Early-Elimination Modification for Priority-First Search Decoding

Shin-Lin Shieh, Po-Ning Chen, *Senior Member, IEEE,*
Yunghsiang S. Han, *Senior Member, IEEE,* and Ting-Yi Wu

***Abstract*—In order to release the *growing* demand for computational complexity with respect to *increasing* information sequence length in the priority-first search decoding algorithm, a path elimination modification is proposed and also analyzed in this work. Specifically, we propose to directly eliminate all paths whose end nodes are Δ-level prior to the farthest node among those that have been visited thus far by the priority-first search. Following the argument on random coding, we then analyze the path elimination window Δ that results in a larger exponent for additional decoding error caused by path elimination than the exponent of the maximum-likelihood error performance, and hence guarantees *exponentially* negligible performance degradation. Our analytical results indicate that under additive white Gaussian noise (AWGN) channels, the path elimination window required for exponentially negligible performance degradation is just three times the code constraint length for rate one-half convolutional codes. It can be further reduced to 1.7-fold of the code constraint length when rate one-third convolutional codes are considered instead. Simulation results confirm these analytical window sizes. As a consequence, the priority-first search decoding algorithm can considerably reduce its computation burden and memory consumption by directly eliminating a large number of paths with nearly no performance degradation. This makes the priority-first search decoding algorithm with path elimination suitable for applications that demand low-complexity software implementation with near optimal performance.**

***Index Terms*—Priority-first search decoding, maximum-likelihood, soft-decision, random coding.**

I. Introduction

ONE of the commonly used decoding algorithms for convolutional codes is the Viterbi algorithm. It operates on a convolutional code trellis and has been shown to be a maximum-likelihood (ML) decoder [8]. When the information sequence is long, *path truncation* is proposed for practical implementation of the Viterbi algorithm [8]. Instead of recording all trellis branches of survivor paths in the decoder memory, only a certain number of the most recently visited trellis branches are retained, and a decision is forced on the oldest trellis branch whenever new data arrives at the decoder. Three strategies have been proposed in regard to this kind of forceful decision: (i) The majority vote strategy traces back from all states, and outputs the decision that occurs most often; (ii) The best state strategy only traces back from the state with the best metric, and outputs the information bits corresponding to the path being traced; (iii) The random state strategy randomly traces back from one state, and outputs the information bits corresponding to the path being traced. Although none of these three strategies guarantee maximum-likelihood performance, their performance degradation can be made negligible as long as the truncation window is sufficiently large.

Forney [4] proved that a truncation window that is 5.8-fold of the code constraint length suffices to provide negligible performance degradation for the best state strategy under a very noisy channel. Hemmati and Costello [7] later derived an upper performance bound as a function of the truncation window and distance property of a given code, and suggested that at very high signal-to-noise ratio, a truncation length larger than the code free distance is sufficient to achieve near-optimal performance for the best state strategy. McEliece and Onyszchuk [11] studied the tradeoff between the truncation window size and performance loss for the random state strategy under both binary symmetric channels and additive white Gaussian noise (AWGN) channels, concluding that the truncation window for the random state strategy should be about twice as large as that for the best state strategy. Onyszchuk [12] continued to investigate the truncation lengths for the best state strategy under AWGN channels, and suggested that the truncation lengths for codes with constraint length greater than seven are in general larger than 5.8 times the code constraint length.

Another well-known decoding algorithm for convolutional codes is the sequential decoding algorithm [8]. Although it is suboptimal in performance, its decoding complexity does not depend on the code constraint length in contrast to the exponential growth of the Viterbi decoder. This makes the sequential decoding algorithm especially suitable for convolutional codes with large constraint length. For this reason, it has recently been used in the decoding of the so-called "supercode" that considers the joint effect of multi-path channel and convolutional code [6]. Furthermore, sequential-type search algorithms have been shown promising for sphere detector in communication systems with high-order modulation and multiple antennae [9][10].

In 2002, by replacing the Fano metric with one derived

Paper approved by A. K. Khandani, the Editor for Coding and Information Theory of the IEEE Communications Society. Manuscript received December 14, 2009; revised March 30, 2010 and July 6, 2010.

S. L. Shieh is with HT mMobile Inc., Hsinchu Science Park, Taiwan, R.O.C., and also with the Graduate Institute of Communication Engineering, National Taipei University, Taipei, Taiwan, R.O.C. (e-mail: slshieh@mail.ntpu.edu.tw).

P. N. Chen and T. Y. Wu are with the Department of Electrical Engineering, National Chiao-Tung University, Hsinchu City, Taiwan, R.O.C. (e-mail: poning@faculty.nctu.edu.tw, mavericktywu@gmail.com).

Y. S. Han is with the Department of Electrical Engineering, National Taiwan University of Science and Technology, Taiwan, R.O.C. (e-mail: yshan@mail.ntust.edu.tw).

Digital Object Identifier 10.1109/TCOMM.2010.101910.090766

from maximum-likelihood criteria, a variant of the sequential decoding algorithm was established in [5]. The same paper showed that with the new metric, optimal performance can be obtained by following the priority-first search procedure of sequential decoding. Thus, the performance of the priority-first search decoding algorithm (PFSDA) proposed therein is exactly the same as that of the Viterbi algorithm. As far as the *software* complexities[1] of the two equal-performance decoding algorithms are concerned, it has been shown by simulations that the algorithmic complexity of the proposed PFSDA is in general smaller [5].

When compared with the sequential decoding algorithm using the Fano metric, the PFSDA has four advantages. Firstly, in contrast to the *asymptotical* maximum-likelihoodness of the sequential decoding algorithm, the PFSDA is an *exact* maximum-likelihood (ML) decoding algorithm for codes of any given finite blocklength. Secondly, by adding another stack for the examination of path merging, the PFSDA can operate over a code trellis without sacrificing its optimality in performance. Thirdly, the decoding metric used by the PFSDA does not depend on the channel signal-to-noise ratio (SNR) under AWGN channels, and hence no estimation of channel SNR is necessary at the receiver end. Fourthly, the decoding metric of the PFSDA is less sensitive to quantization distortion than the sequential decoding algorithm. As such distortion is unavoidable during the practical implementation of a decoding algorithm, it will be shown in this work that 4-bit quantization causes only 0.05 dB loss on block error rate for the PFSDA, while the same quantization distortion leads to an evident performance loss for the sequential decoding algorithm (cf. Figure 10). These factors make the PFSDA suitable for systems that provide only a simple quantizer but no channel SNR estimator.

Analogous to both of the Viterbi algorithm and the sequential decoding algorithm, the computational complexity and memory consumption of the PFSDA grow with the length of information sequence. An effective approach to reduce the growing computational effort with respect to increasing information sequence length is perhaps to restrict the number of paths involved in the computations. Specifically, we propose to directly eliminate the paths whose end nodes are Δ-level-prior to the farthest one among all nodes that have been visited thus far by the priority-first search. This process is referred to as *early elimination*. Following the random coding argument used by Forney [4], we analyze the early-elimination window Δ that achieves *exponentially* negligible performance degradation in the sense that the exponent of additional error introduced by early-elimination is larger than the exponent of the ML error performance. Our analytical results indicate that the required early-elimination window for exponentially negligible performance degradation is around three times the code constraint length for half-rate convolutional codes. This window can be further reduced to 1.7-fold of the code constraint length when rate one-third convolutional codes are considered. Simulations are then performed, and are found to conform to our analytical results. Consequently, the computational burden and memory consumption of the PFSDA can be considerably reduced by introducing early-elimination modification with nearly no sacrifice in performance.

It should be emphasized that the *early-elimination window* for the PFSDA functions differently in its notion from the *path truncation window* for the Viterbi algorithm. The former is the window to eliminate paths from the stacks in order to reduce the maintenance burden of a large stack, while the latter is used by the Viterbi algorithm to truncate the paths under consideration (without eliminating any of them) and its objective is to provide a timely decoding output. As such, path truncation can also be applied to the PFSDA together with the early-elimination scheme when timely decision is also required. Simulations show that by adopting the best state strategy over $(2, 1, 12)$ convolutional codes, the path truncation window for the PFSDA with early elimination window $\Delta = 40 \approx 3(12+1)$ should be no less than $110 \approx 8.5(12+1)$ under AWGN channels, if 0.05 dB performance degradation from the ML decoder is required for block error rate less than 10^{-2}. This number is exactly the same as that required by the Viterbi algorithm under an identical requirement. Thus, the decision delays for both the Viterbi algorithm and the early-elimination PFSDA are comparable.

The rest of the paper is organized as follows. The system model and the PFSDA are introduced in Section II. The early-elimination scheme is presented in Section III. Numerical and simulation results are summarized and remarked in Section IV. Section V concludes the paper.

II. Preliminaries

We use $\mathscr{C}$ to denote a binary (n, k, m) convolutional code with input information sequence of $k \times L$ bits, followed by $k \times m$ zeros for the purpose of clearing the encoder memory. Thus, $\mathscr{C}$ forms an (N, K) linear block code with *effective code rate* $R \triangleq K/N$, where $K \triangleq kL$ and $N \triangleq n(L + m)$. With this setting, the *code rate*, the *memory order* and the *constraint length*[2] of $\mathscr{C}$ are given by k/n, m and $m + 1$, respectively.

