Minimal Trellis Modules and Equivalent Convolutional Codes

Hung-Hua Tang, Mao-Chao Lin, *Member, IEEE*, and Bartolomeu F. Uchôa Filho, *Member, IEEE*

***Abstract*—In this correspondence, it is shown that some convolutional codes with distinct memory sizes of minimal encoders are equivalent in the sense that the minimal trellises of these codes are the shifted versions of one another. For an (n, k) binary convolutional code, the weight spectrum obtained from the minimal trellis may be slightly different from that obtained from the conventional code trellis with n-bit branches. Code search is conducted to find some good $(n, n-1)$ binary convolutional codes. Bounds on the trellis complexity, measured by the number of states and the number of branches in the minimal trellis module, of any convolutional code and its equivalent codes are also derived.**

***Index Terms*—Error-correction coding, convolutional codes, minimal trellis, minimal trellis module, trellis complexity.**

I. Introduction

In most papers in the literature [1]–[3], for the trellis representation of a binary (n, k) convolutional code, the transition between two state nodes is labeled by an n-bit code branch or multiple n-bit code branches, and the number of branches emanating from each node or entering each node is 2^k. Hence, the associated decoding complexity is high for large k. In such a trellis, each n-bit code branch is associated with a k-bit message. In communication systems for which high rate convolutional codes are required, punctured convolutional codes [4]–[7] are usually considered to reduce the trellis complexity. Suppose that a binary (n, k) convolutional code is obtained by puncturing a $(2, 1)$ binary convolutional code. In the associated trellis, each code branch corresponds to one message bit and is labeled by one or two code bits. The trellis of a binary convolutional code can be refined such that the transition between two state nodes (at depth j and depth $j+1$) respectively is labeled by a single code bit in a way similar to the trellis of a binary block code. For the linear block code, Forney introduced the minimal trellis construction, which minimizes the number of vertices (state nodes) at each depth [8][9]. For a binary convolutional code, its minimal trellis can be similarly constructed [10][11], where each one-bit code branch corresponds to one message bit in some sections and no message bit in other sections.

The complexity of the minimal trellis of a linear block code can be described by its state complexity profile. In [8], it has been shown that the state complexity profile of a linear block code is identical to that of its dual code. In [12], similar property has been derived for convolutional codes, i.e., the state complexity profiles of a convolutional code and the reciprocal of its dual code are identical if minimal encoders for both codes are used, where the state complexity profile of a convolutional code is defined over a minimal trellis module, which is a portion (or a period) of the minimal trellis. For an (n, k, ν) binary convolutional code C, its state complexity profile is the n-tuple $(s_0, s_1, \cdots, s_{n-1})$ [12], where s_j is the dimension of state space at depth j of the minimal trellis module of C.

In this correspondence, we will prove that taking a cyclic-shifted version of a minimal trellis module (or distinct period of a minimal trellis) of a convolutional code C will result in an equivalent convolutional code which has the same weight spectra as C but may have a distinct memory size of the minimal encoder. For convolutional codes which are equivalent in the sense that the associated minimal trellis modules are cyclic-shifted versions of one another, the associated state complexity profiles are also cyclic-shifted versions of one another.

For a binary $(n, n-1, \nu)$ code, it [12] can be shown that the component of its state complexity profile is either ν or $\nu + b$, where b is either -1 or 1. All the components with values of $\nu + b$ occur consecutively. The sum of the components of the state complexity profile is $\sum_{j=0}^{n-1} s_j = n\nu - \chi$, where $\chi \in \{-n+1, -n+2, \cdots, 0, \cdots, n-2, n-1\}$, $\chi = -b|\chi|$, and $|\chi|$ is the number of components with values of $\nu + b$. The specified χ is zero if all the components of the state complexity profile are ν. It [12] can be derived that the trellis complexity of a binary $(n, n-1)$ convolutional code is affected by its memory size ν and a parameter χ if the complexity of the minimal trellis is measured by the number of states or the number of branches. The trellis complexity of a binary $(n, n-1, \nu)$ punctured convolutional code obtained from a binary $(2, 1, \nu)$ mother code is identical to that of a binary $(n, n-1, \nu)$ convolutional code with $\chi = -1$, if the complexity is measured by the number of branches. In [12], some good binary $(n, n-1)$ convolutional codes of $\chi \geq -1$ are listed, where some of the codes with $\chi = -1$ are slightly better than the best known punctured convolutional codes [4]–[6] of identical rate and complexity by better weight spectra. Graell i Amat, *et al.* [13], [14] proposed a method to construct recursive systematic $(n, n-1)$ convolutional codes, where for some codes improved free distances are found as compared to the best known convolutional codes of identical rates and identical memory sizes of encoders. In this correspondence, we extend the search of good binary $(n, n-1)$ convolutional codes to codes with χ smaller than -1. We find that the best codes in [13], [14] have weight spectra almost the same as the searched codes with $\chi = -n+1$. There is an interesting phenomenon, which shows that many good binary $(n, n-1, \nu)$ codes with $\chi < 0$ have weight spectra almost the same as the good $(n, n-1, \nu+1)$ codes with $\chi + n$. However, this phenomenon can be easily explained by the equivalence of binary convolutional codes for which the minimal trellis modules are cyclic shift versions of one another.

The new equivalence condition for binary convolutional codes indicates that the conventional impression [3] of characterizing the complexity of a convolutional code by the memory size of its encoder may be insufficient in case the minimal trellis is used for decoding. For the $(n, n-1, \nu)$ binary convolutional code, it is already known [12] that the trellis complexity measured by the number of states and the number of branches is determined by ν and χ. In this correspondence, we derive bounds on the trellis complexity of the general (n, k, ν) convolutional codes and the equivalent codes based on the minimal trellises.

This correspondence is organized as follows. In Section II, some basics of convolutional codes are given. In Section III, we show that taking different portions of a minimal trellis of a convolutional code as minimal trellis modules can yield equivalent codes. In Section IV, the conditions of message mapping of equivalent codes are studied. Some extended results on the search for good $(n, n-1)$ convolutional codes are presented in Section V. In Section VI, bounds on the complexity

Manuscript received June 17, 2004; revised March 27, 2006. This work was supported by National Science Council of Taiwan, R.O.C. under Grants NSC 92-2213-E262-013 and NSC 91-2213-E002-100, and by CNPq (Brazil) under Grant 302286/2004-7. The material in this paper was presented in part at the International Symposium on Information Theory and Its Applications, Parma, Italy, October 10–13, 2004.

H.-H. Tang is with the Department of Electrical Engineering, Lunghwa University of Science and Technology, Taoyuan 333, Taiwan, R.O.C. (e-mail: tang@mail.lhu.edu.tw).

M.-C. Lin is with the Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: mclin@cc.ee.ntu.edu.tw).

B. F. Uchôa-Filho is with the Communications Research Group, Department of Electrical Engineering, Federal University of Santa Catarina, Florianopolis, SC, Brazil, CEP 88040-900 (e-mail: uchoa@eel.ufsc.br).

Communicated by Ø. Ytrehus, Associate Editor for Coding Techniques.

Digital Object Identifier 10.1109/TIT.2006.878225

of convolutional codes are studied. Concluding Remarks are given in Section VII.

II. Preliminaries

An (n,k) binary convolutional code C can be characterized by a $k\times n$ transform domain generator matrix [3] $G(D)=[g_{ij}(D)]$, where $g_{ij}(D)$ is a polynomial in D over $GF(2), 0\le i\le k-1, 0\le j\le n-1$. Let $g_i(D)=(g_{i0}(D),g_{i1}(D),\ldots,g_{i\,n-1}(D))=\sum_{l=0}^{m_i}\underline{g}_i^l D^l$, where $\underline{g}_i^l$ is an n-tuple over $GF(2)$ and $m_i=\max_j\{\deg g_{ij}(D)\}$. Write $G(D)$ as $G(D)=\sum_{l=0}^{m}G_lD^l$, where G_l is a $k\times n$ matrix over $GF(2)$ and $m=\max_i\{m_i\}$. The convolutional code C generated by $G(D)$ can be viewed as a block code over $\mathrm{GF}(2)$ of semi-infinite length that has a generator matrix G_{scalar} over $\mathrm{GF}(2)$, where

$$G_{\mathrm{scalar}}=\begin{pmatrix} G_0 & G_1 & \cdots & G_m & & & \\ & G_0 & G_1 & \cdots & G_m & & \\ & & G_0 & G_1 & \cdots & G_m & \\ & & & & \ddots & & \end{pmatrix}. \tag{1}$$

Let $\mathbf{x}=\{x_0,x_1,x_2,\cdots\}$ be a nonzero sequence. Its *left index*, denoted $L(\mathbf{x})$, is the smallest index j such that $x_j\ne 0$. Similarly, the *right index* of $\mathbf{x}$ (if it exists), denoted $R(\mathbf{x})$ is the largest index j such that $x_j\ne 0$. A nonzero sequence $\mathbf{x}$ is said to be active at depth j if both $j-1$ and j are in the interval of $[L(\mathbf{x}),R(\mathbf{x})]$. A generator matrix of a linear code is said to be a *minimal span generator matrix* (MSGM) [15] if for any two distinct rows $\mathbf{x_p}$ and $\mathbf{x_q}$ of it, we have $L(\mathbf{x_p})\ne L(\mathbf{x_q})$, and $R(\mathbf{x_p})\ne R(\mathbf{x_q})$. If the G_{scalar} is an MSGM, we can obtain the minimal trellis for C [10]. Beginning from depth mn, the minimal trellis is periodic. A period of the trellis is called a minimal trellis module.

