A Coding Algorithm for Constant Weight Vectors: A Geometric Approach Based on Dissections

Chao Tian, *Member, IEEE*, Vinay A. Vaishampayan, *Senior Member, IEEE*, and N. J. A. Sloane, *Fellow, IEEE*

***Abstract*—We present a novel technique for encoding and decoding constant weight binary vectors that uses a geometric interpretation of the codebook. Our technique is based on embedding the codebook in a Euclidean space of dimension equal to the weight of the code. The encoder and decoder mappings are then interpreted as a bijection between a certain hyper-rectangle and a polytope in this Euclidean space. An inductive dissection algorithm is developed for constructing such a bijection. We prove that the algorithm is correct and then analyze its complexity. The complexity depends on the weight of the vector, rather than on the block length as in other algorithms. This approach is advantageous when the weight is smaller than the square root of the block length.**

***Index Terms*—Constant weight codes, encoding algorithms, dissections, polyhedral dissections, bijections, mappings, Dehn invariant.**

I. Introduction

We consider the problem of encoding and decoding binary vectors of constant Hamming weight w and block length n. Such sets of vectors (hereafter referred to as codes) are useful in a variety of applications: a few examples are fault-tolerant circuit design and computing [15], pattern generation for circuit testing [25], identification coding [28], and optical overlay networks [26].

The problem of interest is that of designing the encoder and decoder, i.e., the problem of mapping all binary (information) vectors of a given length onto a subset of length-n vectors of constant Hamming weight w in a one-to-one manner. In this work, we propose a novel geometric method in which information and code vectors are represented by vectors in w-dimensional Euclidean space, covering polytopes for the two sets are identified, and a one-to-one mapping is established by dissecting the covering polytopes in a specific manner. This approach results in an invertible integer-to-integer mapping, thereby ensuring unique decodability. The proposed algorithm has a natural recursive structure, and an inductive proof is given for unique decodability. The issue of efficient encoding and decoding is also addressed. We show that the proposed algorithm has complexity $O(w^2)$, where w is the weight of the codeword, independent of the codeword length n.

Compared to a linear-time encoding algorithm [16], the algorithm presented here is faster only when $w = O(\sqrt{n})$, in which case the rate of the resulting code, $(1/n)\log\binom{n}{w}$, approaches zero as $n \to \infty$. However, we view this paper as the starting point of a new approach to the classic problem of encoding and decoding constant-weight codes, and we believe there is considerable room for improvement. Some steps in this direction can be found in [22], [27]. Further, the geometric view advanced here is also of potential significance to areas such as source coding and simulation.

Dissections are of considerable interest in geometry, partly as a source of puzzles, but more importantly because they are intrinsic to the notion of volume. Of the 23 problems posed by David Hilbert at the International Congress of Mathematicians in 1900, the third problem dealt with dissections. Hilbert asked for a proof that there are two tetrahedra of the same volume with the property that it is impossible to dissect one into a finite number of pieces that can be rearranged to give the other, i.e., that the two tetrahedra are not equidecomposable. The problem was immediately solved by Dehn [6]. In 1965, after 20 years of effort, Sydler [24] completed Dehn's work. The Dehn–Sydler theorem states that a necessary and sufficient condition for two polyhedra to be equidecomposable is that they have the same volume and the same Dehn invariant. This invariant is a certain function of the edge lengths and dihedral angles of the polyhedron. An analogous theorem holds in four dimensions (Jessen [11]), but in higher dimensions it is known only that equality of the Dehn invariants is a necessary condition. In two dimensions, any two polygons of equal area are equidecomposable, a result due to Bolyai and Gerwein (see Boltianskii [1]). Among other books dealing with the classical dissection problem in two and three dimensions we mention in particular Frederickson [7], Lindgren [13], and Sah [19].

The remainder of the paper is organized as follows. We provide background and review relevant previous work in Section II. Section III describes our geometric approach and gives some low-dimensional examples. Encoding and decoding algorithms are then given in Section IV, and the correctness of the algorithms is established. Section V summarizes the paper.

II. Background and Previous Methods

Let us denote the Hamming weight of a length-n binary sequence $\boldsymbol{s} := (s_1, s_2, \ldots, s_n)$ by $w(s) := |\{s_i : s_i = 1\}|$, where $|\cdot|$ is the cardinality of a set.

Definition 1: An (n, w) constant-weight binary code $\mathcal{C}$ is a set of length-n sequences such that any sequence $\boldsymbol{s} \in \mathcal{C}$ has weight $w(\boldsymbol{s}) = w$.

If $\mathcal{C}$ is an (n, w) constant-weight code, then its rate $R := (1/n)\log_2|\mathcal{C}| \leq R(n, w) := (1/n)\log_2\binom{n}{w}$. For fixed β and $w = \lfloor \beta n \rfloor$, we have

$$\overline{R} := \lim_{n\to\infty} R(n, w) = h(\beta) \tag{1}$$

Manuscript received June 08, 2007; revised September 01, 2008. Current version published February 25, 2009.

The authors are with AT&T Shannon Laboratory, Florham Park, NJ 07932 USA (e-mail: tian@reseach.att.com; vinay@reseach.att.com; njas@reseach.att.com).

Communicated by G. Serousi, Associate Editor for Coding Theory.

Digital Object Identifier 10.1109/TIT.2008.2011441

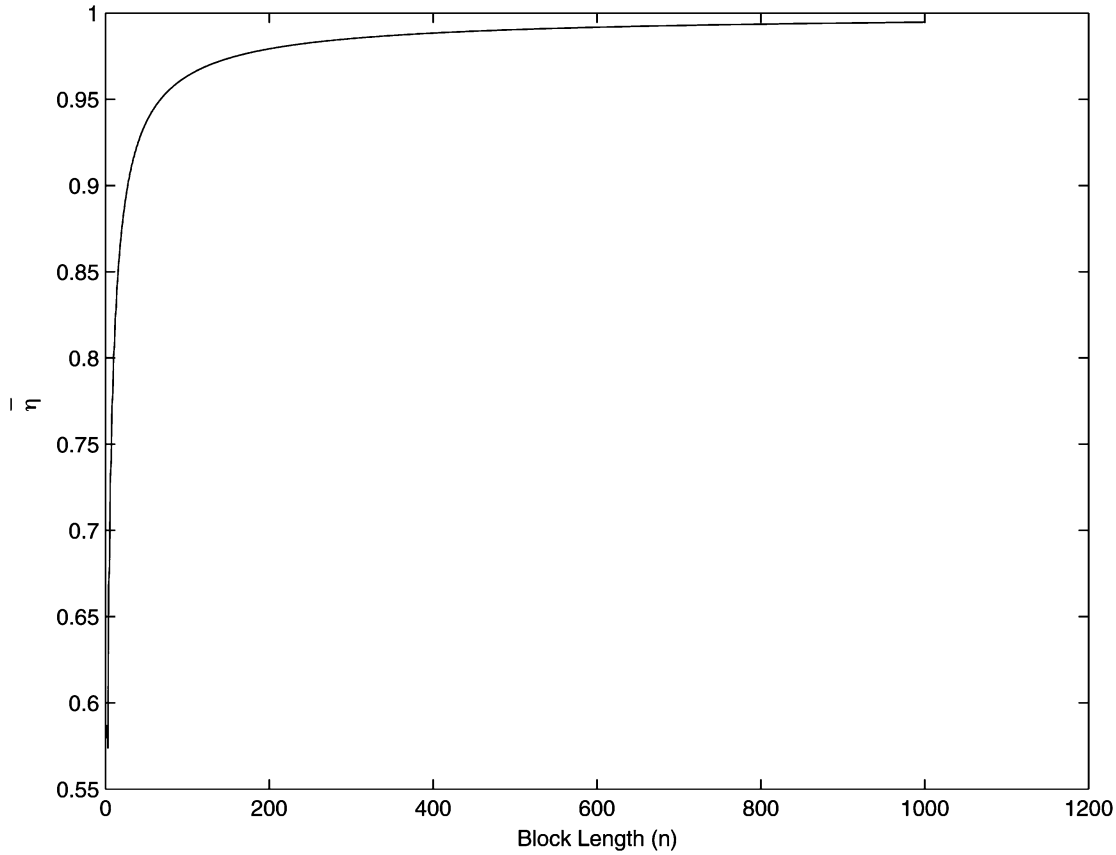

Fig. 1. Efficiency $\overline{\eta}$ as a function of block length when $\beta = 1/2$.

where $h(\beta) := -\beta \log_2(\beta) - (1-\beta)\log_2(1-\beta)$ is the entropy function. Thus, $\overline{R}$ is maximized when $\beta = 1/2$, i.e., the asymptotic rate is highest when the code is balanced.

