will be a DPM for mapping binary sequences to permutation sequences.

Each bit that is 1 is assigned a transposition or a set of transpositions from a certain level. Any other combination of 1s can then be obtained by making use of a linear combination of all the single 1 bits. Since the transpositions on a level are independent (as set out in Remark 1), all the transpositions in the linear combination are going to contribute to the distance for that level. Thus, only considering distances for a certain level, say $k$, it will hold that $d_{ij}(c_{i,k}, c_{j,k}) \geq 2^{d_{ij}(x_{i,k}, x_{j,k})}$.

For different levels, any additional transpositions will contribute a minimum distance of 1, up to the maximum distance, which is $M$. If Remark 2 is taken into account, then it is ensured that there are not more input bits than distance being built anywhere in the mapping (except in the case of a reducing mapping where it is allowed to “lose” distance). Also, as transpositions are done for each level, there is no way for a symbol to return to its original position. Thus, it must always build distance whenever it is transposed.

Since the first transposition contributes 2 to the distance and any subsequent transposition contributes a minimum distance of 1, up to the maximum of $M$, in general the following will hold.

a) If the distance between the input subsequences is $d_i < M$, then the distance between the resulting permutation subsequences will be $d_p \geq d_i + 1$.

b) If the distance between the input subsequences is $d_i \geq M$, then the distance between the resulting permutation subsequences will be $d_p = M$.

Since $n < M$ for DIs, a) will always hold and we get the expression in 2). For DCMs and DRMs both a) and b) will hold and we get the expression in 1).

Since we are working with linear combinations, we will only have to compare sequences to the all zeros input sequence (or alternatively, the identity permutation sequence with no transpositions). The argument can then be generalized to any other sequence combination. Thus, if 1) and 2) holds for all sequences, then a DPM is obtained.

The following steps are used to obtain an $\mathcal{M}(n, M, \delta)$ DPM and to map a binary input sequence to a permutation sequence:

1) determine number of levels, $L = \lceil \log_2 M \rceil$;
2) determine the multilevel permutations, $P_k$, $1 \leq k \leq L$;
3) break input sequence of length $n$ into $L$ subsequences of length $n_k$, $1 \leq k \leq L$;
4) choose the multilevel components, $C_k$, $1 \leq k \leq L$, to satisfy the requirements of Proposition 4;
5) start on level $k = 1$ and swap columns of the multilevel identity matrix, transforming $B \rightarrow B'$, until $b'_{i,k} = c_{i,k}$, $1 \leq i \leq M$;
6) proceed to next level, $k = k + 1$, and repeat the previous step for that level. Continue until last level is reached, $k = L$;
7) determine permutation, $y^{(1)}, y^{(2)}, \ldots, y^{(n_L)}$, according to the columns of $B'$ as

$$y^{(i)} = \sum_{j=1}^{L} 2^{j-1} y^{(j)}.$$  

The following example illustrates how this construction is applied.

**Example 6:** Let $n = 6$ and $M = 6$ and start with the identity sequence

$$012345 \rightarrow \begin{bmatrix} 010101 \\ 001100 \\ 000011 \end{bmatrix}.$$  

As before, transpositions can only take place if a 0 and 1 can be transposed and the bits in the columns below are the same. We can use $\text{swap}(1, 2)$, as both columns have 00 below, $\text{swap}(3, 4)$ as both have 10 below, and similarly we can use $\text{swap}(5, 6)$, as both columns have 01 below.

Considering the second level, any of the columns $1, 2, 3,$ and 4 can be transposed as long as the bits differ, since all these have 0 below them in the third level. We can use $\text{swap}(2, 3), \text{swap}(1, 3), \text{swap}(2, 4), \text{swap}(1, 4)$, as well as $\text{swap}(1, 3)(2, 4)$.

