Low Complexity Iterative Multiuser Detection and Decoding for Real-Time Applications

Suman Das, Elza Erkip, Joseph R. Cavallaro and Behnaam Aazhang

Abstract

In this paper we present a low complexity multiuser decoding technique that can be implemented in real-time for a convolutionally coded direct sequence code division multiple access system. The main contribution, which we call iterative prior updates (IPU) consists of iterative interference cancellation and prior updates on sequences of coded bits, combined with M-algorithm and list decoding. We illustrate the performance gains over other low complexity sequence detection and decoding strategies and argue that the algorithm converges within a few iterations and requires only a small size buffer for keeping track of the priors along iterations. The fact that we can use the existing available architectures for Viterbi decoding with slight modifications and can meet the real-time processing constraints makes the iterative prior updates algorithm an attractive alternative for cellular systems.
1 Introduction

Direct-sequence code-division multiple-access (DS-CDMA) systems assign mobile users different signature waveforms over which information bearing signals are modulated. One important purpose of these signature waveforms is for the base stations to be able to distinguish among different users. The asynchronous nature of the mobile transmissions together with the impossibility of designing mutually orthogonal signature waveforms for all possible delays results in an interference limited uplink cellular system. Multiuser detection techniques have been very effective in combating this multiple access interference [1]. The optimal multiuser detector, investigated by Verdú [2], has exponential complexity in the number of users. This observation has led to a number of suboptimal alternatives with lower complexity [3, 4, 5].

Apart from the multiple access interference, thermal noise is another important source of error for wireless signals. Limited battery life and path loss over large distances make the wireless signal power limited. This weak signal is additionally corrupted by the thermal noise at the receiving base station. Channel coding strategies are essential for protection of information against various sources of error and convolutional coding in conjunction with DS-CDMA has been shown to provide sufficient protection against errors in a wireless system [6, 7].

The optimal detection and decoding strategy for a convolutionally coded uplink DS-CDMA system consists of combining the trellises of all the users. Like the optimal detector, this has exponential complexity in the number of users [8] and thus the feasibility of implementation in real-time systems is severely limited. A straightforward low complexity alternative is to have a suboptimal multiuser detector which makes hard decisions on the coded bits followed by single user decoders. Although this strategy manages to bring the complexity down to linear in the number of users, it results in a considerable loss in performance. Our goal in this paper is to distribute the computation across the detector and the decoder and to achieve acceptable error and delay performance along with real-time implementation speed. This necessitates passing information back and forth in an iterative fashion between the multiuser detector and decoder.
Instead of convolutional codes one can also use turbo codes as the channel code. However, the well-known iterative decoding techniques for turbo codes are an order of magnitude more complex than Viterbi decoding techniques used for convolutional codes. Hence for high data rates or real-time applications convolutional codes are still preferred in many wireless systems. Even though our proposed algorithm has a decoding structure similar to that of a turbo decoder, there are a few important differences between the two systems. Our algorithm is for a multiuser system with single convolutional code per user. The iterative turbo decoder is used to decode a single user data stream.

Turbo codes are composed of two concatenated codes. In turbo decoding, during each iteration, a-posterior probabilities (APP) of the information symbols for one component code are calculated based on the BCJR algorithm proposed by Bahl et. al. [17]. These APP estimates are then used as priors for the second component code and BCJR algorithm (or variants there-of) are used to compute the posterior symbol estimate. These new APP estimates are then used as priors for the first component code and the algorithm proceeds in an iterative fashion. For two dimensional codes like turbo codes, the APP based decoding techniques can give significant gains. However the APP computation requires a significant number of calculations and this makes turbo codes unattractive for real-time systems.

One can calculate similar APP estimates for the individual symbols for convolutional codes as well. Then the symbol sequence with the largest APP will give the final estimate. This is the well known maximum a-posteriori (MAP) sequence decoding algorithm for convolutional codes. This decoding algorithm calculates the information symbol sequence that has the minimum symbol error probability. An alternative and more popular decoding algorithm for convolutional codes is the Viterbi algorithm which minimizes the probability of sequence (or codeword) error. For convolutional codes, the performance of Viterbi algorithms is no worse than the MAP based decoding and it requires significantly fewer operations. Moreover the Viterbi algorithm has a regular decoding structure and naturally lends itself to efficient hardware implementation that exploits its inherent parallelism. In this paper we have restricted our attention to systems that deploy convolutional codes and Viterbi decoding for
error protection.

Researchers have already proposed several suboptimal joint detection and decoding techniques [10, 11]. A number of recent studies are on iterative, or turbo methods [12, 13, 14]. However, most of this research effort has been directed towards turbo codes in conjunction with CDMA. Iterative detection and decoding techniques have also been investigated [9, 15] for systems with convolutional codes. We have two main contributions in this area.