The (asymptotic) efficiency of a code relative to an infinite-length code with the same weight to length ratio w/n, given by $\eta := R/\overline{R}$, can be written as $\eta = \eta_1 \overline{\eta}$ where $\eta_1 := R/R(n,w)$ and $\overline{\eta} := R(n,w)/\overline{R}$. The first term, η_1, is the efficiency of a particular code relative to the best possible code with the same length and weight; the second term, $\overline{\eta}$, is the efficiency of the best finite-length code relative to the best infinite-length code.

From Stirling's formula we have

$$\overline{\eta} \approx 1 - \frac{\log_2(2\pi n\beta(1-\beta))}{2nh(\beta)}. \tag{2}$$

A plot of $\overline{\eta}$ as a function of n is given in Fig. 1 for $\beta = 1/2$. The slow convergence visible here is the reason one needs codes with large block lengths.

The problem of finding efficient algorithms for encoding and decoding constant-weight vectors has been considered by several authors. We briefly discuss two previous methods that are relevant to our work. The first, a general-purpose technique based on the idea of lexicographic ordering and enumeration of codewords in a codebook (Schalkwijk [20], Cover [2]) is an example of ranking/unranking algorithms that are well studied in the combinatorial literature (Nijenhuis and Wilf [14]). We refer to this as the *enumerative* approach. The second (Knuth [12]) is a special-purpose, highly efficient technique that works for balanced codes, i.e., when $w = \lfloor (n/2) \rfloor$, and is referred to as the *complementation* method.

The enumerative approach orders the codewords lexicographically (with respect to the partial order defined by $0 < 1$), as in a dictionary. The encoder computes the codeword from its dictionary index, and the decoder computes the dictionary index from the codeword. The method is effective because there is a simple formula involving binomial coefficients for computing the lexicographic index of a codeword. The resulting code is fully efficient in the sense that $\eta_1 = 1$. However, this method requires the computation of the exact values of binomial coefficients $\binom{n}{k}$, and requires registers of length $O(n)$, which limits its usefulness.

An alternative is to use arithmetic coding (Rissanen and Langdon [18], Rissanen [17]; see also Cover and Thomas [3, Sec. 13.3]). Arithmetic coding is an efficient variable length source coding technique for finite alphabet sources. Given a source alphabet and a simple probability model for sequences $\boldsymbol{x}$, let $p(\boldsymbol{x})$ and $F(\boldsymbol{x})$ denote the probability distribution and cumulative distribution function, respectively. An arithmetic encoder represents $\boldsymbol{x}$ by a number in the interval $(F(\boldsymbol{x}) - p(\boldsymbol{x}), F(\boldsymbol{x})]$. The implementation of such a coder can also run into problems with very long registers, but elegant finite-length implementations are known and are widely used (Witten, Neal and Cleary [30]). For constant-weight codes, the idea is to reverse the roles of encoder and decoder, i.e., to use an arithmetic decoder as an encoder and an arithmetic encoder as a constant-weight decoder (Ramabadran [16]). Ramabadran

gives an efficient algorithm based on an adaptive probability model, in the sense that the probability that the incoming bit is a 1 depends on the number of 1's that have already occurred. This approach successfully overcomes the finite-register-length constraints associated with computing the binomial coefficients and the resulting efficiency is often very high, in many cases the loss of information being at most one bit. The encoding complexity of the method is $O(n)$.

Knuth's complementation method [12] relies on the key observation that if the bits of a length-n binary sequence are complemented sequentially, starting from the beginning, there must be a point at which the weight is equal to $\lfloor n/2 \rfloor$. Given the transformed sequence, it is possible to recover the original sequence by specifying how many bits were complemented (or the weight of the original sequence). This information is provided by a (relatively short) constant-weight check string, and the resulting code consists of the transformed sequence followed by the constant-weight check bits. In a series of papers, Bose and colleagues extended Knuth's method in various ways, and determined the limits of this approach (see [31] and references therein). The method is simple and efficient, and even though the overall complexity is $O(n)$, for $n = 100$ we found it to be eight times as fast as the method based on arithmetic codes. However, the method only works for balanced codes, which restricts its applicability.

The two techniques that we have described above both have complexity that depends on the length n of the codewords. In contrast, the complexity of our algorithm depends only on the weight w, which makes it more suitable for codes with relatively low weight.

As a final piece of background information, we define what we mean by a dissection. We assume the reader is familiar with the terminology of polytopes (see, for example, Coxeter [4], Grünbaum [8], Ziegler [32]). Two polytopes P and Q in $\mathbb{R}^w$ are said to be *congruent* if Q can be obtained from P by a translation, a rotation, and possibly a reflection in a hyperplane. Two polytopes P and Q in $\mathbb{R}^w$ are said to be *equidecomposable* if they can be decomposed into finite sets of polytopes $P_1, \ldots, P_t$ and $Q_1, \ldots, Q_t$, respectively, for some positive integer t, such that P_i and Q_i are congruent for all $i = 1, \ldots, t$ (see Frederickson [7]). That is, P is the disjoint union of the polytopes P_i, and similarly for Q. If this is the case then we say that P can be *dissected* to give Q (and that Q can be dissected to give P).

Note that we allow reflections in the dissection: there are at least four reasons for doing so. i) It makes no difference to the *existence* of the dissection, since if two polytopes are equidecomposable using reflections they are also equidecomposable without using reflections. This is a classical theorem in two and three dimensions [7, Ch. 20] and the proof is easily generalized to higher dimensions. ii) When studying congruences, it is simpler not to have to worry about whether the determinant of the orthogonal matrix has determinant $+1$ or -1. iii) Allowing reflections often reduces the number of pieces. iv) Since our dissections are mostly in dimensions greater than three, the question of "physical realizability" is usually irrelevant. Note also that we do not require that the P_i can be obtained from P by a succession of cuts along infinite hyperplanes. All we require is that P be a disjoint union of the P_i.

One final technical point: when defining dissections using coordinates, as in (3), (4) below, we use a mixture of $\leq$ and $<$ signs in order to have unambiguously defined maps. This is essential for our application. On the other hand, it means that the "pieces" in the dissection may be missing certain boundaries. It should therefore be understood that if we were focusing on the dissections themselves, we would replace each piece by its topological closure.

For further information about dissections see the books mentioned in Section I.

III. The Geometric Interpretation

In this section, we first consider the problem of encoding and decoding a binary constant-weight code of weight $w = 2$ and arbitrary length n, i.e., where there are only two bits set to 1 in any codeword. Our approach is based on the fact that vectors of weight two can be represented as points in two-dimensional Euclidean space, and can be scaled, or normalized, to lie in a right triangle. This approach is then extended, first to weight $w = 3$, and then to arbitrary weights w.

For any weight w and block length n, let $\mathcal{C}_w$ denote the set of all weight w vectors, with $|\mathcal{C}_w| = \binom{n}{w}$. Our codebook $\mathcal{C}$ will be a subset of $\mathcal{C}_w$, and will be equal to $\mathcal{C}_w$ for a fully efficient code, i.e., when $\eta_1 = 1$. We will represent a codeword by the w-tuple $\boldsymbol{y}' := (y'_1, y'_2, \ldots, y'_w)$, $1 \leq y'_1 < y'_2 < \cdots < y'_w \leq n$, where y'_i is the position of the ith 1 in the codeword, counting from the left. If we normalize these indices y'_i by dividing them by n, the codebook $\mathcal{C}$ becomes a discrete subset of the polytope T_w, the convex hull of the points $0^w, 0^{w-1}1, 0^{w-2}11, \ldots, 01^{w-1}, 1^w$. T_2 is a right triangle, T_3 is a right tetrahedron, and in general we will call T_w a *unit orthoscheme*.[1]

The set of inputs to the encoder will be denoted by $\mathcal{R}_w$: we assume that this consists of w-tuples $\boldsymbol{y} := (y_1, y_2, \ldots, y_w)$ which range over a w-dimensional hyper-rectangle or "brick." After normalization by dividing the y_i by n, we may assume that the input vector is a point in the hyper-rectangle or "brick"

$$B_w := [0, 1) \times [1 - 1/2, 1) \times \cdots \times [1 - 1/w, 1).$$

We will use $\boldsymbol{x} := (x_1, x_2, \ldots, x_w) = \boldsymbol{y}/n \in B_w$ and $\boldsymbol{x}' := (x'_1, x'_2, \ldots, x'_w) = \boldsymbol{y}'/n \in T_w$ to denote the normalized versions of the input vector and codeword, respectively, defined by $x_i := y_i/n$ and $x'_i := y'_i/n$ for $i = 1, \ldots, w$.

