Ordered Binary Decision Diagrams and Minimal Trellises

John Lafferty[*] Alexander Vardy[†]

July 30, 1998
CMU-CS-98-162

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[†]Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, 1308 W. Main Street, Urbana IL, 61801.

[*]Research supported in part by the National Science Foundation under award CCR-9805366.
[†]Research supported in part by the Packard Foundation and by the National Science Foundation.

Keywords: binary decision diagrams, trellises, CAD, coding theory, formal verification

Abstract

Ordered binary decision diagrams (OBDDs) are graph-based data structures for representing Boolean functions. They have found widespread use in computer-aided design and in formal verification of digital circuits. Minimal trellises are graphical representations of error-correcting codes that play a prominent role in coding theory. This paper establishes a close connection between these two graphical models, as follows. Let $\mathbb{C}$ be a binary code of length n, and let $f_{\mathbb{C}}(x_1, \ldots, x_n)$ be the Boolean function that takes the value 0 at $x_1, \ldots, x_n$ if and only if $(x_1, \ldots, x_n) \in \mathbb{C}$. Given this natural one-to-one correspondence between Boolean functions and binary codes, we prove that the minimal proper trellis for a code $\mathbb{C}$ with minimum distance $d > 1$ is isomorphic to the single-terminal OBDD for its Boolean indicator function $f_{\mathbb{C}}(x_1, \ldots, x_n)$. Prior to this result, the extensive research during the past decade on binary decision diagrams – in computer engineering – and on minimal trellises – in coding theory – has been carried out independently. As outlined in this work, the realization that binary decision diagrams and minimal trellises are essentially the same data structure opens up a range of promising possibilities for transfer of ideas between these disciplines.

1. Introduction

Algorithms on graphical structures play a central role in both communications and computer engineering. Most modern communications systems make use of error-correcting codes in order to increase reliability and manage resources such as power and spectrum. In this context, trellises [75] and related graphs [37, 38] have emerged as a unifying framework for understanding, manipulating, and decoding error-correcting codes of all types. In computer engineering, ordered binary decision diagrams [14, 16] and their variants have found widespread use for a range of applications, including circuit checking, logic synthesis, and test generation. Binary decision diagrams are at the core of many tools for formal verification, and have been a major reason for recent advances in this area.

In this paper, we show that there is a very close relationship between trellises and binary decision diagrams. In particular, we show that if a binary error-correcting code $\mathbb{C}$ has minimum distance greater than one, then the minimal proper trellis for $\mathbb{C}$ is isomorphic to the single-terminal ordered binary decision diagram (OBDD) for this code, viewed as a Boolean function. Our proof is based on a direct argument using a vertex-merging construction of OBDDs due to Bryant [14, 16], along with some basic results on minimal trellises. We thus establish a bridge between previously disparate areas of research that makes possible coordinated exploration and transfer of ideas between them. One of our goals in this paper is to make the two research communities aware of each other.

Prior to this result, the historical development of ideas surrounding OBDDs and trellises was independent, yet remarkably parallel. In coding, trellises were introduced by Forney [30], and first used to represent and decode block codes by Bahl, Cocke, Jelinek, and Raviv [5]. However, the subject remained dormant until the publication of [34, 63] in 1988, that ignited a flurry of research during the past decade. To date, the study of trellises for block codes encompasses a sizable body of results — a comprehensive bibliography, consisting of some 100 references, may be found in the recent survey [75]. In a similar fashion, the idea of representing Boolean functions as decision graphs was recorded in the early papers of Lee [57] and Akers [2]. However, their widespread use as the data structure of choice for symbolic Boolean manipulation started with the work of Bryant [14] in 1986, who formulated a set of algorithms for constructing binary decision diagrams, and operating upon them. Key to this algorithmic formulation was the requirement that the variables along every path from the root to a leaf occur in a fixed order, which is analogous to the well-defined depth property of trellises. During the past decade, binary decision diagrams have been a very active research topic in automated logic design and verification, and the subject has now accumulated a vast body of literature.

Not surprisingly, the key results and the central research problems in the two areas share much in common. A fundamental theorem in the study of trellises is due to Muder [63], who showed that every block code has a minimal proper trellis representation, and any two minimal proper trellises for the same code are isomorphic. On the other hand, Bryant [14] proved that the OBDD representation of a Boolean function is canonical: for a given ordering of variables, two OBDDs for the same function are isomorphic. We now realize that these are two instances of the same result, described in different languages. A central problem in the study of OBDDs is how to order the variables for a given function so that the size of the resulting decision diagram is minimized. A similar problem for trellises, known [61, 75] as the *art of trellis decoding* or the *permutation problem*, asks how the time axis for a given code should be permuted in order to minimize the complexity of the resulting trellis. Once again, these are essentially two instances of the same problem. In both cases, the research is centered around techniques for combating the exponential growth in the size of the graph; but the methods that have been developed are complementary. The close relationship between OBDDs and minimal trellises that we establish here may therefore lead to useful results for each discipline.

We point out that the possibility of connection between binary decision diagrams and trellises was noted in passing by Horn and Kschischang [43], who wrote that "block-code trellises appear to be closely related to graphs called binary decision diagrams that are used to represent Boolean functions." However, to the best of our knowledge, this connection was never pursued in the literature beyond the single sentence quoted above.

The rest of this paper is organized as follows. In order to make our results accessible to both communities — computer engineering and coding — we start with a brief overview of the basic concepts concerning BDDs and trellises in the next two sections. These two sections also contain pointers to the literature on their respective subjects. In Section 4, we prove our main result: the correspondence between OBDDs and minimal trellises. Some directions for transfer of ideas between the two areas are then discussed in Section 5.

2. Binary decision diagrams

Binary decision diagrams are a graph-based data structure for representing Boolean functions [14, 16]. They have found widespread use in computer-aided design of digital circuits, and form the heart of many tools for formal verification [3, 21, 26]. They are also used extensively in logic synthesis [67], and in various aspects* of circuit testing [9].

The success of binary decision diagrams has led to research efforts on a number of fronts, as surveyed in [18]. First, there have been many improvements to the core technology, refining the algorithms and representation techniques for improved performance [12, 40, 64, 66]. Secondly, a number of extensions to the data structure have been developed, leading to a more general class of representations known as decision diagrams. Some of these extensions attempt to improve the compactness of representation [7, 28], while

*The importance and potential impact of these methods can be gauged by the highly-publicized Intel Pentium floating-point divider bug in 1994, which cost the company an estimated $475 million. It has been shown [17] that Intel could have used ordered binary decision diagrams to detect and correct the erroneous table entries in the Pentium floating-point divider.

others extend the class of functions that can be represented [20, 4, 23, 24, 27, 56]. Finally, decision diagrams have been applied to a wider range of tasks in [60].

In this section, we review the basics of binary decision diagrams, and in particular present the canonical algorithm [14, 16] for building the OBDD for a Boolean function. This algorithm will be used in Section 4 to construct the minimal trellis for a binary code.

2.1. Construction of ordered binary decision diagrams

A binary decision diagram represents a Boolean function as a rooted, directed acyclic graph. The leaves (vertices of degree zero) in this graph are called terminal vertices, or simply *terminals*. The terminals are labeled 0 or 1, corresponding to the possible function values. Each nonterminal vertex v is labeled by a function variable $\mathsf{var}(v)$ and has two outgoing edges, corresponding to the cases where the variable takes on the value 0 or 1 and directed towards the two *children* of v, denoted $\hookrightarrow_0(v)$ and $\hookrightarrow_1(v)$, respectively. For any truth assignment to the variables, the function value is determined by tracing a path from the root to a terminal vertex, following the appropriate edge from each vertex.

One example of a binary decision diagram for a Boolean function $f(x_1,\ldots,x_n)$ is a *full binary decision tree*, which contains 2^n terminals and $2^n - 1$ nonterminals. This is illustrated in Figure 1a for the function* $(x_1 + x_2) \cdot x_3$. However, binary decision diagrams are usually much more compact. For example, a smaller BDD for the same function is illustrated in Figure 1b, while Figure 1c depicts a BDD for the function $x_1 + x_2 + x_3$.

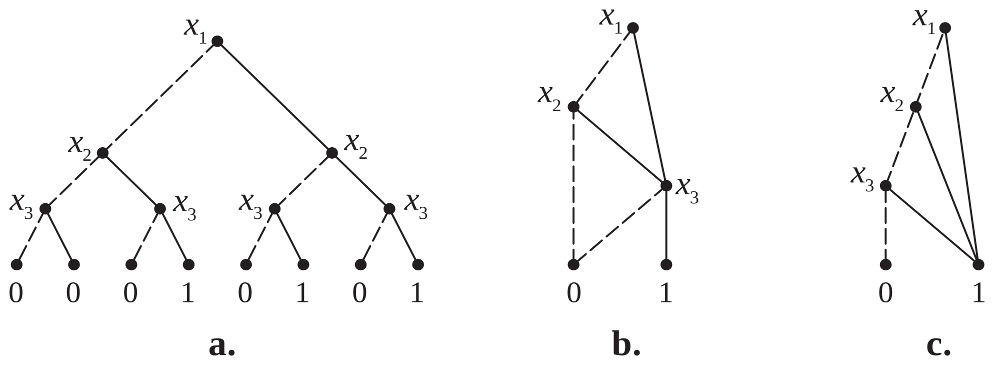


Figure 1. *Examples of binary decision diagrams*

A dashed, respectively solid, line indicates the edge that is followed when the decision variable is 0, respectively 1.

To introduce *ordered* binary decision diagrams, we impose an arbitrary total order $\prec$ on the set of variables $x_1,\ldots,x_n$. Then the *ordered binary decision diagram* $\mathcal{D}$ for a Boolean function $f(x_1,\ldots,x_n)$ is defined by the following properties: a) every path from the root

*We use the symbols $+$, $\cdot$, $\oplus$, and $^{-}$ to denote Boolean OR, AND, EXCLUSIVE-OR, and NOT, respectively.

to a leaf in $\mathcal{D}$ encounters variables in ascending order, and b) it does not contain duplicate terminals or nonterminals, or redundant tests (precise definitions of these terms follow in the next paragraph). For example, the graphs in Figures 1b and 1c are OBDDs, if we consider the variables to have the ordering $x_1 \prec x_2 \prec x_3$.

Bryant [14] proved that the OBDD representation of a given function is *unique* — for a given ordering, two OBDDs for the same function are isomorphic. He also showed that the OBDD for an arbitrary Boolean function $f(x_1, \ldots, x_n)$ can be constructed by applying a set of reduction rules to the full binary decision tree for $f(x_1, \ldots, x_n)$. First, terminals in the decision tree having the same label are merged. This step, known as ***merging duplicate terminals***, results in a directed graph with only two terminals, labeled 0 and 1. A nonterminal v in this graph is said to be a ***redundant test*** if $\hookrightarrow_0(v) = \hookrightarrow_1(v)$. Redundant tests may be ***removed***, without altering the function being represented, by deleting v and redirecting all incoming edges to $\hookrightarrow_0(v)$. Two nonterminals u and v are said to be ***duplicate*** if $\hookrightarrow_0(v) = \hookrightarrow_0(u)$, $\hookrightarrow_1(v) = \hookrightarrow_1(u)$, and $\mathsf{var}(v) = \mathsf{var}(u)$. Duplicate nonterminals can be ***merged*** by deleting one of the two vertices and redirecting all incoming edges to the other vertex. Again, this does not affect the function being represented.

The reduction algorithm proceeds by iteratively merging duplicate nonterminals and removing redundant tests. It terminates when no redundant test or duplicate nonterminals remain. This algorithm is summarized below.

Construction A

Input: Boolean function $f(x_1, \ldots, x_n)$ and variable ordering $x_1 \prec \cdots \prec x_n$.
Output: Ordered binary decision diagram for $f(x_1, \ldots, x_n)$.

Algorithm: Starting with the full binary decision tree for $f(x_1, \ldots, x_n)$, proceed as follows:

Step 1. Merge duplicate terminals.
Step 2. Merge all duplicate nonterminals.
Step 3. Remove all redundant tests.

Iterate steps **2** and **3** until no duplicate nonterminals or redundant tests remain.

