MAXIMUM WEIGHT BASIS DECODING OF CONVOLUTIONAL CODES

Suman Das* suman@rice.edu
Elza Erkip† elza@ee.poly.edu
Joseph R. Cavallaro* cavallar@rice.edu
Behnaam Aazhang* aaz@rice.edu

* Department of Electrical and Computer Engineering
Rice University
Houston, Texas 77005.

† Department of Electrical Engineering
Polytechnic University
Brooklyn, NY 11201.

Abstract— **In this paper we describe a new suboptimal decoding technique for linear codes based on the calculation of maximum weight basis of the code. The idea is based on estimating the maximum number locations in a codeword which have least probability of estimation error without violating the codeword structure. In this paper we discuss the details of the algorithm for a convolutional code. The error correcting capability of the convolutional code increases with the constraint length of the code. Unfortunately the decoding complexity of Viterbi algorithm grows exponentially with the constraint length. We also augment the maximal weight basis algorithm by incorporating the ideas of list decoding technique. The complexity of the algorithm grows only quadratically with the constraint length and the performance of the algorithm is comparable to the optimal Viterbi decoding method.**

I. INTRODUCTION

Background noise is one of the main sources of error in transmission of digital data across a channel. In a multiple access wireless communication system, the reliability of transmitted data is further affected by the interference due to other users and the multipath fading characteristic of the channel. Forward error-correction methods are mandatory to combat the channel errors. Convolutional code is one of the most popular error-control schemes deployed by the wireless system designers. The strength (effectiveness to recover from error) of the code and the availability of computationally efficient optimal decoding algorithm due to Viterbi [1] have contributed to its popularity.

The strength of a convolutional code depends on two parameters - the rate of the code which measures the amount of redundancy added to the raw information bits, and the *constraint length* of the code which is the number of past information bits that affect the current coded bit. The rate of generation of source bits and the available bandwidth usually upper-bounds the rate of the code. Hence it is desirable to have a large constraint length convolutional code to improve the reliability of the system. However, the complexity of Viterbi algorithm grows exponentially with the constraint length. This precludes large constraint length convolutional codes from being practically implemented, especially in a system with real time demands or limited processing power.

Several alternative sequential and stack based techniques [2], [3] have been proposed in the literature, but none of them are as computationally efficient as the Viterbi algorithm. Several suboptimal variations of these stack based algorithms [4], [5] have been proposed that sacrifice performance to reduce the computational complexity. However these stack algorithms require a lot of extra storage. Some dynamic programming based decoding algorithms [6] have been designed whose computation cost is independent of the constraint length. However the number of computations that needs to be done is not deterministic. This is not usually acceptable for a real time implementation. Moreover at low signal to noise ratio (SNR) the number of computations are not only large but can even lead to a decoding failure. Table look-up based decoding [7] has also been investigated to reduce the decoding time at the expense of added memory requirements.

In this paper we propose a new decoding algorithm for convolutional codes based on the maximum weight basis of the code. The computational complexity of this algorithm grows only quadratically with the constraint length and is deterministic. The reduction in complexity is achieved without significantly affecting the performance of the system. It should be mentioned that our algorithm is equally applicable to any block-codes. The decoding idea is similar to the generalized Dijkstra's algorithm [8] used for block codes by several researchers [9], [10]. However these paper do not provide any systematic algorithm to compute the ordered statistic for convolutional codes. We also augment the decoding technique by incorporating the list decoding [11] principles for improvement in performance.

II. SYSTEM DESCRIPTION AND VITERBI DECODING

We consider a system where binary antipodal signals, $\mathbf{b}$, are encoded by a rate R convolutional code. The received signal is given by

$$\mathbf{r} = \sqrt{\mathcal{E}}\mathbf{d} + \eta, \tag{1}$$

where $\mathcal{E}$ is the received energy of the signal, $\mathbf{d}$ is the coded bit sequence and η is the background noise with mean zero and variance σ^2. For decoding we will consider a block of data of length N and the rate of the code to be R.