Let V_0 be the space generated by G_0 and V_{end} be the space generated by G_{end} which is defined as $[(\underline{g}_0^{m_0})^T\cdots(\underline{g}_{k-1}^{m_{k-1}})^T]^T$, where A^T is the transpose of A. Note that $\underline{g}_i^0$ is the leftmost nonzero n-bit block and $\underline{g}_i^{m_i}$ is the rightmost nonzero n-bit block of the i-th row of G_{scalar}, where we assume $\underline{g}_i^0$ is nonzero for each i. Hence, G_0 contains the information of the left index of each row of G_{scalar} and G_{end} contains the information of the right index of each row of G_{scalar}. Let $J=\{0,1,\ldots,n-1\}$ be an *index set*. Define [16] $j^-=\{0,\ldots,j-1\}, j^+=\{j,\ldots,n-1\}$ and $0^+=n^-=J$ and $0^-=n^+=\phi$, the empty set. Let V_{0,j^+} be the subspace of V_0 consisting of all the vectors of V_0 for which the components with indices outside j^+ are zero. Let V_{end,j^-} be the subspace of V_{end} consisting of all the vectors of V_{end} for which the components with indices outside j^- are zero. Since $G_0=[(\underline{g}_0^0)^T\cdots(\underline{g}_{k-1}^0)^T]^T$, there are $\dim(V_{0,j^+})$ rows of G_0 which are inactive at depth j. Similarly, there are $\dim(V_{\mathrm{end},j^-})$ rows of G_{end} which are inactive at depth j. The dimension of state space at depth j for a minimal trellis module of an (n,k,ν) code C is

$$s_j=k+\nu-\dim(V_{0,j^+})-\dim(V_{\mathrm{end},j^-}). \tag{2}$$

The state complexity profile of the convolutional code C is defined as $(s_0,s_1,\cdots,s_{n-1})$ [12].

For the convolutional code C, its trellis complexity can be evaluated by its state complexity profile and its error performance can be evaluated by the code weight spectrum $\{t_w\}$ and information weight spectrum $\{f_w\}$ [3][17], where t_w is the total number of code sequences with weight $d_{\mathrm{free}}+w-1$, f_w is the total number of information bits associated to the code sequences with weight $d_{\mathrm{free}}+w-1$ and d_{free} is the free distance of the code. Suppose that the code is applied over a symmetric and memoryless channel and maximum likelihood decoding is used. The first event error probability of the coding system can be estimated by $P_f\le\sum_{w=1}^{\infty}t_wP_e(d_{\mathrm{free}}+w-1)$ and the symbol error probability can be estimated by $P_s\le(1/k)\sum_{w=1}^{\infty}f_wP_e(d_{\mathrm{free}}+w-1)$, where $P_e(d)$ is the probability of erroneously decoding a code sequence $\mathbf{c}$ into another code sequence $\mathbf{c}'$ which is separated from $\mathbf{c}$ by a distance of d.

III. Equivalent Convolutional Codes

There exist in the literature various definitions of equivalence between two (n,k) binary convolutional codes. As an example [2], a convolutional code C' is said to be equivalent to C if the transform domain generator matrix $G'(D)$ of C' is obtained by applying row operations on the transform domain generator matrix $G(D)$ of C. Row operations on $G(D)$ will affect the message mapping. Hence, the two equivalent codes C and C' have the same code weight spectra but possibly different information weight spectra. As another example, a convolutional code C'' is said to be equivalent to C if the n code bits of each code branch of C'' are obtained by a certain permutation of the n code bits of the corresponding code branch of C. The two equivalent codes C and C'' have the same information weight spectra and the same code weight spectra. In [12], the permutation of the n code bits of each code branch is used to find an equivalent code with minimum trellis of the least complexity for the (n,k) binary convolutional code.

In this section, we will introduce a new form of equivalence. A binary convolutional code C''' is equivalent to C in the sense that the minimal trellis module of C''' is obtained from cyclicly shifting the minimal trellis module of C. We first consider a heuristic example and then consider the general case. Consider three $(5,3)$ binary convolutional codes, all of which have the same free distance $d_{\mathrm{free}}=4$. The associated minimal encoders are shown respectively as follows.

C_I:

$$G_I(D)=\begin{pmatrix}1&1&1&1&0\\0&3&3&0&1\\0&2&1&3&2\end{pmatrix}. \tag{3}$$

C_{II}:

$$G_{II}(D)=\begin{pmatrix}3&3&0&1&0\\2&1&3&2&0\\2&2&2&0&1\end{pmatrix}. \tag{4}$$

C_{III}:

$$G_{III}(D)=\begin{pmatrix}1&3&2&0&1\\2&2&0&1&1\\6&0&2&0&3\end{pmatrix}. \tag{5}$$

Each entry of $G_I(D), G_{II}(D)$ and $G_{III}(D)$ is represented in octal form. For example, "3" represents $1+D$ and "6" represents $D+D^2$. The code weight spectrum and information weight spectrum of C_I are $1,12,32,68,172,488,1364,\ldots$, and $1,32,144,424,1264,4116,13224,\ldots$, respectively. For C_{II}, the spectra are $1,12,32,68,173,506,1484,\ldots$, and $1,32,144,424,1266,4185,13916,\ldots$, respectively. For C_{III}, the spectra are $1,12,32,68,173,508,1512,\ldots$, and $1,32,144,424,1266,4190,14030,\ldots$, respectively. The memory sizes of the encoders of C_I, C_{II}, and C_{III} are $2,3$, and 4, respectively. The state complexity profiles of C_I, C_{II}, and C_{III} are $(2,3,4,4,3)$, $(3,4,4,3,2)$, and $(4,4,3,2,3)$, respectively. Although these three codes have similar weight spectra, the trellis complexities based on their trellises with n-bit code branches are significantly different. However, we notice that there is a cyclic relation among the state complexity profiles and hence there should be some relationship among these codes.

We now examine the decoding complexities in detail through their minimal trellises. A portion of the minimal trellis of C_I, which is the cascade of two minimal trellis modules, is shown in Fig. 1, where the

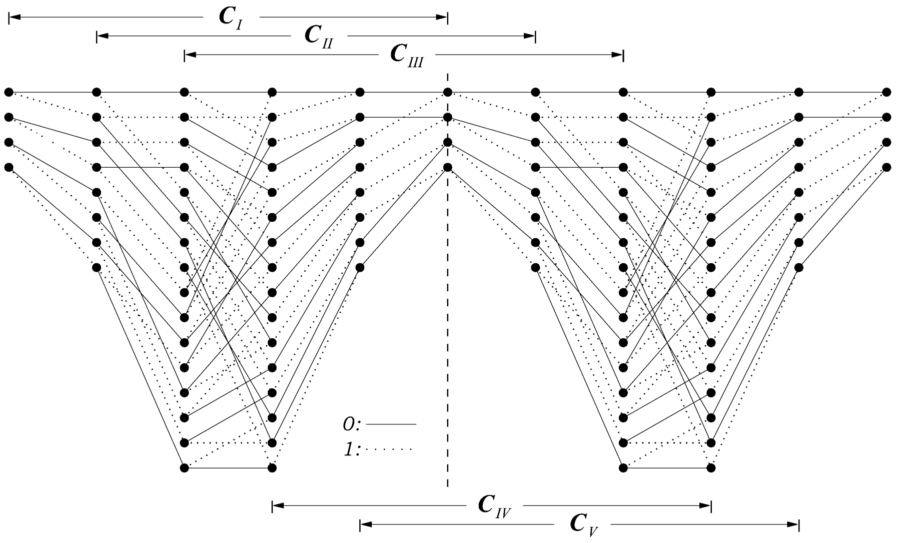

Fig. 1. Two consecutive minimal trellis modules of C_I (and the minimal trellis modules of C_{II}, C_{III}, C_{IV}, and C_V.)