Without level below the third, any transpositions can occur. As there are 15 possibilities we will not list them, however it should be clear from the multilevel permutations, which are

$$P_1 = \begin{bmatrix} 010101, 100101, 011001, 101001 \\ 010110, 100110, 011010, 101010 \end{bmatrix}$$  

$$P_2 = \begin{bmatrix} 001100, 010100, 100100, 011000, 011000, 101000, 110000, 000111, 000101, 010101, 010010, 010001, 001001, 011001, 011010, 001010, 011010, 010100, 010010, 001100 \end{bmatrix}$$  

$$P_3 = \begin{bmatrix} 001100, 010100, 100100, 011000, 011000, 101000, 110000, 000111, 000101, 010101, 010010, 010001, 001001, 011001, 011010, 001010, 011010, 010100, 010010, 001100 \end{bmatrix}.$$  

We choose $n_1 = 3$, $n_2 = 2$, and $n_3 = 1$ and map the input subsequences as follows:

$$\begin{bmatrix} 000, 001, 010 \\ 011, 100, 101 \end{bmatrix} \rightarrow \begin{bmatrix} 010101, 100101, 011101 \\ 101010, 010110, 100110 \end{bmatrix}$$  

$$\begin{bmatrix} 00, 01, 10, 11 \end{bmatrix} \rightarrow \begin{bmatrix} 001100, 100100, 011000, 110000 \end{bmatrix}$$  

$$\{0, 1\} \rightarrow \{000111, 110000\}.$$
We now have three mappings with subsequences of length $n_1, n_2,$ and $n_3$ respectively being mapped to multilevel components of length $M$. When performing these mappings, the requirements of Proposition 4 should be taken into account. If $D$ and $D'$ represent the distances between input subsequences and the distances between multilevel components, respectively, then it can be verified that for all three levels we have $d_{ij}^{D'} \geq 2d_{ij}^D$ for $i \neq j$, satisfying the requirements of Proposition 4.

The multilevel construction for $M(6, 6, 0)$ is then defined by the multilevel components

\begin{align*}
C_1 &= \{010101, 100101, 011001, 101001\} \\
C_2 &= \{001100, 100100, 011000, 110000\} \\
C_3 &= \{000011, 110000\}.
\end{align*}

With the multilevel components known, any binary sequence can be mapped to the corresponding permutation sequence for that mapping. Consider 011101 as the binary sequence and break it up into 011, 10, and 1 to, respectively, select the multilevel components 101001, 011000, and 110000. We start with

$$B = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}.$$ 

We then transpose the columns until the first row is equal to the chosen component for the first level. That is,

\begin{align*}
\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} &\rightarrow \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}, \\
\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} &\rightarrow \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}
\end{align*}

where $\text{swap}(1, 2)/(3, 4)$ was used. Next we transpose the columns until the second row is equal to the chosen component for the second level

\begin{align*}
\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} &\rightarrow \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}, \\
\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} &\rightarrow \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}
\end{align*}

where $\text{swap}(2, 4)$ was used. Finally, we transpose the columns until the third row is equal to the chosen component for the third level

\begin{align*}
\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} &\rightarrow \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}, \\
\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} &\rightarrow \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}
\end{align*}

where $\text{swap}(1, 5)/(2, 6)$ was used. By using the binary to decimal conversion on the columns, we obtain the permutation sequence 453012. Thus, in this mapping 011101 is mapped to 453012.

A more elegant method to obtain the permutations from the multilevel components is shown in the following example. Note that for convenience $x_1, x_2, \ldots$ are used for sequences instead of $x^{(1)}, x^{(2)}, \ldots$ as defined earlier.

**Example 7:** Consider the same multilevel components as in Example 6, with the binary sequence $x_1, x_2, x_3, x_4, x_5, x_6$ being mapped to the permutation sequence $y_1, y_2, y_3, y_4, y_5, y_6$.

The mapping $M(6, 6, 0)$ can then be written more concisely as

**Input:** $(x_1, x_2, x_3, x_4, x_5, x_6)$

**Output:** $(y_1, y_2, y_3, y_4, y_5, y_6)$

\begin{align*}
\text{begin} & \\
(y_1, y_2, y_3, y_4, y_5, y_6) &\leftarrow (0, 1, 2, 3, 4, 5) \\
\text{if } x_3 = 1 \text{ then } &\text{ swap } (y_1, y_2) \\
\text{if } x_2 = 1 \text{ then } &\text{ swap } (y_3, y_4) \\
\text{if } x_1 = 1 \text{ then } &\text{ swap } (y_5, y_6) \\
\text{if } x_5 = 1 \text{ then } &\text{ swap } (y_1, y_3) \\
\text{if } x_4 = 1 \text{ then } &\text{ swap } (y_2, y_4) \\
\text{if } x_6 = 1 \text{ then } &\text{ swap } (y_1, y_5)/(y_2, y_6) \\
\text{end.}
\end{align*}

