Transactions Letters

A Maximum-Likelihood Soft-Decision Sequential Decoding Algorithm for Binary Convolutional Codes

Yunghsiang S. Han, *Member, IEEE*, Po-Ning Chen, *Senior Member, IEEE*, and Hong-Bin Wu

***Abstract*—In this letter, we present a trellis-based maximum-likelihood soft-decision sequential decoding algorithm (MLSDA) for binary convolutional codes. Simulation results show that, for (2, 1, 6) and (2, 1, 16) codes antipodally transmitted over the AWGN channel, the average computational effort required by the algorithm is several orders of magnitude less than that of the Viterbi algorithm. Also shown via simulations upon the same system models is that, under moderate SNR, the algorithm is about four times faster than the conventional sequential decoding algorithm (i.e., stack algorithm with Fano metric) having comparable bit-error probability.**

***Index Terms*—Coding, convolutional codes, decoding, maximum-likelihood, sequential decoding, soft-decision.**

I. Introduction

THE convolutional coding technique is designed to reduce the probability of erroneous transmission over noisy communication channels. A popular decoding algorithm for convolutional codes is the Viterbi algorithm, which is known to be a maximum-likelihood (ML) decoder [9]. Although widely adopted in practice, the Viterbi algorithm suffers from a decoding complexity with constraint length tradeoff, which prevents it from being applied to codes with long constraint length. This somewhat limits its attainable error probability, which decreases exponentially with respect to the code constraint length. Nowadays, the Viterbi algorithm is usually applied to codes with constraint length no greater than 7.

In contrast to the limitation of the Viterbi algorithm, the computational complexity of the sequential decoding algorithm is independent of the code constraint length, but becomes adaptable to the noise level [9]. Nevertheless, the sequential decoding algorithm does not perform ML decoding and thereby is only *suboptimal* in its performance.[1] More improved versions of the sequential decoding algorithm, such as the multiple stack algorithm and the creeper algorithm, can be found in [7] and [9].

In this work, we replace the Fano metric employed in the conventional sequential decoding algorithm by a metric defined based on a variation of the Wagner rule [11]. This results in a new sequential decoding algorithm, which performs *ML* decoding. By the nonnegativity of the new metric, the new sequential decoding algorithm can be made to operate on a code trellis, instead of a code tree on which the conventional sequential decoding algorithm operates. As a consequence, the worst-case computational complexity of the new sequential decoding algorithm, which is defined as the *maximum* number of metric values possibly evaluated, equals the overall branch number of a trellis. Note that the constant computational effort of the Viterbi algorithm is exactly equal to the number of trellis branches. Simulation results show that, for binary (2, 1, 6) convolutional code antipodally transmitted over the AWGN channel, the average number of metric values evaluated by the proposed algorithm is several orders of magnitude less than that of the Viterbi algorithm, when the length of the information bits is 40 and the SNR per information bit (SNR_b) is greater than 3 dB.

It needs to be pointed out that the computational effort of the sequential decoding algorithm is not only determined by the average number of metrics computed, but also decided by the cost of searching and inserting of the stack elements. The latter cost, however, can be made of the same order as the former one [3]. It is therefore justified to consider the metric computation of the sequential decoding algorithm as the key determinant of algorithmic complexity.

We also compare the new ML sequential decoding algorithm with the conventional one, i.e., the stack algorithm with Fano metric. A fair comparison in computational efforts should be proceeded under the premise that both algorithms yield (almost) the same error probability. By simulation results, we obtain that for binary (2, 1, 6) code, the average number of metric values evaluated by the conventional sequential decoding algorithm is four times larger than that of the new ML sequential

Paper approved by M. Fossorier, the Editor for Coding and Communication Theory of the IEEE Communications Society. Manuscript received May 2, 1999; revised August 17, 2000, and June 2, 2001. This work was supported by the National Science Council, Taiwan, R.O.C., under Projects NSC 88-2219-E-009-004 and NSC 88-2213-E-260-006. The paper was presented in part at the 1999 International Symposium on Communications, Kaohsiung, Taiwan, R.O.C., November 1999, and also in the recent result session of the 1998 IEEE International Symposium on Information Theory, Cambridge, MA, August 1998.