The basic idea underlying our approach is to find a dissection of B_w that gives T_w. The encoding and decoding algorithms are obtained by tracking how the points $\boldsymbol{y}$ and $\boldsymbol{y}'$ move during the dissection.

The volume of B_w is $1 \times \frac{1}{2} \times \frac{1}{3} \times \cdots \times \frac{1}{w} = \frac{1}{w!}$. This is also the volume of T_w, as the following argument shows. Classify the points $\boldsymbol{x} = (x_1, \ldots, x_w)$ in the unit cube $[0, 1]^w$ into $w!$ regions according to their order when sorted; the regions are congruent, so all have volume $1/w!$, and the region where the x_i are in nondecreasing order is T_w.

We now return to the case $w = 2$. There are many ways to dissect the rectangle B_2 into the right triangle T_2. We will

[1] An *orthoscheme* is a w-dimensional simplex having an edge path consisting of w totally orthogonal vectors (Coxeter [4]). In a *unit orthoscheme* these edges all have length 1. Hadwiger [9] showed that T_w is equidissectable with a cube.

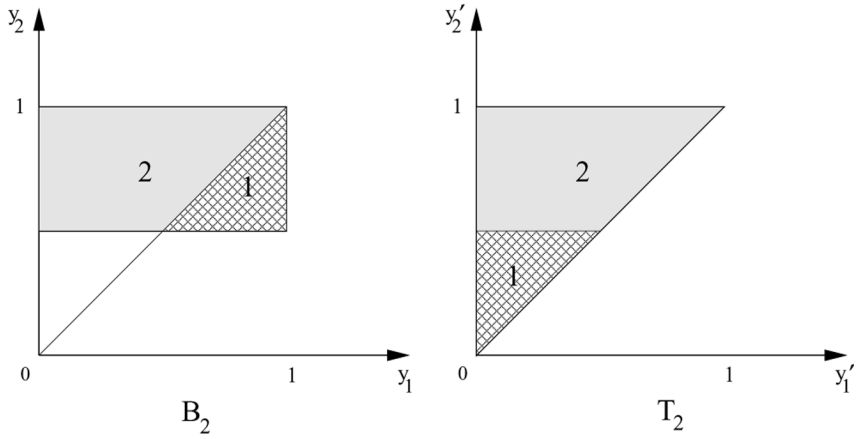

Fig. 2. Two ways to dissect rectangle B_2 to give triangle T_2. Piece 1 may be rotated to its new position, or reflected and translated.

consider two such dissections, both two-piece dissections based on Fig. 2.

In the first dissection, the triangular piece marked 1 in Fig. 2 is rotated clockwise about the center of the square until it reaches the position shown on the right in Fig. 2. In the second dissection, the piece marked 1 is first reflected about the main diagonal of the square and then slid down until it reaches the position shown on the right in Fig. 2. In both dissections the piece marked 2 is fixed.

The two dissections can be specified in terms of coordinates[2] as follows. For the first dissection, we set

$$\begin{cases} (x_1', x_2') := (x_1, x_2), & \text{if } x_1 < x_2 \\ (x_1', x_2') := (1 - x_1, 1 - x_2), & \text{if } x_1 \geq x_2 \end{cases} \tag{3}$$

and for the second, we set

$$\begin{cases} (x_1', x_2') := (x_1, x_2), & \text{if } x_1 < x_2 \\ (x_1', x_2') := \left(x_2 - \frac{1}{2}, x_1 - \frac{1}{2}\right), & \text{if } x_1 \geq x_2. \end{cases} \tag{4}$$

The first dissection involves only a rotation, but seems harder to generalize to higher dimensions. The second one is the one we will generalize; it uses a reflection, but as mentioned at the end of Section II, this is permitted by the definition of a dissection.

We next illustrate how these dissections can be converted into encoding algorithms for constant-weight (weight 2) binary codes. Again there may be several solutions, and the best algorithm may depend on arithmetic properties of n (such as its parity). We work now with the unnormalized sets $\mathcal{R}_2$ and $\mathcal{C}_2$. In each case, the output is a weight-2 binary vector with 1's in positions y_1' and y_2'.

A. First Dissection, Algorithm 1

1) The input is an information vector $(y_1, y_2) \in \mathcal{R}_2$ with $1 \leq y_1 \leq n-1$ and $\lceil n/2 \rceil + 1 \leq y_2 \leq n$.
2) If $y_1 < y_2$, we set $y_1' = y_1$, $y_2' = y_2$, otherwise we set $y_1' = n - y_1$ and $y_2' = n - y_2 + 1$.

For n even, this algorithm generates all possible $n(n-1)/2$ codewords. For n odd it generates only $(n-1)^2/2$ codewords, leading to a slight inefficiency, and the following algorithm is to be preferred.

[2]For our use of a mixture of $\leq$ and $<$ signs, see the remark at the end of Section II.

B. First Dissection, Algorithm 2

1) The input is an information vector $(y_1, y_2) \in \mathcal{R}_2$ with $1 \leq y_1 \leq n$, $\lceil (n+1)/2 \rceil + 1 \leq y_2 \leq n$.
2) If $y_1 < y_2$, we set $y_1' = y_1$, $y_2' = y_2$, otherwise we set $y_1' = n - y_1 + 1$, $y_2' = n - y_2 + 2$.

For n odd, this algorithm generates all $n(n-1)/2$ codewords, but for n even it generates only $n(n-1)/2$ codewords, again leading to a slight inefficiency.

C. Second Dissection

1) The input is an information vector $(y_1, y_2) \in \mathcal{R}_2$ with $1 \leq y_1 \leq n-1$ and $\lceil n/2 \rceil + 1 \leq y_2 \leq n$.
2) If $y_1 < y_2$, we set $y_1' = y_1$, $y_2' = y_2$, otherwise we set $y_1' = y_2 - \lceil n/2 \rceil$, $y_2' = y_1 - \lceil n/2 \rceil + 1$.

For n even, this algorithm generates all $n(n-1)/2$ codewords, but for n odd it generates only $(n-1)^2/2$ codewords, leading to a slight inefficiency. There is a similar algorithm, not given here, which is better when n is odd.

Note that only one test is required in any of the encoding algorithms. The mappings are invertible, with obvious decoding algorithms corresponding to the inverse mappings from $\mathcal{C}_2$ to $\mathcal{R}_2$

We now extend this method to weight $w = 3$. Fortunately, the Dehn invariants for both the brick B_3 and our unit orthoscheme T_3, which is the tetrahedron[3] with vertices $(0,0,0), (0,0,1), (0,1,1)$, and $(1,1,1)$, are zero (since in both cases all dihedral angles are rational multiples of π), and so by the Dehn–Sydler theorem, the polyhedra B_3 and T_3 *are* equidecomposable. As already mentioned in Section I, the Dehn–Sydler theorem applies only in three dimensions. But it will follow from the algorithm given in the next section that B_w and T_w are equidecomposable in all dimensions.

We will continue to describe the encoding step (the map from B_w to T_w) first. We will give an inductive dissection (see Fig. 3), transforming B_3 to T_3 in two steps, effectively reducing the dimension by one at each step. In the first step, the brick B_3 is dissected into a triangular prism (the product of a right triangle, T_2, and an interval), and in the second step, this triangular prism

[3]To solve Hilbert's third problem, Dehn showed that this tetrahedron is not equidecomposable with a regular tetrahedron of the same volume.