Let a binary codeword of $\mathscr{C}$ be represented by $\boldsymbol{v} \triangleq (v_0, v_1, \ldots, v_{N-1})$, where each $v_j \in \{0, 1\}$, and denote a portion of it by $\boldsymbol{v}_{(a,b)} \triangleq (v_a, v_{a+1}, \ldots, v_b)$. For convenience, we drop the subscripts 0 and $N - 1$ whenever they appear in notation $\boldsymbol{v}_{(a,b)}$; hence, $\boldsymbol{v}_{(0,b)}$ and $\boldsymbol{v}_{(0,N-1)}$ are abbreviated respectively as $\boldsymbol{v}_{(b)}$ and $\boldsymbol{v}$. The same abbreviation will be applied to other vector notations.

Assume that the binary codeword is transmitted over a time-discrete channel with channel output $\boldsymbol{r} \triangleq (r_0, r_1, \ldots, r_{N-1})$. Define the hard-decision sequence $\boldsymbol{y} \triangleq (y_0, y_1, \ldots, y_{N-1})$ corresponding to $\boldsymbol{r}$ as:

$$y_j \triangleq \begin{cases} 1, & \text{if } \phi_j < 0; \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where

$$\phi_j \triangleq \log \frac{f(r_j|v_j = 0)}{f(r_j|v_j = 1)},$$

[1] By *software* complexity (as contrast to *hardware* complexity), we mean the decoder is programmed and executed in a sequential fashion, and no parallel processing technique is employed in its implementation.

[2] A formal definition of the *memory order* is $m \triangleq \max_{1 \le i \le k} \nu_i$, where ν_i is the length of the ith shift register in a convolutional encoder. This leads to another known definition of the *constraint length*, i.e., $\sum_{i=1}^{k} \nu_i$ (See for example [8, Definition 11.3]). In this work, by following [16], we adopt $m + 1$ as the constraint length so as to be consistent with the analysis of early-elimination windows (cf. (4)).

and $f(r_j|v_j)$ denotes the channel transition probability density for r_j given v_j.[3] It can then be obtained [5] that the maximum-likelihood decoding output $\hat{\boldsymbol{v}}$ for received vector $\boldsymbol{r}$ is given by:

$$\hat{\boldsymbol{v}} = \boldsymbol{y} \oplus \boldsymbol{e}^*, \tag{2}$$

where $\boldsymbol{e}^*$ is the error pattern with the smallest $\sum_{j=0}^{N-1} e_j|\phi_j|$ among all $\boldsymbol{e} \in \{0,1\}^N$ satisfying $\boldsymbol{e}\mathbb{H}^T = \boldsymbol{y}\mathbb{H}^T$, and $\mathbb{H}$ is the parity check matrix of the (N, K) linear block code $\mathscr{C}$. In the above sentence, "$\oplus$" is the bitwise exclusive-OR operation, and superscript "T" denotes the matrix transpose operation. Based on the observation in (2), a sequential-type decoder has been established in [5] by replacing the Fano metric in the sequential decoding algorithm with a metric defined as:

$$\mu\left(\boldsymbol{x}_{(\ell n-1)}\right) \triangleq \sum_{j=0}^{\ell n-1} \mu(x_j), \tag{3}$$

where $\boldsymbol{x}_{(\ell n-1)} = (x_0, x_1, \ldots, x_{\ell n-1}) \in \{0,1\}^{\ell n}$ represents the code word label of a path ending at level ℓ in the (n, k, m) convolutional code tree, and $\mu(x_j) \triangleq (y_j \oplus x_j)|\phi_j|$ is the bit metric. Since the decoding metric μ is nondecreasing along all code paths, and since finding $\boldsymbol{e}^*$ is equivalent to finding the code path with the smallest metric in the code tree, it was also proven in [5] that the proposed sequential-type decoder (referred to as Priority-First Sequential Decoding Algorithm or PFSDA) guarantees locating the maximum-likelihood codeword.

By adding a second stack, the PFSDA can be made to operate on a code trellis instead of a code tree [5]. The two stacks are referred to as the *Open Stack* and the *Closed Stack*, respectively. The Open Stack contains all paths that end at the frontier part of the trellis being explored thus far, as exemplified in Figure 1. It is called the Open Stack because it functions the same as the single stack in the sequential decoding algorithm and hence its elements are still *open* for further expansion. An efficient management of the Open Stack can be obtained by using the *HEAP* data structure [2]. This structure can reduce the computational burden of path insertion down to the order of $O(\log s)$, where s is the size of the Open Stack. The Closed Stack stores the information of the ending states and the ending levels of paths that had been the top paths of the Open Stack. Analogously, it is named the Closed Stack because its elements will not be further expanded and thus are seemingly *closed*.

In order to facilitate the introduction of early-elimination modification, the trellis-based PFSDA [5] is reproduced below.

Step 1. Load the Open Stack with the path consisting of only the origin node, and initialize its metric to be zero.

Step 2. Put into the Closed Stack both the state and level of the end node of the top path in the Open Stack. Compute the path metric for each of the successor paths of the top path in the Open Stack by adding the branch metric of the extended branch to the path metric of the top path. Delete the top path from the Open Stack.

[3]For discrete channels, it is understood that the log-likelihood ratio ϕ_j becomes $\log[\Pr(r_j|v_j = 0)/\Pr(r_j|v_j = 1)]$. All the analyses in this work are accordingly valid for discrete channels.

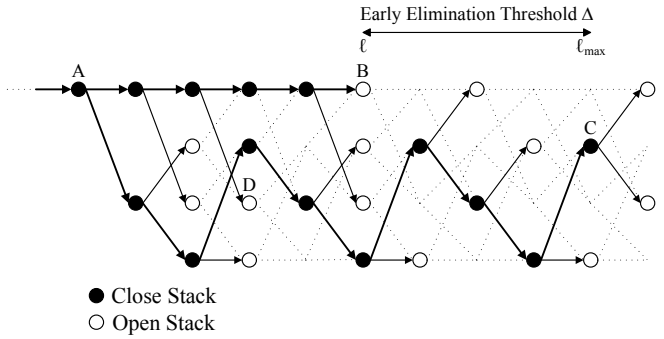

Fig. 1. Early-elimination window Δ in the trellis-based PFSDA.

Step 3. Discard those successor paths in Step 2, which end at a node that has the same state and level as any entry in the Closed Stack. If any successor path ends at the same node as a path already in the Open Stack, eliminate the path with higher path metric.[4]

Step 4. Insert the remaining successor paths into the Open Stack *in order of ascending path metrics. If two (or more) paths in the Open Stack have equal metric, sort them in order of descending levels. If they happen to end at the same level, sort them randomly.*

Step 5. If the top path in the Open Stack reaches the end of the convolutional code trellis, the algorithm stops; otherwise go to Step 2.

III. Early-Elimination Modification for Priority-First Search Decoding Algorithm

The motivation behind early-elimination modification can be indicated by the following two observations. As shown in Figure 1, suppose that the path ending at node C is the portion of the final code path found at the end of priority-first search, and suppose that the path ending at node D happens to be the current top path. Then, expanding node D until its offspring gradually accumulate adequate decoding metrics to exceed the decoding metric of the path ending at node C will consume a considerable but unnecessary amount of computational effort. On the other hand, due to the nondecreasing nature of the adopted decoding metric along paths on the code trellis, a top path ending at a level much smaller than $\ell_{\max}$, where $\ell_{\max}$ is the largest level for all nodes having been expanded thus far by the priority-first search, is with high probability not the final code path located at the end of decoding process. These two observations jointly suggest that by setting a proper level threshold Δ and directly eliminating the top path whose level is no larger than $(\ell_{\max} - \Delta)$, the computational complexity of the priority-first search algorithm may be reduced without sacrificing much of the performance. We call this scheme *early elimination*.

In its implementation, because the decoding complexity is mainly contributed by the branch metric computations following the node expansion of the top path in the Open Stack, we propose to examine the path early-elimination condition only on the top paths prior to their expansions. This can save

[4]For discrete channels, it may occur that the successor path not only ends at the same node as some path already in the Open Stack but also has equal path metric to it. In such case, just randomly eliminate one of them.

the extra effort of keeping tract of those paths momentarily ending at the least level in addition to the maintenance of the top paths with the least metric value. As such, only Step 2 in the trellis-based PFSDA needs to be modified.

Initialize at the beginning of the algorithm $\ell_{\max} = 0$.

Step 2″. Perform the following check before executing the original Step 2.

- *If the top path in the Open Stack ends at a node whose level is less than or equal to* $(\ell_{\max} - \Delta)$*, then directly eliminate the top path and go to Step 5; otherwise, update* $\ell_{\max}$ *if* $\ell_{\max}$ *is smaller than the ending level of the current top path.*

It is worth mentioning that since the decoding metric is monotonically nondecreasing along the path to be searched, it is guaranteed that the path that updates the current $\ell_{\max}$ is exactly the one with the smallest path metric among all paths ending at the same level [5]. This is the key that leads to that the priority-first search using a monotonic maximum-likelihood metric defined in (3) guarantees that the first top path that reaches the last level of the code trellis is the maximum-likelihood code path.