Y. S. Han is with the Department of Computer Science and Information Engineering, National Chi Nan University, Nan Tou, Taiwan, 545, R.O.C. (e-mail: yshan@csie.ncnu.edu.tw).

P.-N. Chen is with the Department of Communications Engineering, National Chiao Tung University, Hsin Chu, Taiwan 30050, R.O.C. (e-mail: poning@cc.nctu.edu.tw).

H.-B. Wu is with MediaTek Inc., Taiwan, 30050, R.O.C. (e-mail: HB_Wu@mtk.com.tw).

Publisher Item Identifier S 0090-6778(02)01355-7.

[1]When the codeword length is sufficiently large, e.g., 1000 bits, the sequential decoding algorithm can actually achieve the performance of an ML decoder below cutoff rate [7], [9]. However, what concerns us in this work is the situation where the codeword length is prohibitively small, such as 80 bits in length, which, to some extent, is more feasible in practice.

decoding algorithm for all SNR_b no smaller than 5 dB. This result justifies the benefit of adopting the new metric to sequential decoding in situation where short information block length is concerned.

Of equal importance to the optimality of a decoder is the consideration of performance degradation due to practical constraint in implementation. The optimality of the new sequential decoding algorithm relies on the assumption of availability of *infinite* stack space. This assumption, however, may not be feasible due to the practical limitation on the system memory, and the system performance is expected to degrade if a finite stack size constraint is applied. A further simulation indicates that the system performance of the proposed algorithm remains almost intact for a feasible stack size constraint.

II. Maximum-Likelihood Soft-Decision Sequential Decoding

The following definitions and notations are similar to those in [1] and [9].

Let $\boldsymbol{C}$ be a binary (n, k, m) convolutional code, where m is the memory order defined as the maximum number of shift-register stages from the encoder input to the encoder output. The constraint length of $\boldsymbol{C}$ is commonly defined as $(m+1)$. Denote by $N \triangleq n(L+m)$ the codeword length of $\boldsymbol{C}$, where L is the length of the information bits.

A trellis is a structure obtained from a code tree by merging the nodes in the same state. The state associated with a node is determined by the contents of the K shift-register stages, where K is the total number of shift-register stages in an encoder. For a binary (n, k, m) convolutional code trellis, the number of states at levels m through L is 2^K, which implies that there are at most 2^K nodes on these levels. Due to node merging, there is only one terminal node in a trellis. A codeword is represented by a path from the origin node at level 0 to the terminal node at level $(L+m)$. Throughout, we will refer to a path from the origin node to the terminal node as a code path.

Let $\boldsymbol{v} \triangleq (v_0, v_1, \ldots, v_{N-1})$ be a binary codeword of an (n, k, m) convolutional code. Denote a portion of codeword $\boldsymbol{v}$ by $\boldsymbol{v}_{(b)} \triangleq (v_0, v_1, \ldots, v_b)$. Define the hard-decision sequence $\boldsymbol{y} = (y_0, y_1, \ldots, y_{N-1})$ corresponding to the received vector $\boldsymbol{r} = (r_0, r_1, \ldots, r_{N-1})$ as

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

where

$$\phi_j \triangleq \ln \frac{\Pr(r_j|0)}{\Pr(r_j|1)}. \tag{1}$$

The syndrome of $\boldsymbol{y}$ is then given by $\boldsymbol{s} \triangleq \boldsymbol{y}H^T$, where $\boldsymbol{H}$ is the parity-check matrix of $\boldsymbol{C}$. Denote by $E(\boldsymbol{s})$ the collection of all error patterns whose syndrome is $\boldsymbol{s}$. Then we obtain by the Wagner rule [11] (cf. Appendix) that the ML decoding output $\hat{\boldsymbol{v}}$ for received vector $\boldsymbol{r}$ is equal to

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

for some $\boldsymbol{e}^* \in E(\boldsymbol{s})$ satisfying

$$\sum_{j=0}^{N-1} e_j^* |\phi_j| \le \sum_{j=0}^{N-1} e_j |\phi_j|, \qquad \text{for all } \boldsymbol{e} \in E(\boldsymbol{s}). \tag{2}$$

We therefore define a new metric as follows.