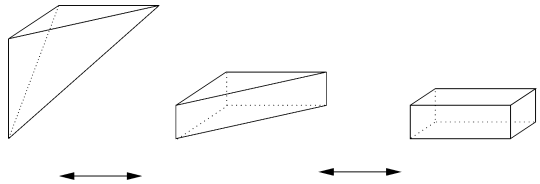

Fig. 3. Transformation from tetrahedron to rectangular prism.

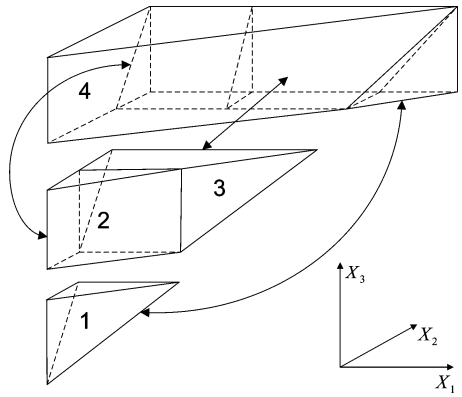

Fig. 4. Four-piece dissection of tetrahedron to triangular prism. Pieces 2 and 3 must be reflected, piece 4 is fixed.

is dissected into the tetrahedron T_3. Note that the first step has essentially been solved by the dissection given in (4).

For the second step, we use a four-piece dissection of the triangular prism to the tetrahedron T_3. This dissection, shown with the tetrahedron and prism superposed in Fig. 4 appears to be new.

There is a well-known dissection of the same pair of polyhedra that was first published by Hill in 1896 [10]. This also uses four pieces, and is discussed in several references: see Boltianskii [1, p. 99], Cromwell [5, p. 47], Frederickson [7, Fig. 20.4], Sydler [23], Wells [29, p. 251]. However, Hill's dissection seems harder to generalize to higher dimensions. Sydler's dissection does have the advantage over ours that it can be accomplished purely by translations and rotations, whereas in our dissection, two of the pieces (pieces labeled 2 and 3 in Fig. 4) must also be reflected. However, as mentioned at the end of Section II, this is permitted by the definition of a dissection, and is not a drawback for our application. Apart from this, our dissection is simpler than Hill's, in the sense that his dissection requires a cut along a skew plane ($x_1 - x_3 = \frac{1}{3}$), whereas all our cuts are parallel to coordinate axes.

To obtain the four pieces shown in Fig. 4, we first make two horizontal cuts along the planes $x_3 = \frac{1}{3}$ and $x_3 = \frac{2}{3}$, dividing the tetrahedron into three slices. We then cut the middle slice into two by a vertical cut along the plane $x_2 = \frac{1}{2}$.

There appears to be a tradition in geometry books that discuss dissections of not giving coordinates for the pieces. To an engineer this seems unsatisfactory, and so in Table I we list the vertices of the four pieces in our dissection. Piece 1 has four vertices, while the other three pieces each have six vertices. (In the Hill dissection, the numbers of vertices of the four pieces are 4, 5, 6, and 6, respectively.) Given these coordinates, it is not difficult to verify that the four pieces can be reassembled to form

TABLE I
COORDINATES OF VERTICES OF PIECES IN DISSECTION OF TETRAHEDRON SHOWN IN FIG. 4

Piece	Coordinates
1	$[0,0,0]$, $[0,0,1/3]$, $[0,1/3,1/3]$, $[1/3,1/3,1/3]$.
2	$[0,0,1/3]$, $[0,1/3,1/3]$, $[1/3,1/3,1/3]$, $[0,0,2/3]$, $[0,1/3,2/3]$, $[1/3,1/3,2/3]$.
3	$[0,1/3,1/3]$, $[1/3,1/3,1/3]$, $[0,1/3,2/3]$, $[0,2/3,2/3]$, $[2/3,2/3,2/3]$, $[1/3,1/3,2/3]$.
4	$[0,0,2/3]$, $[0,2/3,2/3]$, $[2/3,2/3,2/3]$, $[0,0,1]$, $[0,1,1]$, $[1,1,1]$.

the triangular prism, as indicated in Fig. 4. As already remarked, pieces 2 and 3 must be reflected (or "turned over" in a fourth dimension). The correctness of the dissection also follows from the alternative description of this dissection given below.

The dissection shown in Fig. 4 can be described algebraically as follows. We describe it in the more logical direction, going from the triangular prism to the tetrahedron since this is what we will generalize to higher dimensions in the next section. The input is a vector (x_1, x_2, x_3) with $0 \le x_1 \le x_2 < 1$, $\frac{2}{3} \le x_3 < 1$; the output is a vector (x'_1, x'_2, x'_3) with $0 \le x'_1 \le x'_2 \le x'_3 < 1$, given by

$$(x'_1, x'_2, x'_3) = \begin{cases} (x_1, x_2, x_3), & \text{if } x_1 \le x_2 < x_3 \\ \left(x_1 - \frac{1}{3}, x_3 - \frac{1}{3}, x_2 - \frac{1}{3}\right), & \text{if } \frac{1}{3} \le x_1 < x_3 \le x_2 \\ \left(x_3 - \frac{2}{3}, x_2 - \frac{2}{3}, x_1 + \frac{1}{3}\right). & \text{if } x_1 \le \frac{1}{3} < x_3 \le x_2 \\ \left(x_3 - \frac{2}{3}, x_1 - \frac{2}{3}, x_2 - \frac{2}{3}\right), & \text{if } x_3 \le x_1 \le x_2. \end{cases} \quad (5)$$

The four cases in (5), after being transformed, correspond to the pieces labeled 4, 3, 2, 1, respectively, in Fig. 4. We see from (5) that in the second and third cases the linear transformation has determinant -1, indicating that these two pieces must be reflected.

Since it is hard to visualize dissections in dimensions greater than three, we give a schematic representation of the above dissection that avoids drawing polyhedra. Fig. 5 shows a representation of the transformation from the triangular prism to the tetrahedron T_3, equivalent to that given in (5). The steps shown in Fig. 5 may be referred to as "cut and paste" operations, because, as Fig. 5 shows, the vector in the triangular prism is literally cut up into pieces which are rearranged and relabeled. Note that, to complete the transformation, we precede this operation by the dissection given in (4), finally establishing the bijection between B_3 and T_3.

We now describe the mapping shown in Fig. 5 in more detail. The triangular prism is represented by the set of partially ordered triples (x_1, x_2, x_3) with $0 \le x_1 \le x_2 < 1$ and $\frac{2}{3} \le x_3 < 1$, and we wish to transform this into the tetrahedron consisting of the points (x'_1, x'_2, x'_3) with $0 \le x'_1 \le x'_2 \le x'_3 < 1$.

We divide the interval $[0,1)$ into $w = 3$ equal segments of length $1/w = 1/3$, and consider where the points x_1, x_2, and x_3 fall in this interval, given that (x_1, x_2, x_3) is in the triangular prism. There are three possibilities for where x_3 lies in relation to $0 \le x_1 \le x_2 < 1$, and we further divide the case $x_1 \le x_3 < x_2$ into two subcases depending on whether $x_1 \ge \frac{1}{3}$ or $x_1 < \frac{1}{3}$. These are the four cases shown in Fig. 5, and correspond one-to-one with the four dissection pieces in Fig. 4. Fig. 5 shows

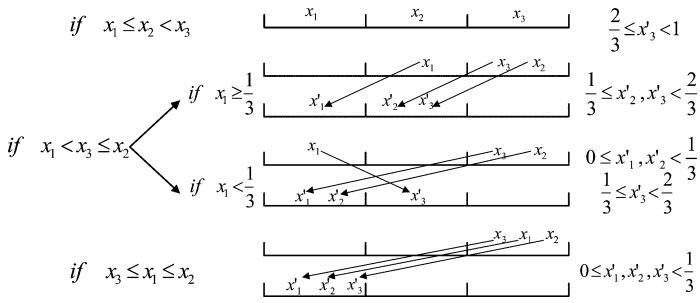

Fig. 5. Cut-and-paste description of the inverse transformation from triangular prism to tetrahedron.

how the triples x_1, x_2, x_3 (reindexed according to their relative positions) are mapped to the triples x'_1, x'_2, x'_3.

The last column of Fig. 5 shows the ranges of the x'_i in the four cases; the fact that these ranges are disjoint guarantees that the mapping from x_1, x_2, x_3 to x'_1, x'_2, x'_3 is invertible. The ranges of the x'_i will be discussed in more detail in the following section after the general algorithms are presented.