The optimal decoding rule that minimizes the probability

Research supported in part by Nokia Inc., by the Texas Advanced Technology Program under grant 1997-003604-044, and by NSF under grant NCR 9506681 and ANI-9979465.

of error for the above system is given by

$$\hat{\mathbf{b}} = \arg\max_{\mathbf{b}} p(\mathbf{b}|\mathbf{r}) = \arg\max_{\mathbf{d}\in\mathcal{C}} p(\mathbf{b}|\mathbf{r}),$$

where $\mathcal{C}$ is the codeword space. For an additive white Gaussian noise channel this optimization criteria is equivalent to finding a codeword sequence that has minimum *Euclidean distance* from the received signal. Essentially one should calculate the likelihoods associated with the 2^{NR} possible codeword sequences and select the one with largest value. Viterbi's algorithm is an iterative method of exploring the trellis to calculate the maximum likelihood codeword.

The complexity of Viterbi's algorithm grows only linearly with the number of stages in the trellis. However at each stage of the trellis, the algorithm has to compute the likelihood of each *state*. The number of states is exponential in the size of the constraint length of the code. Thus Viterbi algorithm for a convolutional code of constraint length κ and codeword length of N requires $N2^{2(\kappa+1)}$ operations indicating that the complexity becomes prohibitive for large constraint lengths. In the following sections we will describe a suboptimal channel decoding algorithm which has quadratic complexity in the constraint length of the convolutional code.

III. Motivation of the maximum weight basis

The maximum likelihood decoding algorithm is computationally complex because in the optimization process we have to restrict ourselves to the code space. On the other hand, if the decoder ignores the fact that the transmitted bit sequence is a codeword and assumes that the received signal results from an unconstrained binary sequence corrupted by noise, then the estimate is given simply by the sign estimates of the received sequence $\hat{\mathbf{d}} = sgn(\mathbf{r})$. This estimation has trivial computational complexity but obviously it is not always guaranteed that the estimated bit sequence will be a codeword. To summarize, the decoding complexity becomes non-trivial when we have to restrict our search to codewords only, while reduction in complexity can be achieved if we can make our search unconstrained but correct solution cannot be guaranteed. We want to achieve a tradeoff between these two extreme options.

We define the likelihood corresponding to the j^{th} bit of the codeword $\hat{d}_j$ as

$$\phi_j = \frac{\Pr(r_j|d_j = \hat{d}_j)}{\Pr(r_j|d_j = -\hat{d}_j) + \Pr(r_j|d_j = \hat{d}_j)},$$

and the likelihood corresponding to the estimated codeword $\hat{\mathbf{d}}$

$$\phi(\hat{\mathbf{d}}) = \prod_{j=1}^{N} \phi_j. \qquad (2)$$

The MLSE algorithm calculates

$$\hat{\mathbf{d}} = \arg\max_{\mathbf{d}\in\mathcal{C}} \phi(\mathbf{d}), \qquad (3)$$

where $\mathcal{C}$ represents the codebook. Additionally, the numerical value measures our confidence in the estimated codeword. Larger the likelihood value corresponding to a codeword, greater are the chances that it was actually transmitted.

For simplicity of discussion, let us first consider a systematic code. The systematic codeword has two parts - the *information bits*, that can take any arbitrary binary values, and the *parity bits*, which are uniquely determined by the information bits. If I represents the set of locations corresponding to the information bits (which are independent) and D represent the locations corresponding to the parity bits (which depend on the information bits), then we can rewrite (3) as

$$\hat{\mathbf{d}} = \arg\max_{\{\mathbf{d}_I,\mathbf{d}_D\}\in\mathcal{C}} \prod_{i\in I}\phi_i \prod_{j\in D}\phi_j, \qquad (4)$$

where $\mathbf{d}_I$ represents the information bits, and $\mathbf{d}_D$, the parity bits.

At this point we would like to distinguish between the two types of likelihood values we will consider. The likelihood function defined in (2) will be referred to as the *total likelihood* of the codeword, while the product corresponding to a few bits in the codeword will be referred to as *partial likelihood* of the codeword. The positions of the bits will be obvious from the context. For example $\prod_{i\in I}\phi_i$ is the partial likelihood of the codeword corresponding to the information bits.

For optimal decoding, we need to maximize the total likelihood of the codeword, but as we have observed this constrained optimization is computationally expensive, while a completely unrestricted optimization will not guarantee a valid codeword. In order to guarantee a valid codeword, we can approximate the optimization problem in (3) by maximizing the partial likelihood corresponding to the information bits. This maximization can be done in an unconstrained manner because the information bits can assume any arbitrary binary sequence. Of course this approximation technique may not always produce the codeword with largest total likelihood as we have ignored the likelihoods corresponding to the parity bits. However we can get further information about our decision if we calculate the total likelihood of the codeword corresponding to the estimated information bit sequence. If the decision is erroneous, the total likelihood value will likely be close to zero. This gives us an approximate algorithm which is computationally simple and an associated metric which signifies when we arrive at wrong decisions.