Firstly, unlike most other available results we use a-posterior decision rule on entire codewords not individual symbols. Our scheme, which we call iterative prior updates (IPU), consists of iterative interference cancellation using estimated codewords. At each step we calculate the a-posteriori estimates of codewords and select the codeword that has the minimum probability of sequence error. While other iterative schemes use multi-user detection scheme to reduce interference and use decoders with uniform priors, we have used the information obtained during the previous iterations of decoding not only to reduce interference but also to update priors of the decoders. We argue, with numerical examples, how this strategy improves over simpler joint detection and decoding schemes.

Our iterative scheme is a multi-step detection and decoding process. During the initial iterations we expect a relatively high decoding errors. With subsequent iterations the performance will of course improve. For multiuser interference cancellation we need access to complete codewords. Algorithms that only estimate best possible information bit sequences need to regenerate the corresponding codewords. However the distance spectrum of Viterbi algorithms dictate that codes that are close in information bit space are usually distant in codeword space. Thus small errors in information domain will lead to large errors in codeword domain and hence will significantly pollute the interference cancellation scheme. This will introduce further errors in subsequent iterations. However, if we chose a decoding scheme that tries to minimize the codeword error than symbol error this error propagation scheme triggered during the initial iterations can be controlled.

Our second main contribution is to augment the iterative prior update scheme to work in a pipelined fashion thereby reducing the decoding delay. Most of the iterative detection and
decoding schemes described in the literature work on a block of symbols and incur a large delay which can be unacceptable for real-time systems. Moreover our iterative prior update scheme uses the already existing architectures designed for the Viterbi algorithm [18]; thus IPU can be implemented with today’s systems only with minor modifications. We argue, with numerical examples, that for real-time applications this strategy provides a receiver structure that has low computational complexity, low delay and buffer size requirements and thus can be deployed in practical systems.

This paper is organized as follows: Section 2 provides a description of the optimal multiuser sequence detection and decoding rule for a convolutionally coded DS-CDMA system. Section 3 describes our proposed iterative prior updates algorithm. Section 4 discusses computational complexity and delay issues, and provides modifications that enable real time implementation of the iterative prior updates scheme. Section 5 includes comparisons with other suboptimal joint sequence detection and decoding techniques both by simulations and by providing an analytical framework. The convergence of the iterative prior updates algorithm can be shown to be fast and storage requirements for keeping track of priors along iterations is low. This section also contains a numerical example illustrating the speed of the iterative prior updates strategy establishing it as an important alternative for real-time implementations.

2 Optimum joint detection and decoding for codeword sequences

2.1 System model

We consider an asynchronous convolutionally coded uplink DS-CDMA system with $K$ users. User $k$ has the normalized spreading code $s_k$ ($s_k s_k >= 1$) and the spreading gain is $N_c$. We let the vector $b_k$ denote the uncoded information bits of user $k$, and the vector $d_k$ denote the corresponding coded bits. We observe the received signal $r(t)$ at the base station for $N$
symbol periods, so \( d_k \) is of length \( N \) for all \( k \). The baseband signal \( r(t) \) can be written as

\[
r(t) = \sum_{k=1}^{K} \sum_{i=1}^{N} A_k d_k(i) s_k(t - iT - \tau_k) + z(t),
\]

where \( A_k \) is the amplitude and \( \tau_k \) is the delay of user \( k \) at the receiver. The signal \( z(t) \) denotes the additive white Gaussian noise.

When \( r(t) \) is passed through a chip matched filter [6], the discrete output \( \mathbf{r} \), a vector of length \( N_c(N+1) \), can be expressed as

\[
\mathbf{r} = \mathbf{S} \mathbf{A} \mathbf{d} + \mathbf{z},
\]

where \( \mathbf{S} \) is the \( N_c(N+1) \times NK \) matrix of the signature waveforms repeated over \( N \) bits, \( \mathbf{A} \) is the \( NK \times NK \) diagonal matrix of user amplitudes, \( \mathbf{d} \) is the \( NK \times 1 \) vector of the coded bits of all users over \( N \) symbol periods and \( \mathbf{z} \) is the \( N_c(N+1) \times 1 \) noise vector. Details about this model of the uplink can be found in [6]. If \( \mathbf{r} \) is further passed through a bank of code matched filters, the output is an \( NK \times 1 \) vector

\[
\mathbf{y} = \mathbf{R_N} \mathbf{A} \mathbf{d} + \mathbf{n},
\]

where \( \mathbf{R_N} = \mathbf{S}^H \mathbf{S} \) is the asynchronous code correlation matrix and \( \mathbf{n} \) is the resulting additive noise vector. Since the output of the bank of code-matched filters provides a sufficient statistic for the estimation of the vector \( \mathbf{d} \), either \( \mathbf{r} \) or \( \mathbf{y} \) can be used to find these estimates.