Definition 1: For any path

$$\boldsymbol{x}_{(\ell n-1)} = (x_0, x_1, \ldots, x_{\ell n-1})$$

ending at level ℓ in a trellis, define the metric associated with it as

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

where $M(x_j) \triangleq (y_j \oplus x_j)|\phi_j|$ is the bit metric.

The path metric defined here is actually equivalent to that given in [3] and [5], which was originally defined over code trees of block codes.

It can be verified from the above definition that finding the $\boldsymbol{e}^*$ in (2) is equivalent to finding the code path $\boldsymbol{v}$ with the smallest metric, since $\boldsymbol{e} = \boldsymbol{y} \oplus \boldsymbol{v} \in E(\boldsymbol{s})$. We thus propose to replace the Fano metric in the conventional sequential decoding algorithm by the new metric and establish a new sequential-type ML decoder (cf. Theorem 1). For convenience, we will refer to the new algorithm as the ML sequential decoding algorithm (MLSDA).

Owing to the nonnegativity of the new metric, the metric is nondecreasing along any path in a trellis (cf. Lemma 1). Accordingly, the MLSDA can be made to operate on a trellis, instead of a code tree on which the conventional stack algorithm operates. This is achieved by introducing a second stack. Note that the Fano metric along a path in a code tree could be locally decreasing in its value.

Equipped with the above definitions and notations, we are now ready to present the MLSDA.

The Trellis-Based MLSDA:

- Step 1: Load the Open Stack with the origin node whose metric is assigned zero.
- Step 2: Compute the metrics of the successors of the top path in the Open Stack and put the ending state and the ending level of this top path into the Closed Stack. Delete the top path from the Open Stack.
- Step 3: Whenever any of the new paths (i.e., the successors of the top path in the Open Stack in Step 2) merges[2] with a path already in the Open Stack, eliminate the one with higher metric value. If any of the new paths merges with a path already in the Closed Stack, discard the new path.
- Step 4: Insert the remaining new paths into the Open Stack and reorder the Open Stack according to ascending metric values.
- Step 5: If the top path in the Open Stack ends at the terminal node in the trellis, the algorithm stops; otherwise go to Step 2.

[2]By "merging," we mean that the two paths have the same ending state and ending level, or equivalently, they end at the same node in a trellis.

As illustrated in the previous algorithm, the *Open Stack* contains all paths having been explored by the MLSDA, which are not the prefix of all other paths in the Open Stack. Apparently, the *Open Stack* functions in a similar fashion as the stack in the conventional sequential decoding algorithm. The *Closed Stack* stores the information of ending states and ending levels of the paths, which had been the top paths of the *Open Stack* at some previous time.

We next prove the optimality of the MLSDA, i.e., the sequential-type searching in the above algorithm can locate the code path with the smallest metric.

Lemma 1: The metric is a nondecreasing function along any path in a trellis.

Proof: The lemma follows directly from the nonnegativity of the bit metric. ∎

The following lemma assures that the function in Step 3 will not eliminate the ML codeword.

Lemma 2: When a new path merges with an existing path in the Closed Stack, the metric value of the new path is greater than or equal to that of the existing path.

Proof: Suppose that a new path $\boldsymbol{x}_{(\ell n-1)}$ merges with an existing path $\hat{\boldsymbol{x}}_{(\ell n-1)}$ in the Closed Stack. Then the new path $\boldsymbol{x}_{(\ell n-1)}$ must be generated from a path $\boldsymbol{x}_{(\bar{\ell} n-1)}$, where $\bar{\ell} < \ell$, which once coexisted with $\hat{\boldsymbol{x}}_{(\ell n-1)}$ in the Open Stack at some previous time. From Lemma 1, the metric value associated with a path is no greater than that of any path generated from it. Also observe that any path in the Closed Stack was once the top path in the Open Stack and among all paths in the Open Stack, the top path carries the smallest metric. The proof is then completed by noting that

$$M\left(\boldsymbol{x}_{(\ell n-1)}\right) \geq M\left(\boldsymbol{x}_{(\bar{\ell} n-1)}\right) \geq M\left(\hat{\boldsymbol{x}}_{(\ell n-1)}\right).$$

∎

Theorem 1: The MLSDA is an ML decoding algorithm.