A natural question that follows is how to analytically determine the early-elimination window size Δ that can result in exponentially negligible performance degradation. For this purpose, we derive a lower bound for the exponent of additional decoding error introduced by early-elimination modification, and choose an early-elimination window, of which the corresponding exponent lower bound is larger than the known error exponent of the maximum-likelihood error performance. This window can therefore guarantee exponentially negligible performance degradation. In particular, we show in Appendix A that for codes with code rate R above the channel cutoff rate, the additional decoding error due to early elimination becomes exponentially negligible if

$$\Delta/(m+1) > E_{\rm c}(R)/E_{\rm el}(R), \tag{4}$$

where $E_{\rm el}(R)$ and $E_{\rm c}(R)$ are functions defined in (14) and (15), respectively. In the next section, we will examine the near-optimal early-elimination windows for various codes by simulations, and compare them with the ones obtained analytically from (4).

IV. Numerical and Simulation Results under AWGN Channel

Inequality (4) is valid only for those code rates no less than the channel cutoff rate R_0. Although lack of general evidence, the cutoff rate is widely believed to be the rate beyond which the communication cost dramatically increases [1, pp. 184]. This is especially true for sequential decoders for which the decoding complexity grows rapidly when the code rate is increased above R_0. The code rate taken into (4) is accordingly suggested to be an "implementation-feasible" number no larger than R_0. As a result, $R = R_0$ is the only choice that can simultaneously meet this requirement and (4). The same choice has also been used by Forney to show that under very noisy channels, 5.8-fold of the code constraint length is suggested for the path truncation window at the cutoff rate [4], [15].

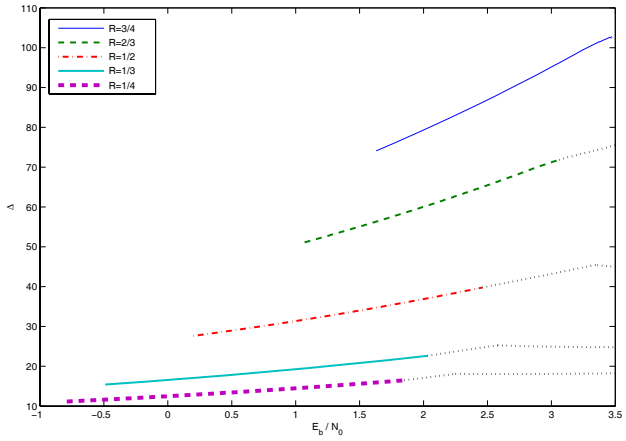

Fig. 2. The lower bound $(m+1)E_{\rm c}(R)/E_{\rm el}(R)$ of the early-elimination window Δ for various code rates, $m = 12$ and $R_0 \le R < C$. The extension dotted black line along each curve corresponds to rates below R_0, for which $(m+1)\min\{E_{\rm sp}(R), E_{\rm x}\}/E_{\rm el}(R)$ is plotted instead (See Footnote 8.)

As a complement to the above discussion, Figure 2 depicts the lower bounds of Δ according to (4) with respect to various code rates and $m = 12$. By considering only the range corresponding to $R_0 \le R < C$, where C is the channel capacity, Figure 2 clearly indicates that $(m+1)E_{\rm c}(R)/E_{\rm el}(R)$ peaks at $R = R_0$ as the SNR varies. Since what we concern is an early-elimination window Δ that guarantees exponentially negligible performance degradation, this figure again suggests taking the peak Δ value at $R = R_0$ is sufficient for all SNRs under consideration. For the SNRs beyond R_0 (i.e., $0 \le R < R_0$), one can perhaps derive an analytically suggested Δ value using (16) as remarked in Footnote 8. The later simulations however consistently show that taking the minimum Δ satisfying (4) at $R = R_0$ has already provided almost exact performance to an ML decoder; hence, it is not necessary to further increase Δ as implied by (16). A side and reasonable observation from the same figure is that Δ should be made larger for higher code rates.

By choosing noise variances $\sigma^2 = 0.567$ and $\sigma^2 = 0.940$ to respectively approach the cutoff rates $1/2$ and $1/3$ under AWGN channels, it can be established from (4) that the suggested early-elimination windows at rates equal to the cutoff rates are respectively:

$$\Delta > \frac{0.500}{0.164} \times (m+1) \approx 3.05(m+1) \text{ for rate 1/2 codes} \tag{5}$$

and

$$\Delta > \frac{0.333}{0.195} \times (m+1) \approx 1.71(m+1) \text{ for rate 1/3 codes.}$$

The exponent functions $E_{\rm el}(R)$ and $E_c(R)$ for the above AWGN channels are plotted in Figure 3.

Condition (5) then indicates that for (2,1,6), (2,1,8), (2,1,10), and (2,1,12) convolutional codes, taking $\Delta = 22$, 28, 34 and 40, respectively, should suffice to result in exponentially negligible performance degradation. Simulations are next performed and summarized in Figure 4, which confirms that the performance of the early-elimination PFSDA with these

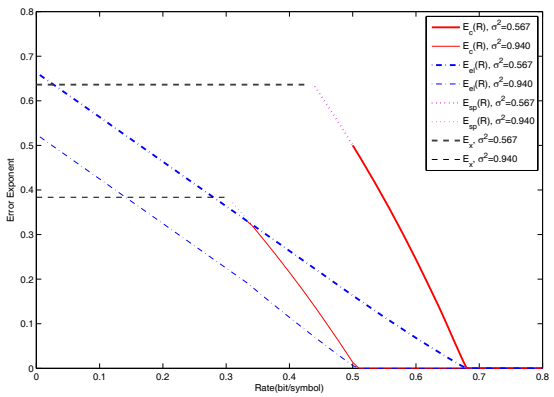

Fig. 3. Exponent lower bound $E_{\rm el}(R)$ of additional decoding error due to early elimination, and exponent lower bound $E_{\rm c}(R)$ and upper bounds $E_{\rm sp}(R)$ and $E_{\rm x}$ of the ML decoding error for time-varying convolutional codes. The solid red curves correspond to the region in which $E_{\rm c}(R) = E_{\rm sp}(R)$, i.e., $R_0 \le R < C$. The noise variances of the AWGN channels considered are $\sigma^2 = 0.567$ and $\sigma^2 = 0.940$, respectively.

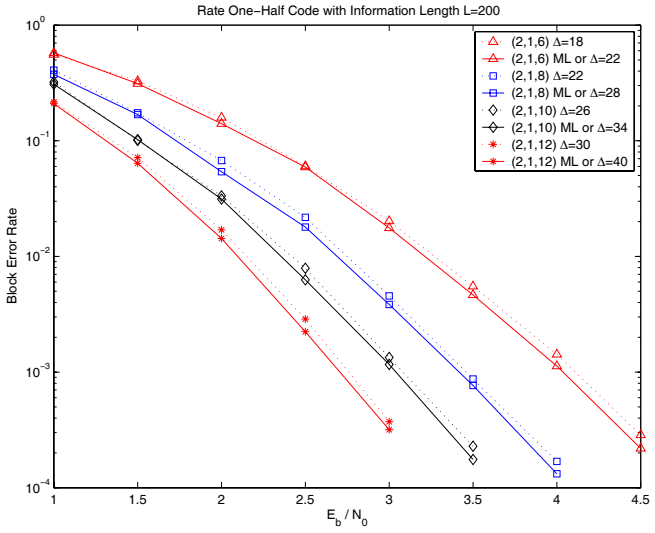

Fig. 4. Performance for rate one-half convolutional codes for the ML decoder and the PFSDA with early-elimination window Δ. The message length is $L = 200$. The generator polynomials for $m = 6, 8, 10, 12$ are [554,744], [561,753], [4672,7542], and [42554,77304] (in octal), respectively.

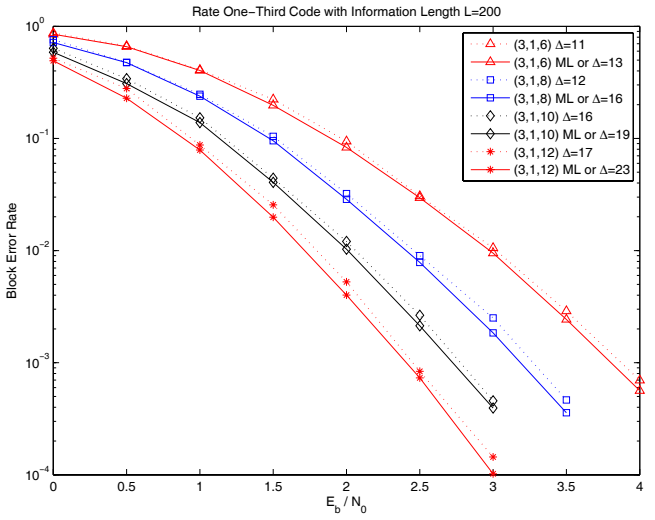

Fig. 5. Performance for rate one-third convolutional codes for the ML decoder and the PFSDA with early-elimination window Δ. The message length is $L = 200$. The generator polynomials for $m = 6, 8, 10, 12$ are [554,624,764], [557,663,711], [4726,5562,6372], and [42554,43364,77304] (in octal), respectively.