### 2.2 MAP sequence decoding

The goal of the receiver is to obtain reliable estimates of the information bit sequences of all the \( K \) users simultaneously. If we let vector \( \mathbf{b} = [b_1, b_2, \ldots, b_K]^H \) denote the information bit sequences transmitted by the \( K \) mobile users and \( \hat{\mathbf{b}} \) its estimate at the receiver, the aim is to minimize the probability of error \( \Pr(\hat{\mathbf{b}} \neq \mathbf{b}) \). It is well known that this is achieved by choosing the maximum-a-posteriori estimate of \( \mathbf{b} \) given \( \mathbf{r} \), that is

\[
\hat{\mathbf{b}} = \arg \max_{\mathbf{b}} p(\mathbf{b}|\mathbf{r}).
\]

The same maximization can be written in terms of the coded bits of all the users by noting that for any useful error correction code, there is a one-to-one relationship between
the information bit sequences and coded bit sequences. Then the equivalent coded bit estimate is given by

\[ \hat{d} = \arg \max_{d \in \mathcal{C}} \log p(d|r), \] (3)

where \( \mathcal{C} \) denotes the collection of all users’ codebooks. We have used the fact that logarithm is an increasing function of its argument and thus does not change the maximization problem. Note that we now have a constrained optimization as the codewords have to be part of their respective codebooks.

The optimal decoding rule necessitates a joint search through the codebooks of all \( K \) users. This search has exponential complexity in \( K \) and is not suitable for real time implementation. In the remainder of this paper we will concentrate on approximating the optimization in (3) by an iterative interference cancellation, MAP sequence decoding and prior update scheme. We will observe that this strategy brings the complexity of joint detection and decoding down to linear in \( K \).

Before we continue with the proposed iterative prior updates algorithm, we would like to elaborate on (3). It is possible to rewrite it as

\[ \hat{d} = \arg \max_{d \in \mathcal{C}} [\log p(d, r) - \log p(r)] \]
\[ = \arg \max_{d \in \mathcal{C}} \log p(d, r) \]
\[ = \arg \max_{d \in \mathcal{C}} [\log p(r|d) + \log p(d)]. \] (4)

Above derivations follow from Bayes’ rule. Also, we removed \( \log p(r) \) as it is common for all \( d \in \mathcal{C} \) and does not affect the optimization process.

The well known maximum likelihood sequence detection rule is a special case of (4) when we have uniform priors on the coded bit sequences of all the users. While this assumption is very reasonable for optimum detection and decoding, we will observe that for a suboptimal iterative scheme it is beneficial to keep track of priors along iterations. This notion will be clarified in the next section where we describe the proposed algorithm.
3 Iterative multiuser detection and decoding

Let us first focus on a particular mobile user, say $k$. Let $I_k = \{1, \ldots, k-1, k+1, \ldots K\}$ be the set of interferers for mobile $k$. We can rewrite (1) as

$$\mathbf{r} = \mathbf{S}_k \mathbf{A}_k \mathbf{d}_k + \mathbf{S}_{I_k} \mathbf{A}_{I_k} \mathbf{d}_{I_k} + \mathbf{z},$$

where $\mathbf{S}_k$, $\mathbf{A}_k$ and $\mathbf{d}_k$ denote the spreading code, amplitude and coded bit matrices of user $k$ spanning $N$ symbol periods, and $\mathbf{S}_{I_k}$, $\mathbf{A}_{I_k}$ and $\mathbf{d}_{I_k}$ are the spreading code matrix, amplitude matrix and coded bit vector for interfering users $I_k$.

If we had accurate estimates of $\hat{\mathbf{d}}_{I_k}$, the coded bit sequences of the interferers, then the signal $\hat{\mathbf{y}}_k$ given by

$$\hat{\mathbf{y}}_k = \mathbf{S}^H_k (\mathbf{r} - \mathbf{S}_{I_k} \mathbf{A}_{I_k} \hat{\mathbf{d}}_{I_k})$$

would provide a sufficient statistics for estimation of the coded bit sequence $\mathbf{d}_k$. The minimum probability of error estimate would then be given by the maximum-a-posteriori sequence rule as

$$\hat{\mathbf{d}}_k = \arg \max_{\mathbf{d}_k \in \mathcal{C}_k} p(\mathbf{d}_k | \hat{\mathbf{y}}_k)$$

$$= \arg \max_{\mathbf{d}_k \in \mathcal{C}_k} [\log p(\hat{\mathbf{y}}_k | \mathbf{d}_k) + \log p(\mathbf{d}_k)].$$

Here $\mathcal{C}_k$ denotes the codebook of user $k$. With uniform priors on all the codeword sequences, this results in maximum likelihood sequence estimates, or single user Viterbi decoders for all the $K$ users.

However, for any practical system there will be a nonzero probability of error associated with the estimates of $\hat{\mathbf{d}}_{I_k}$. This suggests that $\hat{\mathbf{y}}_k$ in (5) will contain residual interference and will no longer be sufficient for the estimation of $\mathbf{d}_k$. It was shown in [10] that one can get satisfactory performance by an iterative interference cancellation scheme. For user $k$, $k \in \{1, \ldots, K\}$ at iteration step $i$, coded bit sequence estimates $\hat{\mathbf{d}}_{I_k}$ from iteration step $i-1$ are used to obtain the semi-interference-free soft signal $\hat{\mathbf{y}}_k$ which in turn is used to obtain the maximum likelihood sequence estimate of $\mathbf{d}_k$. The prior $p(\mathbf{d}_k)$ in (6) is assumed to be uniform for all the iteration steps and $p(\hat{\mathbf{y}}_k | \mathbf{d}_k)$ is approximated assuming complete
interference cancellation. In [11] the performance of this strategy was analyzed for a DS-CDMA system combined with trellis coded modulation.