junction of two modules is marked by a vertical dashed line. This trellis can be easily obtained from the generator matrix of C_I, i.e.,

$$G_{I,\mathrm{scalar}} = \begin{pmatrix} 1&1&1&1&0&&&&&&&&&&& \\ 0&1&1&0&1&0&1&1&0&0&&&&&& \\ 0&0&1&1&0&0&1&0&1&1&&&&&& \\ &&&&&1&1&1&1&0&&&&&& \\ &&&&&0&1&1&0&1&0&1&1&0&0& \\ &&&&&0&0&1&1&0&0&1&0&1&1& \\ &&&&&&&&&&&&&&&\ddots \end{pmatrix}. \tag{6}$$

Suppose that the columns are numbered by integers $0, 1, 2, \ldots$ from the leftmost column and the rows are numbered by integers $0, 1, 2, \ldots$ from the top. The row numbered 0 $(\mathrm{mod} 3)$ of $G_{I,\mathrm{scalar}}$ has a leading "1" at position 0 $(\mathrm{mod} 5)$ that implies two branches emanating from each node at depth 0 $(\mathrm{mod} 5)$ of the trellis, and has a trailing "1" at position 3 $(\mathrm{mod} 5)$ that implies two branches merging to each node at depth 4 $(=3+1)$ $(\mathrm{mod} 5)$ of the trellis. Similarly, the row numbered 1 $(\mathrm{mod} 3)$ of $G_{I,\mathrm{scalar}}$ has a leading "1" at position 1 $(\mathrm{mod} 5)$ that implies two branches emanating from each node at depth 1 $(\mathrm{mod} 5)$ of the trellis, and has a trailing "1" at position 2 $(\mathrm{mod} 5)$ that implies two branches merging to each node at depth 3 $(\mathrm{mod} 5)$ of the trellis. The row numbered 2 $(\mathrm{mod} 3)$ of $G_{I,\mathrm{scalar}}$ has a leading "1" at position 2 $(\mathrm{mod} 5)$ that implies two branches emanating from each node at depth 2 $(\mathrm{mod} 5)$ of the trellis, and has a trailing "1" at position 4 $(\mathrm{mod} 5)$ that implies two branches merging to each node at depth 0 $(\mathrm{mod} 5)$ of the trellis.

Delete the first row (numbered 0) and the first column (numbered 0) of $G_{I,\mathrm{scalar}}$. We have $G_{II,\mathrm{scalar}}$ for a code $C_{II,\mathrm{scalar}}$, where we have

$$G_{II,\mathrm{scalar}} = \begin{pmatrix} 1&1&0&1&0&1&1&0&0&0&&&&&& \\ 0&1&1&0&0&1&0&1&1&0&&&&&& \\ 0&0&0&0&1&1&1&1&0&0&&&&&& \\ &&&&&1&1&0&1&0&1&1&0&0&0& \\ &&&&&0&1&1&0&0&1&0&1&1&0& \\ &&&&&0&0&0&0&1&1&1&1&0&0& \\ &&&&&&&&&&&&&&&\ddots \end{pmatrix}. \tag{7}$$

As indicated in Fig. 1, the minimal trellis module of C_{II} is obtained from cyclicly shifting the minimal trellis module of C_I by one section (or one depth). Delete the first two rows (numbered 0 and 1) and the first two columns (numbered 0 and 1) of $G_{I,\mathrm{scalar}}$. We have $G_{III,\mathrm{scalar}}$ for a code C_{III}, for which its minimal trellis module is obtained by cyclicly shifting the minimal trellis module of C_I by two sections (or two depths) as shown in Fig. 1.

We now see that the minimal trellises of C_I, C_{II}, and C_{III} are identical except for the shifting of one or two sections. Then, we have an interesting result which indicates that convolutional codes with distinct memory sizes of encoders may be equivalent in the sense that they have minimal trellises, each of which is a shifted version of one another. Moreover, the cyclic-shift relation of the minimal trellis modules of these equivalent codes implies the cyclic-shift relation of the associated state complexity profiles.

According to the trellises with n-bit code branches, we obtain the code weight spectra and information weight spectra of C_I, C_{II}, and C_{III} which are listed right after (5). Although these spectra are very close, there are still noticeable differences among them. Hence, we would like to know what factor results in the differences among these equivalent codes.

Conventionally [3], [17], the weight spectra of an (n, k, ν) convolutional code is calculated based on a code trellis of 2^ν states and the transition from one state to a following state is represented by a branch or multiple branches of n code bits. The value t_w of the code weight spectrum of a convolutional code can be computed by the number of code paths of weight $d_{\mathrm{free}} + w - 1$ departing from the zero state of the code trellis at depth 0 and returning to the zero state for the first time. The value f_w of the information weight spectrum of a convolutional code can be computed by the number of message bits on all the code paths of weight $d_{\mathrm{free}} + w - 1$ departing from the zero state of the code trellis at depth 0 and returning to the zero state for the first time.

Since C_I, C_{II}, and C_{III} are equivalent in the sense that the minimal trellises of C_{II} and C_{III} can be obtained from that of C_I by shifting one or two bits (sections), we may consider calculating the weight spectra using the minimal trellis. The number t_w is the sum of the numbers of code paths of weight $d_{\mathrm{free}} + w - 1$ departing from the zero states at depths $0, 1, \ldots, n-1$ of the minimal trellis and returning to the zero state for the first time. The number f_w is similarly modified. The weight spectra derived by using the trellis with n-bit code branches will be lower bounded by those derived by the minimal trellis, since in the trellis with n-bit code branches, a path that returns to the zero state at depth $pn + j_1, 0 < j_1 \leq n - 1$ and leaves the zero state at depth $pn + j_2, 0 < j_1 \leq j_2 \leq n - 1$ will not be considered as returning to the zero state at the time interval of $[pn, (p+1)n - 1]$. With the minimal trellis, we find that the weight spectra of all the three equivalent codes are identical. The code weight spectra and information weight spectra are $1, 12, 32, 68, 172, 488, 1364, \ldots,$ and $1, 32, 144, 424, 1264, 4116, 13224, \ldots,$ respectively.

The weight spectra obtained from the minimal trellis and the weight spectra obtained from the trellis with n-bit branches are the same for the first few terms and are only slightly different for the other terms. In fact, for $w \in \{1, 2, \cdots, d_{\text{free}}\}, t_w$ and f_w obtained from the conventional trellis are identical to those obtained from the minimal trellis, since breaking up a zero-state-to-zero-state path of weight less than $2d_{\text{free}}$ into two zero-state-to-zero-state paths would imply that there exists a zero-state-to-zero-state path with weight less than d_{free}. In case of maximum likelihood decoding such as Viterbi decoding, using the conventional trellis with n-bit branches for decoding will yield the same bit error rate as that obtained from using the minimal trellis for decoding. Note that the zero states in the minimal trellis are different from the zero states in the trellis with n-bit branches. Hence, using the minimal trellis in the Viterbi decoding, some paths will be eliminated earlier as compared to using the trellis with n-bit branches. The first error events in the minimal trellis are in general different from those in the trellis with n-bit branches. Therefore, there will be a little difference between the first error event probability of decoding using the minimal trellis and that of decoding using the trellis with n-bit branches.

In our illustrating examples, the generator matrices of C_I, C_{II}, and C_{III} are all in row echelon form. Suppose that we start with a code $C_{I'}$ with transform domain generator matrix

$$G_{I'}(D) = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 2 & 1 & 3 & 2 \\ 0 & 3 & 3 & 0 & 1 \end{pmatrix} \tag{8}$$

for which its generator matrix

$$G_{I',\text{scalar}} = \begin{pmatrix} 1&1&1&1&0&&&&&&&&&& \\ 0&0&1&1&0&0&1&0&1&1&&&&& \\ 0&1&1&0&1&0&1&1&0&0&&&&& \\ &&&&&1&1&1&1&0&&&&& \\ &&&&&0&0&1&1&0&0&1&0&1&1 \\ &&&&&0&1&1&0&1&0&1&1&0&0 \\ &&&&&&&&&&&&&&\ddots \end{pmatrix} \tag{9}$$

is not in row echelon form. We need to apply row operations on the generator matrix of $G_{I'}$ in advance to obtain $G_{I,\text{scalar}}$, that is in row echelon form, and then apply the illustrated procedure of finding other equivalent codes with cyclicly shifted minimal trellis modules.