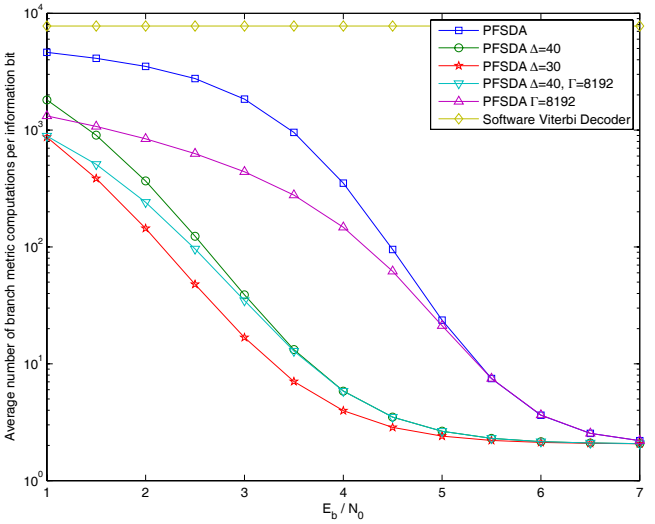

Fig. 6. Average number of branch metric computations per information bit for (2,1,12) convolutional codes decoded by the Viterbi decoder, the PFSDA with and without early elimination and also with and without finite stack size constraint Γ under AWGN channels. The message length is $L = 200$.

suggested window sizes is almost indistinguishable from the maximum-likelihood decoder.[5]

By simulations, we also examine the Δ values that result in at most 0.1 dB performance degradation, and plot their corresponding performance curves also in Figure 4. The simulation results show that the Δ values are reduced down to 18, 22, 26 and 30 for (2,1,6), (2,1,8), (2,1,10) and (2,1,12) convolutional codes, respectively, when performance degradation of 0.1 dB is acceptable. This indicates that although one can adopt the Δ values directly derived from (4), these values tend to be larger than what are usually required from the viewpoint of practical applications.

The discussions in the above two paragraphs can be likewise observed in Figure 5 for rate one-third convolutional codes, and hence their discussions are omitted.

The reduction of computational complexity due to early elimination is illustrated in Figure 6. For sequential-type decoders, the decoding complexity is clearly determined not only by the number of branch metrics evaluated but also by the cost of searching and reordering of the stack elements. The latter cost however has been proved to be of comparable order to the former one [3]. It is therefore justified to consider the branch metric computations as the key determinant of algorithmic complexity. Figure 6 then shows that for $(2, 1, 12)$ convolutional code with message length $L = 200$ and at $E_{\rm b}/N_0 = 2.5$ dB, the average number of branch metric computations to decode one information bit is 123 for the

[5]Since their performance curves are indistinguishable, the near-ML early-elimination PFSDA and the ML decoder are therefore combined in one in the legends of Figures 4 and 5.

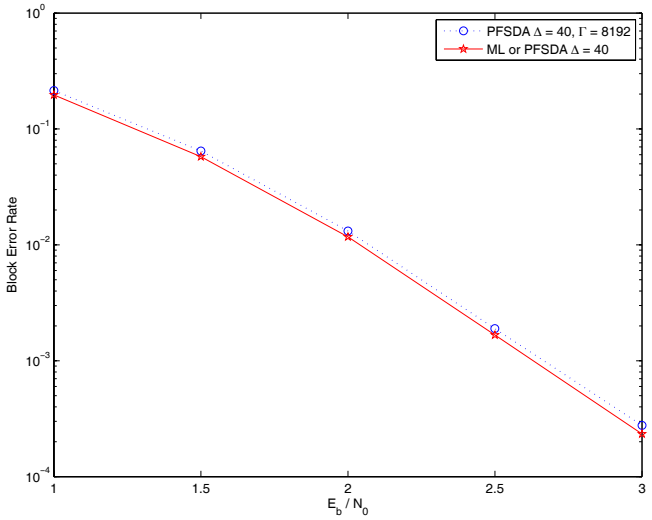

Fig. 7. Performance for (2,1,12) convolutional codes decoded by the ML decoder, the PFSDA with early-elimination window Δ, and the PFSDA with early-elimination window Δ and finite stack size Γ. The message length is $L = 200$.

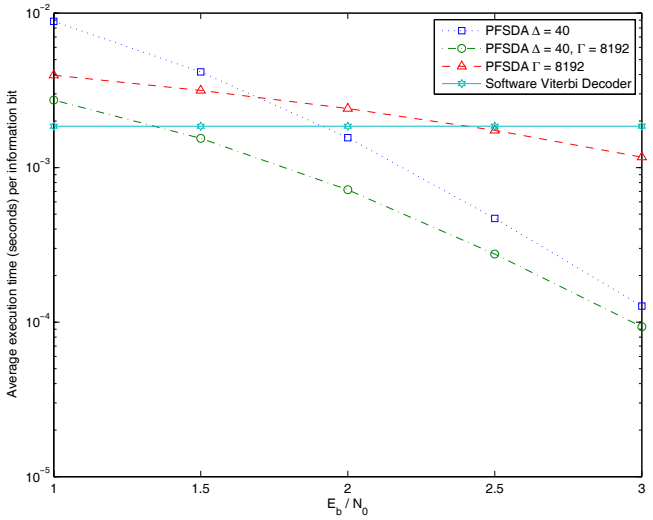

Fig. 8. Average decoding time (in seconds) per information bit for (2,1,12) convolutional codes decoded by the Viterbi decoder, the PFSDA with early-elimination window Δ, and the PFSDA with early-elimination window Δ and finite stack size Γ. The message length is $L = 200$.

PFSDA with early-elimination window $\Delta = 40$. This number is only 4.4% of that (specifically, 2994) for the PFSDA without early elimination. One can further reduce the average decoding complexity down to 53 computations per information bit at a price of 0.1 dB performance degradation, if Δ decreases down to 30. The above simulations confirm our anticipation that with early-elimination modification, the PFSDA can achieve near-optimal performance with considerable reduction of computational complexity.

Although we have anticipated that by early elimination, the stack size requirement for the PFSDA is reduced, we have been surprised by the amount of the reduction. By setting a stack size limit Γ to be the same as that required by the Viterbi decoder, i.e., 2×2^m, and simply deleting the path with the smallest level when the stack size exceeds this limit,[6] only 0.05 dB performance degradation is resulted in comparison with the ML decoder as shown in Figure 7. With such a small stack size limit, the computational complexity can be further reduced from 123 down to 96 computations per information bit at $E_b/N_0 = 2.5$ dB as observed from Figure 6. Notably, this computational complexity is much smaller than the PFSDA with only the stack size limit but without early elimination, where 629 computations are required to decode one information bit at the same E_b/N_0. Thus, the early-elimination modification is more crucial in complexity reduction than the limitation of stack size.

Figure 6 also presents the number of per-information-bit metric computations required by the Viterbi decoder. From this figure, one may question that even though the early-elimination PFSDA has a much smaller number of branch metric computations than the Viterbi decoder, the management of two stacks, such as insertion and deletion of stack elements, may consume more time in each computation. In order to have a better understanding in this practical issue, an experiment was performed on an IBM System x3800 server, and the resulting average execution times per information bit are summarized in Figure 8. In this simulation, two implicit observations that are not shown in this figure can be added. Firstly, by using the HEAP data structure, the execution time per branch computation is nearly a constant with respect to the signal-to-noise ratios for the PFSDA, and this average execution time is around 13 times that of the Viterbi decoder when the PFSDA with $\Delta = 40$ and $\Gamma = 8192$ is regarded. Secondly, the PFSDA without stack size limitation needs a little more time in each computation at low SNR because the stack size may grow slightly larger in a noisier environment. Accordingly, to place a moderately small upper limit (such as $\Gamma = 2 \times 2^{12}$ for $(2, 1, 12)$ convolutional codes) on the stack size together with the early-elimination modification can provide the best decoding efficiency subject to near-optimal performance. Figure 8 substantiates this conclusion, which shows that the PFSDA with early-elimination window $\Delta = 40$ and stack size limit $\Gamma = 8192$ is seven times faster than the Viterbi algorithm at $E_b/N_0 = 2.5$ dB when both decoders are implemented by software.

Timely or on-the-fly decision outputs are sometimes required for certain applications. We next examine the path truncation window T beyond which the corresponding information bits can be forcefully decided or well estimated for the early-elimination PFSDA. Respective comparison with the Viterbi decoder was also performed. The best state strategy that traces back the top path in the Open Stack and outputs the information bits corresponding to the path being traced is adopted. Figure 9 then implies that for $(2, 1, 12)$ convolutional code,

both the early-elimination PFSDA and the Viterbi algorithm require a path truncation window $110 \approx 8.5(12+1)$ to achieve 0.05 dB performance degradation from the ML performance

[6]For the purpose of locating the path with the smallest ending level, an additional HEAP is implemented with the key being the ending levels of the paths in the Open Stack. The maintenance complexity of the original HEAP based on path metric indices still remains logarithm of the stack size. A brief of the HEAP data structure as well as necessary modification for the PFSDA with finite stack size can be found in Appendix B.