In the iterative scheme described above, only hard decisions on the coded bit sequences \( \hat{d} \) are passed along iterations. However, while obtaining those estimates, we in fact calculate the posterior distributions \( p(d_k|\hat{y}_k) \) for all the users. The MAP sequence estimate is simply the mode of this posterior distribution, so it carries less information than the distribution itself. The knowledge of the distribution along with the coded sequence estimate provides robustness against estimation errors. Hence we would like to pass these posteriors along iterations as well. In our proposed \textit{iterative prior updates} algorithm, for every user we set the posterior of iteration step \( i - 1 \) as the prior distribution of step \( i \). This prior and the signal \( \hat{y}_k \) at step \( i \) are both used to calculate the posterior and the MAP sequence estimate for the current iteration according to (6). Iterations are carried on until we reach a reliable estimate of the information bit sequences.

The proposed algorithm is summarized below.

\textbf{Iterative prior updates (IPU) algorithm}

1. Set initial coded sequence estimates \( \hat{d}_k^0 = 0 \), and initial priors \( p^0(d_k) \) uniform for \( k \in \{1, \ldots, K\} \). This corresponds to setting the first iteration estimates based on matched filter outputs.

2. In iteration step \( i \), for \( k \in \{1, \ldots, K\} \)

   - Set
     \[
     \hat{y}_k^i = S_k^H(r - S_{I_k}A_{I_k}\hat{d}_{I_k}^{i-1}),
     \]
     where \( I_k = \{1, \ldots, k-1, k+1, \ldots K\} \).
   - Set \( \hat{y}_k = \hat{y}_k^i \) and \( p(d_k) = p^i(d_k) \) in (6). Calculate the posterior \( p^i(d_k|\hat{y}_k^i) \) and the current estimate \( \hat{d}_k^i \).
   - Set the prior for the next iteration equal to the posterior for this iteration, that
is

\[ p^{i+1}(\mathbf{d}_k) = p^i(\mathbf{d}_k|\hat{\mathbf{y}}^i_k). \]

3. Iterate until there is no further change in successive codeword estimates or for a preset number of iterations

Figure 1 shows a block diagram representation of a particular iteration step of this algorithm for user 1. The “Spread” block corresponds to multiplying by \( \mathbf{S}_k \). “Estimate coded bits” block performs MAP sequence decoding using the updated prior and the signal \( \hat{\mathbf{y}}_k \). Note that it is possible to speed up the algorithm further by immediately using the estimates \( \hat{\mathbf{d}}^i_1, \ldots, \hat{\mathbf{d}}^i_{k-1} \) in obtaining the estimate \( \hat{\mathbf{d}}^i_k \).

Having motivated and mathematically stated the iterative prior updates algorithm, we will next provide a complexity analysis of the proposed scheme. We will also incorporate practical constraints such as decoding delay and storage requirements into the algorithm making it suitable for real-time implementations.

4 Complexity and efficient implementation

The optimum multiuser detection and decoding algorithm is computationally expensive in terms of the number of users \( K \) because it optimizes jointly over the codebooks of all users. The IPU algorithm described above uses \( K \) single user decoders per iteration step and thus brings the complexity down to linear in \( K \) rather than exponential. We will show via simulations that a few iterations are enough for convergence.

The well-known form of the Viterbi algorithm can be used to provide the likelihood value (and thus the posterior) for the codeword with largest likelihood - essentially the mode of the distribution. However our goal is to carry out the information about the entire distribution along iterations. A straightforward modification of Viterbi algorithm that calculates the likelihoods for all codewords would require keeping track of partial codewords at each state of the trellis, and would result in a large number of computations. Furthermore, the algorithm
in Section 3 requires that the receiver store the posterior distribution for all \( d \in C \) along iterations. This means we need a storage space exponential in the block length \( N \).

However, we conjecture that except for a few codewords, for most of the coded sequences the corresponding posteriors are insignificant. In fact, the distribution can be very well approximated by calculating and maintaining the posteriors for only these few codeword sequences. Based on this conjecture our strategy, akin to list decoding [19], we will only calculate and store the probabilities of \( L \) codewords that have the highest posterior distribution thus keeping the computational complexity and storage requirements low. By properly choosing \( L \), we can ensure that the total probability of these top \( L \) codewords is close to one. Through simulations, we show that the above conjecture is true and that small values of \( L \) give satisfactory performance.