It is easy to see that Construction A always produces the unique OBDD for $f(x_1, \ldots, x_n)$. To illustrate this construction, consider the Boolean function:

$$f(x_1, x_2, x_3, x_4, x_5) \;=\; (x_1 \oplus x_2 \oplus x_3) + (x_1 \oplus x_4) + (x_1 \oplus x_2 \oplus x_5)$$

Figure 2 shows the OBDD for this function, during the various stages of its construction: the top part of the figure depicts the binary decision tree with the terminals merged, the center shows the result of merging duplicate nonterminals, and the bottom part shows the BDD obtained after removing redundant tests. In this particular example, there are no additional duplicate nonterminals generated by step **3**, so the algorithm terminates.

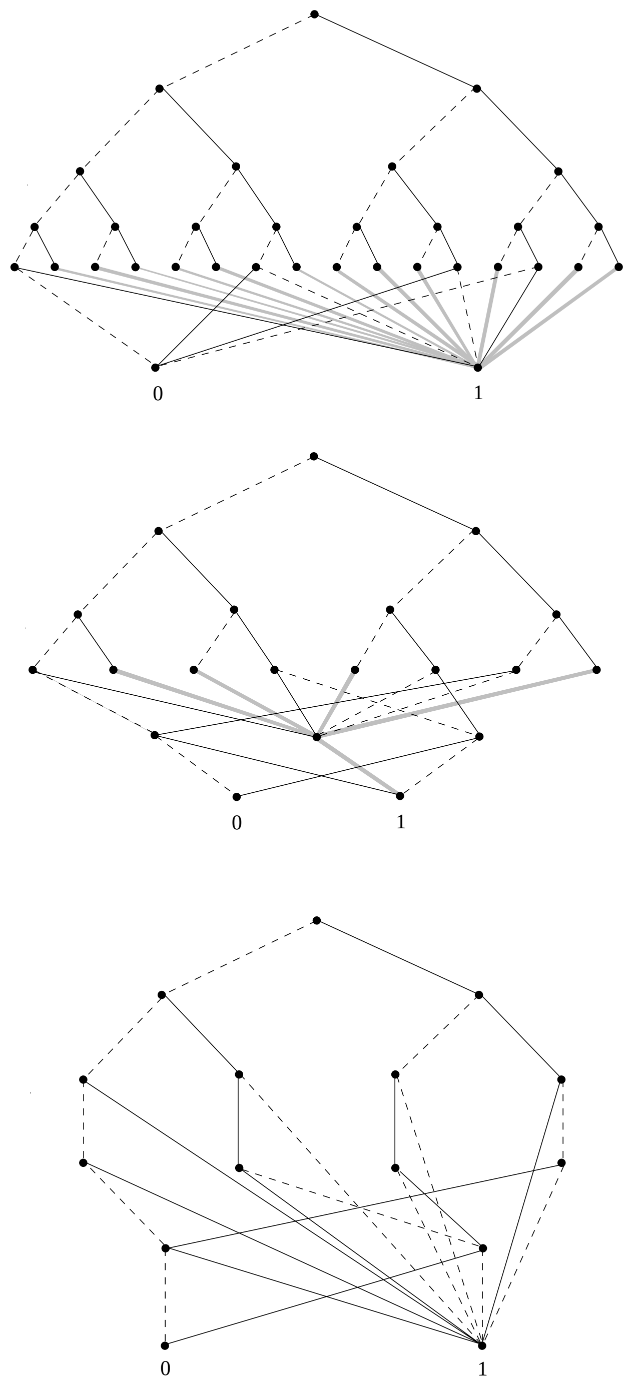


Figure 2. *The OBDD for* $(x_1 \oplus x_2 \oplus x_3) + (x_1 \oplus x_4) + (x_1 \oplus x_2 \oplus x_5)$ *with respect to the ordering* $x_1 \prec x_2 \prec x_3 \prec x_4 \prec x_5$

The OBDD is shown during the various stages of its construction: after step **1** has been carried out (top), after duplicate nonterminals have been merged (center), and after redundant tests have been removed (bottom). Upon completion of steps **1**–**3**, there are no additional duplicate nonterminals, and the algorithm terminates. For clarity, a pair of parallel edges between two vertices, is shown as a shaded line.

2.2. Operations on ordered binary decision diagrams

A number of symbolic operations on Boolean functions can be implemented as simple graph algorithms applied to their OBDD representations [14, 16]. These algorithms typically have complexities that are polynomial in the size of their input. We give just one example here. To describe this example, we will need some more notation and insight into the structure of ordered binary decision diagrams.

Given a Boolean function $f(x_1, \ldots, x_n)$, the function f_x which replaces variable x by the value 1 is called the *positive cofactor* of f with respect to x, while the function $f_{\overline{x}}$ that replaces variable x by the value 0 is called the *negative cofactor*. The Shannon expansion (originally recognized by Boole [13]) expresses $f(x_1, \ldots, x_n)$ as follows:

$$f \;=\; \overline{x} \cdot f_{\overline{x}} \;+\; x \cdot f_x \tag{1}$$

Since at least one of the two terms in the sum above must evaluate to zero, this decomposition splits an arbitrary function into two mutually exclusive cases.

Suppose that a vertex v in an OBDD represents some function f^*. If $\mathsf{var}(v) = x$, then $\hookrightarrow_1(v)$ represents the function f^*_x and $\hookrightarrow_0(v)$ represents the function $f^*_{\overline{x}}$. We postulate that the root vertex in the OBDD for a Boolean function $f(x_1, \ldots, x_n)$ represents the function f itself. Then following a path from the root to a leaf corresponds to taking successive cofactors of $f(x_1, \ldots, x_n)$ until it reduces to a constant. In other words, OBDDs are graphical representations of the Shannon expansion (1) of a Boolean function.

One use of OBDDs is to test the equivalence of two logic circuits [14, 16]. If the circuits are represented as OBDDs corresponding to two functions f and g, then the verification is carried out by computing $f \oplus g$ and testing whether the result is the constant function 0. This can be done efficiently using the fact that the cofactor operations distribute through the Boolean operations; for example $(f \oplus g)_x = f_x \oplus g_x$. Hence, we can compute $f \oplus g$ as $\overline{x} \cdot (f_{\overline{x}} \oplus g_{\overline{x}}) \;+\; x \cdot (f_x \oplus g_x)$. As a consequence, the verification can be efficiently carried out using a recursive graph traversal algorithm. For more details on this and many other applications of OBDDs, we refer the reader to the survey by Bryant [16].

3. Minimal trellises for block codes

Trellises were introduced by Forney [30] in 1967 as a conceptual means to explain the inner workings of the Viterbi algorithm [32] for decoding convolutional codes. IBM researchers Bahl, Cocke, Jelinek, and Raviv [5] were the first to observe that linear *block* codes may be also represented by a trellis, and showed how to construct such a trellis. For a detailed survey of the trellis theory of block codes, we refer the reader to Vardy [75].

Today, trellises are used extensively in the construction and decoding of error-correcting codes, where their applications range from deep-space communications (trellises were used to transmit images from Mars in 1977), through high-speed modems, to household appliances such as CD players. Furthermore, trellises were also found useful in such areas as channel equalization [31], hidden Markov models [29], and speech recognition [6].

In this section we present the definition of a trellis, and only briefly touch on some of its properties. We also define the minimal proper trellis for a given binary code. This notion will be used in the next section to establish the connection with OBDDs.

Loosely speaking, a trellis $T = (V, E, A)$ is an edge-labeled directed graph with the property that every vertex in T has a well-defined depth. We will regard each labeled, directed edge $e \in E$ as an ordered triple (v, v', a), and say that this edge *begins* at $v \in V$, *ends* at $v' \in V$, and has label $a \in A$. With this terminology, we have the following definition.

Definition 1. *A trellis $T = (V, E, A)$ of depth n is an edge-labeled directed graph with the following property: the vertex set V can be partitioned as*

$$V = V_0 \cup V_1 \cup \cdots \cup V_n$$

such that every edge in T that begins at a vertex in V_i ends at a vertex in V_{i+1}, and every vertex in T lies on at least one path from a vertex in V_0 to a vertex in V_n.

For $i = 0, 1, \ldots, n$, we will refer to V_i as the set of vertices at *time* i, and call the ordered index set $\mathcal{I} = \{0, 1, \ldots, n\}$ induced by the partition of the vertex set the *time axis* for T. This temporal terminology is both natural and standard [75] in the study of trellises.

In most cases of interest, the subsets $V_0, V_n \subset V$ each consist of a single vertex, called the *root* and the *toor*, respectively, and this will be assumed in the remainder of this paper. A trellis T is said to be *proper* if the edges beginning at any given vertex of T are labeled distinctly. It is said to be *co-proper* if this condition holds with the direction of all edges reversed: namely, if the edges *ending* at any vertex of T are labeled distinctly. A trellis T is said to be *biproper* if it is both proper and co-proper.

The set of binary n-tuples is denoted $\mathbb{F}_2^n$. For $x, y \in \mathbb{F}_2^n$, the Hamming distance $d(x, y)$ is the number of positions where x and y differ. An $[n, M, d]$ binary *block code* $\mathbb{C}$ is a subset of $\mathbb{F}_2^n$ of cardinality M, such that $\min_{x,y \in \mathbb{C}} d(x, y) = d$. The elements of $\mathbb{C}$ are called codewords. An (n, k, d) binary *linear* code is a *subspace* of $\mathbb{F}_2^n$ of dimension k and minimum distance d. An (n, k, d) binary linear code can be specified either as the row-space of a $k \times n$ binary *generator matrix* or as the kernel of an $(n-k) \times n$ binary *parity-check matrix*. A block code $\mathbb{C}$ is said to be *rectangular* if for all choices of a, b, c, d, the fact that $(a, c), (a, d), (b, c) \in \mathbb{C}$ implies that $(b, d) \in \mathbb{C}$, where $(\cdot\,, \cdot)$ denotes string concatenation. It is easy to see that every linear code is rectangular, but not vice versa.

Definition 2. *Let $T = (V, E, \mathbb{F}_2)$ be a trellis of depth n. Then the sequence of edge labels along each path from the root to the toor in T defines an ordered binary n-tuple. We say that T represents a binary block code $\mathbb{C}$ of length n, or simply that T is a trellis for $\mathbb{C}$, if the set of all such n-tuples is precisely the set of codewords of $\mathbb{C}$.*

The minimal trellis may be defined in a number of different ways which, in most cases, are all equivalent to the following definition. We say that a trellis T for a code $\mathbb{C}$ of length n is *minimal* if it satisfies the following property: for each $i = 0, 1, \ldots, n$, the number of vertices at time i in T is less than or equal to the number of vertices at time i in any other trellis for $\mathbb{C}$. Given a code $\mathbb{C}$, it is not at all obvious that there exists a minimal trellis for $\mathbb{C}$. Although it is known [51, 75, 76] that such a trellis exists (and is, in fact, unique up to graph isomorphism) if the code $\mathbb{C}$ is rectangular, there are also examples [52] of non-rectangular codes that do not admit a minimal trellis representation. On the other hand, this problem does not arise if we restrict our attention to *proper* trellises.

Definition 3. *Let T be a proper trellis for a code $\mathbb{C}$ of length n. We say that T is the minimal proper trellis for $\mathbb{C}$ if it satisfies the following property: for each $i = 0, 1, \ldots, n$, the number of vertices at time i in T is less than or equal to the number of vertices at time i in any other proper trellis for $\mathbb{C}$.*

One of the fundamental results in trellis theory, due to Muder [63], is that every block code, whether it is rectangular or not, has a unique minimal proper trellis. For rectangular codes (and, hence, also for linear codes), it is known [75] that the minimal proper trellis and the minimal trellis coincide. For linear codes, the minimal trellis is sometimes called the *BCJR trellis*, after the authors of [5] who first came up with the construction of such a trellis. We elaborate upon the BCJR construction in Section 5.

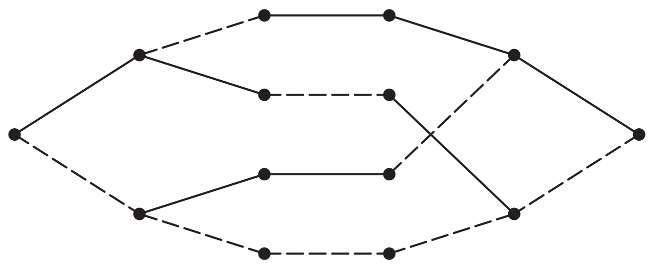


Figure 3. *Minimal trellis for the code* $\mathbb{C} = \{00000, 11010, 01101, 10111\}$

There are several natural measures of complexity for a given trellis, including the state complexity $s = \max_i \log |V_i|$, the edge complexity $|E|$, and the Viterbi decoding complexity $D = 2|E| - |V| + 1$. Recent work has clarified the relationship between these parameters, and to a large extent they can be considered as equivalent, at least as the block length n gets large. The minimal trellis uniquely minimizes *all* of these complexity measures, given a fixed time axis for the code. The precise statement and proof of this and other related facts is the subject of a number of recent papers [33, 62, 73, 76].