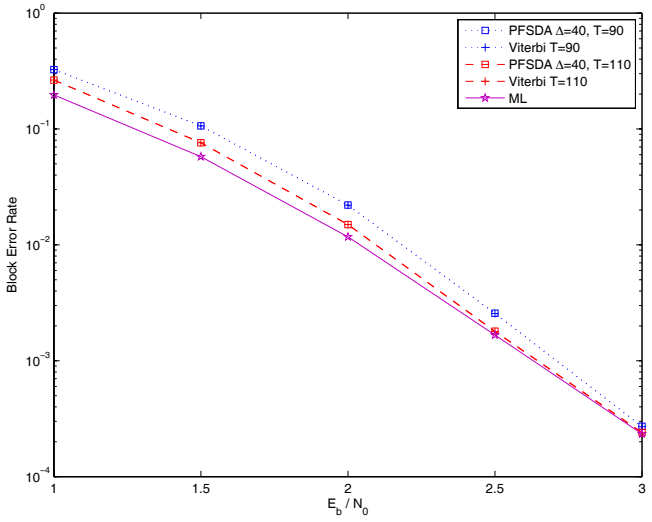

Fig. 9. Performance for (2,1,12) convolutional codes decoded by the ML decoder, the Viterbi decoder with truncation window T, and the PFSDA with early-elimination window $\Delta = 40$ and truncation window T under AWGN channels.

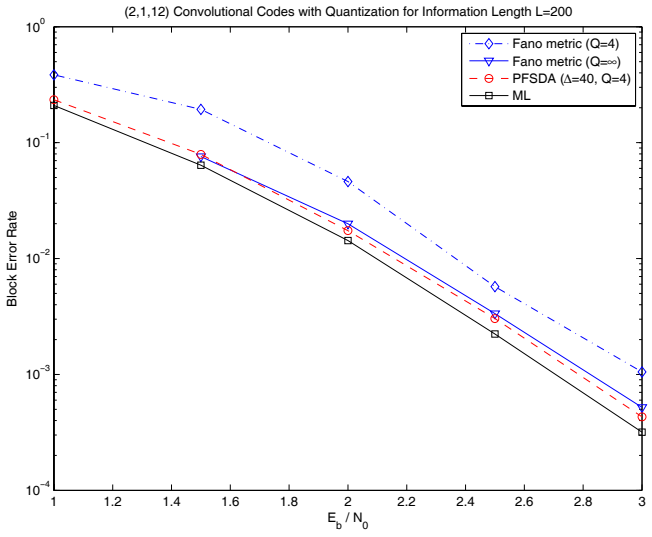

Fig. 10. Performance with metric quantization for (2,1,12) convolutional codes decoded by the sequential decoding algorithm and the PFSDA with early elimination under AWGN channels. The number of quantization levels is 2^Q.

at block error rate (BLER) $= 10^{-2}$. Our result agrees with the conclusion in [12] where the path truncation window of $5.8(m+1)$ as suggested by Forney [4] may not be enough to secure near-optimal performance under AWGN channels. In particular, by simulations, Onyszchuk [12] suggested a path truncation window of $59 \approx 6.6(8+1)$ for $(2,1,8)$ convolutional codes, and hint that the ratio of the near-optimal path truncation window against the code constraint length may further increase for codes of larger constraint length.

One of the advantages of the PFSDA in distinction to the sequential decoding algorithm is that unlike the Fano metric, its decoding metric is independent of channel SNRs. In addition, its decoding metric is less sensitive to quantization distortion. The former advantage of the PFSDA clearly remains after the introduction of early elimination. Therefore, we only examine whether the insensitiveness to quantization distortion remains after applying early-elimination modification. As shown in Figure 10, the PFSDA with early elimination exhibits only 0.05 dB performance degradation under 4-bit quantization. This figure also shows that the sequential decoding algorithm with perfect knowledge on channel SNR results in about 0.3 dB degradation in comparison with the ML performance under 4-bit quantization. In this simulation, the uniform quantization method[7] adopted follows from [13]. Since the decoding metric of the PFSDA is irrelevant to channel SNRs, no further performance degradation will be produced in the particular situation where channel SNRs cannot be accurately estimated.

V. Concluding Remarks

In this work, we propose to improve the computational complexity and memory requirement of the priority-first search decoding algorithm by early elimination. The random coding analysis of the sufficient early elimination window for exponentially negligible performance degradation, as well as the subsequent simulations, confirms our anticipated improvement.

Since our result justifies the fitness of the PFSDA with early elimination for applications that dictate near-ML performance with limited support in computational power and memory, future research of practical interest could apply the PFSDA with early elimination to sphere detector [9][10] as well as joint multi-path channel equalization and convolution decoding [6]. In addition, the hardware implementation of the PFSDA may become feasible with early elimination, and hence, could be another future work of practical interest.

Appendix A
Analysis of the Window Size With Exponentially Negligible Performance Degradation

In the analysis of near-optimal early-elimination window, we first observe, as exemplified in Figure 1, that the current top path that ends at node B with label $\boldsymbol{x}_{(\ell n-1)}$ is early-eliminated if, and only if, node C is expanded earlier than node B, provided that $\ell \le \ell_{\max} - \Delta$. It then follows from the PFSDA algorithm and the nondecreasingness of the path metric along the path to be searched that node C being expanded earlier than node B implies that

$$\mu\left(\boldsymbol{x}_{(\ell n-1)}\right) \ge \mu\left(\tilde{\boldsymbol{x}}_{(\ell_{\max} n-1)}\right),$$

[7]The Q-bit uniform quantization for a decoding metric actually involves two steps: the first step decides the 2^Q uniform partitions on the alphabet of the received scalar, while the second step determines the Q-bit representative metric value for each partition. By following [13], the step size of the uniform partitions that maximize the cutoff rate of the resultant 2-input-2^Q-output quantized channel is adopted, which, for $E_b/N_0 = 1$ dB, is 0.296 in Figure 10 as the step size should be chosen for the lowest operational E_b/N_0 so that the quantizer can work well, even not optimally, for higher SNRs. For the Fano metric μ_{F}, the second step is tricky because it has to map the highly asymmetric metric values (e.g., $-12 \le \mu_{\mathrm{F}} \le 0.5$ for $Q = 4$) to either $-2^{Q-1}, \ldots, 0, \ldots, 2^{Q-1}-1$ or $0, 1, \ldots, 2^Q - 1$. Apparently, one has to scale, round to the nearest integer, shift and possibly clip the original metric value in order to obtain a 4-bit representative metric mapping that performs well (e.g., $\max\{0, \mathrm{Round}(\mu_{\mathrm{F}} \times 2) + 14\}$ for $Q = 4$). Note that the shifted counterpart of the decoding metric in (3) can be directly applied in the PFSDA; however, the shifted Fano metric must subtract the shift constant before it is used to locate the next path to be extended, which introduces additional subtraction operations in metric computations. Since we restrict the system to only use Q-bit arithmetic logics, the shift constant should also be a Q-bit representable number (e.g., 14 for $Q = 4$).

equivalently,

$$\sum_{j=0}^{\ell n-1}\mu(x_j)\geq\sum_{j=0}^{\ell_{\max}n-1}\mu(\tilde{x}_j), \tag{6}$$

where $\tilde{\boldsymbol{x}}_{(\ell_{\max}n-1)}$ labels the path ending at node C. By noting that (6) can be rewritten using $\mu(x_j)=(x_j-y_j)\phi_j$ and $\mu(\tilde{x}_j)=(\tilde{x}_j-y_j)\phi_j$ as:

$$\sum_{j=0}^{\ell n-1}(\tilde{x}_j-x_j)\phi_j+\sum_{j=\ell n}^{\ell_{\max}n-1}\tilde{x}_j\phi_j-\sum_{j=\ell n}^{\ell_{\max}n-1}y_j\phi_j\leq 0,$$

and by reformuating the unequal-length log-likelihood ratio as:

$$\begin{aligned}
&\log\frac{f(\boldsymbol{r}_{(\ell n-1)}|\boldsymbol{x}_{(\ell n-1)})}{f(\boldsymbol{r}_{(\ell_{\max}n-1)}|\tilde{\boldsymbol{x}}_{(\ell_{\max}n-1)})}\\
&=\sum_{j=0}^{\ell n-1}[\log f(r_j|v_j=0)-x_j\phi_j]\\
&\qquad-\sum_{j=0}^{\ell_{\max}n-1}[\log f(r_j|v_j=0)-\tilde{x}_j\phi_j]\\
&=\sum_{j=0}^{\ell n-1}(\tilde{x}_j-x_j)\phi_j+\sum_{j=\ell n}^{\ell_{\max}n-1}\tilde{x}_j\phi_j\\
&\qquad-\sum_{j=\ell n}^{\ell_{\max}n-1}\log f(r_j|v_j=0),
\end{aligned}$$

we found that (6) is equivalent to:

$$\begin{aligned}
&\Phi_1(\boldsymbol{r}_{(\ell n,\ell_{\max}n-1)})\cdot f\left(\boldsymbol{r}_{(\ell n-1)}\middle|\,\boldsymbol{x}_{(\ell n-1)}\right)\\
&\qquad\leq f\left(\boldsymbol{r}_{(\ell_{\max}n-1)}\middle|\,\tilde{\boldsymbol{x}}_{(\ell_{\max}n-1)}\right), \quad (7)
\end{aligned}$$

where $f(\cdot|\cdot)$, ϕ_j and y_j have been defined in (1), and

$$\begin{aligned}
&\log\Phi_1(\boldsymbol{r}_{(\ell n,\ell_{\max}n-1)})\\
&\triangleq\sum_{j=\ell n}^{\ell_{\max}n-1}[(1-y_j)\log f(r_j|v_j=0)+y_j\log f(r_j|v_j=1)]\\
&=\sum_{j=\ell n}^{\ell_{\max}n-1}\log\left(\max\{f(r_j|v_j=0),f(r_j|v_j=1)\}\right).
\end{aligned}$$