In general, the equivalent convolutional codes derived from shifting the minimal trellis of a convolutional code C can be obtained as follows. Suppose C has its G_{scalar} as shown in (1), which is in MSGM form. Permute the first k rows of G_{scalar} so that the first k rows of the resultant matrix is in row echelon form. Then, we permute every following k rows in the same way to obtain G'_{scalar}, which is in row echelon form. Let j_i be the location of the leading "1" of the row numbered i, where $0 \leq i \leq k-1$ and $0 \leq j_i \leq n-1$. For $0 \leq i \leq k-1$, delete the first i rows of G'_{scalar} and the first j_i columns of G'_{scalar}. We then have $G_{\text{eq},i,\text{scalar}}, 0 \leq i \leq k-1$, which is the generator matrix of an equivalent code $C_{eq,i}$. The minimal trellis of $C_{eq,i}$ is obtained from that of C by cyclicly shifting j_i bits (sections). In this way, we can have k equivalent convolutional codes, i.e., $C_{\text{eq},0}, C_{\text{eq},1}, \cdots, C_{\text{eq},k-1}$. In each of these equivalent codes, the associated minimal trellis has two branches emanating from each state at depth $0 \pmod{n}$. By now only k possible cyclic shifts of the minimal trellis module of C are used. In fact, the n possible cyclic shifts of the minimal trellis module will imply n equivalent convolutional codes. Suppose that $j_{i+1} - j_i \geq 2$ for any $i \in \{0, \ldots, k-1\}$, for which $j_k = j_0 + n$. We already have $G_{\text{eq},i,\text{scalar}}$. For each $\ell \in \{1, 2, \cdots, j_{i+1} - j_i - 1\}$, deleting the first row and the first ℓ columns of $G_{\text{eq},i,\text{scalar}}$ yields a generator matrix of an equivalent code. In this way, we can obtain generator matrices of $C_{\text{eq},k+z}, 0 \leq z \leq n-k-1$, which is equivalent to C. For $C_{\text{eq},k+z}, 0 \leq z \leq n-k-1$, the associated minimal trellis has only one branch emanating from each state at depth $0 \pmod{n}$. For the (5, 3) codes illustrated at the beginning of this section, we have $C_{\text{eq},0} = C_I, C_{\text{eq},1}, = C_{II}$ and $C_{\text{eq},2} = C_{III}$. There are two more equivalent codes, i.e., $C_{\text{eq},3} = C_{IV}$ and $C_{\text{eq},4} = C_V$, where C_{IV}

$$G_{IV}(D) = \begin{pmatrix} 2 & 0 & 1 & 1 & 1 \\ 0 & 2 & 0 & 3 & 3 \\ 6 & 4 & 0 & 2 & 1 \end{pmatrix} \tag{10}$$

and C_V

$$G_V(D) = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 2 & 0 & 3 & 3 & 0 \\ 4 & 0 & 2 & 1 & 3 \end{pmatrix}. \tag{11}$$

IV. Message Mapping of Equivalent Codes

Let C be an (n, k) convolutional code with an MSGM G_{scalar}. Consider a subset of C which consists of all the code sequences of C for which the first j code bits are zero, where $0 \leq j < n$. By deleting the first j code bits in each sequence of such a subset, we have a convolutional code equivalent to C. We call it an equivalent code of the jth class. The minimal trellis module of the equivalent code of the jth class is related to the minimal trellis module of C by cyclicly shifting j sections.

Suppose that $C = C_I$ is the code with $G_{\text{scalar}} = G_{I,\text{scalar}}$ given in (6). Then, C_I and $C_{I'}$ (with $G_{I'}(D)$ shown in (8)) are both equivalent codes of the 0th class. All the codes with transform domain generator matrices obtained by permuting the rows of $G_I(D)$ are equivalent codes of the 0th class. The subset of C consisting of code sequences, each of which has the first code bit equal to zero, is the set generated by the matrix obtained by deleting the first row of (6). Deleting the first trivial code bit in each code sequence is equivalent to deleting the first column of the above mentioned matrix. The resultant generator matrix is $G_{II,\text{scalar}}$ given in (7). All the codes with transform domain generator matrices obtained by permuting the rows of $G_{II}(D)$ are equivalent codes of the first class. Similarly, all the codes with transform domain generator matrices obtained by permuting the rows of $G_{III}(D), G_{IV}(D)$, and $G_V(D)$, respectively, are equivalent codes of the second class, the third class and the fourth class, respectively.

Equivalent codes of the jth class of C have generator matrices, each of which is a row-permuted version of one another. The permutation of rows of a generator matrix will result in a distinct message mapping of the convolutional code. Moreover, in Fig. 1, we see that there are two branches emanating from each node at depths 0, 1, and 2 among the five depth positions 0, 1, 2, 3, and 4 of the trellis module. It means that, for each message, its three message bits respectively map at three of the five depth positions of the trellis modules. In general, for an (n, k) binary convolutional code, the k message bits of each message, respectively, map at k of the n depth positions of the trellis modules. We will now show that we can construct equivalent codes for which the message mapping can also be arranged in a cyclic-shift form. We note that the set of left indices of code sequences under modulo-n arithmetic is of size k, where the k elements are also those left indices of the first k rows of the generator matrix. To describe the message mapping, we consider an ordered set. Let $L_0 = \{\ell_{0,0}, \ell_{1,0}, \cdots, \ell_{k-1,0}\}$ be the ordered set, where $\ell_{i,0}$ is the left index of the row numbered $i \pmod k$ of G_{scalar}. The message mapping of an equivalent code of the jth class denoted by $L_j = \{\ell_{0,j}, \ell_{1,j}, \cdots, \ell_{k-1,j}\}, 0 \le j < n$, can be systematically characterized using the following rule.

1) For $j = 0$, let $L_0 = \{\ell_{0,0}, \ell_{1,0}, \cdots, \ell_{k-1,0}\}$.
2) For $0 < j < n$, we have $L_j = \{\ell_{0,j}, \ell_{1,j}, \cdots, \ell_{k-1,j}\}$ such that $\ell_{i,j} = (\ell_{i,j-1} - 1) \pmod n$, where $0 \le i < k$.

The codes $C_I, C_{II}, C_{III}, C_{IV}$, and C_V with encoders $G_I, G_{II}, G_{III}, G_{IV}$, and G_V given in Section III have $L_0 = \{0,1,2\}, L_1 = \{4,0,1\}, L_2 = \{3,4,0\}, L_3 = \{2,3,4\}$, and $L_4 = \{1,2,3\}$ respectively.

If we consider the code of the 0th class with transform domain generator matrix $G_{I'}$ given in (8), we have $L_0 = \{0,2,1\}, L_1 = \{4,1,0\}, L_2 = \{3,0,4\}, L_3 = \{2,4,3\}$, and $L_4 = \{1,3,2\}$ respectively for codes $C_{I'}, C_{II'}, C_{III'}, C_{IV'}$ and $C_{V'}$ with encoders $G_{I'}, G_{II'}, G_{III'}, G_{IV'}$, and $G_{V'}$, respectively, where

$$G_{II'}(D) = \begin{pmatrix} 2 & 2 & 2 & 0 & 1 \\ 2 & 1 & 3 & 2 & 0 \\ 3 & 3 & 0 & 1 & 0 \end{pmatrix} \tag{12}$$

$$G_{III'}(D) = \begin{pmatrix} 2 & 2 & 0 & 1 & 1 \\ 1 & 3 & 2 & 0 & 1 \\ 6 & 0 & 2 & 0 & 3 \end{pmatrix} \tag{13}$$

$$G_{IV'}(D) = \begin{pmatrix} 2 & 0 & 1 & 1 & 1 \\ 6 & 4 & 0 & 2 & 1 \\ 0 & 2 & 0 & 3 & 3 \end{pmatrix} \tag{14}$$

and

$$G_{V'}(D) = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 \\ 4 & 0 & 2 & 1 & 3 \\ 2 & 0 & 3 & 3 & 0 \end{pmatrix}. \tag{15}$$