Another important issue is the decoding delay encountered by the single user decoders used within the iterative prior updates algorithm. The M-algorithm [20] provides a standard method for reducing the decoding delay for a convolutional code. A window of size \( 5\kappa \), where \( \kappa \) is the constraint length of the convolutional code, is used to make a decision on the first branch of the trellis. The window is then slid by one symbol period to continue decoding. However, when combining list decoding with the M-algorithm, storing the top \( L \) paths up to level \( m \) in the trellis does not necessarily lead to the top \( L \) paths up to any subsequent levels. The next lemma shows how to overcome this problem in a storage-efficient manner without the need for an elaborate sorting scheme at each stage.

**Lemma 1:** For a convolutional code of constraint length \( \kappa \), to evaluate the top \( L \) paths (with largest likelihoods) at any level or depth of the trellis we need to store at most \( L2^\kappa \) path metrics.

**Proof:** We start with a few words on the notation. For a convolutional code of constraint length \( \kappa \), we have \( 2^\kappa \) states. The trellis branch connecting state \( i \) to \( j \) is denoted by \( b(i \to j) \). A path up to level \( t \) is a \( t \times 1 \) vector of branches. Note that not all combinations of branches can form a path on the trellis, the terminal state of the branch at time \( t - 1 \) should be the originating state of the branch at time \( t \).
We claim that if we have the $L$ most probable paths into each state at a particular level of the trellis, then we will be able to calculate the top $L$ most probable paths in the next level. Since the number of states is given by $2^s$, the total storage requirement will then be $L2^s$ proving the theorem.

In order to show that the above claim is true, we use induction on the level of the trellis. The claim is true for level 1, since all the paths originate from the same node at level 0. Let us assume the claim is true for level $(t-1)$. We have to prove that if we have the top $L$ paths ending at each state in level $(t-1)$, we will be able to get the top $L$ paths at level $t$.

Let us assume the contrary. Suppose there exists a path at level $t$ that is one of the top $L$ most probable paths ending at a particular state but its predecessor at level $t-1$ does not correspond to one of the top $L$ paths stored for each state at level $t-1$. Note that we calculate the probabilities conditioned on the received signal $y^t = (y_1, \ldots, y_t)$. We denote this path by the vector $P^t_0 = (b(s_0 \rightarrow s_1), \ldots, b(s_{t-1} \rightarrow s_t))$ and its predecessor by $P^t_{0-1}$ where $s_i$ denotes the state that the path goes through at level $i$. Since $P^t_{0-1}$ is not one of the top $L$ paths, there exist L paths $P^{t-1}_1, \ldots, P^{t-1}_L$ at level $t-1$ that end at state $s_{t-1}$ and that have probabilities larger than $P^t_{0-1}$.

Using Bayes’ rule we can write

$$\log p(P^t_0 | y^t) \propto \log p(y^t | P^t_0) + \log (P^t_0)$$

where we have ignored $\log p(y^t)$ since it contributes equally to all the paths. Also

$$\log p(y^t | P^t_0) = \log p(y^{t-1} | P^{t-1}_0) + \log p(y_t | b(s_{t-1} \rightarrow s_t))$$

and

$$\log p(b(s_{t-1} \rightarrow s_t) | P^{t-1}_0) = \log p(b(s_{t-1} \rightarrow s_t))$$

as the channel is memoryless.

Combining the above relations, we have

$$\log p(P^t_0 | y^t) \propto \log p(y^{t-1} | P^{t-1}_0) + \log p(y_t | b(s_{t-1} \rightarrow s_t)) + \log (P^t_0)$$

$$\propto \log p(P^{t-1}_0 | y^{t-1}) + \log p(y_t | b(s_{t-1} \rightarrow s_t))$$

$$< \log p(P^{t-1}_j | y^{t-1}) + \log p(y_t | b(s_{t-1} \rightarrow s_t)),$$
for all $i = 1, \ldots, L$. The last inequality is based on the above definition of $P^t_{i-1}$ as the top $L$ paths into state $s_{i-1}$ at level $t-1$.

The above chain of inequalities suggest that by appending $P^t_{i-1}$ with $b(s_{i-1} \rightarrow s_i)$, we can get $L$ paths at level $t$ that end at state $s_i$ and that have probabilities larger than the probability of $P^t_0$. This is a contradiction to our assumption and proves the original claim and the theorem. \hfill \Box

One of the main difficulties in the implementation of the traditional list decoding algorithm is that it involves elaborate sorting at each level. This can be avoided by using the above Lemma. To compute the $L$ most likely paths ending at state $s_i$ at level $t$ we have to consider all the states $s^i_{t-1}$ at level $t-1$ that have a branch ending at $s_i$. We consider the sorted list of $L$ paths for these states stored at $s^i_{t-1}$. Only the top-most items in those lists are candidates for becoming the next path with highest probability for state $s_i$. We just add the branch metric corresponding to $b(s^i_{t-1} \rightarrow s_i)$ to the topmost candidate of state $s^i_{t-1}$ and compare with other similar candidates for all other states at level $t-1$ that have a branch to state $s_i$. This way we can eliminate the elaborate sorting associated with any list decoding algorithm and slightly modify the vanilla Viterbi algorithm to get sufficient statistics. The efficient combination of list decoding with the M-algorithm reduces amount of storage required to $2L2^K$, and the decoding delay to $5\kappa$. We next provide a modified version of the original iterative prior updates algorithm to incorporate these delay and buffer size reduction techniques.