This operation can now be described without explicitly mentioning the underlying dissection. Each interval of length $1/w$, together with the given x_i values within it, is treated as a single complete unit. In the "cut and paste" operations, these units are rearranged and relabeled in such a way that the operation is invertible.

IV. Algorithms and Proof of Correctness

In the previous section, we provided an encoding and decoding algorithm for weights $w = 2$ and $w = 3$, based on our geometric interpretation of $\mathcal{C}_2$ and $\mathcal{C}_3$ as points in $\mathbb{R}^w$. In this section, the algorithm is generalized to larger values of the weight w. We start with the geometry, and give a dissection of the "brick" B_w into the orthoscheme T_w. We work with the normalized coordinates $x_i = y_i/n$ (for a point in B_w) and $x'_i = y'_i/n$ (for a point in T_w), where $1 \le i \le w$. Later in this section, we discuss the modifications needed to take into account the fact that the y'_i must be integers.

Before we proceed, we make a few remarks. i) The algorithm is described assuming the source word is represented as a point in the "brick." In practice, one may need to convert a given source word or index (possibly in its binary representation) to such a representation; this is a straightforward process—we shall return to this at the end of this section. ii) When calculating the complexity, we assume the computation is performed on operands of length $O(\log n)$. iii) In the complexity analysis, we do not include the cost of setting w positions to 1 during encoding, and extracting the positions of the 1's in the length n constant weight codewords during decoding, for two reasons: a) this is part of the transmission/reception process (or the write/read process), and does not contribute to the overall computation load, and b) during encoding, setting certain positions to 1 can be done efficiently on a codeword preset to all zeros.

A. An Inductive Decomposition of the Orthoscheme

Restating the problem, we wish to find a bijection F_w between the sets B_w and T_w. The inductive approach developed for $w = 3$ (where the $w = 2$ case was a subproblem) will be generalized. Of course the bijection F_1 between B_1 and T_1 is trivial. We assume that a bijection F_{w-1} is known between B_{w-1} and T_{w-1}, and show how to construct a bijection F_w between B_w and T_w.

The last step in the induction uses a map f_w from the prism $T_{w-1} \times [1 - \frac{1}{w}, 1)$ to T_w (f_2 is the map described in (4) and f_3 is described in (5)). The mapping F_w from B_w to T_w is then given recursively by $F_w : (x_1, x_2, \ldots, x_w) \mapsto (x'_1, x'_2, \ldots, x'_w)$, where

$$(x'_1, x'_2, \ldots, x'_w) := f_w(F_{w-1}(x_1, x_2, \ldots, x_{w-1}), x_w). \quad (6)$$

For $w = 1$ we set

$$F_1 := f_1 : B_1 \to T_1, (x_1) \mapsto (x'_1) = (x_1).$$

By iterating (6), we see that F_w is obtained by successively applying the maps $f_1, f_2, \ldots, f_w$.

The following algorithm defines f_w for $w \ge 2$. We begin with an algebraic definition of the mapping and its inverse, and then discuss it further in the following section. The input to the mapping f_w is a vector $\boldsymbol{x} := (x_1, x_2, \ldots, x_w)$, with $(x_1, x_2, \ldots, x_{w-1}) \in T_{w-1}$ and $x_w \in [1 - 1/w, 1)$; the output is a vector $\boldsymbol{x}' := (x'_1, x'_2, \ldots, x'_w) \in T_w$.

Forward mapping $f_w (w \ge 2)$:

1) Let

$$i_0 := \min\{i \in \{1, \ldots, w\} \,|\, x_w \le x_i\}.$$

2) Let

$$j_0 := \min\{i \in \{1, \ldots, i_0\} \,|\, w - i_0 + i - 1 \le w x_i\} - 1.$$

3) Set x'_k equal to

$$\begin{cases} x_{k+j_0} - \frac{w+j_0-i_0}{w}, & \text{for } k = 1, \ldots, i_0 - j_0 - 1 \\ x_w - \frac{w+j_0-i_0}{w}, & \text{for } k = i_0 - j_0 \\ x_{k+j_0-1} - \frac{w+j_0-i_0}{w}, & \text{for } k = i_0 - j_0 + 1, \ldots, w - j_0 \\ x_{k-w+j_0} + \frac{i_0-j_0}{w}, & \text{for } k = w - j_0 + 1, \ldots, w. \end{cases} \quad (7)$$

Equation (7) identifies the "cut and paste" operations required to obtain x'_k for different ranges of the variable k. If the initial index in one of the four cases in (7) is smaller than the final index, that case is to be skipped. A case is also skipped if the subscript for an x_i is not in the range $1, \ldots, w$. Note in Step 1 that $i_0 = w$ if x_w is the largest of the x_i's. This implies that $j_0 = 0$, and then Step 3 is the identity map.

The inverse mapping G_w from T_w to B_w has a similar recursive definition. The wth step in the induction is the map

$g_w : T_w \to T_{w-1} \times [1 - \frac{1}{w}, 1)$ defined below. For $w = 1$ we set

$$G_1 := g_1 : T_1 \to B_1, (x'_1) \mapsto (x_1) = (x'_1).$$

The map G_w is obtained by successively applying the maps $g_w, g_{w-1}, \ldots, g_1$.

Inverse mapping $g_w (w \geq 2)$:

1) Let

$$m_0 := \max\{i \in \{1, \ldots, w\} \,|\, i - 1 \leq w x'_i\}.$$

2) If $m_0 = w$, let $j_0 := 0$, otherwise, let

$$j_0 := w - \max\{i \in \{m_0 + 1, \ldots, w\} \,|\, w x'_i \leq m_0\};$$

in either case, let $i_0 := j_0 + m_0$.

3) Set x_k equal to

$$\begin{cases} x'_{k+w-j_0} - \frac{i_0 - j_0}{w}, & \text{for } k = 1, \ldots, j_0 \\ x'_{k-j_0} + \frac{w + j_0 - i_0}{w}, & \text{for } k = j_0 + 1, \ldots, i_0 - 1 \\ x'_{k-j_0+1} + \frac{w + j_0 - i_0}{w}, & \text{for } k = i_0, \ldots, w - 1 \\ x'_{i_0-j_0} + \frac{w + j_0 - i_0}{w}, & \text{for } k = w. \end{cases} \tag{8}$$

Note that the transformations in (7) and (8) are formal inverses of each other, and that these transformations are volume-preserving. The underlying linear transformations are orthogonal transformations with determinant $+1$ or -1.

Before proceeding further, let us verify that in the case $w = 3$, the mapping $f_w = f_3$ agrees with that given in (5).

- If $x_1 \leq x_2 < x_3$, then $i_0 = 3$, $j_0 = 0$, and the map is the identity, as mentioned above.
- If $x_1 < x_3 \leq x_2$, there are two subcases:
  - If $\frac{1}{3} \leq x_1$ then $i_0 = 2$, $j_0 = 0$.
  - If $x_1 < \frac{1}{3}$ then $i_0 = 2$, $j_0 = 1$.
- If $x_3 \leq x_1 \leq x_2$, then $i_0 = 1$, $j_0 = 0$.

The transformations in (7) now exactly match those in (5).

B. Interpretations and Explanations

In Fig. 6, we give a graphical interpretation of the algorithm, which can be regarded as a generalization of the "cut and paste" description given above. This figure shows the transformation defined by the wth step f_w in the algorithm. At this step, we begin with a list of $w - 1$ numbers $(x_1, x_2, \ldots, x_{w-1})$ in increasing order, and a further number x_w which may be anywhere in the interval $[1 - 1/w, 1)$. This list of w numbers is plotted in the plane as the set of w points (i, wx_i) for $i = 1, 2, \ldots, w$ (indicated by the solid black circles in Fig. 6). In the first step in the forward algorithm, the augmented list $(x_1, x_2, \ldots, x_w)$ is sorted into increasing order. In the sorted list, x_w now occupies position i_0, so the point (w, wx_w) moves to the left, to the new position (i_0, wx_w), and the points (i, wx_i) for $i = i_0 + 1, \ldots, w - 1$ move to the right. This is indicated by the arrows in the figure. The new positions of these points are marked by hollow circles.