V. Good $(n, n-1)$ Convolutional Codes

For C, there exists a dual code $C^\perp$. Let $G(D)$ and $H(D)$ be minimal encoders for C and $C^\perp$ respectively. The product of $G(D)$ and $H(D)$ is a zero matrix. A code $\hat{C}$ is said to be the reciprocal code of C if there is a minimal encoder $\hat{G}(D)$ for $\hat{C}$ that is realized by the generators $\hat{g}_i(D) = \sum_{l=0}^{m_i} \underline{g}_i^{m_i - l} D^l, i = 0, 1, \cdots, k-1$. It can be shown that the state complexity profiles of C and its reciprocal dual $\hat{C}^\perp$ are identical [12]. For an $(n, n-1)$ binary convolutional code with memory size ν, we have $(s_0, s_1, \cdots, s_{n-1}, s_0) = (\nu, \cdots, \nu, \nu + b, \ldots, \nu + b, \nu, \cdots, \nu)$, where the number of consecutive $(\nu + b)$'s is $|\chi|, b \in \{1, -1\}$, and

$$\sum_{j=0}^{n-1} s_j = n\nu - \chi \tag{16}$$

$\chi \in \{-n+1, -n+2, \cdots, 0, \cdots, n-1\}$. The decoding complexity of an (n, k) convolutional code C can be evaluated by the number of states in each minimal trellis module, $\varphi(C)$ and the number of branches in each minimal trellis module, $\psi(C)$. For an $(n, n-1)$ binary convolutional code C with memory size ν, we have

$$\varphi(C) = \begin{cases} n \cdot 2^\nu - \chi \cdot 2^{\nu-1}, & \chi \ge 0 \\ (n - \chi) \cdot 2^\nu, & \chi < 0 \end{cases} \tag{17}$$

and

$$\psi(C) = \begin{cases} (n - \chi) \cdot 2^{\nu+1} + (\chi - 1) \cdot 2^\nu, & \chi \ge 0 \\ (n - \chi - 1) \cdot 2^{\nu+1}, & \chi < 0. \end{cases} \tag{18}$$

For $\chi = -1$, the decoding complexity of C measured by the number of branches will be identical to that of the $(n, n-1)$ punctured code. Based on the simple structure of the $(n, 1)$ convolutional code, many good $(n, n-1)$ convolutional codes were found in [12]. In [12], only codes with $\chi \ge -1$ were searched. Note that a larger χ value will result in lower trellis complexity as (17), (18) show.

In [13], $(7, 6)$ and $(8, 7)$ convolutional codes in recursive form with weight spectra significantly better than those found in [12] were reported. It can be checked that the reciprocal dual codes of the $(7, 6)$ and $(8, 7)$ codes in [13] have χ values of -6 and -7 respectively. By observing this result, we begin to search for good $(n, n-1)$ convolutional codes with $\chi < -1$. The term "good code" is used in this correspondence for three reasons. For the first, the code search is incomplete. We believe the searched code is close to the best one although there may exist codes with better spectra. For the second, there may be two codes for which one has better code weight spectrum $\{t_w\}$ while the other has better information weight spectrum $\{f_w\}$. For the third, the trend is that the currently best $(n, n-1, \nu)$ code with a smaller χ tends to have spectra better than the currently best $(n, n-1, \nu)$ code with a larger χ, although there are some cases of exception. Due to space limitation, we list only some $(3, 2)$ codes and some $(4, 3)$ codes in Tables I and II respectively. In the tables, an entry $g_{ij}(D)$ of each $G(D)$ is represented in octal form.

There is an interesting phenomenon that the spectra of a good $(n, n-1, \nu, \chi)$ code with $\chi < 0$, are very close to those of a good $(n, n-1, \nu+1, \chi+n)$ code. For example, in Table II, the weight spectra of the $(4, 3, \nu = 3, \chi = -3)$ code are the same as those of the $(4, 3, \nu = 4, \chi = 1)$ code. By substituting $\nu + 1$ for ν and substituting $\chi + n$ for χ in the upper part of (17) (or (18)), we can have the lower part of (17) (or (18)). This explains the phenomenon. In the following, we give more details.

In [12], it has been proved that for a binary $(n, n-1, \nu, \chi)$ convolutional code using minimal trellis module, we have either $(s_0, s_1, \cdots, s_{n-1}, s_0) = (\nu, \nu, \cdots, \nu, \nu+1, \cdots, \nu+1, \nu, \cdots, \nu)$ or $(s_0, s_1, \cdots, s_{n-1}, s_0) = (\nu, \nu, \ldots, \nu, \nu-1, \cdots, \nu-1, \nu, \cdots, \nu)$, where the number of consecutive $(\nu+1)$'s or consecutive $(\nu-1)$'s is $|\chi|$. Note that $(s_0, s_1, \cdots, s_{n-1})$ is the state complexity profile. For example, the state complexity profiles of the $(4, 3)$ codes with $\nu = 4$ in Table II, are $(4, 3, 3, 3)$ for $\chi = 3$, $(4, 3, 3, 4)$ for $\chi = 2$, $(4, 4, 3, 4)$ for $\chi = 1$, $(4, 4, 4, 4)$ for $\chi = 0$, $(4, 5, 4, 4)$ for $\chi = -1$, $(4, 5, 5, 4)$ for $\chi = -2$ and $(4, 5, 5, 5)$ for $\chi = -3$, respectively.

For an $(n, n-1, \nu, \chi)$ convolutional code C with $(s_0, s_1, \ldots, s_{n-1}, s_0) = (\nu, \nu, \cdots, \nu, \nu+1, \ldots, \nu+1, \nu, \ldots, \nu)$, we see that the number of states in the minimal trellis module is $\varphi(C) = n \cdot 2^\nu + (-\chi) \cdot 2^\nu = (n - \chi) \cdot 2^\nu$, where χ is a negative integer and $|\chi| = -\chi$ is the number of consecutive $(\nu+1)$'s. Cyclicly shifting the minimal trellis module can result in an equivalent code C', which is an $(n, n-1, \nu+1, n+\chi)$ code with $(s_0, s_1, \cdots, s_{n-1}, s_0) = (\nu+1, \nu+1, \cdots, \nu+1, \nu, \cdots, \nu, \nu+1, \cdots, \nu+1)$, where the number of consecutive ν's is $n + \chi$. We then have $\varphi(C') = n \cdot 2^{\nu+1} + (-(n+\chi)) \cdot 2^\nu = (n - \chi) \cdot 2^\nu = \varphi(C)$.

For an $(n, n-1, \nu, \chi)$ convolutional code C'' with $(s_0, s_1, \cdots, s_{n-1}, s_0) = (\nu, \nu, \cdots, \nu, \nu-1, \ldots, \nu-1, \nu, \ldots, \nu)$, we have $\varphi(C'') = n \cdot 2^\nu - \chi \cdot 2^{\nu-1} = (2n - \chi) \cdot 2^{\nu-1}$, where χ is a positive integer and is the number of consecutive $(\nu-1)$'s. Cyclicly shifting the minimal trellis module will result in an equivalent code C''', which is an $(n, n-1, \nu-1, n-\chi)$ code with $(s_0, s_1, \ldots, s_{n-1}, s_0) = (\nu-1, \nu-1, \ldots, \nu-1, \nu, \ldots, \nu, \nu-1, \ldots, \nu-1)$, where the number of consecutive ν's is $n - \chi$. Thus, we have $\varphi(C''') = n \cdot 2^{\nu-1} + (n - \chi) \cdot 2^{\nu-1} = (2n - \chi) \cdot 2^{\nu-1} = \varphi(C'')$.

Considering the number of branches in each minimal trellis module, we can also compute that $\psi(C) = \psi(C') = 2 \cdot 2^{\nu+1}(-\chi - 1) + 2^{\nu+1} + 2 \cdot 2^\nu (n + \chi) = 2^{\nu+1}(n - \chi - 1)$ and $\psi(C'') = \psi(C''') = 2 \cdot 2^{\nu-1}\chi + 2^\nu + 2 \cdot 2^\nu (n - \chi - 1) = 2^\nu (2n - \chi - 1)$.