**Delay and buffer efficient (DBE) iterative prior updates algorithm**

1. For all of the $N$ coded bits, set initial coded sequence estimates $\hat{d}_k = 0$, and initial priors $p(d_k)$ uniform for $k \in \{1, \ldots, K\}$.

2. Set $n = 1$.

3. In order to decode the $n^{th}$ coded bit for all the users, take a block length of size $5\kappa$ starting from the $n^{th}$ bit. Note that updated starting priors and starting coded bit
estimates for time instants $n - 1, n$, \ldots, $n + 5\kappa - 1$ are already determined from the decoding window corresponding to bit number $n - 1$.

Working with the current block of coded bits iterate as follows:

(a) In iteration step $i$, for user $k \in \{1, \ldots, K\}$

(i) Set

$$\hat{y}_k^i = S_k^H (r - S_{I_k} A_{I_k} \hat{d}_{I_k}^{i-1}),$$

where $I_k = \{1, \ldots, k - 1, k + 1, \ldots, K\}$ is the set of interferers.

(ii) Set $\hat{y}_k = \hat{y}_k^i$ and $p(d_k) = p(d_k^i)$ in (6). Calculate the posterior $p^i(d_k|\hat{y}_k^i)$ and the current estimate $\hat{d}_k^i$. For the posterior calculation, use Lemma 1 to calculate the likelihood for only the top $L$ paths in the trellis. For all other paths use a uniform likelihood that is smaller than the probability of the $L^{th}$ smallest path. Combine likelihood with the prior to find the posterior.

(iii) Set the prior for the next iteration equal to the posterior for this iteration, that is

$$p^{i+1}(d_k) = p^i(d_k|\hat{y}_k^i).$$

(b) Iterate until there is no further change in successive estimates for the first bit of the block for all the users.

4. Increase $n$ by 1. If $n \leq N$, go back to step 3 to repeat the procedure to decode the next coded bit. Otherwise stop.

This modified algorithm requires $O(5\kappa [K - 1 + 2^{\kappa + 1}])$ operations per coded bit per user per iteration, slightly higher than the original algorithm of Section 3 that requires $O(K - 1 + 2^{\kappa + 1})$ operations. The added benefit of the modified algorithm, as explained above, is to reduce the decoding delay to $5\kappa$ and storage requirements to $2L2^{\kappa}$ where $\kappa$ is the constraint length of the convolutional code and $L$ is the number of paths kept in the DBE iterative prior updates scheme. It is to be noted, that to compute the path metrics for bit $n$ we have to remember the path metrics for bit $n - 1$. This is responsible for the multiplicative factor
of 2 in the storage requirement expression. It should be noted, that the complexity of the
standard interference calculation (see [5] for details) that needs to be performed along with
this decoding process is linear in both the number of users as well as the number of bits per
user per bit. Thus the primary difficulty in real time implementation of any joint detection
and decoding scheme is associated with the complexity of the decoder.

We next provide a simulation analysis of the proposed algorithm and a system example
illustrating real-time decoding capabilities.

5 Numerical Studies

5.1 Performance analysis

We first analyze the results of a simulation study on the performance of the proposed algo-
rithm. We also compare it with other low complexity multiuser sequence detection/decoding
schemes. A number of iterative techniques that aim to minimize the probability of symbol
error have recently been investigated in the literature [12, 13]. Since our goal is to reduce
the probability of codeword error, or sequence error, we will primarily compare with other
techniques that are aimed at reducing codeword sequence error.

For the first set of simulations we use Gold code sequences of length 7. Our system has 4
users with the amplitude of all other users being twice that of user 1. For error protection,
the users have a convolutional code of rate 2/3 and constraint length 2. The delays of the
users were assumed to be distributed uniformly over the bit period. Figure 2 provides the
bit error rate performance of user 1 for various multiuser detection/decoding algorithms.

The curve labeled “MF+Viterbi” is for a system where hard decisions on coded bits
are made based on matched filter outputs, then passed through $K$ single user Viterbi de-
coders. “Hard2stage+conv” has two stage hard output multistage detector followed by
single user Viterbi decoders. Since the multistage detector is better in mitigating multiple
access interference, the performance of “Hard2stage+conv” is superior to “MF+Viterbi”.
“2stage+trellis” refers to the algorithm described in [11]. A multistage detector in conjunc-
tion with $K$ single user soft Viterbi decoders is used. We have iterated the algorithm two
times. Since their algorithm combines detection and decoding, performance is better than “Hard2stage+conv”. Finally “2stage+IPU” refers to the DBE iterative prior updates algorithm described in Section 4 with two iteration steps. We observe that the iterative prior updates consistently outperforms all the other schemes of comparable complexity. It provides about 0.5dB gain over the best algorithm (“2stage+trellis”) for a bit error rate of 10^{-3} (Figure 2). It also comes to within 0.5dB of the single user bound for this loaded system. Simulation results for a system with 12 users, spreading gain of 31 and a convolutional code of rate 2/3, constraint length 5 follow a similar trend and can be found in Figure 3.