The point (i_0, wx_w) now lies between the grid points (i_0, w) and $(i_0, w - 1)$ (it may coincide with the latter point), since $x_w \geq 1 - \frac{1}{w}$. We draw the line $y = x + w - i_0 - 1$ (shown as the dashed-and-dotted line in Fig. 6). This has unit slope and passes through the points $(i_0, w - 1)$ and $(0, w - i_0 - 1)$. The algorithm then computes $j_0 + 1$ to be the smallest index i for which x_i is on or above this line. Once i_0 and j_0 have been determined, the forward mapping proceeds as follows. The points (i, wx_i) for $i = 1, \ldots, j_0$ are shifted to the right of the figure and are moved upwards by the amount $(i_0 - j_0)/w$, their new positions being indicated by crosses in the figure. Finally, the origin is moved to the grid point $(j_0, w - i_0 + j_0)$ and the points are reindexed. The $m_0 := i_0 - j_0$ points which originally had indices $j_0 + 1, \ldots, i_0$ become points $1, \ldots, m_0$ after reindexing. In the new coordinates, the final positions of the points lie inside the square region $[1, w) \times [1, w)$. The reader can check that this process is exactly equivalent to the algebraic description of f_w given above.

To recover i_0 and j_0, we first determine the value of $m_0 := i_0 - j_0$. This can indeed be done since m_0 is precisely the index of the largest wx'_i that lies on or above the line $y = x - 1$ in the new coordinate system. Note that the position of this line is independent of i_0 and j_0 and $(x'_1, x'_2, \ldots, x'_w)$. This works because the points $wx_1, \ldots, wx_{j_0}$ in the original coordinate system, before the origin is shifted, are moved right by w units and upwards by w units, so points below the dashed-and-dotted line remain below the line. Furthermore, observe that in the new coordinate system the number of points (i, wx'_i) below the line $y = m_0$ is equal to $w - j_0$. Thus, the correct i_0 and j_0 values may be recovered, and the inverse mapping can be successfully performed.

The following remarks record two properties of the algorithm that will be used later.

Remark 1: Step 2 of the forward algorithm implies that $x_{j_0} < \frac{w - i_0 + j_0 - 1}{w}$ and $x_{j_0+1} \geq \frac{w - i_0 + j_0}{w}$. It follows that there is no i in the range $1 \leq i \leq w$ for which

$$\frac{w - i_0 + j_0 - 1}{w} \leq x_i < \frac{w - i_0 + j_0}{w}.$$

Remark 2: The forward algorithm produces a vector $\boldsymbol{x'}$ whose components satisfy

$$0 \leq x'_1 \leq \cdots \leq x'_{i_0 - j_0} \leq \cdots \leq x'_{w - j_0} < \frac{i_0 - j_0}{w} \tag{9}$$

$$\frac{i_0 - j_0}{w} \leq x'_{w-j_0+1} \leq x'_{w-j_0+2} \leq \cdots \leq x'_w < 1 \tag{10}$$

and

$$x'_k < \frac{k - 1}{w}, \quad \text{for } w - j_0 + 1 \leq k \leq w. \tag{11}$$

Equations (9) and (10) follow from the minimizations in Steps 1 and 2 of the forward algorithm, respectively. The right-hand side of (11) expresses the fact, already mentioned, that the first j_0 points remain below the dotted-and-dashed line after they are shifted.

C. Proof of Correctness

We now give the formal proof that the algorithm is correct. This is simply a matter of collecting together facts that we have already observed.

Theorem 1: For any $w \geq 1$, the forward mapping f_w is a one-to-one mapping from $T_{w-1} \times [1 - \frac{1}{w}, 1)$ to T_w with inverse g_w.

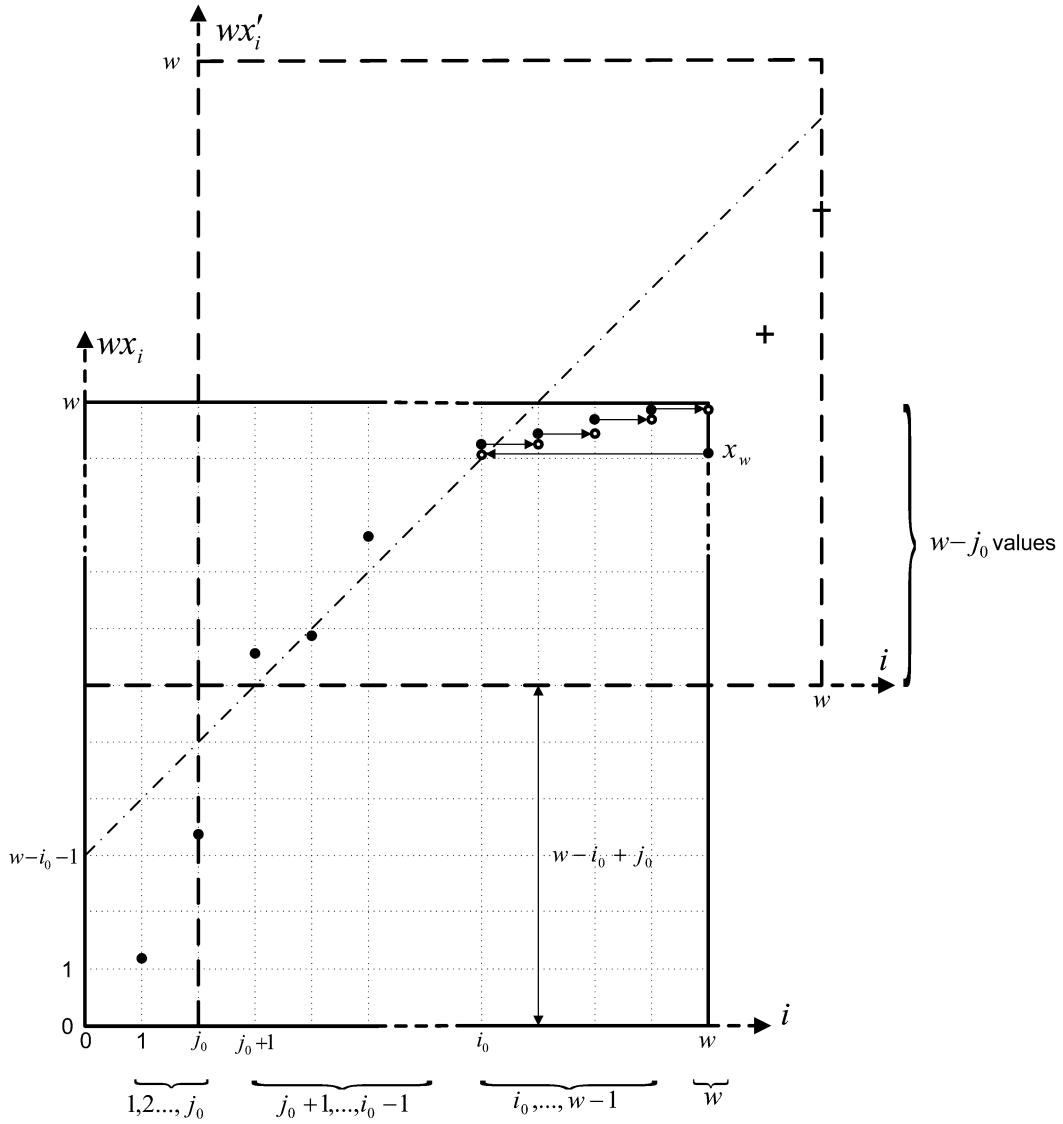

Fig. 6. A graphical illustration of the forward and inverse mapping.

Proof: First, it follows from Remark 2 that, for $\boldsymbol{x} \in T_{w-1} \times [1-\frac{1}{w}, 1)$, $\boldsymbol{x}' = (x'_1, x'_2, \ldots, x'_w)$ satisfies $0 \le x'_1 \le x'_2 \le \cdots \le x'_w < 1$, and so is an element of T_w.

Suppose there were two different choices for $\boldsymbol{x}$, say $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$, such that

$$f_w(\boldsymbol{x}^{(1)}) = f_w(\boldsymbol{x}^{(2)}) = \boldsymbol{x}'.$$

We know that $\boldsymbol{x}'$ determines m_0, j_0 and i_0. So $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ have the same associated values of i_0 and j_0. But for a given pair (i_0, j_0), (7) is invertible. Hence, $\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(2)}$, and f_w is one-to-one.