VI. Trellis Complexity

In Section V, we have seen the possible trellis complexity of an $(n, n-1, \nu)$ binary convolutional code. In this section, we will check

TABLE I
GOOD $(3, 2)$ CONVOLUTIONAL CODES

n	ν	$G(D)$	d_{free}	χ	Spectra $\frac{t_1,t_2,\cdots}{f_1,f_2,\cdots}$
3	2	$\begin{pmatrix} 2 & 0 & 1 \\ 2 & 3 & 0 \end{pmatrix}$	2	2	$\frac{1,2,4,8,16,32,64,\cdots}{1,3,12,36,96,240,576,\cdots}$
3	2	$\begin{pmatrix} 0 & 2 & 3 \\ 2 & 1 & 0 \end{pmatrix}$	2	1	$\frac{1,2,4,10,20,40,80,\cdots}{1,3,12,41,112,284,688,\cdots}$
3	2	$\begin{pmatrix} 0 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}$	3	0	$\frac{3,7,15,34,76,171,384,\cdots}{5,20,65,191,530,1418,3696,\cdots}$
3	2	$\begin{pmatrix} 0 & 2 & 3 \\ 3 & 3 & 1 \end{pmatrix}$	3	-1	$\frac{1,4,14,40,115,331,953,\cdots}{1,10,54,226,853,3038,10423,\cdots}$
3	2	$\begin{pmatrix} 3 & 2 & 0 \\ 2 & 1 & 3 \end{pmatrix}$	3	-2	$\frac{1,4,14,40,116,339,991,\cdots}{1,10,57,240,911,3275,11366,\cdots}$
3	3	$\begin{pmatrix} 2 & 3 & 0 \\ 6 & 2 & 3 \end{pmatrix}$	3	2	$\frac{1,4,14,40,115,331,953,\cdots}{1,10,54,226,853,3038,10423,\cdots}$
3	3	$\begin{pmatrix} 0 & 3 & 2 \\ 6 & 2 & 1 \end{pmatrix}$	3	1	$\frac{1,4,14,40,116,339,991,\cdots}{1,10,57,240,911,3275,11366,\cdots}$
3	3	$\begin{pmatrix} 2 & 3 & 2 \\ 4 & 4 & 3 \end{pmatrix}$	4	0	$\frac{10,0,86,0,739,0,6425,\cdots}{30,0,507,0,6470,0,74398,\cdots}$
3	3	$\begin{pmatrix} 3 & 3 & 1 \\ 4 & 2 & 5 \end{pmatrix}$	4	-1	$\frac{2,9,28,97,324,1091,3685,\cdots}{6,33,161,723,2976,11951,46726,\cdots}$
3	3	$\begin{pmatrix} 3 & 2 & 1 \\ 2 & 5 & 5 \end{pmatrix}$	4	-2	$\frac{1,5,24,71,238,862,2991,\cdots}{1,11,108,417,1857,7948,32335,\cdots}$
3	4	$\begin{pmatrix} 6 & 1 & 7 \\ 7 & 7 & 2 \end{pmatrix}$	5	-2	$\frac{2,13,45,143,534,2014,7336,\cdots}{5,51,248,1048,4685,20691,86666,\cdots}$
3	5	$\begin{pmatrix} 6 & 7 & 3 \\ 13 & 7 & 14 \end{pmatrix}$	6	-2	$\frac{6,27,70,285,1103,4063,15359,\cdots}{26,129,494,2446,10878,46500,198453,\cdots}$
3	6	$\begin{pmatrix} 6 & 17 & 13 \\ 13 & 13 & 6 \end{pmatrix}$	7	-2	$\frac{17,53,133,569,2327,8624,32412,\cdots}{86,360,1148,5767,27277,114524,481710,\cdots}$
3	7	$\begin{pmatrix} 6 & 13 & 13 \\ 37 & 21 & 16 \end{pmatrix}$	8	-2	$\frac{41,0,528,0,7497,0,111071,\cdots}{234,0,4854,0,93342,0,1741474,\cdots}$
3	8	$\begin{pmatrix} 16 & 37 & 23 \\ 27 & 31 & 6 \end{pmatrix}$	8	-2	$\frac{6,42,153,510,1853,7338,28378,\cdots}{20,284,1312,5164,22192,99382,428364,\cdots}$

the possible trellis complexity of general (n, k, ν) binary convolutional codes based on the minimal trellis.

Let C be an (n, k, ν) binary convolutional code and $\hat{C}^{\perp}$ is its reciprocal dual. In [12], it has been shown that the state complexity profiles of C and $\hat{C}^{\perp}$ are identical if minimal encoders for both codes are used. Using (2) for C, we see that the dimension of state space s_j is lower bounded by $\max\{0, -k+\nu\}$ and is upper bounded by $k+\nu$. Using (2) for $\hat{C}^{\perp}$, we have $\max\{0, -(n-k)+\nu\} \le s_j \le n-k+\nu$. Combining these two inequalities, we have

$$\max\{0, \nu - q\} \le s_j \le \nu + q \tag{19}$$

where

$$q = \min\{k, n-k\}. \tag{20}$$

For brevity, we only consider bounds on the complexity of convolutional codes with $\nu \ge q$.

We first consider the case that the state complexity profile of C provides the largest $\varphi(C)$ and the largest $\psi(C)$. In this case, the set of locations of leading "1" of G_0 (i.e., the set of leftmost indices for rows of G_0) is $\{0, 1, \ldots, k-1\}$ and the set of locations of trailing "1" of G_{end} (i.e., the set of rightmost indices for rows of G_{end}) is $\{n-k, n-k+1, \ldots, n-1\}$. Hence, the first $q+1$ components of the state complexity profile will be $\nu, \nu+1, \ldots, \nu+q-1, \nu+q$ respectively and the last q components of the state complexity profile will be $\nu+q, \nu+q-1, \ldots, \nu+2, \nu+1$ respectively. The state complexity profile will be $(\nu, \nu+1, \ldots, \nu+q, \ldots, \nu+q, \ldots, \nu+2, \nu+1)$. We can derive that

$$\varphi(C) = 2^{\nu}[2^{q}(n-2q+3)-3] \tag{21}$$

TABLE II
GOOD $(4,3)$ CONVOLUTIONAL CODES

n	ν	$G(D)$	d_{free}	χ	Spectra $\frac{t_1,t_2,\cdots}{f_1,f_2,\cdots}$
4	3	$\begin{pmatrix} 2 & 2 & 0 & 3 \\ 0 & 2 & 3 & 1 \\ 3 & 3 & 1 & 1 \end{pmatrix}$	4	-1	$\frac{29,0,532,0,9853,0,182372,\cdots}{124,0,4504,0,124337,0,3059796,\cdots}$
4	3	$\begin{pmatrix} 2 & 2 & 2 & 3 \\ 2 & 1 & 3 & 1 \\ 3 & 2 & 1 & 1 \end{pmatrix}$	4	-2	$\frac{10,42,194,886,4091,18855,86898,\cdots}{32,212,1476,8564,48760,265391,1414806,\cdots}$
4	3	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 2 & 3 & 1 \\ 4 & 5 & 3 & 0 \end{pmatrix}$	4	-3	$\frac{5,36,152,708,3424,16300,77640,\cdots}{9,136,1012,6380,38136,218916,1219708,\cdots}$
4	4	$\begin{pmatrix} 2 & 0 & 3 & 1 \\ 0 & 3 & 1 & 1 \\ 6 & 2 & 2 & 3 \end{pmatrix}$	4	3	$\frac{29,0,532,0,9853,0,182372,\cdots}{124,0,4504,0,124337,0,3059796,\cdots}$
4	4	$\begin{pmatrix} 2 & 2 & 2 & 3 \\ 2 & 3 & 1 & 1 \\ 6 & 2 & 1 & 2 \end{pmatrix}$	4	2	$\frac{10,42,194,886,4091,18855,86898,\cdots}{32,212,1476,8564,48760,265391,1414806,\cdots}$
4	4	$\begin{pmatrix} 2 & 2 & 1 & 1 \\ 3 & 1 & 0 & 1 \\ 6 & 0 & 4 & 5 \end{pmatrix}$	4	1	$\frac{5,36,152,708,3424,16300,77640,\cdots}{9,136,1012,6380,38136,218916,1219708,\cdots}$
4	4	$\begin{pmatrix} 2 & 2 & 2 & 3 \\ 2 & 3 & 1 & 1 \\ 2 & 6 & 7 & 0 \end{pmatrix}$	4	0	$\frac{5,39,151,690,3545,16935,80141,\cdots}{15,216,1188,6964,43347,244875,1339220,\cdots}$
4	4	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 4 & 6 & 1 \\ 4 & 6 & 3 & 0 \end{pmatrix}$	4	-1	$\frac{3,44,160,638,3556,17130,79234,\cdots}{6,296,1354,6891,47092,263191,1396139,\cdots}$
4	4	$\begin{pmatrix} 2 & 1 & 3 & 1 \\ 3 & 1 & 0 & 2 \\ 6 & 6 & 2 & 1 \end{pmatrix}$	4	-2	$\frac{2,22,96,464,2353,11736,58539,\cdots}{3,93,680,4461,28214,170374,996609,\cdots}$
4	4	$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 4 & 6 & 3 & 1 \\ 4 & 3 & 5 & 0 \end{pmatrix}$	4	-3	$\frac{1,16,84,376,1912,9728,48592,\cdots}{1,48,544,3244,20984,128496,755008,\cdots}$
4	5	$\begin{pmatrix} 3 & 3 & 1 & 1 \\ 6 & 4 & 5 & 5 \\ 6 & 1 & 6 & 1 \end{pmatrix}$	5	-3	$\frac{7,45,223,1066,5612,29010,148269,\cdots}{25,270,1854,11190,71024,429927,2522656,\cdots}$