Using the same input sequence as in Example 6, 011101, and using the steps above we have

\begin{align*}
x_3 &= 1 \rightarrow \text{ swap } (y_1, y_2) \rightarrow 012345 \\
x_2 &= 1 \rightarrow \text{ swap } (y_3, y_4) \rightarrow 102345 \\
x_1 &= 0 \rightarrow \rightarrow 102345 \\
x_5 &= 0 \rightarrow \rightarrow 102345 \\
x_4 &= 1 \rightarrow \text{ swap } (y_2, y_4) \rightarrow 123045 \\
x_6 &= 1 \rightarrow \text{ swap } (y_1, y_5)/(y_2, y_6) \rightarrow 453012
\end{align*}

where the same result is obtained.

In this way, the components of any multilevel construction can be written in terms of input bits and swaps.

In verifying Proposition 4, the multilevel components of various mappings are explicitly listed in Tables I–III. The multilevel components are presented for conserving, increasing and reducing mappings with $M \leq 8$, also listing the sum of distances and the upper bound on the sum of distances according to (7) for each mapping.

With the exception of Construction 2 [10], where the position function can be varied, all the other constructions have a set structure, i.e., the same mapping structure will always be obtained. In this new construction the mapping structure can be changed by either choosing different multilevel components or by assigning the number of input bits differently. Thus, we have a tradeoff where greater flexibility is gained in the mapping with a slightly more complex construction. Simulation results in [9] and [13] showed why this flexibility can be an important property. Four $M(8, 8, 0)$ mappings are presented in Table I to illustrate this flexibility.

### V. Distance Optimality of New DPMs

When $M$ is some integer power of 2, such that $M = 2^l, l \in \{1, 2, 3, \ldots \}$, the resulting DPM will achieve the upper bound on the distance sum, provided the maximum distances were obtained in the multilevel construction.

**Proposition 5:** Any multilevel DPM with $M = 2^l, l \in \{1, 2, 3, \ldots \}$ will attain the upper bound $|E_{\max}|$, provided the maximum distances were achieved between all the multilevel components.

**Proof:** The maximum distance between multilevel components is achieved when each component’s complement is also in the set. The only values of $M$ for which it is possible to have complements with the same weight, is for $M = 2^l, l \in \{1, 2, 3, \ldots \}$. For instance, for $M = 4$ the complement of 0011 is 1100, however for $M = 5$ the complement of 00011 is 11000. With $M = 2^l, \alpha$ and $\beta$ in (7) becomes $\alpha = 2^{n-l}$ and $\beta = 0$. Each symbol will appear $2^{n-l}$ times in all the positions. Thus, we need to prove that if every component’s complement is present among the multilevel components, then all the symbols will appear $2^{n-l}$ times in all the positions.

Recall that we start with the multilevel identity matrix and then transpose the columns until we get the required permutation. Now consider
starting with the permutation identity sequence with symbol 0 in position 1 for each of the 2^n sequences. Since each component’s complement is present, we have that C_1 = \{01\ldots01,\ldots10 \ldots10\}. Thus, of the 2^n sequences, 2^{n-1} components will still have symbol 0 in position 1 and 2^{n-1} will have symbol 0 in position 2 (recalling that for the first level \text{swap}(1, 2) is used). For the second level, we have C_2 = \{00\ldots11, \ldots00, 01\ldots11\} and \text{swap}(1, 3). Of the 2^{n-1} sequences that still have symbol 0 in position 1, 2^{n-2} will now have symbol 0 in position 1 and 2^{n-2} will have symbol 0 in position 3.

Thus, for each additional level the number of symbol 0s in position 1 is halved. With M = 2^l, the number of levels is L = l. After the transpositions are done for all levels, symbol 0 will be in position 1, 2^n times. This same reasoning can be used for symbol 0 in any other position, as well as for any other symbol in any position. Therefore, all the symbols will appear 2^n times in all the positions.