Proof: Assume that $\boldsymbol{v}$ is the first top path in the Open Stack ending at the terminal node in the trellis. Obviously, $\boldsymbol{v}$ is a code path and the metric of $\boldsymbol{v}$ is the smallest among all paths currently in the Open Stack. By Lemma 1, the metric value associated with a path is no greater than that of any path generated from it, which implies that the metric of any other code path (which should be the offspring of a path currently in the Open Stack) can not be smaller than the metric of $\boldsymbol{v}$. Consequently, $\boldsymbol{v}$ has the smallest metric value among all code paths, which, from (2), validates the Theorem. ∎

A similar argument to that given in Theorem 1 can be used to derive the following corollary, which is useful when a truncated decoder search length [9] is implemented.

Corollary 1: If $\boldsymbol{x}_{(\ell n-1)}$ is the first top path in the Open Stack ending at level ℓ, then $\boldsymbol{x}_{(\ell n-1)}$ is the path with the smallest metric among all paths ending at level ℓ.

III. Simulation Results Over the AWGN Channel

In this section, we examine the computational effort and the bit error rate (BER) of the MLSDA by simulations over the AWGN channel. We assume that the binary codeword is antipodally transmitted. Hence for the AWGN channel, the received vector is given by

$$r_j = (-1)^{v_j}\sqrt{\varepsilon} + \lambda_j$$

for $0 \leq j \leq N-1$, where ε is the signal energy per channel bit and $\{\lambda_j\}_{j=0}^{N-1}$ are independent noise samples of a white Gaussian process with single-sided noise power per hertz N_0. The SNR for the channel is therefore $\mathrm{SNR} \triangleq \varepsilon/N_0$. In order to account for the code redundancy for different code rates, we will use the SNR per information bit, i.e.,

$$\mathrm{SNR_b} = \frac{\frac{\mathrm{N}\varepsilon}{\mathrm{kL}}}{\mathrm{N}_0} = \frac{\mathrm{N}}{\mathrm{kL}}\left(\frac{\varepsilon}{\mathrm{N}_0}\right)$$

in the following discussions.

For the AWGN channel, we can also simplify the metric associated with a path $\boldsymbol{x}_{(\ell n-1)}$ to

$$M\left(\boldsymbol{x}_{(\ell n-1)}\right) \triangleq \sum_{j=0}^{\ell n-1} (y_j \oplus x_j)|r_j|$$

where

$$y_j \triangleq \begin{cases} 1, & \text{if } r_j < 0 \\ 0, & \text{otherwise.} \end{cases}$$

As indicated in the algorithm, the computational efforts of the MLSDA are determined not only by the numbers of metrics evaluated, but also by the cost of searching and reordering of the stack elements. However, we can adopt a balanced-tree data structure [2] in the stack implementation so that the latter cost becomes of comparable order to the former one. In addition, one can further employ a hardware-based stack structure [8] and attain constant complexity on insertion operation. It is therefore justified to consider the metric computation of the MLSDA as the key determinant of algorithmic complexity.

We now turn to the empirical investigation of the average decoding complexity. Two types of binary convolutional codes are considered. One is a (2, 1, 6) code with generators 634, 564 (octal) and the other is a (2, 1, 16) code with generators 1632044, 1 145 734 (octal). The lengths of the information bits used in our simulations are 40, 100 and 200, respectively. For ease of notations, we will respectively abbreviate the Viterbi algorithm and the conventional sequential decoding algorithm as VA and SA in the sequel; also, a number enclosed by parentheses that immediately follows the acronym of an algorithm, such as MLSDA (40) and SA (200), specifically designates the length of the information bits taken for the algorithm. Without loss of generality, we only count the metric operations up to level L. For all simulations, at least 10 word errors have been reported to ensure that there is no bias on the simulation results. We summarize the simulation results in Tables I and II and Figs. 1 and 2.

