A Multistack Algorithm for Soft MIMO Demodulation

Mehran Nekuii and Timothy N. Davidson

***Abstract*—We propose a family of list-based soft demodulators for multiple-input–multiple-output (MIMO) communication systems based on a multistack algorithm for traversing the tree structure that is inherent in the MIMO demodulation problem. The existing stack algorithm for MIMO soft demodulation stores a single stack of visited nodes in the tree and expands the stack using the "best-first" principle. In the proposed multistack algorithm, the single stack is partitioned into a stack for each level of the tree, and the algorithm proceeds by performing one best-first search step in each of these stacks in the natural ordering of the tree. By assigning appropriate priorities to the level at which this "best-first search per level" processing restarts once a leaf node has been obtained, the proposed demodulators can achieve tradeoffs between performance and complexity that dominate those of several existing methods, including the stack algorithm, in the low-complexity region.**

***Index Terms*—Iterative demodulation and decoding (IDD), list demodulation, sphere decoder, stack algorithm, tree-search decoding, turbo principle.**

I. Introduction

Multiple-input–multiple-output (MIMO) wireless communication systems are attractive because they provide the potential for reliable communication at substantially higher data rates than the corresponding single-antenna system. However, the computational effort required to achieve these high spectral efficiencies is often beyond the capabilities of the envisioned devices, and hence there has been a considerable interest in the development of transceivers that balance the competing demands of spectral and computational efficiency. A popular transceiver architecture for balancing these demands is a MIMO version of the bit interleaved coded modulation (BICM) with block-by-block transmission and iterative "soft" demodulation and decoding (IDD), e.g., [1]. A key computational bottleneck in these schemes is the demodulation step; that is, the extraction of the log-likelihood ratio (LLR), or an approximation thereof, of each of the bits transmitted in a given block from the corresponding output block of the MIMO channel. The design of list-based techniques to manage this computational burden is the core topic of this paper.

The goal of list-based soft demodulation for block-based MIMO transmission is to (efficiently) obtain a list of candidate bit vectors that generate the dominant components of the likelihoods for a given block, and then to approximate the LLR of each bit transmitted in that block using the members of the list, e.g., [1]–[8]. A popular class of approaches to an efficient list generation [1]–[7] is based on the tree-search representation of the MIMO demodulation problem [9], [10]. In that representation, the metric that is used to assess the significance of each bit vector corresponds to an additive path metric in a tree with nonnegative branch metrics. The leaf nodes in the tree represent the complete bit vectors, and the dominant bit vectors correspond to leaf nodes with small path metrics. A feature of the tree-search representation is that, in an IDD receiver, the extrinsic information provided by the previous iteration of the decoder can easily be incorporated into the branch metric [2]–[7].

Manuscript received January 10, 2008; revised July 11, 2008. First published August 26, 2008; current version published May 11, 2009. This work was supported in part by a Premier's Research Excellence Award from the Government of Ontario. The work of T. N. Davidson was supported in part by the Canada Research Chairs program. This paper was presented in part at the 2007 International Conference on Acoustics, Speech, and Signal Processing, Honolulu, HI, April 15–20, 2007. The review of this paper was coordinated by Prof. N. Arumugan.

The authors are with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, ON L8S 4K1, Canada (e-mail: nekuii@ieee.org; davidson@mcmaster.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2008.2004961

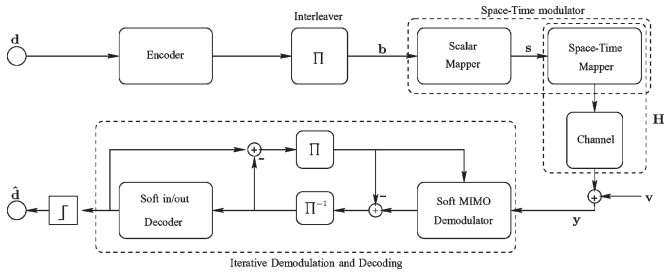

Fig. 1. MIMO-BICM-IDD transceiver.

Once the list generation problem has been associated with the search for leaf nodes with small path metrics, a number of conventional tree-search algorithms can be applied. Of particular interest is the extension of the stack algorithm for "hard" demodulation [10], [11] to the process of list generation [2], [3], [7]. The stack algorithm adopts a "best-first" search strategy in which the exposed nodes of the tree are stored in a global stack, and the algorithm proceeds by expanding the node in the stack with the smallest path metric until a leaf node is selected for expansion. (That leaf node is the best leaf node in the tree.) The natural extension of the stack algorithm to the list generation simply involves continuing the search for the next best leaf node [2], [3], [7]. Hence, the stack algorithm generates bit vectors in the order of their path metrics.