Note that the transformations in (7) and (8) are inverses of each other. Hence, f_w is also an onto map, and g_w is its inverse. □

D. Number of Pieces

The map f_w, which dissects the prism $T_{w-1} \times [1-\frac{1}{w}, 1)$ to give the orthoscheme T_w, has one piece for each pair (i_0, j_0). If $i_0 = w$ then $j_0 = 0$, while if $1 \le i_0 \le w-1$, j_0 takes all values from 0 to $i_0 - 1$. (It is easy to write down an explicit point in the interior of the piece corresponding to a specified pair of values of i_0 and j_0. Assume $i_0 < w$ and set $\delta = 1/w^3$. Take the point with coordinates $(x_1, \ldots, x_w)$ given by $x_w = (w-1)/w + \delta$; $x_i = x_w + \delta(i - i_0)$ for $i = i_0 + 1, \ldots, w-1$; $x_i = (i + w - i_0 - 1 - \delta)/w$ for $i = 1, \ldots, j_0$; $x_i = (i + w - i_0 - 1 + \delta)/w$ for $i = j_0 + 1, \ldots, i_0 - 1$.) The total number of pieces in the dissection is therefore

$$1 + 1 + 2 + 3 + \cdots + (w-1) = \frac{w^2 - w + 2}{2}$$

which is $1, 2, 4, 7, 11, \ldots$ for $w = 1, 2, 3, 4, 5, \ldots$. This is a well-known sequence, entry A124 in [21], which by coincidence also arises in a different dissection problem: it is the maximal number of pieces into which a circular disk can be cut with $w-1$ straight cuts. For example, with three cuts, a pizza can be cut into a maximum of seven pieces, and this is also the number of pieces in the dissection defined by f_4.

E. The Algorithms for Positive Integers

To apply the above algorithm to the problem of encoding and decoding constant-weight codes, we must work with positive

integers rather than real numbers, which entails a certain loss in rate, although the algorithms remain largely unchanged. Let $\mathbb{N} := \{1, 2, 3, \ldots\}$, and let n and w be given with $2w < n$. In a manner analogous to the real-valued case, we find a bijection between a finite hyper-rectangle or brick $B_w^{\mathbb{N}} \subset \mathbb{N}^w$ and a subset of the finite orthoscheme $T_w^{\mathbb{N}} \subset \mathbb{N}^w$, where $B_w^{\mathbb{N}}$ is the set of vectors $(y_1, y_2, \ldots, y_w) \in \mathbb{N}^w$ satisfying

$$n - (w - i) - \left\lfloor \frac{n - (w - i)}{i} \right\rfloor + 1 \le y_i \le n - (w - i)$$

for $i = 1, 2, \ldots, w$, and $T_w^{\mathbb{N}}$ is the set of vectors $(y_1, y_2, \ldots, y_w) \in \mathbb{N}^w$ satisfying

$$1 \le y_1 < y_2 < \cdots < y_w \le n.$$

Note that usually $|B_w^{\mathbb{N}}| < |T_w^{\mathbb{N}}|$, which entails a loss in rate.

The forward mapping f_w is now replaced by the map $f_w^{\mathbb{N}}$, which sends $(y_1, y_2, \ldots, y_w)$ with $(y_1, y_2, \ldots, y_{w-1}) \in T_{w-1}^{\mathbb{N}}$ and $n - \lfloor \frac{n}{w} \rfloor + 1 \le y_w \le n$ to an element of $T_w^{\mathbb{N}}$. Let us write $n = pw + q$, where $p \ge 0$ and $0 \le q \le w - 1$. We partition the range $1, 2, \ldots, n$ into w parts, where the first $n - w - 1$ parts each have p elements, the next q parts each have $p+1$ elements, and the last part has p elements (giving a total of n elements). This is similar to the real-valued case, where each interval had length $1/w$.

1) Let

$$i_0 := \min\{i \in \{1, \ldots, w\} \,|\, y_w \le y_i\}.$$

2) Let

$$j_0 := \min\{i \in \{1, \ldots, i_0\} \,|\, V_i < y_i\} - 1$$

where $V_i := (w - i_0 + i - 1)p + \max\{q - i_0 + i, 0\}$.

3) Set y'_k equal to

$$\begin{cases} y_{k+j_0} - V_{j_0+1}, & \text{for } k = 1, \ldots, i_0 - j_0 - 1 \\ y_w - V_{j_0+1}, & \text{for } k = i_0 - j_0 \\ y_{k+j_0-1} + 1 - V_{j_0+1}, & \text{for } k = i_0 - j_0 + 1, \ldots, w - j_0 \\ y_{k-w+j_0} + n - V_{j_0+1}, & \text{for } k = w - j_0 + 1, \ldots, w. \end{cases}$$

The inverse mapping g_w is similarly replaced by the map $g_w^{\mathbb{N}} : T_w^{\mathbb{N}} \to \{(y_1, y_2, \ldots, y_w) : (y_1, y_2, \ldots, y_{w-1}) \in T_{w-1}^{\mathbb{N}}, n - \lfloor \frac{n}{w} \rfloor + 1 \le y_w \le n\}$, defined as follows. Again, assume $n = pw + q$.

1) Let

$$m_0 := \max\{i \in \{1, \ldots, w\} \,|\, W_i < y'_i\}$$

where $W_i := q + (i - 1)p + \min\{i - q - 1, 0\}$.

2) If $m_0 = w$, let $j_0 := 0$, otherwise, let

$$j_0 := w - \max\{i \in \{m_0 + 1, \ldots, w\} \,|\, y'_i \le W_{m_0} + p\};$$

in either case, let $i_0 := j_0 + m_0$.

3) Set y_k equal to

$$\begin{cases} y'_{k+w-j_0} - p - W_{m_0}, & \text{for } k = 1, \ldots, j_0 \\ y'_{k-j_0} + n - p - W_{m_0}, & \text{for } k = j_0 + 1, \ldots, i_0 - 1 \\ y'_{k-j_0+1} - 1 + n - p - W_{m_0}, & \text{for } k = i_0, \ldots, w - 1 \\ y'_{i_0-j_0} + n - p - W_{m_0}, & \text{for } k = w. \end{cases}$$

We omit the proofs, since they are similar to those for the real-valued case.

F. Comments on the Algorithm

The overall complexity of the transform algorithm is $O(w^2)$, because at each induction step, the complexity is linear in the weight at that step. Recall that the complexities of the arithmetic coding method and Knuth's complementation method are both $O(n)$. Thus, when the weight w is larger than $\sqrt{n}$, the geometric approach is less competitive. When the weight is low, the proposed geometric technique is more efficient, because Knuth's complementation method is not applicable, while the dissection operations of the proposed algorithm makes it faster than the arithmetic coding method. Furthermore, due to the structure of the algorithm, it is possible to parallelize part of the computation within each induction step to further reduce the computation time.

So far little has been said about mapping a binary sequence to an integer sequence $y_1, y_2, \ldots, y_w$ such that $y_i \in [L_i, U_i]$, where L_i and U_i are the lower and upper bound of the valid range as specified by the algorithm. A straightforward method is to treat the binary sequence as an integer number and then use "quotient and remainder" method to find such a mapping. However, this requires a division operation, and when the binary sequence is long, the computation is not very efficient. A simplification is to partition the binary sequence into short sequences, and map each short binary sequence to a pair of integers, as in the case of a weight two constant-weight codes. Through proper pairing of the ranges, the loss in the rate can be minimized.

The overall rate loss has two components, the first from the rounding involved in using natural numbers, the second from the loss in the above simplified translation step. However, when the weight is on the order of $\sqrt{n}$, and n is in the range of 100–1000, the rate loss is usually 1–3 bits/block. For example, when $n = 529$, $w = 23$, then the rate loss is 2 bits/block compared to the best possible code which would encode $k_0 = 132$ information bits.

V. Conclusion

We propose a novel algorithm for encoding and decoding constant-weight binary codes, based on dissecting the polytope defined by the set of all binary words of length n and weight w, and reassembling the pieces to form a hyper-rectangle corresponding to the input data. The algorithm has a natural recursive structure, which enables us to give an inductive proof of its correctness. The proposed algorithm has complexity $O(w^2)$, independent of the length of the codewords n. It is especially suitable for constant weight codes of low weight.