To recover from wrong decisions we will make two adjustments. When we consider only the partial likelihood, we are trying to optimize the whole problem by only looking at the partial problem. The basis of this assumption is that the codeword with total largest total likelihood will also have the largest partial likelihood corresponding to the "independent" bits. This may not be entirely true, but it is quite likely that the optimal codeword will have a "large" partial likelihood value. So we will consider all the codewords that have large partial likelihoods corresponding to the independent bits.

However, it must be remembered we are not classifying the set of locations corresponding to the information bits only as independent bits. In a systematic codeword of rate R and block-length N there are in general at most NR locations that can assume unconstrained binary values without violating the codeword constraint. We call the bits corresponding to these locations as independent bits. It is true that any arbitrary NR locations cannot assume unconstrained values for a convolutional codeword, but in general there are multiple options.

Moreover, since the likelihood ϕ_j depends on the corresponding received signal, not all locations can be decoded with uniform confidence. We would like to make decisions about those NR bits which we are most confident about; i.e. those NR independent bits which have the largest likelihood. In fact, we want to generate M possible *partial codewords* which have large partial likelihood and from them choose the one with largest total likelihood.

To summarize, our sub-optimum decoding rule is based on using the sign detector for only NR independent bits of the N coded bits. The remaining $N(1-R)$ bits of the codeword can be obtained through the structure of the codebook, $\mathcal{C}$. The choice of these NR independent bits will be based on the the *weight* of the location $i \in \{1, \ldots, N\}$ denoted by w_i which is equal to $|r_i|$, the absolute value of the received vector at that location. Since large weight corresponds to lower probability of error in the sign detector, the estimates for these NR bits will be reliable. Effectively, we want to chose an independent set of bits that has the largest reliability. It is to be noted that this idea is not unique to convolutional codes only and can also be applied to any linear block codes. In the rest of the paper we will describe a systematic method to identify M sets of independent basis each of length NR that has the M largest partial likelihood.

IV. SUBOPTIMAL DECODING OF CONVOLUTIONAL CODES

Before we describe our algorithm let us define a few terminologies. Let $\mathcal{N} = \{1, \cdots, N\}$ represent the set of locations of the coded bits.

Definition 4.1: A set $I \subset \mathcal{N}$ is defined to be an *independent subset* of $\mathcal{N}$ if the bits corresponding to locations in I can be chosen independently without violating any of the conditions for being part of a codeword in $\mathcal{C}$.

Definition 4.2: A set J is said to be *maximally independent*, if it is an independent subset of $\mathcal{N}$ and there does not exist any element e, which is in $\mathcal{N}$ but not in J such that $J + e$ is also an independent subset.
In our subsequent discussion we will use I, J, $\mathcal{N}$ to represent not only the set of locations but also the coded bits corresponding to those locations when there is no ambiguity. Also as set operators, '+' will denote the set union and '-' the set difference operation.

In case of a convolutional code of block length N, the maximally independent subset of coded bits represent those bits which can assume unconstrained binary values and can be determined independently without violating the properties of convolutional code. Additionally, given the values of the bits in the maximally independent subset, the entire codeword can be *uniquely* determined. In other words, the bits in the maximally independent subset of a convolutional code form a *basis* for the codeword space. For example, in case of a systematic codeword, the set of information bits form a basis for the codewords. In fact, a convolutional code is described by a set of relations that define the parity bits in terms of the information bits.

We will propose an algorithm to calculate the *maximum weight basis* and we will show it is easy to regenerate the codewords from this basis, given the original representation of the convolutional code in terms of the information bits. Before we formally define weight and maximum weight basis, let us prove a few properties of the independent subset and maximally independent subset of a code. Though many of these results are true for any linear codes we will study them particularly in the context of convolutional codes.

Lemma 1: Any independent subset satisfies the following properties.
1. If $I' \subset I$, and I is an independent subset of $\mathcal{N}$, then so is I'.
2. For a convolutionally coded system of block length N and rate R there are at most NR locations whose values can be chosen independently. In other words the *rank* of a convolutional code of block-length N and rate R is NR.
3. The cardinality of all maximally independent subsets are equal.