and

$$\psi(C) = 2^{\nu+2}(2^q - 1) + 2^{\nu+q}(n + k - 3q). \quad (22)$$

For an (n,k,ν) binary convolutional code with this highest complexity minimal trellis, the memory sizes of encoders of equivalent codes can be $\nu, \nu+1, \ldots, \nu+q-1$ and $\nu+q$. The code C_I considered in Section III has state complexity profile (2,3,4,4,3), which has the highest state complexity and branch complexity among all the binary (5,3) convolutional codes. The possible memory sizes of encoders of equivalent codes can be $\nu = 2$ (for C_I), $\nu + 1 = 3$ (for C_{II}) and $\nu + 2 = 4$ (for C_{III}), respectively.

We then consider the case that the state complexity profile of C provides the smallest $\varphi(C)$ and the smallest $\psi(C)$. In this case, the set of locations of leading "1" of G_0 is $\{n-k, n-k+1, \ldots, n-1\}$ and the set of locations of trailing "1" of G_{end} is $\{0, 1, 2, \ldots, k-2, k-1\}$. The first $q+1$ components of the state complexity profile will be $\nu, \nu-1, \ldots, \nu-q+1, \nu-q$ respectively and the last q components of the state complexity profile will be $\nu-q, \nu-q+1, \ldots, \nu-2, \nu-1$ respectively. The state complexity profile will be $(\nu, \nu-1, \ldots, \nu-q, \ldots, \nu-q, \ldots, \nu-2, \nu-1)$. We can derive that

$$\varphi(C) = 2^{\nu}[2^{-q}(n - 2q - 3) + 3] \quad (23)$$

and

$$\psi(C) = 2^{\nu-q+2}(2^q - 1) + 2^{\nu-q}(n + k - 3q). \quad (24)$$

For an (n, k, ν) binary convolutional code with this lowest complexity minimal trellis, the memory sizes of encoders of equivalent codes can be $\nu, \nu - 1, \ldots, \nu - q + 1$ and $\nu - q$ respectively. The code C with transform domain generator matrix

$$G(D) = \begin{pmatrix} 0 & 2 & 3 & 0 & 1 \\ 2 & 2 & 0 & 1 & 1 \\ 4 & 0 & 2 & 0 & 3 \end{pmatrix} \quad (25)$$

has state complexity profile $(4, 3, 2, 2, 3)$, which has the lowest state complexity and branch complexity among all the binary $(5, 3)$ convolutional code. The possible memory sizes of encoders of equivalent codes can be $\nu = 4, \nu - 1 = 3$ and $\nu - 2 = 2$ respectively.

For the general (n, k, ν) binary convolutional code, the complexity measured by the number of states in the minimal trellis module is upper bounded by (21) and is lower bounded by (23), while the complexity measured by the number of branches in the minimal trellis module is upper bounded by (22) and is lower bounded by (24).

It is generally accepted that, the best (n, k, ν) binary convolutional code will have weight spectra better than the best (n, k, ν') code if $\nu > \nu'$. That means by increasing the complexity of the trellis with n-bit branches, it is likely that we can find a convolutional code with better weight spectra. Intuitively, we hope that this rule will apply if we consider the complexity based on the minimal trellis. If the answer is positive, then it will be helpful in the code search for the best (n, k, ν) binary convolutional codes since we do not have to check the weight spectra of codes which have low minimal-trellis complexity. At least, we know that we do not need to check those codes with state complexity profile in the form of $(\nu, \nu-1, \cdots, \nu-q, \ldots, \nu-q, \ldots, \nu-2, \nu-1)$, since such codes will have equivalent codes with memory sizes smaller than ν. In fact, we do not need to check any code for which at least one component in the state complexity profile has value smaller than ν.

By now, we only have some data for $(n, n - 1)$ binary convolutional codes. From Tables I and II for $(n, n - 1)$ codes, we see that for a given ν, larger $\varphi(\cdot)$ and $\psi(\cdot)$, i.e., small χ usually yields better weight spectra, although cases of exception are found in the pair of $(3, 2, \nu = 2, \chi = -1), (3, 2, \nu = 2, \chi = -2)$ codes and the pair of $(4, 3, \nu = 4, \chi = 1), (4, 3, \nu = 4, \chi = 0)$ codes, where in a pair, the code with a larger χ provides weight spectra slightly better than those for another code with a smaller χ. In [14], some best $(n, n - 1)$ convolutional codes were found. The codes with the largest $\varphi(\cdot)$ and $\psi(\cdot)$ (i.e., $\chi = 1 - n$) in Tables I and II have weight spectra almost the same as those of comparable codes listed in [14]. We observe that a convolutional code with the best weight spectra has its $\varphi(\cdot)$ and $\psi(\cdot)$ very close to the largest possible values for $(n, n - 1)$ binary convolutional codes. Whether this phenomenon will occur in (n, k) convolutional codes with k other than $n - 1$ is an interesting and open problem.

VII. Concluding Remarks

By examining the properties of minimal trellises of the convolutional codes, we find that convolutional codes with distinct memory sizes of minimal encoders may be equivalent. The associated weight spectra and decoding complexity of equivalent convolutional codes are identical. In general, for an (n, k, ν) convolutional code, we can have n equivalent codes for which the minimal trellis module of each code is a cyclicly shifted version of the minimal trellis module of another code. The memory sizes of encoders of equivalent codes may vary in the range of $\{\max\{0, \nu - \min\{k, n-k\}\}, \max\{0, \nu - \min\{k, n-k\}\} + 1, \ldots, \nu-1, \nu, \nu+1, \ldots, \nu+\min\{k, n-k\}-1, \nu+\min\{k, n-k\}\}$.

On the other hand, for the (n, k, ν) convolutional code, the complexity measured by the number of states and the number of branches in the minimal trellis module may vary according to the structure of the minimal trellis module. We derive upper bounds and lower bounds on these complexity measures.

In practical applications of decoding, besides state complexity and branch complexity, other measures for complexity such as the complexity of obtaining branch metric, trellis regularity and the number of comparisons may need to be considered. In [14], a soft-in–soft-out (SISO) decoder for the binary $(n, n - 1)$ convolutional code C based on the trellis of its dual $C^{\perp}$, i.e., the trellis of a binary $(n, 1)$ convolutional code, is proposed. Such a decoder is definitely superior to the SISO decoder using the minimal trellis of C considering the number of branches and the number of comparisons and the regularity. However, the SISO decoder using the trellis of $C^{\perp}$ requires some additional complexity in obtaining the branch metric as compared to the SISO decoder using the trellis of C.

Among the various measures of complexity, we guess that the information of the state complexity and the branch complexity of the minimal trellis can help us reduce some effort in the search for (n, k, ν) convolutional codes with best weight spectra. We do not need to consider any code for which at least one component in the state complexity profile has value smaller than ν. From the tabulated results of $(3, 2)$ and $(4, 3)$ convolutional codes, we observe the trend that $(n, n - 1, \nu)$ binary convolutional codes with the best weight spectra have the highest (or close to the highest) state complexity and branch complexity. For the general (n, k, ν) convolutional codes, whether the trend still exists is an interesting and open problem. The state complexity and branch complexity of the minimal trellis are also important factors in the search for convolutional codes with maximum possible code length under given $n - k, \nu$ and d_{free} [18], [19].

Acknowledgment

The authors wish to thank the associate editor and the anonymous reviewers for many helpful comments and suggestions.

References

[1] G. C. Clark and J. B. Cain, *Error Correction Coding for Digital Communications*. New York: Plenum, 1981.

[2] M. Bossert, *Channel Coding for Telecommunications*. New York: Wiley, 1999.

[3] S. Lin and D. J. Costello, Jr., *Error Control Coding*. Upper Saddle River, NJ: Pearson/Prentice Hall, 2004.