While the stack algorithm generates an ordered list, the rapid growth of the stack size and the consequent complexity of finding the next best list member are significant impediments to its implementation in list-based soft demodulation [7]. The goal of this paper is to propose demodulators that provide greater control over the complexity–performance tradeoff by constructing a tree-search algorithm that generates a sizeable collection of "good" list members in the early stages (though not necessarily an ordered collection) so that a good performance can be obtained even if the algorithm is terminated for reasons of complexity. The key aspect of the proposed multistack algorithm is that the (global) stack is partitioned into one stack per level in the tree. The algorithm then proceeds by performing one best-first search step per level of the tree in the natural ordering of the tree. We will show that by assigning appropriate priorities to the level at which this "best-first search per level" processing restarts, and by incorporating natural termination criteria, the proposed demodulators can achieve tradeoffs between performance and complexity that dominate those of several existing methods in the low-complexity region.

II. System Model

We will consider the coherent narrowband MIMO-BICM-IDD transceiver structure illustrated in Fig. 1 [1], where the space–time modulator is the concatenation of a scalar constellation mapper and any (widely) linear space–time block code [12]. We will consider scalar constellations of size 2^M, and the space–time block code will transmit K such symbols per block channel use. We will let $\mathbf{b}^{(n)}$ denote the vector of MK bits from the interleaved encoded bit streams that are mapped to the K symbols at the nth channel use

$\mathbf{s}^{(n)} = \mathcal{M}(\mathbf{b}^{(n)})$, where $\mathcal{M}(\cdot)$ is the corresponding mapping. Hence, the vector of received samples at the nth channel use can be written as

$$\mathbf{y}^{(n)} = \mathbf{H}^{(n)}\mathbf{s}^{(n)} + \mathbf{v}^{(n)} = \mathbf{H}^{(n)}\mathcal{M}(\mathbf{b}^{(n)}) + \mathbf{v}^{(n)} \tag{1}$$

where $\mathbf{H}^{(n)}$ is the equivalent channel matrix [12] at the nth channel use, and $\mathbf{v}^{(n)}$ is a vector of noise samples, which will be assumed to be from a zero-mean additive white circular Gaussian noise (AWGN) model with variance σ^2 per real scalar dimension. We will focus on cases in which the space–time block code is configured so that $\mathbf{H}^{(n)}$ is square or tall.

Since the emphasis of this paper is on the demodulation step, the outer encoder in Fig. 1 can be any binary encoder, and we will adopt its corresponding soft-input–soft-output decoder at the receiver. The role of the soft MIMO demodulator in Fig. 1 is to compute the LLRs of the interleaved encoded bits based on the channel measurements and the extrinsic information from previous decoder iterations. This "soft information" is then passed to the outer soft decoder. Since the channel model in (1) is memoryless, the soft demodulator can operate on a block-by-block basis. For notational simplicity, we will drop the superscript $(\cdot)^{(n)}$ in (1) and consider a generic block channel use. In that case, the soft demodulator computes (or approximates) the (conditioned) LLR for each element b_i of $\mathbf{b}$ [1] as

$$\log \frac{p(b_i = 1|\mathbf{y}, \mathbf{H})}{p(b_i = 0|\mathbf{y}, \mathbf{H})} = \log \frac{\Sigma_{\mathcal{L}_{i,1}} p(\mathbf{y}|\mathbf{b}, \mathbf{H}) p(\mathbf{b})}{\Sigma_{\mathcal{L}_{i,0}} p(\mathbf{y}|\mathbf{b}, \mathbf{H}) p(\mathbf{b})} \tag{2}$$

where $\mathcal{L}$ is the list of all 2^{MK} binary vectors $\mathbf{b}$, $\mathcal{L}_{i,b} \triangleq \{\mathbf{b} \in \mathcal{L} | b_i = b\}$, and under the assumed AWGN noise model $p(\mathbf{y}|\mathbf{b}, \mathbf{H}) \propto e^{-\|\mathbf{y} - \mathbf{H}\mathcal{M}(\mathbf{b})\|_2^2/(2\sigma^2)}$. Assuming that the interleaver is good enough for the conventional approximation $p(\mathbf{b}) \approx \prod_{i=1}^{MK} p(b_i)$ to hold, the summands in (2) can be written as $e^{-D(\mathbf{b})/(2\sigma^2)}$ [5], where

$$D(\mathbf{b}) \triangleq \|\mathbf{y} - \mathbf{H}\mathcal{M}(\mathbf{b})\|_2^2 - 2\sigma^2 \sum_{i=1}^{MK} \log p(b_i). \tag{3}$$

Since each list $\mathcal{L}_{i,b}$ contains 2^{MK-1} terms, there has been a considerable interest in schemes that enable the approximation of (2) by replacing $\mathcal{L}$ with a carefully selected reduce-sized list $\hat{\mathcal{L}}$ that contains the dominant summands in (2) [1]–[8].