As a simple example, consider the $(5, 2, 3)$ linear code $\mathbb{C} = \{00000, 11010, 01101, 10111\}$. The minimal trellis for this code is shown in Figure 3. The complexity measures for this trellis, namely $s = 2$, $|E| = 16$, and $D = 19$, can be easily found by inspection.

4. The main result: OBDDs and minimal trellises

In this section, we rigorously establish the connection between minimal proper trellises and ordered binary decision diagrams. In doing so, we will make frequent use of concepts, constructions, and theorems discussed in the foregoing two sections.

First, we observe that there is a natural one-to-one correspondence between Boolean functions of n variables and binary codes of length n. Let $\mathbb{C}$ be a such a code, not necessarily linear or rectangular. We define the Boolean function $f_{\mathbb{C}}(x_1,\ldots,x_n)$ as follows:

$$f_{\mathbb{C}}(x_1,\ldots,x_n) \overset{\text{def}}{=} \begin{cases} 0 & \text{if } (x_1,\ldots,x_n) \in \mathbb{C} \\ 1 & \text{otherwise} \end{cases}$$

We call $f_{\mathbb{C}}(x_1,\ldots,x_n)$ the *indicator function* of $\mathbb{C}$. To make the terminology concise, we will often refer to a binary decision diagram for the indicator function of $\mathbb{C}$ simply as a BDD for $\mathbb{C}$. Equivalently, given a Boolean function $f(x_1,\ldots,x_n)$, we define the binary block code $\mathbb{C}_f$ of length n as the set of all truth assignments to $x_1,\ldots,x_n$ such that $f(x_1,\ldots,x_n) = 0$. Thus $\mathbb{C}_f$ is just the *off-set* of f, and f is the indicator function of $\mathbb{C}_f$.

Next, we define the *single-terminal* OBDD for a Boolean function $f(x_1,\ldots,x_n)$ by the following procedure, analogous to Construction A. In fact, this procedure is exactly the same as Construction A, except for one extra step, as summarized below.

Construction B

Input: Boolean function $f(x_1,\ldots,x_n)$ and variable ordering $x_1 \prec \cdots \prec x_n$.
Output: Single-terminal ordered binary decision diagram for $f(x_1,\ldots,x_n)$.

Algorithm: Starting with the full binary decision tree for $f(x_1,\ldots,x_n)$, proceed as follows:

- **Step 1.** Merge duplicate terminals.
- **Step X.** Prune away the 1-terminal.
- **Step 2.** Merge all duplicate nonterminals.
- **Step 3.** Remove all redundant tests.

Iterate steps **2** and **3** until no duplicate nonterminals or redundant tests remain.

Recall that after merging the duplicate terminals in step **1**, we have a directed graph with exactly two terminal vertices, labeled 0 and 1. We then recursively remove all the edges and vertices leading only to the terminal labeled 1. This is the step of *pruning the one-terminal* in Construction B. Each nonterminal vertex in the resulting graph has either one or two children. If a given vertex v has only one child, we set $\hookrightarrow_0(v) = \varnothing$ or $\hookrightarrow_1(v) = \varnothing$, by convention. With this convention, the definitions of redundant tests and duplicate nonterminals remain as before, and the algorithm then continues as in Construction A.

The resulting decision diagram has a single terminal vertex, corresponding to all the sequences that evaluate to 0 by $f(x_1,\ldots,x_n)$, or equivalently all of the codewords of $\mathbb{C}_f$. It is important to note that since $f(x_1,\ldots,x_n)$ is binary, this does not discard any information, and the complete OBDD can be reconstructed from the single-terminal OBDD.

This observation shows that the single-terminal OBDD can be also obtained in a slightly different manner. Namely, the operation of pruning away the 1-terminal (step **X**) can be carried out after the full OBDD for $f(x_1,\ldots,x_n)$ is constructed. We will refer to this variation as Construction C. Indeed, it is not difficult to show that the graphs $\mathcal{D}_{\rm B}$ and $\mathcal{D}_{\rm C}$ produced by Constructions B and C, respectively, are isomorphic. Each nonterminal vertex in these graphs has out-degree one or two. In every instance where the out-degree is one, the missing edge must correspond to a sequence that belongs to the on-set of $f(x_1,\ldots,x_n)$. Hence, by first appending a terminal labeled 1, and then adding an edge from each unary vertex to this 1-terminal, labeling this edge so that the resulting graph is proper, we obtain a complete OBDD for $f(x_1,\ldots,x_n)$ from both $\mathcal{D}_{\rm B}$ and $\mathcal{D}_{\rm C}$. However, two complete OBDDs for the same function are isomorphic, and hence so are $\mathcal{D}_{\rm B}$ and $\mathcal{D}_{\rm C}$.

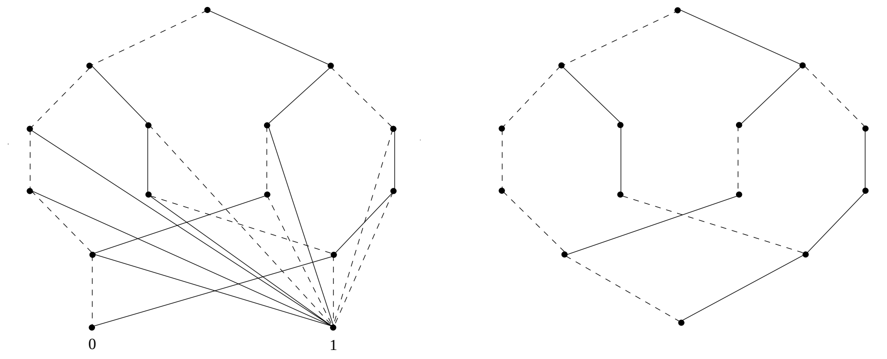


Figure 4. *The OBDD and the single-terminal OBDD for the function f in* (2)*, or equivalently for the code $\mathbb{C}$ defined by* (3)

As a simple example for Construction B (or for Construction C), consider again the Boolean function that was used in Section 2 to illustrate Construction A, namely:

$$f(x_1,x_2,x_3,x_4,x_5) \;=\; (x_1\oplus x_2\oplus x_3)+(x_1\oplus x_4)+(x_1\oplus x_2\oplus x_5) \qquad (2)$$

Notice that $f(x_1,x_2,x_3,x_4,x_5)$ is also the indicator function of the $(5,2,3)$ linear code $\mathbb{C}=\{00000,11010,01101,10111\}$ used as an example in Section 3. This becomes immediately clear upon observing that a parity-check matrix for $\mathbb{C}$ is given by

$$H \;=\; \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} \qquad (3)$$

The OBDD and the single-terminal OBDD for $\mathbb{C}$ are shown in Figure 4. Notice that the single-terminal OBDD is the same as the minimal proper trellis for $\mathbb{C}$, shown in Figure 3. Our main result is the following theorem, proving that this must always be the case.

Theorem 1. *Let $\mathbb{C}$ be an arbitrary binary code with minimum distance $d > 1$. Then the single-terminal OBDD for $\mathbb{C}$ is the unique minimal proper trellis for $\mathbb{C}$.*

Proof. It is easy to see that the graph resulting after steps **1** and **X** in Construction B is a trellis for $\mathbb{C}$. By the $\hookrightarrow_0(v) = \hookrightarrow_0(u)$ and $\hookrightarrow_1(v) = \hookrightarrow_1(u)$ property of duplicate nonterminals, the merging procedure in step **2** does not create any new paths that are not codewords. Furthermore, by the $\mathsf{var}(v) = \mathsf{var}(u)$ property, this procedure also preserves depth. Hence, the graph resulting after step **2** is still a trellis for $\mathbb{C}$. Now, since $d > 1$, there can be no redundant tests in any trellis for $\mathbb{C}$. Thus step **3** in Construction B is vacuous, and the single-terminal OBDD is a trellis for $\mathbb{C}$. Furthermore, it is obvious that the outgoing edges of every vertex in any binary decision diagram must be labeled distinctly. Hence the single-terminal OBDD for $\mathbb{C}$ is a proper trellis for $\mathbb{C}$.

It remains to show that the single-terminal OBDD is the *minimal* proper trellis for $\mathbb{C}$. To this end, we need to introduce some more notation and results from trellis theory. For $i = 1, 2, \ldots, n-1$, we define the projection of $\mathbb{C}$ on the past at time i as follows:

$$\mathcal{P}_i(\mathbb{C}) \overset{\text{def}}{=} \Big\{(c_1, c_2, \ldots, c_i) \; : \; (c_1, \ldots, c_i, c_{i+1}, \ldots, c_n) \in \mathbb{C} \;\text{ for some } c_{i+1}, \ldots, c_n \in \mathbb{F}_2\Big\}$$

For each $c \in \mathcal{P}_i(\mathbb{C})$, we define the future of c as $\mathcal{F}(c) = \{x \in \mathbb{F}_2^{n-i} : (c, x) \in \mathbb{C}\}$, and say that $c_1, c_2 \in \mathcal{P}_i(\mathbb{C})$ are *future-equivalent* if $\mathcal{F}(c_1) = \mathcal{F}(c_2)$. It is shown in [63] that a proper trellis $T = (V, E, \mathbb{F}_2)$ for $\mathbb{C}$ is minimal if and only if for all $i = 1, 2, \ldots, n-1$, the number of vertices at time i in T is equal to the number of future-equivalence classes defined by this relation. From this, we can derive an alternative necessary and sufficient condition for minimality as follows. Given a vertex $v \in V_i$, we define:

$$\mathcal{F}_T(v) \overset{\text{def}}{=} \Big\{x \in \mathbb{F}_2^{n-i} : x \text{ is a sequence of edge labels along a path in } T \text{ starting at } v\Big\}$$

Then a proper trellis T is minimal if and only if for all $i = 1, 2, \ldots, n-1$ and for every pair of vertices $v, v' \in V_i$, we have $\mathcal{F}_T(v) \neq \mathcal{F}_T(v')$. Indeed, this condition implies that $c_1, c_2 \in \mathcal{P}_i(\mathbb{C})$ are equivalent if and only if the paths corresponding to c_1 and c_2 end at the same vertex of V_i. Thus $|V_i|$ must be equal to the number of future-equivalence classes.

Now consider the single-terminal OBDD for $\mathbb{C}$. We already know that this is a proper trellis for $\mathbb{C}$. Call this trellis $T = (V, E, \mathbb{F}_2)$, and assume to the contrary that there exist two distinct vertices $v, v' \in V_i$ with $\mathcal{F}_T(v) = \mathcal{F}_T(v')$. By Construction B, at least one of $\{\hookrightarrow_0(v), \hookrightarrow_0(v')\}$ or $\{\hookrightarrow_1(v), \hookrightarrow_1(v')\}$ must be a pair of distinct vertices, otherwise v and v' would have been merged as duplicate nonterminals. Notice that we allow for the possibility that some of $\hookrightarrow_0(v), \hookrightarrow_1(v), \hookrightarrow_0(v'), \hookrightarrow_1(v')$ may be $\varnothing$, which means that they are not present in the single-terminal OBDD. However, if one of $\{\hookrightarrow_0(v), \hookrightarrow_0(v')\}$ is $\varnothing$ then so is the other one, since otherwise $\mathcal{F}_T(v) \neq \mathcal{F}_T(v')$. By a similar argument, either $\hookrightarrow_1(v) = \hookrightarrow_1(v') = \varnothing$ or both are present in the OBDD. Thus we may assume, w.l.o.g., that $u = \hookrightarrow_0(v)$ and $u' = \hookrightarrow_0(v')$ are both present in the single-terminal OBDD, and $u \neq u'$. But then $\mathcal{F}_T(v) = \mathcal{F}_T(v')$ implies that $\mathcal{F}_T(u) = \mathcal{F}_T(u')$. Thus, from the existence of distinct vertices $v, v' \in V_i$ with $\mathcal{F}_T(v) = \mathcal{F}_T(v')$, we have deduced the existence of distinct vertices $u, u' \in V_{i+1}$ with $\mathcal{F}_T(u) = \mathcal{F}_T(u')$. Iterating this argument, we arrive at a contradiction, since V_n consists of a single vertex by construction. ▮

As an immediate corollary to Theorem 1 and the fact that the minimal proper trellis is actually minimal for rectangular codes, we conclude that if $\mathbb{C}$ is rectangular and $d > 1$ then the single-terminal OBDD for $\mathbb{C}$ is isomorphic to the unique minimal trellis for $\mathbb{C}$.