Proof: : The first property is true from the definition of the independent subset. We know that convolutional code is a *linear code*, i.e it is defined by a linear system of equations. The number of independent locations in a convolutional code is given by the rank of the linear system. In general for a systematic convolutional code there are $N(1-R)$ equations involving N variables and rank of the system is NR. Thus at most NR locations can be chosen independently. In fact the third property says that exactly NR locations can be chosen independently, which is true because any basis of a linear system will have the same number of elements. □

A. Maximum weight basis

Before proceeding any further let us define a few concepts which we will be using frequently in the rest of our paper.

Definition 4.3: We define the *weight* of a particular location i, $wt(i)$, to be a measure of reliability of making decision on that particular bit location. Quantitatively, $wt(i) = | r_i |$ and is non-negative. The weight of a set I is given the sum of the weights of each individual member of the set $wt(I) = \sum_{i \in I} wt(i)$.

When the transmitted bits are corrupted by additive white Gaussian noise, the likelihood, ϕ_j, is a monotonically increasing function of the $wt(i)$. We want to find an independent subset of bit positions whose partial likelihood is the largest among all possible independent subsets.

Definition 4.4: The *maximum weight basis* (MWB), denoted by $J^\star$, of $\mathcal{N}$ with respect to the received signal $\mathbf{r}$ is an independent subset whose weight is maximum.
Since the partial likelihood is given by $\prod_i \phi_i$, and

$$J^\star = \arg\max_I \prod_{i \in I} \phi_i = \arg\max_I \sum_{i \in I} wt(i),$$

where I is an independent subset of $\mathcal{N}$, the maximum weight basis gives us an independent subset of bits whose partial likelihood is maximum among all the independent subsets. It is clear that since all the weights $wt(i) \geq 0$, our chosen set must be a maximally independent subset; otherwise we could add elements to it without violating the independence condition and thereby form another independent subset whose weight is larger than that of our set.

We now present an algorithm to find the MWB $J^\star$ of $\mathcal{N}$ with respect to $\mathbf{r}$.

Algorithm I: Maximally independent subset

. Set $I = \emptyset$
. Sort the weights $wt(i)$ of elements in $\mathcal{N}$ based on the received vector $\mathbf{r}$.
. While $|I| < NR$
- Chose the location $e \notin I$ from $\mathcal{N}$ with largest weight such that $I + e$ is still an independent subset of $\mathcal{N}$
- $I = I + e$

This is essentially a greedy algorithm. We keep on adding elements to our independent subset in decreasing order of their weight unless they violate the independence relation. Since the block-length is finite, the algorithm terminates after a finite number of steps. The following theorem shows that the greedy algorithm actually finds the maximum weight basis.

Theorem 1: The greedy algorithm above gives the maximal weight basis of $\mathcal{N}$.
Proof: Suppose the above lemma is false. Let $J_1 = \{e_1, e_2, \cdots, e_k\}$ be the maximal subset given by the above algorithm and let $J = \{q_1, q_2, \cdots, q_k\}$ be a maximally independent subset with total weight larger than J_1. Since J_1 and J are both maximally independent subsets of $\mathcal{N}$ their cardinalities are the same. Let this cardinality be k.

Without loss of generality, we assume $wt(e_1) \geq wt(e_2) \geq \cdots \geq wt(e_k)$ and $wt(q_1) \geq wt(q_2) \geq \cdots wt(q_k)$. Of course there might be some elements common in J_1 and J. Chose the least index m such that $wt(q_m) > wt(e_m)$. Thus the element added to the set I at the m^{th} step of the algorithm is not one of $q_1, q_2, \cdots, q_m$ and for $1 \leq i \leq m$, either $q_i \in \{e_1, \cdots, e_{m-1}\}$ or $\{e_1, \cdots, e_{m-1}, q_i\}$ is not an independent subset of $\mathcal{N}$.