In Table I, we compare the average computational efforts of the MLSDA with those of the VA and the SA. By the simulation results, when SNR_b is greater than or equal to 3 dB, the MLSDA has a much smaller average computational complexity than the Viterbi algorithm. For example, the average computational effort of the MLSDA for binary (2, 1, 6) convolutional code is about two order of magnitude smaller than that of the Viterbi algorithm when $\mathrm{SNR}_b \geq 6$ dB (cf. Table I). Indeed, when $\mathrm{SNR}_b \geq 6$ dB,

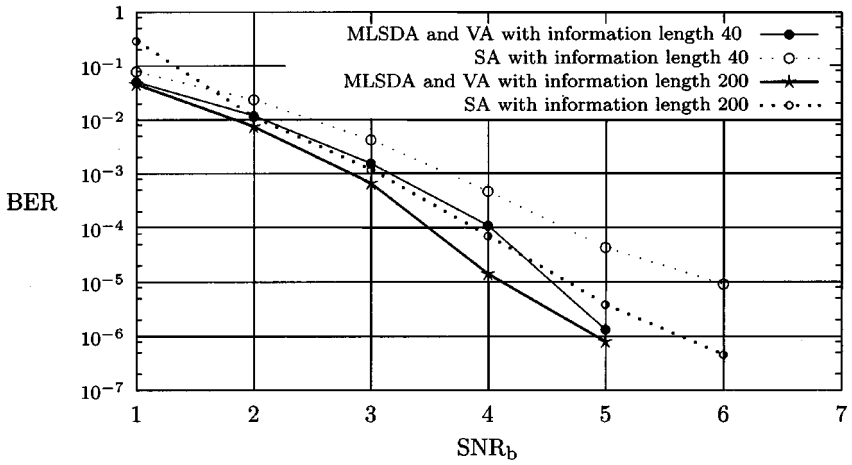

Fig. 1. BER of the MLSDA, the Viterbi algorithm (VA), and the stack algorithm (SA) for (2, 1, 6) code with information lengths 40 and 200.

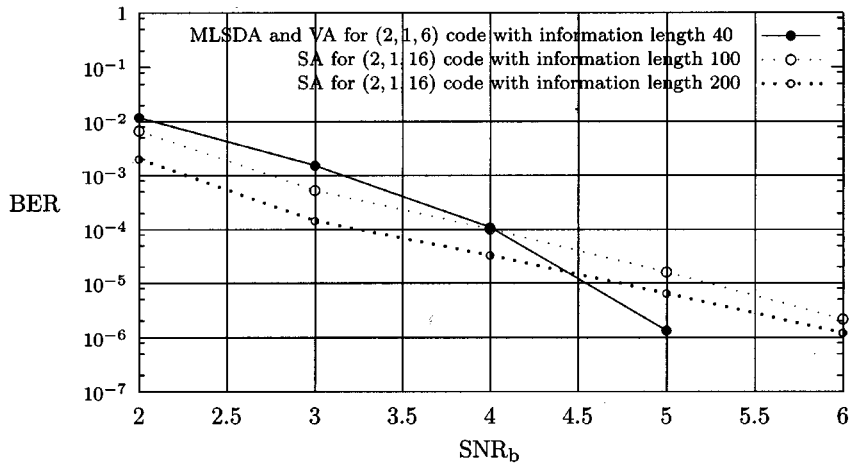

Fig. 2. BER of the MLSDA, the VA, and SA for (2, 1, 6) code and (2, 1, 16) code.

TABLE I
AVERAGE AND MAXIMUM NUMBERS OF METRIC COMPUTATION FOR (2, 1, 6) CODE. VA(L), SA(L), AND MLSDA (L), RESPECTIVELY, REPRESENT THE VITERBI ALGORITHM, THE STACK ALGORITHM WITH FANO METRIC, AND OUR PROPOSED ALGORITHM, APPLIED TO THE CODE WITH INFORMATION LENGTH L

SNR_b	1 dB		2 dB		3 dB		4 dB		5 dB		6 dB		7 dB	
	max	ave	max	ave	max	ave	max	ave	max	ave	max	ave	max	ave
VA(40)	4478	4478	4478	4478	4478	4478	4478	4478	4478	4478	4478	4478	4478	4478
SA(40)	17009	441	6511	204	8846	115	2328	89	3406	83	729	81	116	80
MLSDA(40)	4478	2124	4466	1133	4462	456	4226	178	2072	102	650	84	270	81
VA(200)	24958	24958	24958	24958	24958	24958	24958	24958	24958	24958	24958	24958	24958	24958
SA(200)	1130569	7685	68960	1037	14838	516	1572	426	1004	407	1340	402	438	401
MLSDA(200)	24958	21774	24926	18974	24916	14357	22240	7088	19734	1794	6682	556	1878	413