We point out that an alternative way to view these results comes from considering a binary code $\mathbb{C}$ or a Boolean function $f_{\mathbb{C}}$ as defining a regular set in $\mathbb{F}_2^n$. As such, the Myhill-Nerode theorem [42] guarantees the uniqueness of the minimal deterministic finite-state automaton (DFA) accepting this set. It follows that when the distance of $\mathbb{C}$ is larger than one, the state diagram of its DFA is the same as the minimal proper trellis, or the single-terminal OBDD. This viewpoint is briefly mentioned in the *multilingual dictionary* of coding, systems theory, symbolic dynamics, and automata theory [35].

5. Directions for transfer of ideas

The connection between binary decision diagrams and trellises established in the previous section makes it possible to translate knowledge accumulated in one discipline into the language of the other. We will give just a few examples of this in what follows. In light of the extensive work that has been done in each of these areas, many other possibilities for transfer of results and ideas between the two disciplines surely exist.

5.1. From trellises to binary decision diagrams

We use results from trellis theory to analyze a certain structural property of binary decision diagrams, provide lower bounds on the size of OBDDs, and derive a new type of decision diagrams that are often more compact than OBDDs. We also comment on the complexity of the variable ordering problem, and on alternative graphical models for Boolean functions that may follow from the recent research in coding theory.

Biproper binary decision diagrams. Let $\mathcal{D}$ be an ordered binary decision diagram for a Boolean function $f(x_1, \ldots, x_n)$. It is obvious that the outgoing edges of every nonterminal vertex in $\mathcal{D}$ must be labeled distinctly. When is it that the *incoming* edges of every nonterminal vertex in $\mathcal{D}$ are also labeled distinctly? The following proposition, which follows directly from Theorem 1, provides an answer to this question.

Proposition 2. *Let $f(x_1, \ldots, x_n)$ be a Boolean function, and let $x_1 \prec \cdots \prec x_n$ be an ordering of its variables. If the corresponding binary code $\mathbb{C}_f$ is rectangular then the incoming edges of every nonterminal vertex in the OBDD for $f(x_1, \ldots, x_n)$ are labeled distinctly.*

Proof. It is known [69, 51, 75, 76] that the minimal proper trellis for a rectangular code is biproper. Thus if $\mathbb{C}_f$ is a rectangular code with minimum distance $d(\mathbb{C}_f) > 1$, then the single-terminal OBDD for $f(x_1,\ldots,x_n)$ is isomorphic to the minimal biproper trellis for $\mathbb{C}_f$, and hence all the incoming edges in the single-terminal OBDD are labeled distinctly. Since every nonterminal vertex in the complete OBDD for $f(x_1,\ldots,x_n)$ is also a vertex in the single-terminal OBDD, the proposition follows.

Now assume that $\mathbb{C}_f$ is rectangular and $d(\mathbb{C}_f) = 1$. Then the graph resulting after step **2** of Construction B is still a biproper trellis for $\mathbb{C}_f$. It remains to observe that removing redundant tests in a biproper trellis does not create duplicate nonterminals, and that the resulting graph remains biproper. ▮

Borrowing the terminology of trellis theory, we will say that a binary decision diagram in which the incoming edges of every nonterminal vertex are labeled distinctly is *biproper*. A biproper single-terminal OBDD has the curious property that it can be used to evaluate the function in two different ways: either traversing from top to bottom — as is the standard practice — or traversing from bottom to top. In other words, the root and the single-terminal are interchangeable in a biproper single-terminal OBDD. This, in particular, implies that the variable orderings $x_1 \prec \cdots \prec x_n$ and $x_n \prec \cdots \prec x_1$ produce isomorphic decision diagrams in this case.

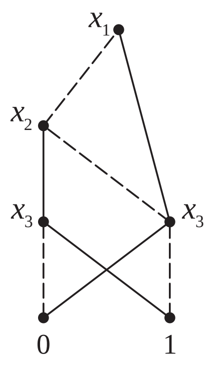


Figure 5. *A biproper OBDD for* $x_1\overline{x}_3 + \overline{x}_1(x_2x_3 + \overline{x}_2\overline{x}_3)$

Notice that whether the OBDD for $f(x_1,\ldots,x_n)$ is biproper depends not only on the function $f(x_1,\ldots,x_n)$ itself, but also on the ordering of its variables. Indeed, there exist codes [69] that are rectangular for some orderings of the time axis and non-rectangular for other orderings. Finally, we observe that the sufficient condition for biproperness given in Proposition 2 is "almost" necessary as well. It is known [51] that a code is rectangular if and *only if* it admits a biproper trellis representation. Thus a Boolean function $f(x_1,\ldots,x_n)$ whose off-set $\mathbb{C}_f$ has distance $d(\mathbb{C}_f) > 1$ can be represented by a biproper OBDD if and *only if* $\mathbb{C}_f$ is rectangular. However, if $d(\mathbb{C}_f) = 1$, this is no longer true in general. As an example, consider the Boolean function:

$$f(x_1,x_2,x_3) \;=\; x_1\overline{x}_3 \;+\; \overline{x}_1(x_2x_3 + \overline{x}_2\overline{x}_3)$$

whose off-set is given by $\mathbb{C}_f = \{001, 010, 101, 111\}$. This function has a biproper OBDD, shown in Figure 5, even though $\mathbb{C}_f$ is not rectangular under any ordering.

Lower bounds on the size of binary decision diagrams. Much work in coding theory [33, 47, 53, 54, 63, 65] has been devoted to lower bounds on the size of the minimal trellis for a given code, under all possible permutations of the time axis. Here, we translate some of these bounds into the language of binary decision diagrams.

To this end, we first need to introduce the appropriate notation. Given a Boolean function $f(x_1,\ldots,x_n)$, we let $\Theta_0(f)$ and $\Theta_1(f)$ denote the cardinalities of the off-set and the on-set of f, respectively. Thus $\Theta_0(f)$ is just the number of codewords in $\mathbb{C}_f$. Next, we elaborate the notation for cofactors of $f(x_1,\ldots,x_n)$ that was introduced in Section 2.2. Given a fixed string $(a_1,\ldots,a_m)\in\{0,1\}^m$ and a subset $\{i_1,i_2,\ldots,i_m\}\subseteq\{1,2,\ldots,n\}$, we let $f|_{x_{i_1},\cdots,x_{i_m}=a_1,\cdots,a_m}$ denote the function obtained from $f(x_1,\ldots,x_n)$ by replacing the variable x_{i_1} by the value a_1, the variable x_{i_2} by the value a_2, and so forth.

For each subset $\mathcal{J}=\{i_1,i_2,\ldots,i_m\}\subseteq\{1,2,\ldots,n\}$, we can now define a discrete random variable $\mathcal{X}_\mathcal{J}$ as follows: $\mathcal{X}_\mathcal{J}$ takes on values in $\{0,1\}^m$ with probabilities given by:

$$\Pr\{\mathcal{X}_\mathcal{J}=(a_1,\ldots,a_m)\} \overset{\text{def}}{=} \frac{\Theta_0\big(f|_{x_{i_1},\cdots,x_{i_m}=a_1,\cdots,a_m}\big)}{\Theta_0(f)} \tag{4}$$

Notice that for some values of $a_1,\ldots,a_m$, the function $f|_{x_{i_1},\cdots,x_{i_m}=a_1,\cdots,a_m}$ may be a tautology, in which case $\Pr\{\mathcal{X}_\mathcal{J}=(a_1,\ldots,a_m)\}=0$. Thus the number of different values that $\mathcal{X}_\mathcal{J}$ takes on may be less than 2^m.

We next recall the definition of entropy. If $\mathcal{X}$ is a discrete random variable taking M values with nonzero probabilities $p_1,p_2,\ldots,p_M$, the *entropy* of $\mathcal{X}$ is given by:

$$H(\mathcal{X}) \overset{\text{def}}{=} p_1\log\frac{1}{p_1} + p_2\log\frac{1}{p_2} + \cdots + p_M\log\frac{1}{p_M}$$

In terms of the notation introduced in the foregoing paragraphs, we are finally ready to define the *entropy profile* of a Boolean function.

Definition 4. *Let $f(x_1,\ldots,x_n)$ be a Boolean function of n variables. We define $\eta_i(f)$ as the minimum possible entropy of a set of i function variables, namely:*

$$\eta_i(f) \overset{\text{def}}{=} \min_{\mathcal{J}} H(\mathcal{X}_\mathcal{J}) \qquad \text{for } i=1,2,\ldots,n$$

where the minimum is taken over all subsets $\mathcal{J}\subseteq\{1,2,\ldots,n\}$ with $|\mathcal{J}|=i$. The sequence $\eta_1(f),\eta_2(f),\ldots,\eta_n(f)$ will be called the entropy profile of $f(x_1,\ldots,x_n)$.

A powerful lower bound on the size of the OBDD for a Boolean function $f(x_1,\ldots,x_n)$ can be derived from its entropy profile, providing $d(\mathbb{C}_f)>1$. Notably, this bound limits the size of the smallest OBDD that can be obtained under *all possible orderings* of the variables $x_1,\ldots,x_n$. The bound was proved by Reuven and Be'ery [65] in the context of trellises; it constitutes a culmination of a long line of work in trellis theory [63, 33, 54]. All we have done here is recast this result in the framework of binary decision diagrams, using the correspondence between BDDs and trellises established in Theorem 1.

Theorem 3. *Let $f(x_1,\ldots,x_n)$ be a Boolean function such that $d(\mathbb{C}_f) > 1$. Then the number of vertices at level i in the OBDD for $f(x_1,\ldots,x_n)$ is lower bounded by*

$$\frac{2^{\eta_i(f)} \cdot 2^{\eta_{n-i}(f)}}{\Theta_0(f)} \tag{5}$$

where $\eta_1(f), \eta_2(f), \ldots, \eta_n(f)$ is the entropy profile of f. This holds for all $i = 1, 2, \ldots, n$ and for any total order on the support $\{x_1,\ldots,x_n\}$.

We believe that it should be possible to extend the scope of Theorem 3 to functions that do not satisfy the requirement $d(\mathbb{C}_f) > 1$. One such extension is immediate. It is obvious, by symmetry, that the same result holds if we look at the *on-set* of the function rather than at the off-set, and replace $\Theta_0(\cdot)$ by $\Theta_1(\cdot)$ in equations (4) and (5). Thus to apply Theorem 3, it would suffice to require that either $\mathbb{C}_f$ or its complement in $\mathbb{F}_2^n$ have minimum distance greater than one. Another possible extension might follow by observing that this requirement essentially ensures that no redundant tests are encountered in the construction of the OBDD. If $\mathbb{C}_f$ is a *rectangular* code with $d(\mathbb{C}_f) = 1$, then removing redundant tests does not create duplicate nonterminals (as noted in the proof of Proposition 2). Thus, in most cases, this step will not reduce the size of the graph significantly. Exploring to what extent the removal of redundant tests can reduce the size of the OBDD beyond the bound of (5) would be an interesting problem for future research.

Sectionalized decision diagrams. Variable orderings for binary decision diagrams correspond to permutations of the time axis for binary codes. Indeed, the problem of finding the best variable ordering for a given function, or equivalently the best permutation of the time axis for a given code, is key in both areas. In trellis theory, another operation on the time axis, called *sectionalization*, has been found useful in a variety of contexts. To the best of our knowledge, the counterpart of this operation for binary decision diagrams has not been investigated previously in the BDD literature.

In trellis theory, a sectionalization corresponds to a choice of the symbol alphabet at each time index. For example, a binary code of length $2n$ may be thought of as a quaternary code of length n if pairs of consecutive bits are grouped together. A wide variety of such granularity adjustments [36] is possible, and each may substantially affect the number of vertices, the number of edges, and the overall structure of the trellis.