In other words the set $\{e_1, \cdots, e_{m-1}\}$ of cardinality $(m-1)$ is a maximally independent subset of $\{e_1, \cdots, e_{m-1}, q_1, \cdots q_m\}$. But the set $\{q_1, \cdots, q_m\}$ is an independent subset of $\{e_1, \cdots, e_{m-1}, q_1, \cdots q_m\}$ and is of cardinality m. This contradicts the third property of *Lemma* 1, that the cardinality of all maximally independent subsets are equal, and proves that the above algorithm gives the desired maximally independent subset. □

However it is not enough to just identify the maximum weight basis corresponding to a given received sequence. This information will only allow us to compute the partial likelihood. In order to have an idea of our confidence in our decision we need to compute the total likelihood. For that we will need to generate the entire codeword and not just the basis for the codeword. As mentioned before, the bit locations of the MWB may not correspond to only the information bits. To compute the total likelihood we will also need to find an algorithm to find the codeword corresponding to the MWB. For that we note that a convolutional code can be represented by a system of linear equations relating the coded bits to the information bits. For any equation involving n variables, as soon as $n-1$ of these variables have been determined, the last variable can be found uniquely.

Based on these observations, we next present an algorithm that selects the maximum weight independent subset for a convolutional code along with the associated codeword.

Algorithm II: Independent subset/codeword

- Set $I = \emptyset$
- Sort the weights $wt(i)$ of elements in $\mathcal{N}$ based on the received vector $\mathbf{r}$.
- While $|I| < NR$

- Chose the location $e \notin I$ from $\mathcal{N}$ with largest weight such that $I + e$ is still an independent subset of $\mathcal{N}$. The received signal corresponding to the location e is r_e.
- Set $\hat{d}_e = sgn(r_e)$.
- Consider the convolutional code equations. In all the equations that location e appears, reduce the number of unknowns. If any of the equations has only one unknown left, solve for that unknown.

- $I = I + e$

B. M largest weight bases

So far we have designed an algorithm to calculate the MWB of a convolutional code for a given received signal. However the codeword corresponding to the MWB may not have the largest total likelihood, but we expect it to have a large partial likelihood. Intuitively, instead of finding just one basis that has the largest partial likelihood if we build a *feasible set* of a number of large weight bases and then chose the codeword with the largest total likelihood from that particular set, then our probability of not finding the optimal codeword will be reduced. In other words, we want to find a set of M bases that have large associated weights and then find the one that gives the highest likelihood among these M bases. This idea of creating a list of feasible partial solutions has also been used in the decoding of turbo codes.

We have already calculated the basis that has the largest partial likelihood. If we have to recompute the rest of the $M-1$ largest weight bases from scratch then it will be computationally inefficient. In the rest of the paper we will show how we can generate the M largest weight bases from the maximal weight basis $J^\star$. We will first quantify the number of places the m^{th} largest weight basis differs from the MWB. We will use the following two lemmas that we state without proof (because of lack of space) to estimate this number.

Lemma 2: If J_m is a basis of $\mathcal{N}$, then there exists a monotone sequence consisting of single exchanges from J_m to the MWB. In other words if $J_0 = J^\star = \{e_1, e_2, \cdots, e_k\}$ is the MWB and J_m is another basis which differs from J_0 in m places, then we can find a sequence of bases $J_1, J_2, \cdots, J_{m-1}$ such that J_i differ from J_{i-1} in only one location and $wt(J_m) \leq wt(J_{m-1}) \leq \cdots \leq wt(J_1) \leq wt(J_0)$.

Lemma 3: If $\mathcal{M}_m$ is the set of m maximum weight bases and the basis $J, (\notin \mathcal{M}_m)$, is the $(m+1)^{th}$ largest weight basis, then there exists at least one basis $J' \in \mathcal{M}_m$ such that J differs from J' in exactly one location.

From the above two lemmas, it is evident that for any basis J_m which differs from the MWB $J^\star$ at m locations there are at least m bases whose weight are greater than or equal to J_m. So the m^{th} largest basis cannot differ from the MWB at more than m locations.

The MWB $J^\star$ has NR elements and all the M largest bases will differ from it in at most M places. But it will mean that we have to consider $\binom{NR}{M}$ locations. Since the rank of the basis is always NR, these replaced basis elements should be filled up with elements from $\mathcal{N} - J^\star$ to create other bases. There are $N - NR$ non basis elements and hence $\binom{N-NR}{M}$ choices for replacement. This can be quite large for large N. We would like to prune down the the number of locations which should be verified to obtain the desired M-largest weight bases.

Definition 4.5: A pair of elements $[e, f]$, where $e \in J$ and $f \in \mathcal{N} - J$, is defined to be a *J-exchange*, if $(J - e + f)$ is a basis when J is a basis.