Since the path that updates the current $\ell_{\max}$ is exactly the one with the smallest path metric among all paths ending at the same level [5], (7) can be equivalently rewritten as:

$$\begin{aligned}
&\Phi_1(\boldsymbol{r}_{(\ell n,\ell_{\max}n-1)})\cdot f\left(\boldsymbol{r}_{(\ell n-1)}\middle|\,\boldsymbol{x}_{(\ell n-1)}\right)\\
&\qquad\leq\max_{\tilde{\boldsymbol{x}}_{(\ell_{\max}n-1)}\in\mathcal{C}_{\ell_{\max}}}f\left(\boldsymbol{r}_{(\ell_{\max}n-1)}\middle|\,\tilde{\boldsymbol{x}}_{(\ell_{\max}n-1)}\right), \quad (8)
\end{aligned}$$

where $\mathcal{C}_{\ell_{\max}}$ is the set of all path labels of length $\ell_{\max}n$, whose corresponding paths consist of different branches from path AB after node A. Consequently, new decoding error (in addition to the usual ML decoding error) is introduced by early elimination if (8) is valid for some ℓ and $\ell_{\max}$, satisfying $\ell\leq\ell_{\max}-\Delta$, and the transmitted $\boldsymbol{x}$ labels the ML code path. Since our goal is to find an upper probability bound for additional decoding error due to early elimination, we can include those cases in which $\boldsymbol{x}$ does not label the ML code path when evaluating the probability of the occurrence of (8).

Continue the derivation by replacing $\ell_{\max}$ with β to simplify the notation. The probability $\xi(\ell,\beta)$ that (8) occurs is given by:

$$\xi(\ell,\beta)=\int_{\Re^{\beta n}}\Phi_0\left(\boldsymbol{r}_{(\beta n-1)}\right)f\left(\boldsymbol{r}_{(\beta n-1)}\middle|\,\boldsymbol{x}_{(\beta n-1)}\right)d\boldsymbol{r}_{(\beta n-1)},$$

where $\Phi_0\left(\boldsymbol{r}_{(\beta n-1)}\right)=1$ if (8) is valid, and 0, otherwise. From

$$\begin{aligned}
&\Phi_0\left(\boldsymbol{r}_{(\beta n-1)}\right)\\
&\leq\left[\frac{\sum_{\tilde{\boldsymbol{x}}_{(\beta n-1)}\in\mathcal{C}_\beta}f\left(\boldsymbol{r}_{(\beta n-1)}\middle|\,\tilde{\boldsymbol{x}}_{(\beta n-1)}\right)^{1/(1+\rho)}}{\Phi_1(\boldsymbol{r}_{(\ell n,\beta n-1)})^{1/(1+\rho)}f\left(\boldsymbol{r}_{(\ell n-1)}\middle|\,\boldsymbol{x}_{(\ell n-1)}\right)^{1/(1+\rho)}}\right]^{\rho}\\
&\qquad\text{for }\rho\geq 0,
\end{aligned}$$

we obtain:

$$\begin{aligned}
&\xi(\ell,\beta)\\
&\leq\int_{\Re^{\beta n}}\left[\frac{\sum_{\tilde{\boldsymbol{x}}_{(\beta n-1)}\in\mathcal{C}_\beta}f\left(\boldsymbol{r}_{(\beta n-1)}\middle|\,\tilde{\boldsymbol{x}}_{(\beta n-1)}\right)^{1/(1+\rho)}}{\Phi_1(\boldsymbol{r}_{(\ell n,\beta n-1)})^{1/(1+\rho)}f\left(\boldsymbol{r}_{(\ell n-1)}\middle|\,\boldsymbol{x}_{(\ell n-1)}\right)^{1/(1+\rho)}}\right]^{\rho}\\
&\quad\times f\left(\boldsymbol{r}_{(\beta n-1)}\middle|\,\boldsymbol{x}_{(\beta n-1)}\right)d\boldsymbol{r}_{(\beta n-1)}.
\end{aligned}$$

Taking the expected value of $\xi(\ell,\beta)$ with respect to random selection of codewords of length βn according to code bit selection distribution $\boldsymbol{p}=(p_0,p_1)$, where p_0 and p_1 are the probabilities respectively for bits 0 and 1, yields that:

$$\begin{aligned}
\overline{\xi(\ell,\beta)}&\leq\int_{\Re^{\beta n}}\Phi_1(\boldsymbol{r}_{(\ell n,\beta n-1)})^{-\rho/(1+\rho)}\\
&\quad\times\overline{\left(\sum_{\tilde{\boldsymbol{x}}_{(\beta n-1)}\in\mathcal{C}_\beta}f\left(\boldsymbol{r}_{(\beta n-1)}\middle|\,\tilde{\boldsymbol{x}}_{(\beta n-1)}\right)^{1/(1+\rho)}\right)^{\rho}}\\
&\quad\times\overline{f\left(\boldsymbol{r}_{(\ell n-1)}\middle|\,\boldsymbol{x}_{(\ell n-1)}\right)^{1/(1+\rho)}}\\
&\quad\times\overline{f\left(\boldsymbol{r}_{(\ell n,\beta n-1)}\middle|\,\boldsymbol{x}_{(\ell n,\beta n-1)}\right)}d\boldsymbol{r}_{(\beta n-1)} \quad (9)\\
&\leq\int_{\Re^{\beta n}}\Phi_1(\boldsymbol{r}_{(\ell n,\beta n-1)})^{-\rho/(1+\rho)}\\
&\quad\times\left(\sum_{\tilde{\boldsymbol{x}}_{(\beta n-1)}\in\mathcal{C}_\beta}\overline{f\left(\boldsymbol{r}_{(\beta n-1)}\middle|\,\tilde{\boldsymbol{x}}_{(\beta n-1)}\right)^{1/(1+\rho)}}\right)^{\rho}\\
&\quad\times\overline{f\left(\boldsymbol{r}_{(\ell n-1)}\middle|\,\boldsymbol{x}_{(\ell n-1)}\right)^{1/(1+\rho)}}\\
&\quad\times\overline{f\left(\boldsymbol{r}_{(\ell n,\beta n-1)}\middle|\,\boldsymbol{x}_{(\ell n,\beta n-1)}\right)}d\boldsymbol{r}_{(\beta n-1)} \quad (10)\\
&=|\mathcal{C}_\beta|^{\rho}\int_{\Re^{\beta n}}\Phi_1(\boldsymbol{r}_{(\ell n,\beta n-1)})^{-\rho/(1+\rho)}
\end{aligned}$$

$$\times \left(\overline{f\left(\boldsymbol{r}_{(\beta n-1)} \middle| \tilde{\boldsymbol{x}}_{(\beta n-1)}\right)^{1/(1+\rho)}} \right)^{\rho}$$

$$\times \overline{f\left(\boldsymbol{r}_{(\ell n-1)} \middle| \boldsymbol{x}_{(\ell n-1)}\right)^{1/(1+\rho)}}$$

$$\times \overline{f\left(\boldsymbol{r}_{(\ell n,\beta n-1)} \middle| \boldsymbol{x}_{(\ell n,\beta n-1)}\right)} d\boldsymbol{r}_{(\beta n-1)}$$

$$= |\mathcal{C}_\beta|^\rho \Bigg\{ \int_{\Re^{\ell n}} \left(\overline{f\left(\boldsymbol{r}_{(\ell n-1)} \middle| \tilde{\boldsymbol{x}}_{(\ell n-1)}\right)^{1/(1+\rho)}} \right)^{\rho}$$

$$\times \overline{f\left(\boldsymbol{r}_{(\ell n-1)} \middle| \boldsymbol{x}_{(\ell n-1)}\right)^{1/(1+\rho)}} d\boldsymbol{r}_{(\ell n-1)} \Bigg\}$$

$$\times \Bigg\{ \int_{\Re^{(\beta-\ell)n}} \Phi_1(\boldsymbol{r}_{(\ell n,\beta n-1)})^{-\rho/(1+\rho)}$$

$$\times \left(\overline{f\left(\boldsymbol{r}_{(\ell n,\beta n-1)} \middle| \tilde{\boldsymbol{x}}_{(\ell n,\beta n-1)}\right)^{1/(1+\rho)}} \right)^{\rho}$$

$$\times \overline{f\left(\boldsymbol{r}_{(\ell n,\beta n-1)} \middle| \boldsymbol{x}_{(\ell n,\beta n-1)}\right)} d\boldsymbol{r}_{(\ell n,\beta n-1)} \Bigg\},$$

where (9) holds since any labels in $\mathcal{C}_\beta$ are selected independently, and (10) is valid due to the concavity of function $\varphi(x) = x^\rho$ with $0 \le \rho \le 1$ and Jensen's inequality. Finally, by noting that $|\mathcal{C}_\beta| \le 2^{k\beta} = 2^{n\beta R}$, we obtain:

$$\overline{\xi(\ell,\beta)} \le 2^{-\ell n[-\rho R + E_0(\rho,\boldsymbol{p})]} \cdot 2^{-(\beta-\ell)n[-\rho R + E_1(\rho,\boldsymbol{p})]}, \quad (11)$$

where

$$E_0(\rho,\boldsymbol{p}) \triangleq -\log_2 \left[\int_{\Re} \left(\sum_{i=0}^{1} p_i f(r|v=i)^{1/(1+\rho)} \right)^{1+\rho} dr \right]$$

and

$$E_1(\rho,\boldsymbol{p}) \triangleq -\log_2 \left[\int_{\Re} \left(\max\{f(r|v=0), f(r|v=1)\}\right)^{-\rho/(1+\rho)} \times \left(\sum_{i=0}^{1} p_i f(r|v=i)^{\frac{1}{(1+\rho)}} \right)^{\rho} \left(\sum_{i=0}^{1} p_i f(r|v=i) \right) dr \right].$$

Inequality (11) provides an upper probability bound that a top path ending at level ℓ is early-eliminated. Based on (11), we can proceed to derive the bound for the probability $P_{\rm el}$ that an incorrect codeword is claimed at the end of the priority-first search because the transmitted path is early-eliminated during the decoding process.