The dominant summands in (2) correspond to binary vectors $\mathbf{b}$ that yield small values for $D(\mathbf{b})$. The search for such vectors is significantly simplified when the QR decomposition is used to make the inherent M-ary tree structure explicit [1]–[3], [5], [6], [9], [10]. In particular, if we let $\mathbf{HE} = \mathbf{QR}$ denote the QR decomposition[1] of $\mathbf{HE}$, where $\mathbf{E}$ is a column permutation matrix that determines the arrangement of the symbols in the tree, and if we define $\tilde{\mathbf{y}} \triangleq \mathbf{Q}^\dagger \mathbf{y}$, $\tilde{\mathbf{v}} \triangleq \mathbf{Q}^\dagger \mathbf{v}$, $\tilde{\mathbf{s}} \triangleq \mathbf{E}^\dagger \mathbf{s}$, and $\mathbf{R}_j$ to be the jth row of $\mathbf{R}$, then (3) can be rewritten as

$$D(\mathbf{b}) = \sum_{j=0}^{K-1} |\tilde{y}_{K-j} - \mathbf{R}_{K-j}\tilde{\mathbf{s}}|^2 - 2\sigma^2 \log p(\tilde{s}_{K-j}). \tag{4}$$

Here, $p(\tilde{s}_i) = \prod_{\ell=(i-1)M+1}^{iM} p(b_\ell)$, where the product is over those bits that index the symbol $\tilde{s}_i$. In (4), the jth summand only depends on symbols $K-j$ to K, and hence, the inherent tree structure is exposed [10]. In particular, we can assign the possible values for the jth summand to be the metrics of the branches emanating from the nodes at the jth level of the tree (with level 0 being the root). Since each $\tilde{s}_{K-j}$ comes from an M-ary constellation, there will be M branches emanating from each node. For later convenience, we observe that the path metric for a node at level L in the tree is the sum of the first L terms in (4), and this path metric is additive, with nonnegative branch metrics.

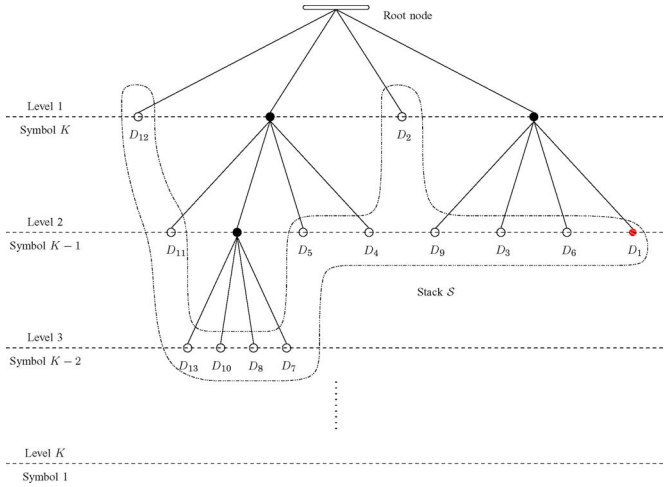

Fig. 2. Snapshot of an instance of the stack algorithm for a system in which K symbols from a 4-ary constellation are transmitted in each (block) channel use.

One approach to searching a tree for the leaf node with the smallest path metric is to employ the stack algorithm [10], [11], in which all the exposed nodes of the tree are stored in a stack $\mathcal{S}$. This algorithm proceeds in a "best-first" manner by selecting the node in the stack with the smallest metric and replacing it by its child nodes. The first leaf node that the algorithm selects for expansion corresponds to the bit vector with the smallest value for $D(\mathbf{b})$. A snapshot of an instance of the stack algorithm is provided in Fig. 2, where the exposed nodes have been labeled in increasing order of their path metrics. The next step in the algorithm would be to expand the node marked D_1.

If the stack algorithm is continued to search for the "next best" leaf nodes, it will produce leaf nodes in increasing order of $D(\mathbf{b})$, and the corresponding bit vectors constitute a candidate list for demodulation purposes [2], [3], [7]. However, such a scheme may explore many internal nodes in the tree before it reaches the leaf node with the next smallest value for $D(\mathbf{b})$ and hence may expend significant computational effort and memory resources to find only a few dominant leaf nodes. The multistack algorithm proposed in the following section provides greater control over the tradeoff between performance and complexity, so a large subset of the dominant leaf nodes can be obtained for a lower computational cost.