Definition 4.6: For every element $e \in J^\star$ there is a set of elements $R(e) = \{f | [e, f]$ is a J-exchange$\}$. We call this set the *replacement set* of e. For all the members of the replacement set there is a *replacement element*, $r(e) = f \in R(e)$, such that among all the elements of $R(e)$, $wt(e) - wt(f)$ is the minimum.

Just as we have defined the replacement set for the elements in the MWB we can also define replacement set, $\tilde{R}(f)$, and replacement element, $\tilde{r}(f)$ for the element $f \in \mathcal{N} - J^\star$.

Theorem 2: Given a maximum weight basis $J^\star$ of $\mathcal{N}$ and the replacement elements $r(e)$ for all elements in $J^\star$, the set of $NR - M$ elements that will remain unchanged can be computed in linear time in M.

Proof: For each element $e \in J^\star$ let $w'(e) = wt(e) - wt(r(e))$. In other words $w'(e)$ is the minimum weight that we lose by replacing the element e from the MWB. We can find the $M-1$ smallest values of w' using a linear time (in M) selection algorithm [8]. Let Q be the set of the remaining $NR - M$ elements.

Then for each element $e \in Q$ there are at least $M-1$ elements e' with $wt\big(J^\star - e' + r(e')\big) \leq wt\big(J^\star - e + r(e)\big)$, and therefore together with $J^\star$ there are at least M bases better than $(J^\star - e + r(e))$. Therefore every basis in the M maximum weight basis must contain the element e. □

Similarly for every element $f \in \mathcal{N} - J^\star$ once we know the replacement element $\tilde{r}(f) \in J^\star$, we can show a similar result. Now Algorithm II not only gives us a maximal weight basis, it can easily be augmented to compute the relationship of all non-basis elements in terms of the maximal weight basis $J^\star$. We claim that the basis elements that describe each non-basis element constitute the replacement set. Moreover the replacement set relation is *symmetric*; i.e. if f_i is a member of the replacement set of e_j then e_j is a member of the replacement set of f_i. So once we have constructed the replacement set of all the non basis elements we can automatically calculate the replacement set of all basis elements.

It should be noted that the two replacement sets, along with the MWB $J^\star$, form a *necessary and sufficient statistic* for the calculation of the M largest weight bases $\mathcal{M}_M$. The set of M elements from $J^\star$ and the corresponding replacement elements alone are not enough for the computation as there might be two elements e and e' in this set such that there exists two elements e_1, e_2 in the replacement set of e with $wt(e) - wt(e_1)$ and $wt(e) - wt(e_2)$ both smaller than $wt(e') - wt(r(e'))$. In that case we have to consider both e_1 and e_2 before replacing e'.

Once we have calculated the MWB $J^\star$, the M potential replaceable candidates in $J^\star$ and $\mathcal{N} - J^\star$, we prune the replacement sets to include only these candidates. Thus for each of the M basis elements e_i that can be replaced, we calculate $wt(e_i) - wt(f_j)$ where f_j is member of both the replacement set of e_i and is one of the M elements in the non basis set which are candidates for replacement.

C. MWB decoding algorithm and its complexity

We now summarize the steps that are to be followed for our maximum weight basis decoding algorithm. We also analyze the cost to compute each of the step and hence calculate the computational complexity of the algorithm.

1. Upon receiving N received symbols, sort them in the decreasing order of their absolute value. Using the *Algorithm* II described in section 4A, compute the maximum weight basis from the sorted list and the dependence relationship of the various non-basis elements with respect to the basis. Also evaluate the first candidate codeword corresponding to the maximum weight basis.
2. For each element in the basis set, calculate the corresponding replacement element. Similarly for each element in the non-basis set, compute the replacement elements.
3. Using *Theorem* 2, select the M elements from both the basis and non-basis set that has the least amount of replacement penalty.
4. Calculate the table of replacement penalty for these two set of elements.
5. Calculate the M largest weight bases from the maximum weight basis and this table.
6. Calculate the codewords corresponding to these M largest weights. Compute the likelihood corresponding to each codeword and select the codeword with the largest likelihood.

The first step of the algorithm requires us to sort N absolute value of the received symbols. Using a radix sort algorithm [8] this computation can be done in O(N) operations. In order to calculate the maximum weight basis in each step of *Algorithm* II, we will need to find whether an element added to an independent subset still keeps it independent. Since each equation has κ variable and each variable can appear in at most κ equations this check can be accomplished in κ^2 operations. Since the cardinality of the maximum weight basis is NR we will need to perform each of these operations O(N) times. Thus we may need at most $N\kappa^2$ operations to calculate the maximum weight basis and the corresponding codeword. This particular step also helps us compute the replacement sets.