As only linear codes are considered in this paper, we may assume without loss of generality that the all-zero codeword $\mathbf{0}$ is transmitted. Then,

$$P_{\rm el} \le \Pr\left(\bigcup_{\ell=1}^{L-\Delta} \mathbf{0}_{n\ell-1} \text{ is early-eliminated} \right) \le \sum_{\ell=1}^{L-\Delta} 2^{-\ell n[-\rho R + E_0(\rho)]} 2^{-\Delta n[-\rho R + E_1(\rho)]}, \quad (12)$$

where the last inequality follows from (11) by letting $E_0(\rho) \triangleq \max_{\boldsymbol{p}} E_0(\rho,\boldsymbol{p}) = E_0(\rho,\boldsymbol{p}^*)$ and $E_1(\rho) \triangleq E_1(\rho,\boldsymbol{p}^*)$, and also from the fact that $\beta - \ell \ge \Delta$. Denoting $\lambda \triangleq E_0(\rho) - \rho R$, we continue the derivation from (12):

$$\begin{aligned} P_{\rm el} &\le 2^{-\Delta n[-\rho R + E_1(\rho)]} \sum_{\ell=1}^{L-\Delta} 2^{-\ell n\lambda} \\ &\le 2^{-\Delta n[-\rho R + E_1(\rho)]} \sum_{\ell=1}^{\infty} 2^{-\ell n\lambda} \\ &= K_n \cdot 2^{-\Delta n[-\rho R + E_1(\rho)]}, \end{aligned}$$

where $K_n = 2^{-n\lambda}/(1-2^{-n\lambda})$ is a constant, independent of Δ. Consequently,

$$\begin{aligned} \liminf_{n\to\infty} -\frac{1}{n}\log_2 P_{\rm el} &\ge \Delta[-\rho R + E_1(\rho)] + \lambda \\ &\ge \Delta[-\rho R + E_1(\rho)], \end{aligned}$$

subject to $\lambda = -\rho R + E_0(\rho) > 0$ over $0 \le \rho \le 1$, which immediately implies:

$$\liminf_{n\to\infty} -\frac{1}{n}\log_2 P_{\rm el} \ge \Delta \cdot E_{\rm el}(R), \quad (13)$$

where

$$E_{\rm el}(R) \triangleq \max_{\{\rho\in[0,1]\,:\,E_0(\rho)>\rho R\}} [-\rho R + E_1(\rho)]. \quad (14)$$

Note that by

$$\begin{aligned} &\sum_{i=0}^{1} p_i f(r|v=i)^{1/(1+\rho)} \\ &= \sum_{i=0}^{1} p_i f(r|v=i) f(r|v=i)^{-\rho/(1+\rho)} \\ &\ge \left(\max\{f(r|v=0), f(r|v=1)\}\right)^{-\rho/(1+\rho)} \times \left(\sum_{i=0}^{1} p_i f(r|v=i) \right), \end{aligned}$$

we obtain that $E_1(\rho) \ge E_0(\rho)$; hence, $E_{\rm el}(R) \ge 0$.

It is noteworthy that the above analysis can also be applied to discrete channels by replacing $f(\cdot|\cdot)$ with the binary-input-and-J-ary-output channel probability mass function. In such case, (13) is valid in the sense that $E_{\rm el}(R)$ is redefined according to

$$E_0(\rho,\boldsymbol{p}) \triangleq -\log_2 \left[\sum_{j=1}^{J} \left(\sum_{i=0}^{1} p_i \Pr(r=j|v=i)^{1/(1+\rho)} \right)^{1+\rho} \right]$$

and

$$E_1(\rho, \boldsymbol{p}) \triangleq -\log_2 \Bigg[\sum_{j=1}^{J} \left(\max\left\{\Pr(r=j|v=0), \Pr(r=j|v=1)\right\}\right)^{-\rho/(1+\rho)} \times \left(\sum_{i=0}^{1} p_i \Pr(r=j|v=i)^{\frac{1}{(1+\rho)}}\right)^{\rho} \times \left(\sum_{i=0}^{1} p_i \Pr(j=j|v=i)\right) \Bigg].$$

After the establishment of the lower bound of additional decoding error exponent introduced by early elimination, we can subsequently quote the result from [14] that the exact error exponent of the maximum-likelihood decoding error for convolutional codes of rates above channel cutoff rate R_0 is given by $(m+1)E_{\rm c}(R)$, where

$$E_{\rm c}(R) \triangleq \max_{\{\rho \in [0,1] \,:\, E_0(\rho) > \rho R\}} E_0(\rho). \tag{15}$$

We then follow a similar argument in [4] to conclude that for code rates above channel cutoff rate R_0, the additional decoding error due to early elimination in the PFSDA becomes exponentially negligible[8] if

$$\Delta \cdot E_{\rm el}(R) > (m+1)E_{\rm c}(R). \tag{17}$$

Appendix B
A Brief Introduction of HEAP Data Structure

HEAP is a tree-based data structure with a key assigned to each node. It satisfies the following two properties: i) If A is a child node of B, then the key associated with A is no less than the key associated with B, and ii) all leaves are located at the same tree level, and the leaf nodes are filled from left to right. The first property implies that the root node is always the one with the smallest key. Since the PFSDA needs to extend the node with the smallest metric, this makes the HEAP a suitable structure for its implementation.

A binary HEAP tree can be easily implemented using an array structure. As a simple example shown in Figure 11, the indexes of the parent, left child and right child of a node with index i are given by $\lfloor (i-1)/2 \rfloor$, $2i+1$ and $2i+2$, respectively. Insertion and deletion of a node can then be done over the array. Specifically, the insertion process will first place the inserted node x at the end of the array, and then repeat exchanging the new node with its parent until either its key is no less than the key of its parent or x becomes a root node. The deletion of an existing node y requires three steps: i) Set the key of the node to be deleted as $-\infty$ (the minimum value), and then keep exchanging the node with its parent until it becomes a root node; ii) Exchange the root node y with the last node z in the array, and then delete y; iii) Repeat exchanging z with its child of smaller key until either no children have a smaller key than z or z becomes a leaf. Both processes can be completed within $O(\log s)$ exchanges, where s is the number of nodes in the current HEAP.

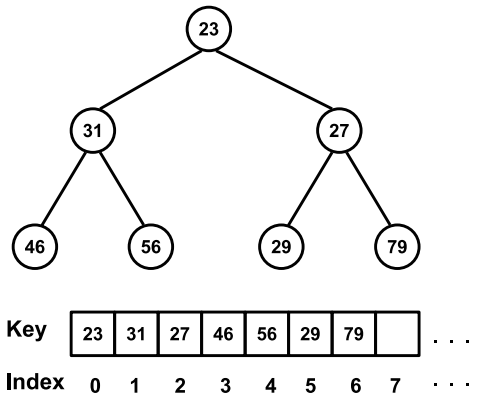

Fig. 11. Example of a binary HEAP and its respective array.

Now, for the PFSDA with finite stack size, the path with the smallest level should be efficiently located and deleted once the stack size exceeds its limit. To fulfill this goal, two binary HEAPs are implemented for the Open Stack. The metric HEAP uses the path metric as its key, while the level HEAP associates its nodes with level keys. An additional mutual link is then maintained for each node in these two HEAPs. In other words, the node in the metric HEAP will be linked to its counterpart in the level HEAP, and vice versus. By this way, the PFSDA can efficiently locate, for example, the node with the smallest level, and delete it as well as its counterpart in the metric HEAP within $O(\log s)$ exchanges.