The analogous operation for binary decision diagrams consists of grouping consecutive variables together, and taking non-binary decisions at each level, based on the value of all the variables that correspond to this level. Let us illustrate this idea by an example. Consider the following Boolean function:

$$(x_1 \oplus x_2 \oplus x_3 \oplus x_4) + (x_3 \oplus x_4 \oplus x_5 \oplus x_6) + (x_5 \oplus x_6 \oplus x_7 \oplus x_8) + (x_2 \oplus x_3 \oplus x_6 \oplus x_7) \tag{6}$$

The conventional single-terminal OBDD for this function corresponds to grouping its variables into singletons $\{x_1\},\ldots,\{x_8\}$. This decision diagram is shown in Figure 6a. Instead, suppose that we group the variables into pairs $\{x_1,x_2\}, \{x_3,x_4\}, \{x_5,x_6\}, \{x_7,x_8\}$

and take four-way decisions at each of the resulting four levels, depending upon whether the value of the variables in the corresponding pair is 00, 01, 10, or 11. The resulting singe-terminal decision diagram is shown in Figure 6b. It is easy to see that this diagram is substantially more compact than the conventional OBDD, although we have not changed the *order* of the variables (in fact, this order is known [75] to be optimal). Also notice that a complete decision diagram for the function $f(x_1, \ldots, x_8)$ in (6) can be recovered from Figure 6b by adding 28 more edges, in such a way that the out degree of each nonterminal vertex becomes 4, and directing all these edges to the 1-terminal.

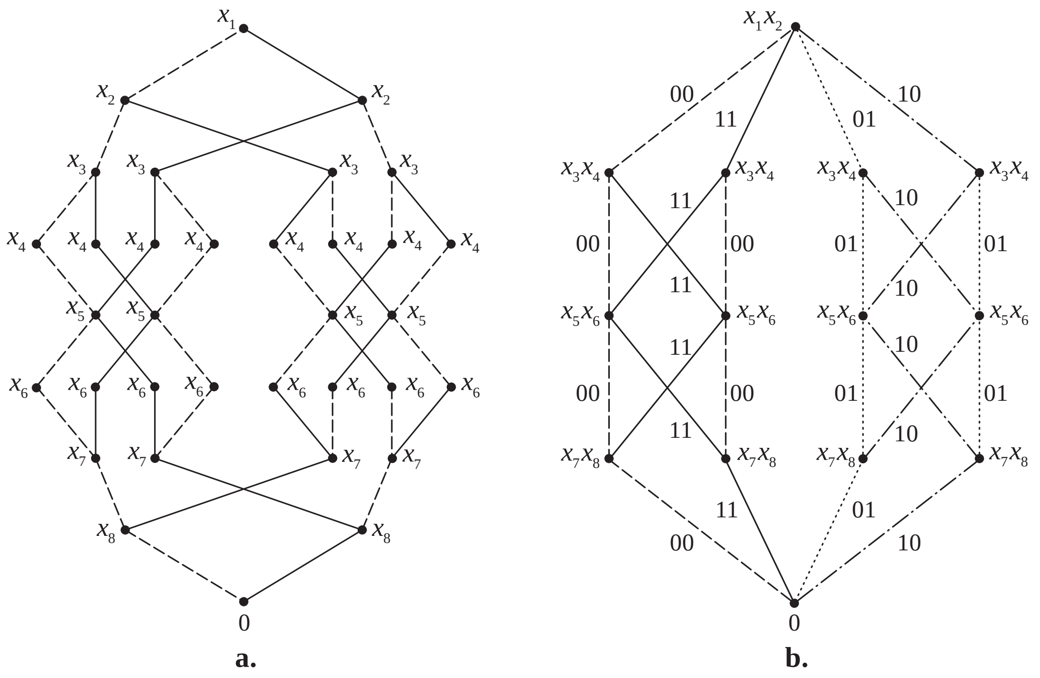


Figure 6. *Two decision diagrams for the Boolean function in* (6)

The edge labels in Figure 6b correspond to the values of the decision variables that result in the traversal of the edge.

In general, there are many different ways to sectionalize a given BDD — that is, to parse the variables $x_1, \ldots, x_n$ into groups: the number of distinct parsings, or sectionalizations, of $x_1, \ldots, x_n$ is about 2^{n-1}. The ***sectionalization problem*** thus consists of finding the optimal parsing among the 2^{n-1} possibilities. In contrast to the variable ordering problem, which is known to be NP-complete for OBDDs, it turns out that the sectionalization problem has a polynomial-time solution. Lafourcade and Vardy [55] devised a *sectionalization algorithm*, based on a dynamic programming approach, that finds the optimal sectionalization of an arbitrary trellis in polynomial time. The algorithm of [55] works for both linear and nonlinear codes, and easily accommodates a broad range of optimality criteria. With some modifications, this algorithm can be applied to binary decision diagrams. If a given single-terminal OBDD represents a function f such that $d(\mathbb{C}_f) > 1$, then the algorithm of [55] works as is. Otherwise, one would need a slightly more complicated book-keeping mechanism for the *composition* and *amalgamation* operations defined in [55]. We leave the details of this modification for future work.

For verification purposes, one of the most important properties of OBDDs is that they are canonical: two functions $f(x_1, \ldots, x_n)$ and $g(x_1, \ldots, x_n)$ are equal if and only if their (single-terminal) OBDDs are isomorphic for the same order on $x_1, \ldots, x_n$. Thus the sectionalization operation would be less useful if it did not preserve canonicity. However, the algorithm of Lafourcade and Vardy [55] can be easily refined in such a way that canonicity is preserved under sectionalization. If we start with two isomorphic trellises and sectionalize them using the algorithm of [55], with respect to the same optimality criterion, then the resulting decision diagrams will be isomorphic. The converse is also true: if two sectionalized decision diagrams are isomorphic, they represent the same function.

Complexity of the variable ordering problem. It is known [10, 11, 43, 45, 74] that the variable ordering problem for binary decision diagrams and the permutation problem for trellises are both computationally hard. However, the known NP-hardness results establish the intractability of *different aspects* of these equivalent problems.

The primary intractability result in the OBDD literature is due to Bollig and Wegener [11], who show that the following decision problem is NP-complete.

Instance: A Boolean function $f(x_1, \ldots, x_n)$ specified in terms of an ordered binary decision diagram, and a positive size bound s.

Question: Is there an ordering of $x_1, \ldots, x_n$ such that the corresponding OBDD for $f(x_1, \ldots, x_n)$ has at most s vertices?

Notice that an implicit assumption in this result is that $f(x_1, \ldots, x_n)$ can be specified by an OBDD whose size is polynomial in n. Indeed, if a function $f(x_1, \ldots, x_n)$ is specified in terms of an OBDD with $N = \Omega(2^n)$ vertices, then the complexity of examining all $n!$ possible orderings of $x_1, \ldots, x_n$ is only $O(N^{\log\log N})$. Furthermore, the reduction used in the proof of [11] explicitly constructs an OBDD whose size is polynomial in n. Thus the hard instances of the foregoing problem are those functions that have a compact OBDD representation. On the other hand, it is known (see [58, 77] and the discussion in the next subsection) that the fraction of such functions becomes vanishingly small as $n \to \infty$.

The hardness results for trellises have a somewhat different flavor. Specifically, Horn and Kschischang [43] prove that the following decision problem is NP-complete.

Instance: A binary *linear* code $\mathbb{C}$ of length n, specified by its parity-check or generator matrix, a positive integer $i < n$, and a positive size bound s.

Question: Is there a permutation of the time axis, such that the number of vertices at time i in the corresponding minimal trellis for $\mathbb{C}$ is less than s?

It is furthermore shown in [74] that this problem remains NP-complete if the size bound is restricted to $s = 2^i$. When translated into the context of binary decision diagrams, using Theorem 1, this implies the following result. Suppose we are given a positive integer $i < n$ and a Boolean function $f(x_1, \ldots, x_n)$ specified in terms of a data structure, *other than* OBDD, whose size is polynomial in n. Then deciding whether there exists an ordering of $x_1, \ldots, x_n$ such that the corresponding OBDD for $f(x_1, \ldots, x_n)$ has less than 2^i vertices at level i is NP-complete.

Alternative graphical models for Boolean functions. In recent years, a number of new graphical models have emerged in coding theory, and evolved into a far-reaching general framework for representing a code by a graph. In this context, one encounters various generalizations of a trellis, such as tail-biting trellises [22] and trellis formations [49, 50], as well as Tanner graphs [71] that are in some sense diametrically opposite to trellises. All these representations are special cases of the general concept of a *factor graph*. We refer the reader to [1, 37, 38, 79] for a detailed treatment of factor graphs and the associated iterative manipulation algorithms: the min-sum and the sum-product.

The success of these graphical models in coding theory and communications has been spectacular. For example, tail-biting trellis representations have been found in [22, 48] for several well-known codes, whose complexity is the square root of the lowest complexity achievable with the conventional minimal trellis. On a grander scale, turbo codes [8] represented by a factor graph and decoded with an iterative sum-product algorithm have been shown to approach channel capacity with feasible complexity, a goal that eluded the research community for almost 50 years. More recently, similar results have been established [59, 70] for low-density parity-check codes, represented by a Tanner graph.

It remains to be seen whether any of the graphical models mentioned in the foregoing paragraphs can be used to efficiently represent Boolean functions in the context of logic synthesis and verification. As an example, consider the well-known *hidden weighted bit* Boolean function, defined by

$$f_{\mathrm{hw}}(x_1, \ldots, x_n) \overset{\mathrm{def}}{=} \begin{cases} 0 & \text{if } \mathrm{wt}(x) = 0 \\ x_{\mathrm{wt}(x)} & \text{if } \mathrm{wt}(x) > 0 \end{cases}$$

where $\mathrm{wt}(x)$ is the number of non-zeros in $(x_1, x_2, \ldots, x_n)$. Bryant [15] proved that any OBDD representation of this function requires at least $\Omega(1.14^n)$ vertices, yet there exists an alternative implementation of $f_{\mathrm{hw}}(x_1, \ldots, x_n)$ with area-time complexity of $O(n^{1+\epsilon})$. We point out here that this alternative implementation is essentially a factor-graph implementation [49]. We refrain from pursuing this any further in this paper. However, we believe that this line of research holds great promise.

5.2. From binary decision diagrams to trellises

Since 1986, when ordered binary decision diagrams were introduced for verification problems [14], many refinements and variations of the basic data structures and algorithms have been proposed. Here, we discuss how these and other results pertaining to binary decision diagrams may be applied to trellises.

Almost all codes have exponential trellis complexity. It is known [39, 58, 77] that almost all n-variable Boolean functions cannot be represented by an OBDD with less than $2^n/2n$ vertices, regardless of the variable ordering. More precisely, Liaw and Lin [58] establish* the following result. Let $\omega(n) = 2^{2^n}$ be the total number of n-variable Boolean functions, or equivalently binary codes of length n, and let $\gamma(n)$ denote the number of n-variable functions whose OBDD, under optimal variable order, has less than $2^n/2n$ vertices. It is shown in [58] that

$$\lim_{n\to\infty} \frac{\gamma(n)}{\omega(n)} \;=\; 0 \tag{7}$$

We know from Theorem 1 that the minimal proper trellis for $\mathbb{C}$ has at least as many vertices as the OBDD for $f_{\mathbb{C}}$. Thus the result of (7) transfers directly to minimal proper trellises. This was not previously known in the trellis literature. It is known that *all* asymptotically good codes have exponential trellis complexity [53], and almost all *linear* codes are asymptotically good [72, p.77]. However, it is not difficult to see that almost all nonlinear codes are *not* asymptotically good.

Liaw and Lin [58] also consider *quasi-reduced* OBDDs, obtained by applying only the merging rule (step **2** in Construction A) and not the redundant-tests deletion rule (step **3** in Construction A). It is obvious from Theorem 1 that a quasi-reduced OBDD for a function f is precisely the minimal proper trellis for $\mathbb{C}_f$, whether $d(\mathbb{C}_f) > 1$ or $d(\mathbb{C}_f) = 1$. Asymptotically as $n \to \infty$, Liaw and Lin [58] observe that for virtually all Boolean functions, the merging rule contributes a factor of $1/n$ to the overall reduction in the size of the OBDD, whereas the redundant-test deletion rule contributes only a constant factor. For fixed n, Liaw and Lin [58] find empirically that the merging rule alone accounts for over 99% of the average reduction in the size of the OBDD, whenever $n \geq 15$. They thus suggest that under certain circumstances, it is more advantageous to use quasi-reduced OBDDs (namely, trellises!), since then the level-index field can be eliminated from the vertex record, resulting in more significant savings in the overall storage space than those obtained by the redundant-tests deletion rule.