[4] J. B. Cain, G. C. Clark, Jr., and J. M. Geist, "Punctured convolutional codes of rate $(n, n-1)$ and simplified maximum likelihood decoding," *IEEE Trans. Inf. Theory*, vol. 25, pp. 97–100, Jan. 1979.

[5] P. J. Lee, "Constructions of rate $(n - 1)/n$ punctured convolutional codes with minimal required SNR criterion," *IEEE Trans. Commun.*, vol. 36, pp. 1171–1173, Oct. 1988.

[6] I. E. Bocharova and B. D. Kudryashov, "Rational rate punctured convolutional codes for soft-decision Viterbi decoding," *IEEE Trans. Inf. Theory*, vol. 43, pp. 1305–1313, Jul. 1997.

[7] G. Bégin and D. Haccoun, "High rate punctured convolutional codes: Structure properties and construction technique," *IEEE Trans. Commun.*, vol. 37, pp. 1381–1385, Dec. 1989.

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

[9] D. J. Muder, "Minimal trellises for block codes," *IEEE Trans. Inf. Theory*, vol. 34, pp. 1049–1053, Sep. 1988.

[10] R. J. McEliece and W. Lin, "The trellis complexity of convolutional codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 1855–1864, Nov. 1996.

[11] V. Sidorenko and V. Zyablov, "Decoding of convolutional codes using a syndrome trellis," *IEEE Trans. Inf. Theory*, vol. 40, no. 5, pp. 1663–1666, Sep. 1994.

[12] H.-H. Tang and M.-C. Lin, "On $(n, n-1)$ convolutional codes with low trellis complexity," *IEEE Trans. Commun.*, vol. 50, no. 1, pp. 37–47, Jan. 2002.

[13] A. Graell i Amat, G. Montorsi, and S. Benedetto, "A new approach to the construction of high-rate convolutional codes," *IEEE Commun. Lett.*, vol. 5, no. 11, pp. 453–455, Nov. 2002.

[14] ——, "Design and decoding of optimal high-rate convolutional codes," *IEEE Trans. Inf. Theory*, vol. 50, no. 5, pp. 867–881, May 2004.
[15] R. J. McEliece, "On the BCJR trellis for linear block codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 4, pp. 1072–1092, Jul. 1996.
[16] G. D. Forney, Jr., "Dimension/length profiles and trellis complexity of linear block codes," *IEEE Trans. Inf. Theory*, vol. 40, no. 6, pp. 1741–1752, Nov. 1994.
[17] A. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun.*, vol. COM-19, no. 5, pp. 751–772, Oct. 1971.
[18] P. CharnKeitKong, H. Imai, and K. Yamaguchi, "On classes of rate $k/(k+1)$ convolutional codes and their decoding techniques," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 2181–2193, Nov. 1996.
[19] E. Rosnes and Ø. Ytrehus, "Maximum length convolutional codes under a trellis complexity constraint," *J. Complex.*, vol. 20, pp. 372–408, Mar.–Jun. 2004.

Iterative Soft-Input Soft-Output Decoding of Reed–Solomon Codes by Adapting the Parity-Check Matrix

Jing Jiang, *Student Member, IEEE*, and Krishna R. Narayanan, *Member, IEEE*

***Abstract*—An iterative algorithm is presented for soft-input soft-output (SISO) decoding of Reed–Solomon (RS) codes. The proposed iterative algorithm uses the sum–product algorithm (SPA) in conjunction with a binary parity-check matrix of the RS code. The novelty is in reducing a submatrix of the binary parity-check matrix that corresponds to less reliable bits to a sparse nature before the SPA is applied at each iteration. The proposed algorithm can be geometrically interpreted as a two-stage gradient descent with an adaptive potential function. This adaptive procedure is crucial to the convergence behavior of the gradient descent algorithm and, therefore, significantly improves the performance. Simulation results show that the proposed decoding algorithm and its variations provide significant gain over hard-decision decoding (HDD) and compare favorably with other popular soft-decision decoding methods.**

***Index Terms*—Adapting the parity-check matrix, gradient descent, iterative decoding, Reed–Solomon (RS) codes, soft-decision decoding.**

I. Introduction

Reed–Solomon (RS) codes are one of the most popular error-correction codes in many state-of-the-art communication and recording systems. In most of these existing systems, RS codes are decoded via an algebraic hard-decision decoding (HDD) algorithm. When soft information about the channel output is available, HDD can incur a significant performance loss compared to optimal soft decision decoding. For example, for the additive white Gaussian noise (AWGN) channel, the loss is believed to be about 2–3 dB. Moreover, in some situations, it is desirable to obtain soft output from the decoder. A typical example is when turbo equalization is employed at the receiver and soft outputs from the decoder have to be fed back to the equalizer. Consequently, soft-input soft-output (SISO) decoding algorithms for RS codes are of research interest both for theoretical and practical reasons.

In the literature, there are several classes of soft-decision decoding algorithms. Enhanced HDD algorithms such as generalized minimum distance (GMD) decoding [1], Chase decoding [2] and a hybrid of Chase and GMD algorithms (CGA) [3] use reliability information to assist HDD decoding. Enhanced HDD usually gives a moderate performance improvement over HDD with reasonable complexity. Recently, algebraic soft interpolation based decoding (the Koetter–Vardy (KV) algorithm [4]), which is a list decoding technique that uses the soft information from the channel to interpolate each symbol, has become popular [5]–[7]. The KV algorithm can significantly outperform HDD for low-rate RS codes. However, to achieve large coding gain, the complexity can be prohibitively large. For detailed discussions of the complexity performance tradeoff of the KV algorithm, we refer interested readers to [7]. Another approach is decoding RS codes using their binary image expansions. Vardy and Be'ery showed that RS codes can be decomposed into Bose–Chaudhuri–Hocquenghem (BCH) subfield subcodes which are glued together using glue vectors [8]. Even though this decomposition significantly reduces the trellis complexity of maximum-likelihood (ML) decoding of RS codes, the complexity still grows exponentially with the code length and $d_{\min}$ and it is thus infeasible for practical long codes. Recent work [9] has reduced the complexity and modified the algorithm in [8] to generate soft output efficiently. By using the binary image expansion of RS codes, we can also use decoding algorithms for general linear block codes such as reliability based ordered statistics decoding (OSD) [10] and its variations [11] for soft-decision decoding of RS codes. Previous such works include the hybrid algorithm by Hu and Lin [12] and the box and match algorithm (BMA) [13] by Fossorier and Valembois. OSD-based algorithms are quite efficient for practical RS codes even though they do not take the structure of the RS codes into account.

Iterative decoding [14] algorithms are of emerging interest for soft decision decoding of RS codes [15]–[17]. The main difficulty in directly applying iterative decoding techniques to RS codes is that the parity-check matrix of an RS code is in general not sparse. In order to deal with such dense parity-check matrices, Yedidia *et al.* [18] proposed a "generalized belief propagation" (GBP) algorithm that introduces hidden states in iterative decoding. However, their results show that this technique does not work well for high-density parity-check (HDPC) codes (such as RS codes) over the AWGN channel. We observe from the simulations that the iterative decoder fails mainly due to some of the unreliable bits "saturating" most of the checks which causes iterative decoding to be stuck at some pseudo-equilibrium points. In [16], the cyclic structure of RS codes is taken advantage of and a sum product algorithm (SPA) is applied to a random shift of the received vector at each iteration to avoid pseudo-equilibrium points (see [16] for details). While significant improvement in performance over HDD was obtained for short codes, the performance improvement diminishes for long RS codes.

In this correspondence, we present an iterative SISO decoding algorithm (which is based on the SPA) for RS codes. The main novelty in the proposed scheme is to adapt the parity-check matrix at each iteration according to the bit reliabilities such that the unreliable bits correspond to a sparse submatrix and the SPA is then applied to the adapted parity-check matrix. This adaptation prevents the iterative decoder from getting stuck at pseudo-equilibrium points and, hence, the convergence behavior of the iterative decoder is significantly improved. Simulation results show that the proposed iterative decoding scheme performs well for RS codes with reasonable decoding complexity, even though the parity-check matrices are not sparse. While the approach in [16] is also one of adapting the parity-check matrix, the adaptation there

Manuscript received March 8, 2005; revised October 17, 2005. This work was supported in part by the National Science Foundation under Grant CCR0093020 and by Seagate Technology, Pittsburgh, PA 15222 USA.

The authors are with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: jjiang@ece.tamu.edu; krn@ece.tamu.edu).

Communicated by M. P. C. Fossorier, Associate Editor for Coding Techniques.
Digital Object Identifier 10.1109/TIT.2006.878176