Even though bit-error rate is a popular performance metric to compare detection and decoding algorithms, since Viterbi based decoding schemes are aimed at reducing sequence error rates, we also simulated the block-error-rate performances. For these calculations we simulated blocks of length 100 for each user and marked the block in error if any one of the information bits were in error. We simulated the performance of “MF+Viterbi”, “2stage+trellis” and our “2stage+IPU” algorithms for 12 users and spreading gain of 31. The simulations results in Figure 4 show that the relative performances of these algorithms remain unchanged even when we consider block error rate as the metric for comparison.

We also illustrate how the normalized probability distribution for the top 10 paths (L = 10) evolves over iterations in Figure 5. Initially, at iteration step 0, all paths are equally likely. However, as the number of iterations increase, it is possible to distinguish the most likely path with higher reliability. Note that we can get a good estimate of the top path after 2-3 iterations.

It is clear that further performance improvement over the IPU algorithm, which attempts to minimize the sequence error, can be achieved if we consider algorithms that minimize probability of symbol error. However this comes at the expense of added complexity and decoding delays. We consider another system that reduces the probability of symbol error for decoding. The standard BCJR algorithm [17] is used to generate the a-priori probabilities and then the symbol based MAP criteria is used to identify the information bit sequence. We then generate the corresponding codewords for these information bit sequences and use
them for interference cancellation.

Our simulations show that the initial MF estimate is too noisy for a symbol MAP based decoder. The performance improves significantly if we start the iterations for joint detection and decoding after a few steps of initial interference cancellation. The simulation comparison (Figure 6) shows that the performance of this symbol MAP based joint detection and decoding algorithm (labeled “BCJR”) is better than our proposed IPU for low SNR ranges. However, for high SNR values the performance difference is almost insignificant. It should be noted that the BCJR algorithm requires significantly larger number of operations (see [16] for details) and storage ($2N2^k$ where $N$ is the block length). Moreover, the BCJR algorithm is a block decoding scheme and may not be suitable for systems that cannot tolerate large decoding delays.

5.2 Framework for comparison with other algorithms

We now provide a systematic way of investigating the differences and similarities between the strategies compared in Figures 2 and 3. We will consider the original iterative prior updates algorithm in Section 3.

In the $i^{th}$ iteration of the algorithm, the coded bit sequence for user $k$ is estimated using (6). This optimization can be rewritten as follows:

$$\hat{d}_k = \underset{d_k \in \mathcal{C}_k}{\arg\max} \{ \log p(\hat{y}_k^i | d_k) + \log p_i^j(d_k) \}$$

$$= \underset{d_k}{\arg\max} \{ \log p(\hat{r} | d_k) + \log p_i^j(d_k) \} \mathcal{I}(d_k \in \mathcal{C}_k) \mathcal{I}(d_{I_k} = \hat{d}_{I_k}^{i-1}) \},$$

where $\mathcal{I}$ denotes the indicator function. The indicator functions ensure that we restrict our sequence search to codewords of the $k^{th}$ codebook and we cancel interference using estimates of interferers from the previous iteration. The algorithm also updates priors $p_i^j(d_k)$ using posteriors from the previous iteration step.

The “2stage+trellis” type scheme assumes a uniform prior distribution $p_i^j(d_k)$ for all the iteration steps $i$. We observed from the simulation results that updating the prior and passing along extra information improves the performance. The “Hard2stage+conv” strategy ignores the constraint $\mathcal{I}(d_k \in \mathcal{C}_k)$ in (7) and uses a uniform prior $p_i^j(d_k)$ to obtain a hard
decision on the coded bits. Then the information bit sequence is extracted using single user Viterbi decoders. The “MF+Viterbi” scheme ignores both the constraints $I(d_k \in C_k)$ and $I(d_k = d_k^{i=1})$ and assumes a uniform prior in making a hard decision on the coded bits. Similar to “Hard2stage+conv”, information is then extracted using Viterbi decoding. Since no information is fed back, there are no iterations.

5.3 Cost of implementation of IPU algorithm: Storage and computational requirements

Each iteration of the IPU algorithm improves the performance but introduces extra computational cost. Figure 7 illustrates the sensitivity of the performance of the IPU algorithm to the number of iterations. The simulation parameters are the same as in Figure 3 and the SNR is fixed at 6 dB. (Similar results are also obtained for parameters described in Figure 2.) We observe that IPU achieves its near optimal performance only after 2-3 iterations. This was also observed in Section 5.1. Thus relatively few iterations are needed and the complexity is essentially linear in the number of users.