TABLE II
AVERAGE AND MAXIMUM NUMBERS OF METRIC COMPUTATION FOR (2, 1, 16) CODE. VA(L), SA(L), AND MLSDA (L), RESPECTIVELY, REPRESENT THE VITERBI ALGORITHM, THE STACK ALGORITHM WITH FANO METRIC, AND OUR PROPOSED ALGORITHM, APPLIED TO THE CODE WITH INFORMATION LENGTH L

SNR_b	2 dB		3 dB		4 dB		5 dB		6 dB	
	max	ave	max	ave	max	ave	max	ave	max	ave
VA(100)	11141118	11141118	11141118	11141118	11141118	11141118	11141118	11141118	11141118	11141118
SA(100)	1565327	8920	494199	677	616819	304	210884	217	45645	203
MLSDA(100)	9644668	1259192	5687634	158477	1618542	9334	284136	931	5952	285
VA(200)	24248318	24248318	24248318	24248318	24248318	24248318	24248318	24248318	24248318	24248318
SA(200)	1619241	9254	1002629	898	858849	489	156597	417	41778	404

the average number of metric values evaluated by the MLSDA reduces to $2^k L$, which is the smallest possible number of metric values evaluated by any decoding algorithm.

In Fig. 1, we compare the BER of the MLSDA with those obtained by the VA and the SA. Since both the MLSDA and the VA are ML decoders, it is reasonable that they yield the same BER. Also noted from Fig. 1 is that the MLSDA provides around a 1.5-dB advantage over the SA at $\mathrm{BER} = 10^{-5}$, when both algorithms employ the same information length 40. Even when we extend the information length of the SA to 200, the MLSDA

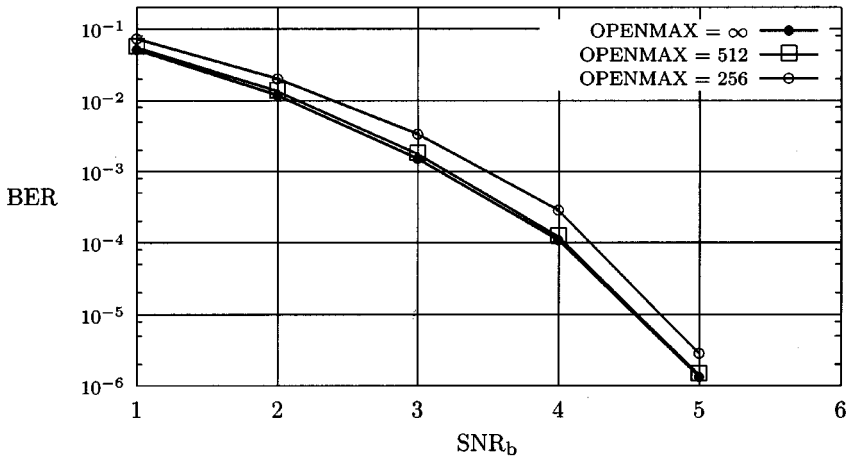

Fig. 3. BER of the MLSDA for (2, 1, 6) code with information length 40. OPENMAX is the upper limit on the size of the Open Stack.

with information length 40 still provides around a 0.5-dB advantage at $\mathrm{BER} = 10^{-6}$.

As mentioned earlier, the comparison in computational complexity between the MLSDA and the SA should be performed under a fixed BER for both algorithms. Observe from Fig. 1 that the MLSDA (40) has almost the same BER as the SA (200). We then note from Table I that the average computational complexity of the SA (200) is at least four times larger than that of the MLSDA (40) when $\mathrm{SNR_b} \geq 5$ dB. Since the memory space required by a path entry in the stack is proportional to the information length, the memory required by the SA (200) is about 20 times larger than that of the MLSDA (40).