In Tables I–III one can see that for all M = 4 and M = 8 mappings the upper bound on the distance sum is achieved. For other values of M, it is also possible to increase the distance between multilevel components to get mappings that have \(|E|\) values closer to the upper bound than those listed in Tables I–III, however this is on a “trial and error” basis as a DPM cannot be guaranteed. As example, we can have a mapping with
\[
C_1 = \{010101, 010011, 011001, 011011, 101001, 101101\}
\]
\[
C_2 = \{001110, 100001\}
\]
\[
C_3 = \{000011, 000110, 011000, 110000\}.
\]

In this case the transpositions for C_3 cannot be written as combinations of others, as in Example 7, and some experimentation is necessary to obtain the correct swaps. The distance between components in C_3 is increased, thereby increasing the overall distance for the mapping with \(|E| = 20224\).

Another possibility is to use a different starting matrix. Instead of using the matrix derived from the permutation identity sequence, one can use any arbitrary starting matrix, provided all the columns are unique. For instance, for M = 9
\[
\begin{pmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{pmatrix}
\]
can be used as the starting matrix. The multilevel permutations can be determined using the same rules as set out previously, from which the multilevel components can then be chosen. We do not explicitly state the multilevel components, but a mapping can be found with \(|E| = 2023424\), which is slightly larger than the 1982464 achieved when using the multilevel identity matrix.

These examples suggest that it could be possible to obtain constructions that attain the upper bound for all values of M, however at present this remains an open problem.

In Table IV we compare the distance sums of known mappings for 4 \leq M \leq 16 and n = M with the upper bound and in Fig. 2 we compare the corresponding mappings’ distance optimality using (8). Mappings are used from [5], using the prefix method, from [10] using Constructions 2 and 3, from [11] and from [12] using Construction 2. (Note that the prefix construction was only defined for M \leq 8, Construction 3 [10] is only defined for even M and the construction from [11] only for odd M.)

The distance optimality of all the previous mappings is decreasing as M is increasing, while the distance optimality of our new construction generally increases toward 1 as M increases. This is due to the set structure in previous constructions where the same number of transpositions are used, regardless of the size of M. In contrast, the multilevel construction makes use of the additional transpositions as M increases.

Although the construction of Wadayama and Vinck [14] was not used to create DPMs and can therefore not be compared directly, we...
### TABLE II
#### Distance-Increasing Mappings

<table>
<thead>
<tr>
<th>M(n, M, δ)</th>
<th>C₁</th>
<th>C₂</th>
<th>Multilevel construction</th>
<th>E₁</th>
<th>E₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>M(3, 4, 1)</td>
<td>{0101, 1010}</td>
<td>{0011, 0110, 1001, 1100}</td>
<td>192</td>
<td>192</td>
<td></td>
</tr>
<tr>
<td>M(3, 6, 3)</td>
<td>{010101, 101010}</td>
<td>{001100, 110000}</td>
<td>304</td>
<td>312</td>
<td></td>
</tr>
<tr>
<td>M(3, 7, 4)</td>
<td>{0101010, 1010100}</td>
<td>{0011001, 1100100}</td>
<td>368</td>
<td>378</td>
<td></td>
</tr>
<tr>
<td>M(3, 8, 5)</td>
<td>{01010101, 10101010}</td>
<td>{00110011, 11001100}</td>
<td>448</td>
<td>448</td>
<td></td>
</tr>
<tr>
<td>M(4, 5, 1)</td>
<td>{010100, 100101, 011000, 101000}</td>
<td>{001100, 110000}</td>
<td>928</td>
<td>1020</td>
<td></td>
</tr>
<tr>
<td>M(5, 6, 1)</td>
<td>{0101010, 1010100}</td>
<td>{0011000, 0101000, 0110000, 1100000}</td>
<td>4864</td>
<td>5112</td>
<td></td>
</tr>
<tr>
<td>M(5, 8, 3)</td>
<td>{01010101, 10101010}</td>
<td>{00110011, 01101100, 11000011, 11001100}</td>
<td>7168</td>
<td>7168</td>
<td></td>
</tr>
<tr>
<td>M(6, 7, 1)</td>
<td>{0101010, 1001010, 0110010, 1010100, 0101100, 0110100, 0111000}</td>
<td>{0011001, 1001010, 0110010, 1100100}</td>
<td>23552</td>
<td>24570</td>
<td></td>
</tr>
<tr>
<td>M(6, 8, 2)</td>
<td>{01010101, 01011010, 10010101, 10101010}</td>
<td>{00110011, 01101100, 11000011, 11001100}</td>
<td>28672</td>
<td>28672</td>
<td></td>
</tr>
<tr>
<td>M(7, 8, 1)</td>
<td>{01010101, 01010101, 01101010, 01111000, 11001001, 10011010, 10100110, 10101010}</td>
<td>{00110011, 01101010, 10011001, 11001100}</td>
<td>114688</td>
<td>114688</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE III
#### Distance-Reducing Mappings