From *Theorem* 2 it is apparent that we will need to select M top replacement elements from this list of replacement elements. A linear selection algorithm can be used to achieve the top M replacement elements in O(M) steps. The table of replacement penalty has M^2 entries and they can be calculated in M^2 steps. We then have to sort these entries using another radix sort algorithm. The computation of each of the other M bases and their codes from the MWB and the replacement sets will require at most N operations. So the total complexity of the algorithm is O($N\kappa^2 + MN + M^2$).

It should be recalled that Viterbi's algorithm has a decoding complexity given by O($N2^{\kappa+1}$). The complexity of our decoding algorithm grows only linearly with the block length N as Viterbi's algorithm. But the complexity of our algorithm only increases only quadratically with the constraint length. This will enable us to decode convolutional codes of large constraint length in real time.

V. SIMULATION RESULTS

In this paper we have proposed a maximum weight basis algorithm. Our analysis of the computational complexity shows that it requires much fewer operations than the Viterbi's algorithm. Since the algorithm is not necessarily an optimal algorithm we need to study the performance loss due to this suboptimal decoding technique. We first compare the performance of the suboptimal channel decoding algorithm with Viterbi algorithm in an AWGN channel. Figure 1 shows the performance of a systematic convolutional code of rate $1/2$ and constraint length 7. We decode 100 information bits at a time. For our simulation we have $M = 6$, that is we only have kept a codeword list of size 6 in the suboptimal algorithm. The simulation results show there is very little performance loss using the suboptimal algorithm over the optimal decoding algorithm.

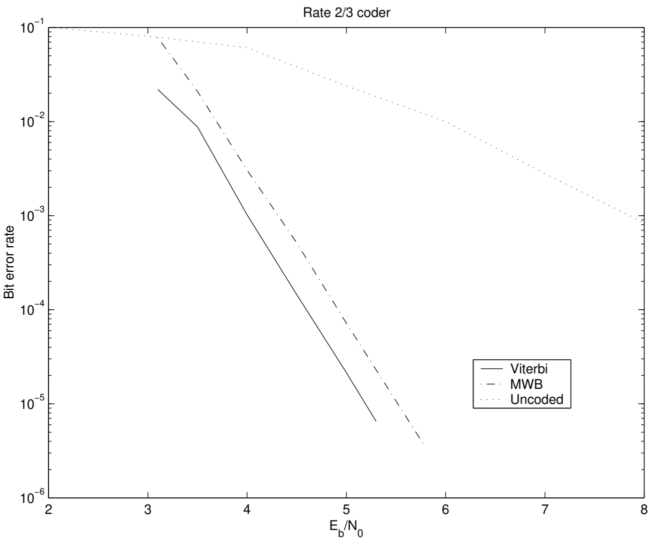

Fig. 1. Suboptimal decoding algorithm versus Viterbi decoding for rate $1/2$ convolutional code with $\kappa = 7$.

But the performance gain is more pronounced for codes with larger constraint length. In fact even though the performance of our algorithm is suboptimal compared to the Viterbi algorithm, since the computational complexity of our algorithm is much lower, we can afford to decode a stronger code with larger constraint length at the same time in which the Viterbi algorithm can decode a codeword with smaller constraint length. We study such a situation in Figure 2. In this example we decode a convolutional code of rate 1/2 and constraint length 11 which is decoded by an optimal Viterbi decoder and another codeword of same rate but constraint length 18 which is decoded by our algorithm and compare their performance. We use $M = 10$. For the Viterbi algorithm we need to perform $(2^{12}N)$ operations while our algorithm requires only $(18^2 + 100)N$ operations. Yet the performance is almost indistinguishable. This is because of the better error correcting capability of the code with larger constraint length. It should however be noted we haven't counted the exact number of operations but merely the order, and such a study of implementation aspects is planned for the future, but the potential of our system is apparent.