Now by examining the simulations on both (2, 1, 6) and (2, 1, 16) codes (cf. Tables I– II), we observe that the *average* computational complexity of the SA for (2, 1, 6) code and (2, 1, 16) code are close to each other, while the *maximum* computational complexity grows as the code constraint length increase. The former observation re-confirms that the average computational complexity of the SA does not depend on the code constraint length [9]; yet, the latter observation indicates that the SA for code with large code constraint length may suffer a buffer overflow.

Fig. 2 collects the results on the MLSDA (40) for (2, 1, 6) code and the SA (100) and the SA (200) for (2, 1, 16) code. The three curves indicate that the MLSDA (40) for (2, 1, 6) code provides around a 1.0–dB advantage over the other two SAs at $\mathrm{BER} = 10^{-6}$.

To summarize from the above simulations, the conventional sequential decoding algorithm usually requires long information block length to obtain a low BER, which unavoidably results in long decoding delay and large stack space demand. By adopting a new metric in sequential decoding, the MLSDA can achieve the same performance by a shorter information block length; and hence, the decoding delay, as well as the stack space, can be fairly reduced. In additional to the above observations, we also found that the new metric adopted in the MLSDA does not rely on the knowledge (i.e., SNR) of the channel, when codes are transmitted over the AWGN channel. This somehow hints that the MLSDA and the Viterbi algorithm share a common nature that their performances are not sensitive to the accuracy of the estimation of the channel SNR. More comprehensive comparison along this line between the Viterbi algorithm and the SA can be found in [6].

The final issue that we touched on in this work is the consideration of a feasible stack-size demand. In practice, one often needs to restrict the stack size due to the limitation of the system memory. Such limitation on the stack size, although reducing the computational efforts, unavoidably degrade the system performance. Accordingly, investigation of the dependence of the system degradation, as well as the reduction of average computational complexity, on the maximum stack size allowable becomes necessary.

The Trellis-Based MLSDA With Finite Stack-Size Constraint:

Steps 1–4:

These four steps are identical to the trellis-based MLSDA without stack-size constraint.

Step 5:

If the size of the Open Stack is larger than OPENMAX, discard the path with the maximum metric in the Open Stack. Repeat this step until the size of the Open Stack is no larger than OPENMAX.

Step 6:

If the top path in the Open Stack ends at the terminal node in the trellis, the algorithm stops; otherwise go to Step 2.

We now use the binary (2, 1, 6) convolutional code with information length 40 to examine the dependence of the MLSDA performance on stack size constraint. We set two different upper limits (OPENMAX) on the size of the Open Stack, which are 512 and 256. The simulation results are depicted in Fig. 3. We conclude from this figure that the MLSDA performance remains almost intact for a feasible stack size constraint of 512. For a smaller stack size limit, such as 256, the BER of the MLSDA is still close to the ML performance. We close the discussion on the stack size constraint by remarking that since the largest size of the Closed Stack is the number of nodes in a trellis and it stores only the information of ending states and ending levels for each entry, the memory size that the Closed Stack requires

tends to be small. Accordingly, to place restriction on the Close Stack size is unnecessary because the memory consumption of the MLSDA is indeed mostly contributed by the Open Stack.

IV. Conclusion

In this work, we present an ML soft-decision sequential decoding algorithm (MLSDA) for binary convolutional codes, where a new metric other than the traditional Fano metric is used for sequential searching of codewords. By the nonnegativity of the new metric, the path metric along any path in a code trellis or code tree of a convolutional code is nondecreasing. Together with the fact that the code path with minimum path metric exactly gives the ML codeword, the MLSDA turns out to be an ML decoding algorithm for any code length as contrary to the suboptimality (or *asymptotic* optimality in code length) of the conventional sequential decoding algorithm that employs the Fano metric. Also, unlike the conventional sequential decoding algorithm which searches codewords on a code tree, the MLSDA can be made to operate on a convolutional code trellis. As a result, the trellis-based MLSDA has the same bit error probability as the Viterbi algorithm, but is with much smaller metric computational complexity. We conclude from our simulations in Section III that the computational complexity of the MLSDA is much less than that of the Viterbi algorithm, when $\mathrm{SNR_b}$ is moderately high. Under a comparable BER, the average computational complexity of the stack algorithm with Fano metric is also four times larger than that of the MLSDA for $\mathrm{SNR_b}$ no less than 5 dB. These results, to some extent, justify the benefit of adopting the new metric to the sequential decoding.