<table>
<thead>
<tr>
<th>M(n, M, δ)</th>
<th>C₁</th>
<th>C₂</th>
<th>Multilevel construction</th>
<th>E₁</th>
<th>E₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>M(7, 6, 1)</td>
<td>{010101, 010110, 011001, 011101, 100110, 101011, 101010}</td>
<td>{001100, 101000, 100100, 110000}</td>
<td>77824</td>
<td>81912</td>
<td></td>
</tr>
<tr>
<td>M(8, 7, 1)</td>
<td>{0101010, 1001010, 0101010, 1010100, 0110100, 1100110, 1010101, 0111010}</td>
<td>{0011001, 1001001, 0110110, 1100011}</td>
<td>376832</td>
<td>393204</td>
<td></td>
</tr>
<tr>
<td>M(9, 7, 2)</td>
<td>{01010100, 10010100, 01010100, 10101100, 01101100, 11001110, 10101110, 11101110}</td>
<td>{00110011, 11001101, 01110011, 11010011, 11010110, 11101110}</td>
<td>1507328</td>
<td>1572858</td>
<td></td>
</tr>
<tr>
<td>M(9, 8, 1)</td>
<td>{01010101, 01010101, 01011010, 01100101, 10010010, 10101100, 10101101, 01010011}</td>
<td>{00110011, 01101101, 10011100, 10101110, 11011100, 11101110}</td>
<td>1835008</td>
<td>1835008</td>
<td></td>
</tr>
<tr>
<td>M(10, 8, 2)</td>
<td>{01010101, 01010101, 01010101, 10010101, 10101010, 10101010, 10101010, 10101010}</td>
<td>{00110011, 01101101, 01110110, 10011110, 10101110, 11011110, 11101110}</td>
<td>7340032</td>
<td>7340032</td>
<td></td>
</tr>
<tr>
<td>M(11, 8, 3)</td>
<td>{01010101, 01010101, 01010101, 01010101, 01010101, 01010101, 01010101, 01010101}</td>
<td>{00110011, 01101101, 01111100, 10011110, 10101110, 11011110, 11101110, 11111110}</td>
<td>29360128</td>
<td>29360128</td>
<td></td>
</tr>
<tr>
<td>M(12, 8, 4)</td>
<td>{01010101, 01010101, 01010101, 01010101, 01010101, 01010101, 01010101, 01010101}</td>
<td>{00110011, 00111100, 01011010, 01011010, 10011010, 10011010, 10101010, 10101010}</td>
<td>117440512</td>
<td>117440512</td>
<td></td>
</tr>
</tbody>
</table>
can make the following observations. Each constant weight code word in their construction can be assigned input bits, thereby creating a mapping from all binary sequences to constant weight codes to permutation codes. However, since the $n$-length constant weight codes are subsets of the code containing all the $2^n$ binary sequences, the length of the input sequences in this case would have to be smaller than $n$ and therefore only DIMs can result. Using their construction, $\mathcal{M}(2, 4, 2)$ and $\mathcal{M}(3, 8, 5)$ DIMs can be created, both attaining the upper bound on the distance sum.

VI. CONCLUSION

We showed that an upper bound exists on the sum of Hamming distances between sequences when mapping from one set of sequences to another set of sequences, which proves to be useful in comparing the distance optimality of different mappings. An upper bound on the sum of distances for mappings from binary sequences to permutation sequences was shown to be a special case of this upper bound, and this was used to compare different DPMs.

We also presented a new method to obtain DPMs by making use of a multilevel construction. It was shown that using the multilevel construction one can obtain the upper bound on the distance sum when $M = 2^l$, with $l$ any positive integer. In almost every case the new construction’s distance sum is greater than that of the previous known mappings. The importance of this was shown by performance results \cite{13} of distance optimal mappings on a power-line communications channel. In addition, this construction can also be used for mappings from binary sequences to permutation sequences with repeating symbols, such as $123455$, using the same principles.