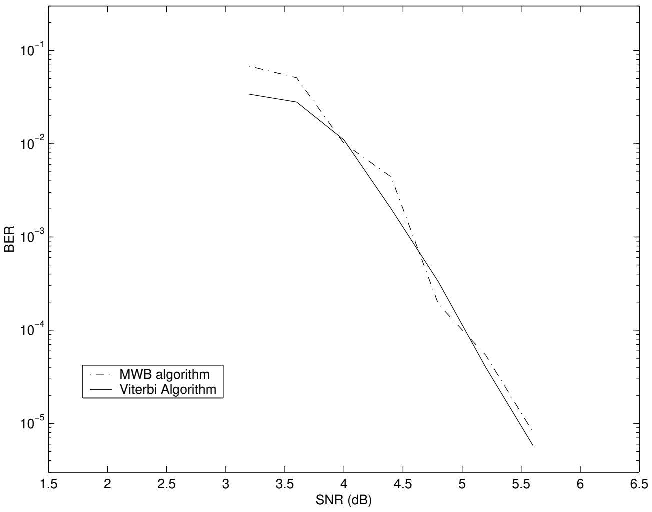

Fig. 2. Performance of Viterbi algorithm on codes of $\kappa = 11$ and MWB algorithm on codes with $\kappa = 18$

VI. Conclusions

The optimal Viterbi decoding algorithm for convolutional codes has a complexity that grows exponentially with the constraint length. This prohibits implementation of "strong" convolutional codes for a practical system. In this paper we have proposed an algorithm that has a complexity quadratic in the constraint length size and yet performs close to the optimal decoding algorithm. This algorithm approximates the process of computing the codeword with largest total likelihood by significantly pruning the search space in a computationally efficient manner. We compute all the codewords which have large partial likelihoods corresponding to the independent bases and from these candidate solutions finally obtain the one with largest likelihood. The algorithm is a two step process. In the first step we compute the codeword with the maximum weight basis and from it compute M largest weight basis. We have also used our results in a multiuser environment [12] where we have combined this computationally efficient decoding algorithm with the iterative interference cancellation technique.

In this paper we have only considered a systematic convolutional codeword. However most of these results can be easily extended to the non-systematic codewords. Also in this paper we have only investigated situations where we make "hard decisions" on the independent bits. More benefits can probably be gained if we make soft decisions. Finally we would like to extend our results to turbo codes.

References

[1] A.J. Viterbi, "Error Bound on Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Transactions on Information Theory*, vol. 13, pp. 260–269, April 1967.

[2] R. M. Fano, "A Heuristic Discussion of Probabilistic Decoding," *IEEE Transactions on Information Theory*, vol. 9, pp. 64–74, 1963.

[3] J. L. Massey, *Threshold Decoding*, MIT Press, 1998, Cambridge, MA.

[4] P. R. Chevillat and D. J. Costello, Jr, "A Multiple Stack Algorithm for Erasure Free Decoding of Convolutional Codes," *IEEE Transactions on Communications*, vol. 25, pp. 1460–1470, 1977.

[5] H. H. Ma, "The Multiple Stack Algorithm Implemented on a Zilog Z-80 Microcomputer," *IEEE Transactions on Communications*, vol. 28, no. 11, pp. 1876–1882, 1980.

[6] D. Haccoun and M. J. Ferguson, "Generalized Stack Algorithms for Decoding Convolutional Codes," *IEEE Transactions on Information Theory*, vol. 21, pp. 638–651, 1975.

[7] M. A. Vouk A. Dhalokia and D. L. Bitzer, "Table-driven Decoding of Binary One-half Rate Non Systematic Convolutional Codes," in *IEEE International Symposium on Information Theory*, 1993, p. 270.

[8] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press,, 1995.

[9] Y. S. Han, C. R. P. Hartmann, and C. Chen, "Efficient Priority-First Search Maximum-Likelihood Soft-Decision Decoding of Linear Block Codes," *IEEE Transactions on Information Theory*, vol. 39, pp. 1514–1523, 1993.

[10] M. P. C. Fossorier and S. Lin, "Soft-Decision Decoding of Linear Block Codes Based on Ordered Statistics," *IEEE Transactions on Information Theory*, vol. 41, pp. 1379–1396, 1995.

[11] K. R. Narayanan and G. L. Stuber, "List Decoding of Turbo Codes," *IEEE Transactions on Communications*, vol. 41, pp. 754–762, 1998.

[12] S. Das, E. Erkip, J.R. Cavallaro, and B. Aazhang, "Computationally Efficient Iterative Multiuser Detection and Decoding," in *Thirty Second Annual Asilomar Conference on Signals, Systems and Computers*, 1998, pp. 631–634.