Liaw and Lin [58] also show that for *all* n-variable Boolean functions, the quasi-reduced OBDD has at most $(2+\epsilon)(2^n/n)$ vertices for all sufficiently large n, regardless of the variable ordering. Clearly, this bound transfers directly to trellises. An intersting conclusion from this result, in conjunction with (7), is that the complexity of the minimal proper trellis for almost all binary codes is not sensitive to permutations of the time axis: the trellis has at least $2^n/2n$ vertices for the best possible permutation, and at most 4 times as many vertices for the worst possible permutation. This insensitivity phenomenon, well-known [39, 58, 77] in the OBDD literature, was not previously observed for trellises.

*A similar result was established by Shannon [68] in the context of two-terminal contact networks.

Multi-terminal trellises/syndrome decision diagrams. Multi-terminal binary decision diagrams [23] are extensions of BDDs for representing functions $f : \{0,1\}^n \mapsto \mathcal{S}$, where $\mathcal{S}$ is any finite set. A multi-terminal BDD differs from a conventional OBDD only in that it may have multiple terminals, rather than two terminals labeled 0 and 1.

The notion of multi-terminal BDDs can be exploited to construct a *multi-terminal trellis* that simultaneously represents a binary linear code $\mathbb{C}$ as well as *all the cosets* of $\mathbb{C}$, in $\mathbb{F}_2^n$ or in a given subspace of $\mathbb{F}_2^n$. Multi-terminal trellises were used by Ytrehus [41, 80] to represent the *parallel branch codes* encountered in the decoding of partial unit memory convolutional codes. In general, such trellises are useful whenever one needs to decode a partition of a given space into cosets of a given code. This task is at the core of the coset-decoding technique [25] and is frequently encountered in multilevel coding [34, 44].

Another application of multi-terminal trellises is as an attractive alternative to the well-known *standard array decoding* technique, which we now briefly describe. Let $\mathbb{C}$ be an (n,k,d) binary linear code, and let $H = [h_1, \ldots, h_n]$ be a parity-check matrix for $\mathbb{C}$. The *standard array* for $\mathbb{C}$ is the $2^{n-k} \times 2^k$ matrix with entries from $\mathbb{F}_2^n$, having the cosets of $\mathbb{C}$ in $\mathbb{F}_2^n$ as its rows. For each coset, we may pre-compute the *coset leader* v, defined as the vector of minimal Hamming weight in the coset. For each $x \in \mathbb{F}_2^n$, the *syndrome* of x with respect to H is defined as Hx^t. Given the channel output $y \in \mathbb{F}_2^n$, we first compute the syndrome $s = Hy^t$, and then decode y to $\widehat{c} = y - v \in \mathbb{C}$, where v is the coset leader of the coset consisting of all the vectors whose syndrome with respect to H is s. This procedure, known as standard array decoding and illustrated in Figure 7, achieves hard-decision maximum-likelihood decoding of $\mathbb{C}$ on a binary symmetric channel.

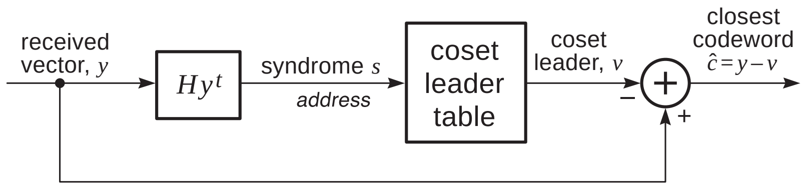


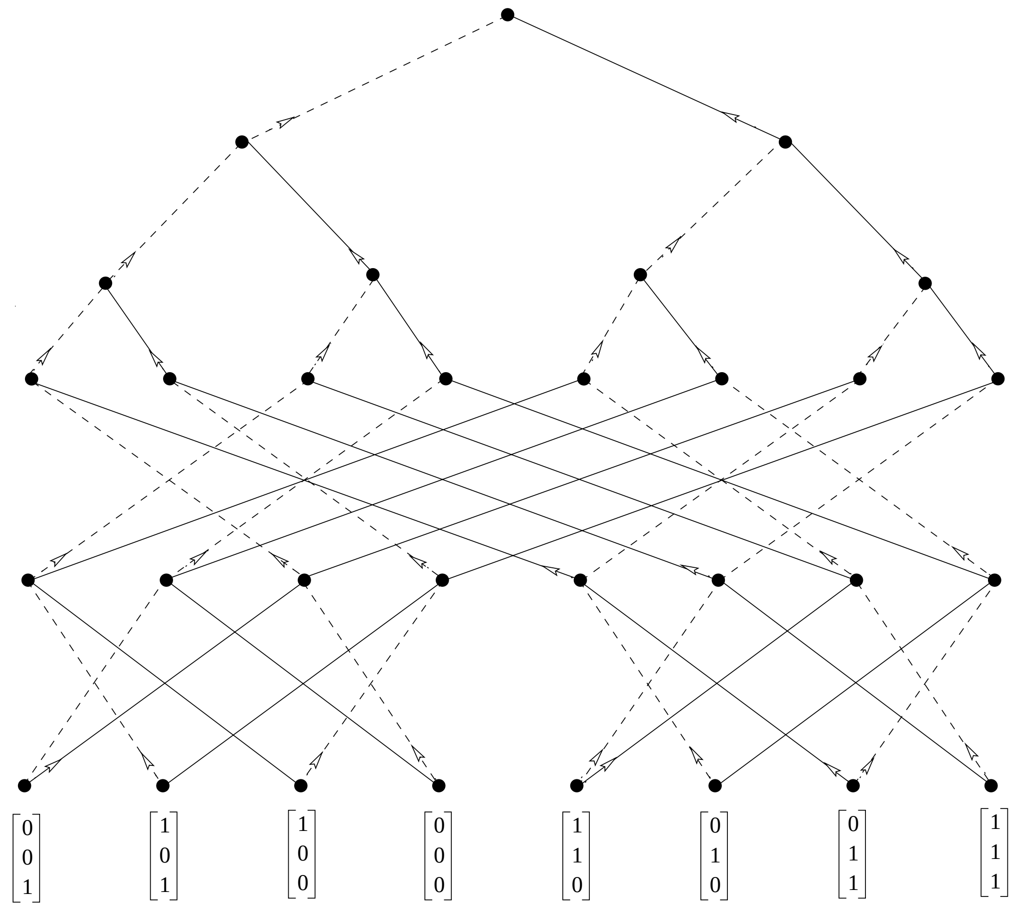
Figure 7. *Standard array decoding*

With a multi-terminal trellis, we can represent the standard array compactly, avoid the brute-force enumeration of cosets, and obtain a linear time procedure for both syndrome computation and decoding. The idea is to construct a multi-terminal BDD for the function $h_{\mathbb{C}}(x_1, \ldots, x_n) = Hx^t$, using a procedure analogous to the BCJR construction [5]. In addition, we will carry out dynamic programming during the construction to label each vertex by the minimum weight path that leads to it.

The BCJR trellis $T = (V, E, \mathbb{F}_2)$ for a linear code $\mathbb{C}$ of length n is constructed [5] by identifying vertices with partial codeword syndromes:

$$V_i \overset{\text{def}}{=} \Big\{ c_1h_1 + \cdots + c_ih_i \ : \ (c_1, \ldots, c_n) \in \mathbb{C} \text{ for some } c_{i+1}, \ldots, c_n \in \mathbb{F}_2 \Big\} \qquad (8)$$

with $V_0 = \mathbf{0}$ by convention. There is an edge $e \in E_i$ from $v \in V_{i-1}$ to $u \in V_i$ if and only if there exists a codeword $(c_1, c_2, \ldots, c_n) \in \mathbb{C}$ such that $c_1h_1 + \cdots + c_{i-1}h_{i-1} = v$ and



$\begin{bmatrix}0\\0\\1\end{bmatrix}$
$\begin{bmatrix}1\\0\\1\end{bmatrix}$
$\begin{bmatrix}1\\0\\0\end{bmatrix}$
$\begin{bmatrix}0\\0\\0\end{bmatrix}$
$\begin{bmatrix}1\\1\\0\end{bmatrix}$
$\begin{bmatrix}0\\1\\0\end{bmatrix}$
$\begin{bmatrix}0\\1\\1\end{bmatrix}$
$\begin{bmatrix}1\\1\\1\end{bmatrix}$

Figure 8. *The syndrome decision diagram for* $\mathbb{C} = \{00000, 11010, 01101, 10111\}$

The terminal corresponding to $y \in \mathbb{F}_2^n$ is labeled by the syndrome Hy^t, and the arrows indicate paths to be taken to obtain a minimum weight error vector. The syndrome may be calculated by trickling the received vector down the diagram. For example, if the vector $y = (10011)$ is received, the corresponding path from the root ends in the terminal labeled $Hy^t = (100)^t$. Following the backpointers from this terminal, the error vector is determined to be (00100).

$c_1h_1 + \cdots + c_{i-1}h_{i-1} + c_ih_i = u$. The label of this edge is $\alpha(e) = c_i$. The multi-terminal trellis $\mathcal{D} = (V', E', \mathbb{F}_2)$ may be constructed in a manner analogous to (8) by computing partial syndromes for *all* vectors in $\mathbb{F}_2^n$, not just the codewords of $\mathbb{C}$. Thus, we define

$$V_i' \overset{\text{def}}{=} \left\{ x_1h_1 + \cdots + x_ih_i \ : \ (x_1, \ldots, x_n) \in \mathbb{F}_2^n \text{ for some } x_{i+1}, \ldots, x_n \in \mathbb{F}_2 \right\} \qquad (9)$$

with $V_0 = \mathbf{0}$ by convention. There is an edge $e \in E_i'$ from $v \in V_{i-1}'$ to $u \in V_i'$ with label $\alpha(e) = x \in \mathbb{F}_2$ if and only if $u = v + xh_i$. It is easy to see that the resulting graph $\mathcal{D}$ is the multi-terminal BDD for the function $h_{\mathbb{C}}(x_1, \ldots, x_n) = Hx^t$. Note that the minimal trellis for $\mathbb{C}$ is contained in $\mathcal{D}$ as a proper subgraph. Also notice that by replacing $\mathbb{F}_2^n$ in (9) by an arbitrary subspace $\mathcal{S}$ such that $\mathbb{C} \subseteq \mathcal{S} \subseteq \mathbb{F}_2^n$, we obtain the multi-terminal trellis that represents the cosets of $\mathbb{C}$ in that subspace.

By carrying out a simple dynamic programming algorithm on $\mathcal{D}$ during its construction, maintaining for each vertex the minimum weight path reaching that vertex and a corresponding pointer back to the previous level, we can determine the minimum weight path to every vertex in $\mathcal{D}$, and therefore to every syndrome. The straightforward details are omitted. The resulting data structure, which we call the *syndrome decision diagram* for $\mathbb{C}$, is illustrated in Figure 8 for the $(5,2,3)$ linear code $\mathbb{C} = \{00000, 11010, 01101, 10111\}$.

Given a syndrome decision diagram $\mathcal{D}$, a maximum-likelihood decoder for $\mathbb{C}$ can be implemented as follows. First we evaluate the received vector y in $\mathcal{D}$, thus computing the function $h_{\mathbb{C}}(y_1, \ldots, y_n) = Hy^t$ which gives the syndrome of y, and then trace back from the corresponding terminal to find a coset leader v in the coset of y.

The standard-array decoding procedure, illustrated in Figure 7, has space complexity $O(2^{n-k})$ and decoding complexity $O(n^2)$, since the computation of the syndrome $s = Hy^t$ is in general quadratic in the block length. Construction of the syndrome decision diagram requires $O(n2^{n-k})$ space and time complexity. However, once the diagram is available, both syndrome computation and decoding can be accomplished in linear time.

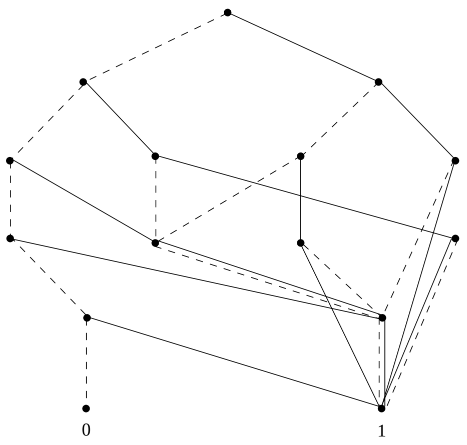
In general, as a computational device that computes syndromes in linear time, syndrome decision diagrams would be useful in many different contexts in coding theory.