[8]For rate below the cutoff rate, an early elimination window Δ that sufficiently guarantees exponentially negligible performance degradation can be obtained using the sphere-packing bound $E_{\rm sp}(R)$ and straight-line bound $E_{\rm x}$ [14] as:

$$\Delta \cdot E_{\rm el}(R) > (m+1)\min\{E_{\rm sp}(R), E_{\rm x}\} \tag{16}$$

where $E_{\rm sp}(R) \triangleq \max_{\{\rho \in [0,\infty) \,:\, \hat{E}_0(\rho) > \rho R\}} \hat{E}_0(\rho)$ with $\hat{E}_0(\rho)$ being the concave hull of $E_0(\rho)$, and $E_{\rm x} = 1/[4\sigma^2 \log(2)]$ for antipodal-input AWGN channels. These two bounds may suggest a slightly larger window size than the one derived from (17) at $R = R_0$. As an example from Figure 2, (16) indicates that for $(2, 1, 12)$ convolutional codes, Δ could be as large as 45, while (17) only requires $\Delta = 40$ at $R = R_0$. Our simulations however consistently show that taking Δ values that equate (17) at $R = R_0$ are already sufficient to secure ML performance even for $E_{\rm b}/N_0$ outside the valid region of (17) (cf. Figures 4 and 5), and (16) indeed overestimates the required Δ. This conforms to the general impression that the sphere-packing bound as well as the straight-line bound is perhaps loose at low rates. This paper will then focus on the Δ value derived from (17) specifically at $R = R_0$ as adopted similarly in [4].

References

[1] R. E. Blahut, *Principles and Practice of Information Theory*. Addison-Wesley Publishing Company, 1988.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.

[3] L. Ekroot and S. Dolinar, "A decoding of block codes," *IEEE Trans. Commun.*, vol. 44, pp. 1052-1056, Sep. 1996.

[4] G. D. Forney, Jr., "Convolutional codes II: maximum likelihood decoding," *Inform. Control*, 25:222-66, July 1974.

[5] Y. S. Han, P.-N. Chen, and H.-B. Wu, "A maximum-likelihood soft-decision sequential decoding algorithm for binary convolutional codes," *IEEE Trans. Commun.*, vol. 50, no. 2, pp. 173-178, Feb. 2002.

[6] C. Heegard, S. Coffey, S. Gummadi, E. J. Rossin, M. B. Shoemake and M. Wilhoyte, "Combined equalization and decoding for IEEE 802.11b devices," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 2, pp. 125-138, Feb. 2003.

[7] F. Hemmati and D. J. Costello, Jr., "Truncation error probability in Viterbi decoding," *IEEE Trans. Commun.*, vol. 25, no. 5, pp. 530-532, May 1977.
[8] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, 2nd edition. Upper Saddle River, NJ: Pearson Prentice-Hall, 2004.
[9] M. Myllylä, J. Cavallaro, and M. Juntti, "A list sphere detector based on Dijkstra's algorithm for MIMO-OFDM systems," in *Proc. IEEE Int. Symp. Pers., Indoor, Mobile Radio Commun. (PIMRC)*, Athens, Greece, Sep. 2007.
[10] M. Myllylä, J. Cavallaro, and M. Juntti, "Implementation aspects of list sphere detector algorithms," in *Proc. IEEE Global Telecommun. Conf. (GLOBECOM)*, Washington, D.C., USA, Nov. 2007.
[11] R. J. McEliece and I. M. Onyszchuk, "Truncation effects in Viterbi decoding," in *Proc. MILCOM '89*, Oct. 1989, pp. 29.3.1-29.3.5.
[12] I. M. Onyszchuk, "Truncation length for Viterbi decoding," *IEEE Trans. Commun.*, vol. 39, no. 7, pp. 1023-1026, July 1991.
[13] I. M. Onyszchuk, K.-M. Cheung, and O. Collins, "Quantization loss in convolutional decoding," *IEEE Trans. Commun.*, vol. 41, no. 2, pp. 261-265, Feb. 1993.
[14] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 2, pp. 260-269, Apr. 1967.
[15] A. J. Viterbi and J. K. Omura, *Priciples of Digital Communications and Coding*. New York: McGraw-Hill, 1979.
[16] S. B. Wicker, *Error Control Systems for Digital Communications and Storage*. Englewood Cliffs, NJ: Prentice-Hall, 1995.

Shin-Lin Shieh was born in Kinmen, Taiwan, R.O.C., in 1977. He received the B.Sc. and M.Sc. degrees in electrical engineering from the National Tsing-Hua University, Hsinchu, Taiwan, in 1999 and 2001, respectively, and a Ph.D. degree in the department of communications engineering from National Chiao-Tung university, Hsinchu, Taiwan.

After serving in the military for three months, he joined the Wireless Communication Technology Department, Computer and Communication Laboratory (CCL), Industrial Technology Research Institute (ITRI), Taiwan, in 2002, where he was engaged in several projects on the development of baseband communication algorithms. In 2005, he switched to the WidebandCode Division Multiple Access (WCDMA) Project, Sunplus Technology Company, Ltd. He continued to work with the HT mMobile, Inc., Hsinchu, a subsidiary spun-off from the Sunplus Technology Company, Ltd. in 2008, and was involved in designing baseband algorithms of enhanced data rates for global system for mobile communications evolution (EDGE), WCDMA, high-speed downlink packet access (HSDPA), long-term evolution (LTE) and other communication systems. Started from August 2010, he joined the Graduate Institute of Communication Engineering at National Taipei university, Taipei, Taiwan, as an assistant professor.

Po-Ning Chen (S'93-M'95-SM'01) was born in Taipei, R.O.C. in 1963. He received the B.S. and M.S. degrees in electrical engineering from the National Tsing-Hua University, Taiwan in 1985 and 1987, respectively, and the Ph.D. degree in electrical engineering from University of Maryland, College Park, in 1994.

From 1985 to 1987, he was with Image Processing Laboratory in National Tsing-Hua University, where he worked on the recognition of Chinese characters. During 1989, he was with Star Tech. Inc., where he focused on the development of finger-print recognition systems. After the reception of Ph.D. degree in 1994, he jointed Wan Ta Technology Inc. as a vice general manager, conducting several projects on Point-of-Sale systems. In 1995, he became a research staff in Advanced Technology Center, Computer and Communication Laboratory, Industrial Technology Research Institute in Taiwan, where he led a project on Java-based Network Managements. Since 1996, he has been an Associate Professor in the Department of Communications Engineering at the National Chiao-Tung University, Taiwan, and was promoted to a full professor since 2001. He was elected to be the Chair of the IEEE Communications Society Taipei Chapter in 2006 and 2007, during which the IEEE ComSoc Taipei Chapter has won the 2007 IEEE ComSoc Chapter Achievement Awards (CAA) and 2007 IEEE ComSoc Chapter of the Year (CoY).

He has served as the chairman of the Department of Communications Engineering, National Chiao-Tung University, during 2007-2009. Dr. Chen received the annual Research Awards from the National Science Council, Taiwan, R.O.C., five years in a row since 1996. He then received the 2000 Young Scholar Paper Award from Academia Sinica, Taiwan. His Experimental Handouts for the course of Communication Networks Laboratory have been awarded as the Annual Best Teaching Materials for Communications Education by the Ministry of Education, Taiwan, R.O.C., in 1998. He has been selected as the Outstanding Tutor Teacher of the National Chiao-Tung University in 2002. He was also the recipient of the Distinguished Teaching Award from the College of Electrical and Computer Engineering, National Chiao-Tung University, Taiwan, in 2003. His research interests generally lie in information and coding theory, large deviation theory, distributed detection and sensor networks.

Yunghsiang S. Han was born in Taipei, Taiwan, on April 24, 1962. He received B.Sc. and M.Sc. degrees in electrical engineering from the National Tsing Hua University, Hsinchu, Taiwan, in 1984 and 1986, respectively, and a Ph.D. degree from the School of Computer and Information Science, Syracuse University, Syracuse, NY, in 1993.

He was from 1986 to 1988 a lecturer at Ming-Hsin Engineering College, Hsinchu, Taiwan. He was a teaching assistant from 1989 to 1992, and a research associate in the School of Computer and Information Science, Syracuse University from 1992 to 1993. He was, from 1993 to 1997, an Associate Professor in the Department of Electronic Engineering at Hua Fan College of Humanities and Technology, Taipei Hsien, Taiwan. He was with the Department of Computer Science and Information Engineering at National Chi Nan University, Nantou, Taiwan from 1997 to 2004. He was promoted to Professor in 1998. He was a visiting scholar in the Department of Electrical Engineering at University of Hawaii at Manoa, HI from June to October 2001, the SUPRIA visiting research scholar in the Department of Electrical Engineering and Computer Science and CASE center at Syracuse University, NY from September 2002 to January 2004, and the visiting scholar in the Department of Electrical and Computer Engineering at University of Texas at Austin, TX from August 2008 to June 2009. He was with the Graduate Institute of Communication Engineering at National Taipei University, Taipei, Taiwan from August 2004 to July 2010. From August 2010, he is with the Department of Electrical Engineering at National Taiwan University of Science and Technology. His research interests are in error-control coding, wireless networks, and security. Dr. Han was a winner of the 1994 Syracuse University Doctoral Prize.

Ting-Yi Wu was born in Tainan, Taiwan, in 1983. He received the B.Sc. and M.Sc. degrees in Computer Science and Information Engineering from National Chi-Nan University, Nantou, Taiwan, in 2005 and 2007, respectively. From 2007 to 2009, he was a research assistant of the Graduate Institute of Communication Engineering, National Taipei University, Taipei, Taiwan. He is currently pursuing the Ph.D. degree in Institute of Communications Engineering, National Chiao-Tung University, Hsinchu, Taiwan. His current research interests include error-control coding and information theory.