Although this construction does not produce a mapping for $M$ in general, it simplifies the process of finding one by breaking it down into smaller mappings. For instance, to map $n = 16$ to $M = 16$ requires one to choose $65536$ permutation sequences for the binary sequences to map to, while preserving the distance. With the new construction this
can be broken down to four mappings of $n = 4$ to $M = 16$ which only requires 16 binary sequences in each smaller mapping to be mapped while preserving the distance.

ACKNOWLEDGMENT

The authors would like to thank Prof. I. Broere for his helpful comments and verifying of the proofs as well as the anonymous reviewers for comments and criticism that improved this correspondence, in particular Reviewer B for suggesting an alternative proof to Proposition 1.

REFERENCES


Improving the Alphabet-Size in Expander-Based Code Constructions
Eran Rom and Amnon Ta-Shma

Abstract—Various code constructions use expander graphs to improve the error resilience. Often the use of expanding graphs comes at the expense of the alphabet size. In this correspondence, we show that by replacing the balanced expanding graphs used in the above constructions with unbalanced dispersers the alphabet size can be dramatically improved.

Index Terms—Disperser graphs, expander graphs, extractor codes, list decoding, randomness extractors.

I. INTRODUCTION

Error-correcting codes were built to deal with the task of correcting errors in transmission over noisy channels. Formally, an $(N, n, d)$, $q$ error correcting code over alphabet $\Sigma$, where $|\Sigma| = q$, is a subset $C \subseteq \Sigma^N$ of cardinality $q^n$ in which every two elements are distinct in at least $d$ coordinates. $n$ is called the dimension of the code, $N$ the block length of the code, and $d$ the distance of the code. We call $\frac{d}{N}$ the relative distance of the code. If $C$ is a linear subspace of $[\mathbb{F}_q]^N$, where $\Sigma$ is associated with some finite field $\mathbb{F}_q$ we say that $C$ is a linear code, and denote it $[N, n, d]_q$ code. From the definition we see that one can uniquely identify a codeword in which at most $\frac{d-1}{N}$ errors occurred during transmission. Moreover, since two codewords from $\Sigma^N$ can differ in at most $N$ coordinates, the largest number of errors from which unique decoding is possible is $N/2$.

This motivates the list decoding problem, first defined in [4]. In list decoding we give up unique decoding, allowing potentially more than $N/2$ errors, and require that there are only few possible codewords having some modest agreement with any received word. Formally, we say that an $(N, n, d)_{q, e}$ code $C$ is $(p, K)$-list decodable, if for every $r \in \Sigma^N$, $|\{c \in C \mid \Delta(r, c) \leq pN\}| \leq K$, where $\Delta(x, y)$ is the number of coordinates in which $x$ and $y$ differ. That is, the number of codewords which agree with $r$ on at least $(1-p)N$ coordinates is at most $K$. We call the ratio $n/N$ the rate of the code, and $p$ the error rate.

We can demonstrate the difference between unique decoding and list decoding with Reed-Solomon codes. Reed-Solomon codes are linear $[N, n, N-n+1]_q$ codes, defined for every $q$ such that $F_q$ is a finite field, and $n \leq N \leq q$. Every $(N, n, d)_{q, e}$ code is $(1-\sqrt{1-d/N}, qN)$-list decodable [5, Lecture 8]. For Reed–Solomon codes there exists an efficient list decoding algorithm [6]. Thus, unique decoding is possible with at most $N/2$ errors, while by [6] list decoding is possible with up to $N - \sqrt{Nn}$ errors, and the number of all possible decodings is small.

Manuscript received December 17, 2004; revised December 25, 2005. This work was supported by the Israel Science Foundation, by the Binational Science Foundation, and by the EU Integrated Project QAP. The material in this correspondence was presented in part at STACS 2005, Stuttgart, Germany, February 2005.

E. Rom is at 38 Anatot St., Tel-Aviv 69080, Israel (e-mail: eranrom@post.tau.ac.il).
A. Ta-Shma is with the Computer Science Department, Tel-Aviv University, Ramat-Aviv, Tel-Aviv 69978, Israel (e-mail: amnon@post.tau.ac.il).
Communicated by A. Ashikhmin, Associate Editor for Coding Theory.
Digital Object Identifier 10.1109/TIT.2006.878166

"We will use $n$ to denote the dimension of a code to avoid confusion with with the min-entropy parameter of extractors and dispersers, for which $k$ is usually reserved.

0018-9448/2006/$20.00 © 2006 IEEE