Reed-Muller expansions and OFDDs. One approach to obtaining more compact representations of Boolean functions has been to change the interpretation of the vertices within the data structure. As discussed in Section 2, OBDDs represent the Shannon expansion (1) of a Boolean function. An alternative expansion of a Boolean function can be expressed in terms of the EXCLUSIVE-OR operation:

$$f \;=\; f_{\overline{x}} \oplus x \cdot f_{\delta x} \;=\; f_x \oplus \overline{x} \cdot f_{\delta x} \tag{10}$$

where $f_{\delta x} = f_x \oplus f_{\overline{x}}$ is the *Boolean difference* of f with respect to x. The first equality in (10) is known as either the *Reed-Muller* or the *negative Davio* expansion, while the second equality is referred to as the *positive Davio* expansion. These decompositions are analogous to the Taylor expansion of a differentiable function.

The Reed-Muller expansion can be used [46] as the basis for graphical representations called *ordered functional decision diagrams*. This representation is analogous to OBDDs, except that the outgoing edges from a vertex represent the negative cofactor and Boolean difference of the function with respect to the vertex variable. Ordered functional decision diagrams (OFDDs) have many properties in common with OBDDs. For example, the representation is canonical, and can be constructed using a similar algorithm for merging and eliminating vertices, with a different reduction rule for removing redundant tests. The OFDD for our example code $\mathbb{C} = \{00000, 11010, 01101, 10111\}$ is shown in Figure 9.



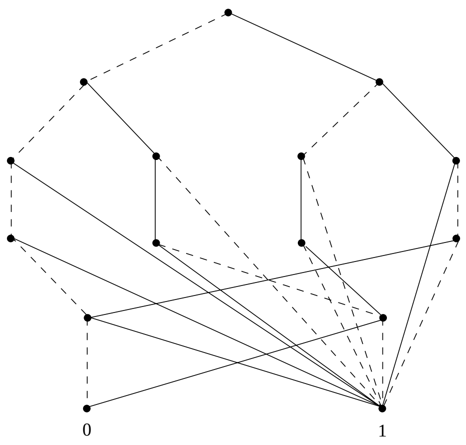


Figure 9. *The OFDD and the OBDD for* $\mathbb{C} = \{00000, 11010, 01101, 10111\}$

For some classes of functions, OFDDs are exponentially smaller than the corresponding OBDDs, although the reverse can also hold. One interesting direction for future research would be to explore an application of OFDDs for channel coding, by investigating decoding algorithms based on this representation.

One useful decoding algorithm for trellises is the *forward-backward* algorithm, also called the BCJR algorithm after the authors of [5], who first developed this algorithm in the trellis context. The forward-backward algorithm is widely used in practice, for example in the decoding of turbo codes [8], to obtain maximum a posteriori likelihood (MAP) decoding of each code symbol. The complexity of this algorithm is polynomial in the size of the trellis, or equivalently the size of the single-terminal OBDD for the code. However, we will now show that it is unlikely that the calculations required in the forward-backward algorithm can be carried out efficiently using the OFDD representation.

The forward-backward algorithm assumes knowledge of the channel transition probability function $p(y|x)$, where $x \in \mathbb{C}$ is the channel input and y is the vector observed at the channel output. For a given binary code $\mathbb{C}$, the algorithm effectively computes

$$\mathcal{S}_0(x_i) \overset{\text{def}}{=} \sum_{\substack{x\in\mathbb{C}\\ x_i=0}} p(y|x) \qquad \text{and} \qquad \mathcal{S}_1(x_i) \overset{\text{def}}{=} \sum_{\substack{x\in\mathbb{C}\\ x_i=1}} p(y|x) \tag{11}$$

for all $i = 1, 2, \ldots, n$, and decodes the i-th code bit x_i to either 0 or 1, according as $\mathcal{S}_0(x_i) > \mathcal{S}_1(x_i)$ or $\mathcal{S}_1(x_i) > \mathcal{S}_0(x_i)$. Although the formulation of the forward-backward algorithm in (11) is general, we will restrict our attention to the simplest possible channel model: the binary symmetric channel with cross-over probability θ. Thus the channel output is binary, and the transition probability function is given by:

$$p_\theta(y|x) = \theta^{d(x,y)} (1-\theta)^{n-d(x,y)} \tag{12}$$

where $d(x,y)$ is the Hamming distance and $\theta \in [0,1]$ is a real constant. The decoding algorithm that we seek must work for any θ, thought of as a parameter of the algorithm.

Proposition 4. *Let $\mathbb{C}$ be an arbitrary binary block code, and let $\mathcal{F}$ be an OFDD for $\mathbb{C}$. Then there is no polynomial-time algorithm in the size of $\mathcal{F}$ for computing the expressions $\mathcal{S}_0(x_i)$ and $\mathcal{S}_1(x_i)$ in* (11) *for the function $p_\theta(y|x)$ in* (12), *unless* P = NP.

Proof. The key idea of the proof is to observe that on a binary symmetric channel with $\theta = 0.5$, the forward-backward algorithm simply *counts* the number of codewords that have 0, respectively 1, in the specified position. Indeed, for $\theta = 0.5$, we have

$$\mathcal{S}_0(x_i) + \mathcal{S}_1(x_i) \;=\; \sum_{\substack{x\in\mathbb{C}\\ x_i=0}} (0.5)^{d(x,y)}\,(0.5)^{n-d(x,y)} \;+\; \sum_{\substack{x\in\mathbb{C}\\ x_i=1}} (0.5)^{d(x,y)}\,(0.5)^{n-d(x,y)} \;=\; \frac{|\mathbb{C}|}{2^n}$$

Thus, as a special case, such an algorithm could be used to compute the size of the code. It is shown in [78], however, that the problem of computing $|\mathbb{C}_f|$ using the OFDD representation of a Boolean function $f(x_1,\ldots,x_n)$ is #P-complete. ▮

We conclude that OFDDs are not suitable for the kind of calculations required in the forward-backward algorithm, at least for general binary codes. It is still possible that OFDDs can be used efficiently in the context of the forward-backward algorithm in the special case of linear codes. It is also possible that maximum-likelihood decoding, as opposed to symbol-by-symbol MAP decoding, can be implemented efficiently with OFDDs.

Binary moment diagrams. There have been several efforts to extend the concept of BDDs to represent functions over Boolean variables, but having non-Boolean ranges, such as the integers or the real numbers.

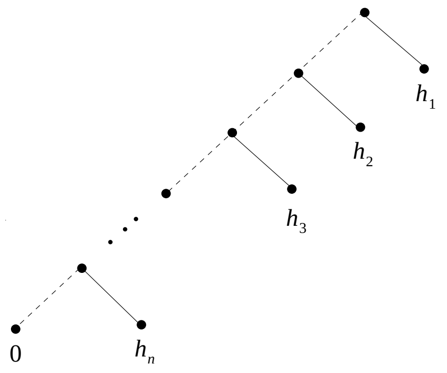


Figure 10. *The BMD for a linear code with parity-check matrix $H = [\,h_1, h_2, \ldots, h_n]$*

One approach to representing numeric functions, especially those encountered in arithmetic circuit verification, involves changing the function decomposition with respect to its variables, in a manner analogous to the use of Reed-Muller expansions for FDDs. The moment decomposition of a function is obtained as

$$f \;=\; f_{\overline{x}} \;+\; x\cdot(f_x - f_{\overline{x}}) \;=\; f_{\overline{x}} \;+\; x\cdot f_{\partial x}$$

where $f_{\partial x} = f_x - f_{\overline{x}}$ is called the *linear moment* of f with respect to x. The resulting representation is known [20] as the *binary moment diagram* or BMD.

Our conclusion with regard to binary moment diagrams is, again, negative. As pointed out to us by Randy Bryant [19], this representation turns out not to be useful for codes. Indeed, let $\mathbb{C}$ be an (n, k, d) linear code with parity-check matrix $H = [h_1, h_2, \ldots, h_n]$. Then the binary moment diagram of the $\mathbb{F}_2^{n-k}$-valued function $h_{\mathbb{C}}(x_1, \ldots, x_n) = Hx^t$ is the tree shown in Figure 10. Using the fact that the BMD representation is canonical [20], this statement follows by a simple induction on the block length of the code.

6. Conclusions and discussion

We have established a correspondence between ordered binary decision diagrams and minimal trellises, proving that the single-terminal OBDD for a binary code $\mathbb{C}$, viewed as a Boolean function, is isomorphic to the minimal proper trellis for $\mathbb{C}$, provided $d(\mathbb{C}) > 1$.

Although we have emphasized the similarities between the two data structures throughout this paper, one should also be aware of the differences between them. It appears that the major distinction between trellises and OBDDs results from the elimination of redundant tests, which does not preserve the depth structure of a trellis. This distinction becomes vacuous if $d(\mathbb{C}) > 1$. The restriction $d(\mathbb{C}) > 1$ does not have much of an impact in coding theory: any useful code will have minimum distance greater than 1. However, there is no reason why the off-set of a useful Boolean function should satisfy this requirement. Thus every reasonable trellis is an OBDD, but not vice versa.

Another significant distinction between the theory of binary decision diagrams and trellis theory stems from a difference in emphasis. While most of the research in channel coding is focused on linear codes, the corresponding class of Boolean functions has not received much attention in logic synthesis and formal verification.

Despite the dissimilarities discussed above, we have demonstrated that the connection between trellises and OBDDs opens up many possibilities for leveraging the extensive work that has been carried out independently in two previously unconnected disciplines. We hope that this paper will stimulate further research in this direction.

Acknowledgment. We would like to thank Randy Bryant for several helpful and informative discussions on binary decision diagrams.

References

[1] S.M. Aji and R.J. McEliece, "The generalized distributive law," *IEEE Trans. Inform. Theory*, submitted for publication, July 1998.

[2] S.B. Akers, "Binary decision diagrams," *IEEE Trans. Computers*, vol. 27, pp. 509–516, August 1978.

[3] D.P. Appenzeller and A. Kuehlmann, "Formal verification of a PowerPC microprocessor," in *Proc. Int. Conference on Computer-Aided Design* (ICCAD), pp. 79–84, San Jose, CA, November 1995.

[4] R.I. Bahar, E.A. Frohm, C.M. Gaona, G.D. Hachtel, E. Macii, A. Prado, and F. Somenzi, "Algebraic decision diagrams and their application," in *Proc. Int. Conf. on Computer-Aided Design* (ICCAD), pp. 188–191, Santa Clara, CA, November 1993.

[5] L.R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284–287, March 1974.

[6] L.R. Bahl, F. Jelinek, and R.L. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 5, pp. 179–190, March 1983.

[7] B. Becker and R. Drechsler, "How many decomposition types do we need?" in *Proc. European Design and Test Conference*, pp. 438–443, Paris, France, March 1995.

[8] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo codes," in *Proc. IEEE Int. Conf. on Communications*, pp. 1064-1070, Geneva, Switzerland, July 1993.

[9] D. Bhattacharya, P. Agrawal, and V.D. Agrawal, "Delay fault test generation for scan/hold circuits using boolean expressions," in *Proc. Design Automation Conference* (DAC), pp. 159–164, Anaheim, CA, June 1992.

[10] B. Bollig, M. Löbbing, M. Sauerhoff, and I. Wegener, "Complexity theoretical aspects of OFDDs," in REPRESENTATIONS OF DISCRETE FUNCTIONS, T. Sasao and M. Fujita (Eds.), Boston: Kluwer Academic, 1996.

[11] B. Bollig and I. Wegener, "Improving the variable ordering of OBDDs is NP-complete," *IEEE Trans. Computers*, vol. 45, pp. 993–1002, September 1996.

[12] K.S. Brace, R.L. Rudell, and R.E. Bryant, "Efficient implementation of a BDD package," in *Proc. Design Automation Conf.* (DAC), pp. 40–45, Orlando, FL, June 1990.

[13] F.M. Brown, *Boolean Reasoning*, Boston: Kluwer Academic Publishers, 1990.

[14] R.E. Bryant, "Graph-based algorithms for boolean function manipulations," *IEEE Trans. Computers*, vol. 35, pp. 677–691, August 1986.

[15] R.E. Bryant, "On the complexity of VLSI implementations and graph representations of Boolean functions with application to integer multiplication," *IEEE Trans. Computers*, vol. 40, pp. 205–213, February 1991.

[16] R.E. Bryant, "Symbolic boolean manipulation with ordered binary decision diagrams," *ACM Computing Surveys*, vol. 24, pp. 293–318, September 1992.