References

[1] V. G. Boltianskii, *Hilbert's Third Problem*. New York: Wiley, 1978, Translated from the Russian by R. A. Silverman.

[2] T. M. Cover, "Enumerative source encoding," *IEEE Trans. Inf. Theory*, vol. IT-19, no. 1, pp. 73–77, Jan. 1973.

[3] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2rd ed. New York: Wiley-Interscience, 2006.

[4] H. S. M. Coxeter, *Regular Polytopes*, 3rd ed. New York: Dover, 1973.

[5] P. R. Cromwell, *Polyhedra*. Cambridge, U.K.: Cambridge Univ. Press, 1997.
[6] M. Dehn, "Über den Rauminhalt," *Nachr. Gesell. Wiss. Göttingen, Math.-Phys. Kl.*, pp. 345–354, 1900, Also, *Math. Ann.,* vol. 55, pp. 465–478, 1902.
[7] G. N. Frederickson, *Dissections: Plane and Fancy*. Cambridge, U.K.: Cambridge Univ. Press, 1997.
[8] B. Grünbaum, *Convex Polytopes*, 2nd ed. New York: Springer-Verlag, 2003.
[9] H. Hadwiger, "Hillsche hypertetraeder," *Gazeta Matemática (Lisboa)*, vol. 12, no. 50, pp. 47–48, 1951.
[10] M. J. M. Hill, "Determination of the volumes of certain species of tetrahedra without employment of the method of limits," *Proc. London Math. Soc.*, vol. 27, pp. 39–53, 1896.
[11] B. Jessen, "The algebra of polyhedra and the Dehn-Sydler theorem," *Math. Scand.*, vol. 22, pp. 241–256, 1968.
[12] D. E. Knuth, "Efficient balanced codes," *IEEE Trans. Inf. Theory*, vol. IT-32, no. 1, pp. 51–53, Jan. 1986.
[13] H. Lindgren, *Geometric Dissections*. New York: Dover, 1972.
[14] A. Nijenhuis and H. S. Wilf, *Combinatorial Algorithms*, 2nd ed. New York: Academic, 1978.
[15] D. K. Pradhan and J. J. Stiffler, "Error correcting codes and self-checking circuits in fault-tolerant computers," *IEEE Comp. Mag.*, vol. 13, pp. 27–37, Mar. 1980.
[16] T. V. Ramabadran, "A coding scheme for m-out-of-n codes," *IEEE Trans. Commun.*, vol. 38, no. 8, pp. 1156–1163, Aug. 1990.
[17] J. Rissanen, "Arithmetic codings as number representations," *Topics in Systems Theory. Acta Polytech. Scand. Math. Comput. Sci.*, vol. 31, pp. 44–51, 1979.
[18] J. Rissanen and G. G. Langdon, Jr., "Arithmetic coding," *IBM J. Res. Devel.*, vol. 23, no. 2, pp. 149–162, 1979.
[19] C.-H. Sah, *Hilbert's Third Problem: Scissors Congruence*. London, U.K.: Pitman, 1979.
[20] J. P. M. Schalkwijk, "An algorithm for source coding," *IEEE Trans. Inf. Theory*, vol. IT-18, no. 3, pp. 395–399, May 1972.
[21] N. J. A. Sloane, The On-Line Encyclopedia of Integer Sequences [Online]. Available: www.research.att.com/~njas/sequences/, 1996-2009
[22] N. J. A. Sloane and V. A. Vaishampayan, "Generalizations of Schöbi's tetrahedral dissection," *Discr. Comput. Geom.*, to be published.
[23] J.-P. Sydler, "Sur les tétraèdres équivalents à un cube," *Elem. Math.*, vol. 11, pp. 78–81, 1956.
[24] J.-P. Sydler, "Conditions nécessaires et suffisantes pour l'équivalence des polyèdres de l'espace euclidien à trois dimensions," *Commun. Math. Helv.*, vol. 40, pp. 43–80, 1965.
[25] D. T. Tang and L. S. Woo, "Exhaustive test pattern generation with constant weight vectors," *IEEE Trans. Comput.*, vol. C-32, no. 12, pp. 1145–1150, Dec. 1983.
[26] V. A. Vaishampayan and M. D. Feuer, "An overlay architecture for managing lightpaths in optically routed networks," *IEEE Trans. Commun.*, vol. 53, no. 10, pp. 1729–1737, Oct. 2005.
[27] V. A. Vaishampayan and N. J. A. Sloane, "Dissections and constant weight codes," in *Proc. IEEE Information Theory Workshop*, Chengdu, China, Oct. 2006, pp. 16–20.
[28] S. Verdú and V. K. Wei, "Explicit construction of optimal constant-weight codes for identification via channels," *IEEE Trans. Inf. Theory*, vol. 39, no. 1, pp. 30–36, Jan. 1993.
[29] D. Wells, *The Penguin Dictionary of Curious and Interesting Geometry*. London, U.K.: Penguin Books, 1991.
[30] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Commun. ACM*, vol. 30, pp. 520–540, Jun. 1987.
[31] J.-H. Youn and B. Bose, "Efficient encoding and decoding schemes for balanced codes," *IEEE Trans. Comp.*, vol. 52, no. 9, pp. 1229–1232, Sep. 2003.
[32] G. M. Ziegler, *Lectures on Polytopes*. New York: Springer-Verlag, 1995.

Chao Tian (S'00–M'05) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2000 and the M.S. and Ph.D. degrees in electrical and computer engineering from Cornell University, Ithaca, NY, in 2003 and 2005, respectively.

He was a Postdoctoral researcher at Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, from 2005 to 2007. He joined AT&T Labs–Research, Florham Park, NJ, in 2007, where he is now a Senior Member of Technical Staff. His research interests include multiuser information theory, joint source–channel coding, quantization design and analysis, as well as image/video coding and processing.

Vinay Vaishampayan (S'85–M'85–SM'03) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Delhi, in 1981 and the Ph.D. degree in electrical engineering from the University of Maryland, College Park, in 1989.

He a is a Distinguished Member of Technical Staff at AT&T's Shannon Laboratory, Florham Park, NJ. Before joining AT&T in 1996, he was on the faculty of the Electrical Engineering Department at Texas A&M University, College Station, TX. His research interests include communication and information theory.

Neil J. A. Sloane (S'62–M'66–SM'77–F'78–LF'05) received the Ph.D. degree from Cornell University, Ithaca, NY, in 1967.

He was Assistant Professor, Electrical Engineering Department, Cornell University from 1967 to 1969; Member of Technical Staff, Bell Laboratories, Murray Hill, NJ, from 1969 to 1995; Principal Member of Technical Staff, AT&T Labs–Research, Murray Hill, NJ, 1996; Technology Leader, AT&T Labs-Research, Florham Park, NJ, 1997 to the presemt. He is the author of several books, including: *A Handbook of Integer Sequences*, Academic Press, 1973; *The Theory of Error-Correcting Codes* (with F. J. MacWilliams), North-Holland, 1977; *Hadamard Transform Optics* (with M. Harwit), Academic Press, 1979; *Sphere-Packings, Lattices and Groups* (with J. H. Conway), Springer-Verlag, 3rd edition, 1998; *Claude Elwood Shannon: Collected Papers* (edited, with A. D. Wyner), IEEE Press, 1993; *The Encyclopedia of Integer Sequences* (with S. Plouffe), Academic Press, 1995; *Orthogonal Arrays* (with A. S. Hedayat and J. Stufken), Springer-Verlag, 1999; *Self-Dual Codes and Invariant Theory* (with G. Nebe and E. M. Rains), Springer-Verlag, 2006;

Dr. Sloane is a member of the National Academy Engineering and an AT&T Fellow. He was the Editor-in-Chief of IEEE TRANSACTIONS INFORMATION THEORY from 1978 to 1980. He received the Chauvenet Prize, from the Mathematical Association of America, 1979; the 1984 Earle Raymond Hedrick Lecturer of the Mathematical Association of America; the 1987 Information Theory Society Paper Award (with J. H. Conway); the 1995 IEEE Information Theory Society Prize Paper Award (with A. R. Calderbank, A. R. Hammons, Jr., P. V. Kumar, and P. Solé); he was the 1997 Shannon Lecturer of IEEE Information Theory Society; and received the IEEE Richard W. Hamming Medal, 2005; and the David R. Robbins Prize, Mathematical Association of America, 2008.