Due to the real-time demand (or memory constraint) in practical applications, the decoding system may need to force the decision on the information bit after a fixed delay, which is often referred to as truncated decoder search length or survivor truncation. It is shown in [4] that the performance degradation of the Viterbi algorithm due to survivor truncation is almost negligible, if the fixed delay is taken at leave five times of the code memory size. By Corollary 1, the MLSDA can also be applied to the situation of truncated decoder search length, where the best state decoding [10] is taken instead. It will be testified in an upcoming work that the MLSDA behaves similarly in performance degradation to the Viterbi algorithm, when the best state decoding is considered.

Appendix
Variation of Wagner Rule

From [9] (also [11]), the ML decoding output $\widehat{v}$ for a binary convolutional code $\boldsymbol{C}$ transmitted over a time-discrete memoryless channel can be given by

$$\widehat{v} = \boldsymbol{v}^*$$

where $\boldsymbol{v}^*$ satisfies

$$\sum_{j=0}^{N-1} \left(\phi_j - (-1)^{v_j^*}\right)^2 \le \sum_{j=0}^{N-1} (\phi_j - (-1)^{v_j})^2 \qquad (4)$$

for all $\boldsymbol{v} \in \boldsymbol{C}$ and ϕ_j is defined in (1). The condition in (4) can be rewritten as

$$\begin{aligned}
&\sum_{j=0}^{N-1} \left(\phi_j - (-1)^{v_j^*}\right)^2 \le \sum_{j=0}^{N-1} (\phi_j - (-1)^{v_j})^2 \\
\Leftrightarrow &\sum_{j=0}^{N-1} -(-1)^{v_j^*}\phi_j \le \sum_{j=0}^{N-1} -(-1)^{v_j}\phi_j \\
\Leftrightarrow &\frac{1}{2}\sum_{j=0}^{N-1} \left[(-1)^{y_j} - (-1)^{v_j^*}\right]\phi_j \le \frac{1}{2}\sum_{j=0}^{N-1} [(-1)^{y_j} - (-1)^{v_j}]\phi_j \\
\Leftrightarrow &\sum_{j=0}^{N-1} (y_j \oplus v_j^*)|\phi_j| \le \sum_{j=0}^{N-1} (y_j \oplus v_j)|\phi_j| \\
\Leftrightarrow &\sum_{j=0}^{N-1} e_j^*|\phi_j| \le \sum_{j=0}^{N-1} e_j|\phi_j|
\end{aligned}$$

where y_j and e_j are defined as in Section II. The alternative expression of the Wagner rule in (2) is then validated.

Acknowledgment

The authors wish to thank the anonymous reviewers for their valuable suggestions and comments that greatly helped to improve the paper. The suggestions from Dr. M. P. C. Fossorier on this work is gratefully appreciated.

References

[1] P. R. Chevillat and D. J. Costello Jr, "An analysis of sequential decoding for specific time-invariant convolutional codes," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 443–451, July 1978.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 1991.

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

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

[5] Y. S. Han, "A new treatment of priority-first search maximum-likelihood soft-decision decoding of linear block codes," *IEEE Trans. Inform. Theory*, vol. 44, pp. 3091–3096, Nov. 1998.

[6] J. A. Heller and I. W. Jacobs, "Viterbi decoding for satellite and space communication," *IEEE Trans. Commun. Technol.*, pp. 835–848, 1971.

[7] R. Johannesson and K. Sh. Zigangirov, *Fundamentals of Covolutional Coding*. Piscataway, NJ: IEEE Press, 1999.

[8] P. Lavoie, D. Haccoun, and Y. Savaria, "A systolic architecure for fast stack sequential decoders," *IEEE Trans. Commun.*, pp. 324–335, 1994.

[9] S. Lin and D. J. Costello Jr, *Error Control Coding: Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[10] 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.

[11] J. Snyders and Y. Be'ery, "Maximum likelihood soft decoding of binary block codes and decoders for the Golay codes," *IEEE Trans. Inform. Theory*, vol. 35, pp. 963–975, Sept. 1989.