We also study the sensitivity of DBE-IPU (with 3 iterations) to $L$, the number of paths stored. By Theorem 1, $L$ is directly related to the storage requirements of the system. Our simulation results in Figure 8 show that by storing probabilities for only a few (5-6) number of paths, we can achieve the limiting performance. Even though both the above parameters depend on the number of users in the system, the spreading gain $N_c$ and the constraint length of the code $\kappa$, our analysis shows that this dependence is quite weak. In fact for most systems with realistic spreading gains and number of users we have simulated, the number of paths needed to achieve the limiting performance was less than 10. Hence as claimed, the algorithm has low storage requirements.

5.4 Real-time implementation

We now consider a numerical example to illustrate the real-time capabilities of the iterative prior updates algorithm. We consider a system with 15 users each transmitting at 20 Kbits/sec. Each user has a convolutional code of rate 2/3 and constraint length 5. If we
consider a block-length of 1024 data bits the receiver will have 0.05 seconds to decode all the information bits of all the users.

For the optimal joint multiuser trellis decoding technique, the complexity of the algorithm per user is given by $N2^{K+\kappa+1}/K$, where $N$ is the number of coded bits in a block, $K$ is the number of users and $\kappa$ is the constraint length. Using $N = 1024$, $K = 15$, $\kappa = 5$ as above, we need a total of $128 \times 10^6$ operations per user. If we use a state of the art TI DSP processor (TMS320C6701) running at 200Mhz per user, it will require 0.64 seconds to complete all the operations. Even if the processor can utilize all the 8 available functional units all the time, which in most applications is not possible, we still would need 0.08 seconds.

On the other hand the DBE iterative prior updates algorithm requires $N(K - 1 + 5\kappa 2^{\kappa+1})$ operations per user resulting in a total of $12 * 10^6$ operations which can be completed in 0.06 seconds. If we assume a very realistic 2 way parallelism (2 out of 8 possible functional units used on the average), the total decoding time becomes 0.03 seconds which is well within the real-time bound. Moreover the decoding delay is only 25 ($5\kappa$) bit periods (i.e., only 1.25msec) and we need less than 1 Kbyte of storage space. Coupled with the fact that the performance is better than other low complexity multiuser schemes, we argue that the iterative prior updates algorithm is an attractive alternative for real-time implementations.

6 Conclusions

In this paper, we have described a low complexity multiuser joint detection and decoding algorithm for a convolutionally coded DS-CDMA system. Our main goal was to show that this algorithm is suitable for systems requiring real-time communications, such as cellular voice. The optimal joint detector/decoder has exponential complexity in the number of mobile users and is not practical for such real-time applications. The proposed iterative prior updates algorithm is based on iterative interference cancellation together with MAP decoding for sequences of coded bits and prior updates at every iteration. It requires a small storage space for storing priors along iterations, has low decoding delay and fast convergence. The complexity can be shown to be linear in the number of users. It also has the added benefit
of utilizing the already existing hardware for Viterbi decoding. Through simulations, we show that the performance is superior to other low complexity sequence detection strategies. We also provide a numerical real-life example illustrating the benefits of the algorithm in real-time applications.

This work deals with lowering the complexity of detection and decoding from exponential to linear in the number of users. Individual users would still use a Viterbi-type decoding algorithm that has exponential complexity in the constraint length of the convolutional code. This prohibits using convolutional codes with larger constraint length (> 10) and better error correction capability. To this end, in a separate work we examine suboptimal single user decoding algorithms that have complexity quadratic in the constraint length [21]. We also show that when this suboptimal decoding scheme is used together with the iterative multiuser algorithm described in this paper, performance close to the single user bound can be achieved [22].

References


Figure 1: Block diagram of the iterative prior updates algorithm. (A) shows the iterative nature of the algorithm (B) represents in detail the interference cancellation and decoding scheme along with iterative prior update for each individual user.
Figure 2: Comparative study of various joint detection and decoding algorithms. “2stage+IPU” refers to the algorithm described in Section 4. Number of users $(K)=4$, Spreading gain $(N_c)=7$.

Figure 3: Comparative study of various joint detection and decoding algorithms with a 12 user system and spreading gain 31.
Figure 4: Block error rates of various joint detection and decoding algorithms with a 12 user system and spreading gain 31.

Figure 5: Evolution of the normalized probability distribution of the top 10 paths with the number of iterations.
Figure 6: Comparison of decoding algorithms based on BCJR algorithms (that minimize bit error rate) and IPU decoding scheme for a 12 user system and spreading gain 31.

Figure 7: Convergence study of the iterative prior update algorithm, $K = 12$, $N_c = 31$, $L = 6$, convolutional code of rate $R = 2/3$, $\kappa = 5$. 

26
Figure 8: Sensitivity study of the DBE iterative prior update algorithm to storage space, $K = 12$, $N_c = 31$, $R = 2/3$, $\kappa = 5$. 