[17] R.E. Bryant, "Division, Pentium style: An analysis of Intel's mistake(s)," CMU Computer Systems Seminar, February 22, 1995.

[18] R.E. Bryant, "Binary decision diagrams and beyond: Enabling technologies for formal verification," in *Proc. Int. Conference on Computer-Aided Design* (ICCAD), pp. 236–243, San Jose, CA, November 1995.

[19] R.E. Bryant, personal communication, November 1997.

[20] R.E. Bryant and Y.-A. Chen, "Verification of arithmetic functions with binary moment diagrams," in *Proc. Design Automation Conference* (DAC), pp. 535–541, San Francisco, CA, June 1995.

[21] J.R. Burch, E.M. Clarke, D.L. Dill, and K. McMillan, "Sequential circuit verification using symbolic model checking," in *Proc. Design Automation Conference* (DAC), pp. 46–51, Orlando, FL, June 1990.

[22] A.R. Calderbank, G.D. Forney, Jr., and A. Vardy, "Minimal tail-biting trellises: the Golay code and more," *IEEE Trans. Inform. Theory*, submitted for publication.

[23] E. Clarke, M. Fujita, P. McGeer, K. McMillan, J. Yang, and X. Zhao, "Multi-terminal binary decision diagrams: An efficient data structure for matrix representation," in *Proc. Int. Workshop on Logic Synth.*, pp. P6a:1–15, Louvain, Belgium, July 1993.

[24] E.M. Clarke, M. Fujita, and X. Zhao, "Hybrid decision diagrams — overcoming the limitations of MTBDDs and BMDs," in *Proc. Int. Conference on Computer-Aided Design* (ICCAD), pp. 159–163, San Jose, CA, November 1995.

[25] J.H. Conway and N.J.A. Sloane, "Soft decoding techniques for codes and lattices, including the Golay code and the Leech lattice," *IEEE Trans. Inform. Theory*, vol. 32, pp. 41–50, January 1986.

[26] O. Coudert, C. Berthet, and J.C. Madre, "Verification of sequential machines based on symbolic execution," in *Automatic Verification Methods for Finite State Systems*, Lect. Notes Computer Science, vol. 407, pp. 365–373, Berlin: Springer-Verlag 1989.

[27] R. Drechsler, B. Becker, and S. Ruppertz, "K*BMDs: A new data structure for verification," in *Proc. European Design and Test Conference*, Paris, March 1996.

[28] R. Drechsler, A. Sarabi, M. Theobald, B. Becker, and M.A. Perkowski, "Efficient representation and manipulation of switching functions based on Ordered Kronecker Functional Decision Diagrams," in *Proc. Design Automation Conference* (DAC), pp. 415–419, San Diego, CA, June 1994.

[29] J.D. Ferguson (Editor), *Proc. Symp. Application of Hidden Markov Models to Text and Speech*, Institute for Defense Analyses, Princeton, NJ, 1980.

[30] G.D. Forney, Jr., "Final report on a coding system design for advanced solar missions," Contract NAS2–3637, NASA Ames Research Center, CA, December 1967.

[31] G.D. Forney, Jr., "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference," *IEEE Trans. Inform. Theory*, vol. 18, pp. 363–378, May 1972.

[32] G.D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, 1973.

[33] G.D. Forney, Jr., "Dimension/length profiles and trellis complexity of linear block codes," *IEEE Trans. Inform. Theory*, vol. 40, pp. 1741–1752, November 1994.

[34] G.D. Forney, Jr., "Coset codes II: Binary lattices and related codes," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1152–1187, September 1988.

[35] G.D. Forney, Jr., B.H. Marcus, N.T. Sindhushayana, and M.D. Trott, "Multilingual dictionary: System theory, coding theory, symbolic dynamics and automata theory," in DIFFERENT ASPECTS OF CODING THEORY, A.R. Calderbank (Editor), vol. 50, pp. 109–138, *Proc. Symp. Applied Mathematics*, Providence, RI: AMS Press, 1995.

[36] G.D. Forney, Jr. and M.D. Trott, "The dynamics of group codes: state spaces, trellises and canonical encoders," *IEEE Trans. Inform. Theory*, vol. 39, pp. 1491–1513, September 1993.

[37] B.J. Frey, *Graphical Models for Machine Learning and Digital Communication*, Cambridge, MA: MIT Press, 1998.

[38] B.J. Frey, F.R. Kschischang, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, submitted for publication, 1998.

[39] C. Gröpl, H.J. Prömel, and A. Srivastav, "Size and structure of random OBDDs," preprint, June 1997.

[40] A. Hett, R. Drechsler, and B. Becker, "MORE: Alternative implementation of BDD-packages by multi-operand synthesis," in *Proc. European Design and Test Conference*, Paris, France, March 1996.

[41] M.F. Hole and Ø. Ytrehus, "Two-step trellis decoding of partial unit memory convolutional codes," *IEEE Trans. Inform. Theory*, vol. 43, pp. 324–325, January 1997.

[42] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Reading, MA: Addison-Wesley 1979.

[43] G. Horn and F.R. Kschischang, "On the intractability of permuting a block code to minimize trellis complexity," *IEEE Trans. Inform. Theory*, vol. 42, pp. 2042–2048, November 1996.

[44] H. Imai and S. Hirakawa, "A new multilevel coding method using error-correcting codes," *IEEE Trans. Inform. Theory*, vol. 23, pp. 371–377, 1977.

[45] K. Jain, I. Măndoiu, and V.V. Vazirani, "The "art of trellis decoding" is computationally hard — for large fields," *IEEE Trans. Inform. Theory*, vol. 44, pp. 1211–1214, May 1998.

[46] U. Kebschull, E. Schubert, and W. Rosentiel, "Multilevel logic based on functional decision diagrams," in *Proc. European Design and Test Conference*, pp. 43–47, Paris, France, March 1992.

[47] A.B. Kiely, S. Dolinar, R.J. McEliece, L. Ekroot, and W. Lin, "Trellis decoding complexity of linear block codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1687–1697, November 1996.

[48] R. Kötter and A. Vardy, "Construction of minimal tail-biting trellises," in *Proc. IEEE Int. Workshop on Inform. Theory*, pp. 73–75, Killarney, Ireland, June 1998.

[49] R. Kötter and A. Vardy, "Minimality of factor graphs," in *Proc. Math. Theory Networks and Syst. Conference* (MTNS), Padova, Italy, July 1998.

[50] R. Kötter and A. Vardy, "Factor graphs: classification, constructions, and bounds," manuscript in preparation, 1998.

[51] F.R. Kschischang, "The trellis structure of maximal fixed-cost codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1828–1838, November 1996.

[52] F.R. Kschischang and V. Sorokine, "On the trellis structure of block codes," *IEEE Trans. Inform. Theory*, vol. 41, pp. 1924–1937, November 1995.

[53] A. Lafourcade and A. Vardy, "Asymptotically good codes have infinite trellis complexity," *IEEE Trans. Inform. Theory*, vol. 41, pp. 555–559, March 1995.

[54] A. Lafourcade and A. Vardy, "Lower bounds on trellis complexity of block codes," *IEEE Trans. Inform. Theory*, vol. 41, pp. 1938–1954, November 1995.

[55] A. Lafourcade and A. Vardy, "Optimal sectionalization of a trellis," *IEEE Trans. Inform. Theory*, vol. 42, pp. 689–703, May 1996.

[56] Y.-T. Lai and S. Sastry, "Edge-valued binary decision diagrams for multi-level hierarchical verification," in *Proc. Design Automation Conference* (DAC), pp. 608–613, Anaheim, CA, June 1992.

[57] C. Lee, "Representation of switching circuits by binary decision programs," *Bell Syst. Tech. J.*, vol. 38, pp. 985–999, July 1959.

[58] H.T. Liaw and C.S. Lin, "On the OBDD-representation of general Boolean functions," *IEEE Trans. Computers*, vol. 41, pp. 661–664, July 1992.

[59] D.J.C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, January 1999, to appear.

[60] J.C. Madre and O. Coudert, “A logically complete reasoning maintenance system based on a logical constraint solver,” in *Proc. Int. Joint Conference on Artificial Intelligence*, pp. 294–299, Sydney, Australia, August 1991.

[61] J.L. Massey, “Foundation and methods of channel encoding,” *Proc. Int. Conf. Information Theory and Systems*, vol. 65, pp. 148–157, NTG-Fachberichte, Berlin, 1978.

[62] R.J. McEliece, “On the BCJR trellis for linear block codes,” *IEEE Trans. Inform. Theory*, vol. 42, pp. 1072–1092, July 1996.

[63] D.J. Muder, “Minimal trellises for block codes,” *IEEE Trans. Inform. Theory*, vol. 34, pp. 1049–1053, September 1988.

[64] R.K. Ranjan, J.V. Sanghavi, R.K. Brayton, and A. Sangiovanni-Vincentelli, “High performance BDD package based on exploiting memory hierarchy,” in *Proc. Design Automation Conference* (DAC), Las Vegas, NV, June 1996.

[65] I. Reuven and Y. Be’ery, “Entropy/length profiles, bounds on the minimal covering of bipartite graphs, and trellis complexity of nonlinear codes,” *IEEE Trans. Inform. Theory*, vol. 44, pp. 580–598, March 1998.

[66] R.L. Rudell, “Dynamic variable ordering for ordered binary decision diagrams,” in *Proc. Int. Conf. on Computer-Aided Design* (ICCAD), pp. 42–47, November 1993.

[67] H. Sato, Y. Yasue, Y. Matsunaga, and M. Fujita, “Boolean resubstitution with permissible functions and binary decision diagrams,” in *Proc. Design Automation Conference* (DAC), pp. 284–289, Orlando, FL, June 1990.

[68] C.E. Shannon, “The synthesis of two-terminal switching circuits,” *Bell Syst. Tech. J.*, vol. 28, pp. 59–98, January 1949.

[69] V.R. Sidorenko, I. Martin, and B. Honary “On the rectangularity of nonlinear block codes,” *IEEE Trans. Inform. Theory*, vol. 45, March 1999, to appear.

[70] M. Sipser and D.A. Spielman, “Expander codes,” *IEEE Trans. Inform. Theory*, vol. 42, pp. 1710–1722, November 1996.

[71] R.M. Tanner, “A recursive approach to low-complexity codes,” *IEEE Trans. Inform. Theory*, vol. 27, pp. 533–547, September 1981.

[72] M.A. Tsfasman and S.G. Vlăduts, *Algebraic-Geometric Codes*, Dordrecht: Kluwer Academic Publishers 1991.

[73] A. Vardy, “Trellis complexity and soft-decision decoding: Conventional and unconventional results on conventional trellises,” in *Proc. IEEE Int. Workshop on Inform. Theory*, pp. 67–69, Svalbard, Norway, July 1997.

[74] A. Vardy, “The intractability of computing the minimum distance of a code,” *IEEE Trans. Inform. Theory*, vol. 43, pp. 1757–1766, November 1997.

[75] A. Vardy, “Trellis structure of codes,” to appear in Handbook of Coding Theory, V.S. Pless and W.C. Huffman (Editors), Elsevier, Amsterdam, 1998.

[76] A. Vardy and F.R. Kschischang, “Proof of a conjecture of McEliece regarding the expansion index of the minimal trellis,” *IEEE Trans. Inform. Theory*, vol. 42, pp. 2027–2033, November 1996.

[77] I. Wegener, “The size of reduced OBDDs and optimal read-once branching programs for almost all Boolean functions,” *IEEE Trans. Computers* vol. 43, pp. 1262–1269, December 1994; see also “Graph-theoretic concepts in computer science,” *Lecture Notes Comput. Science*, vol. 790, pp. 252–263, Berlin: Springer-Verlag 1994.

[78] R. Wercherner, R. Harich, R. Drechsler, and B. Becker, “Satisfiability problems for OFDDs,” in Representations of Discrete Functions, T. Sasao and M. Fujita (Eds.), Boston: Kluwer Academic, 1996.

[79] N. Wiberg, H.-A. Loeliger and R. Kötter, “Codes and iterative decoding on general graphs,” *Euro. Trans. Telecommun.*, vol. 6, pp. 513–526, May 1995.

[80] Ø. Ytrehus, “Trellis complexity and generalized Hamming weights,” in *Proc. IEEE Int. Workshop on Inform. Theory*, pp. 71–72, Svalbard, Norway, July 1997.