III. Multistack Algorithm

While the conventional stack algorithm [10], [11] employs a single-ordered stack of nodes $\mathcal{S}$ for the whole tree, we propose to partition the stack into separate stacks $\mathcal{S}_k$ for each level in the tree. In each step of the conventional stack algorithm, the node in the (global) stack with the smallest path metric is removed and replaced by its child nodes. In each step of the proposed multistack algorithm,[2] one of the stacks is selected for processing, and the node with the smallest path metric in that stack is removed, and its child nodes are placed in the stack at the next lower level of the tree. A snapshot of an instance of the multistack algorithm is provided in Fig. 3, where the exposed nodes have been labeled in increasing order of their path metrics within each

[1]We consider the conventional QR decomposition in which $Q^\dagger Q = I$, where $(\cdot)^\dagger$ denotes the (conjugate) transpose, and R is an upper triangular matrix with nonnegative diagonal elements [13].

[2]The proposed multistack algorithm is substantially different from the multiple-stack algorithm that was developed in [14] for the decoding of convolutional codes.

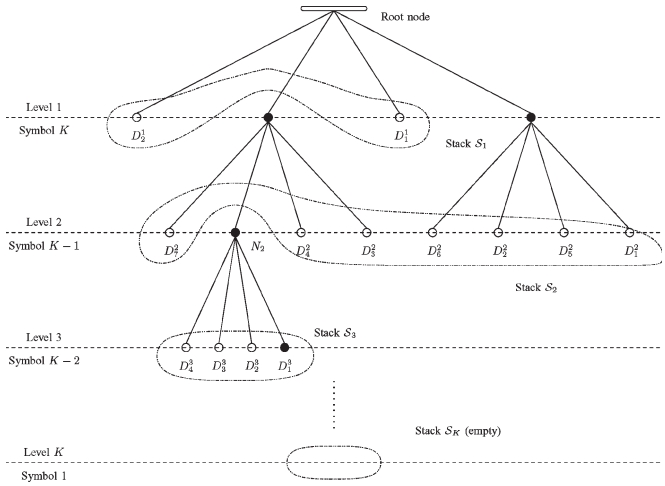

Fig. 3. Snapshot of an instance of the proposed multistack algorithm for a system in which K symbols from a 4-ary constellation are transmitted in each (block) channel use.

stack. If the stack at level 3 is selected for processing, the node to be expanded will be that labeled D_1^3.

The partitioning of the (global) stack into a stack for each level naturally generates an additional degree of freedom: the order in which the stacks are processed. In an attempt to obtain "good" leaf nodes without excessive processing, we will focus on orderings in which the next stack to be processed is the stack at the next lower level of the tree. As there are K levels in the tree, this guarantees that the next leaf node will be found in at most K steps.[3] Once a leaf node has been obtained, there is a degree of freedom in the level at which the search is restarted, and this choice provides some control over the way in which the tree is explored (see Section III-A).

Although this degree of design freedom is a distinct advantage of the proposed algorithm, the leaf nodes are no longer produced in increasing order of $D(\mathbf{b})$, and hence, we need to ensure that the algorithm does not expend computational effort to exploring paths with large metrics. To do so, we only consider those nodes with path metrics below a certain threshold B that is computed using a preliminary greedy depth-first search of the tree (see Section III-B).

A key feature of the proposed algorithm is that it generates a sizeable collection of good leaf nodes in the early stages. We will exploit this feature by providing explicit termination criteria based on the size of the list and/or the number of nodes visited in the tree. These termination criteria enable the algorithm to be tailored to the computational resources at hand (see Section III-C).

A formal statement of the proposed algorithm is provided in Table I, and in the following sections, we will discuss some of the features of the algorithm in more detail.

A. Symbol and Restart Orderings

The conventional stack algorithm has a single degree of freedom, namely the ordering of the symbols in the tree. In our notation, this is controlled by the permutation matrix $\mathbf{E}$ that is implicit in (4). The multistack algorithm introduces an additional degree of freedom, which is the order in which we search for a nonempty stack after having obtained a leaf node.[4] That ordering will be denoted by $\mathbf{t}$,

[3]Note that, by themselves, these K steps do not necessarily create a contiguous path to a leaf node. They merely expose the child nodes of the best node in the stack at each level of the tree.

[4]Or after having encountered a stack whose nodes all have path metrics greater than B.

TABLE I
PROPOSED LIST CONSTRUCTION ALGORITHM

Input data: $\mathbf{y}$; $\mathbf{H}$; $p(b_k)$; K; a bound on the list size, L; a bound on the number of nodes visited, N.
Variables: $\mathbf{E}$; one stack per level, $\mathcal{S}_k$; the search order of the stacks, $\mathbf{t}$; a bound on the path metric, B.
Output: the list, $\hat{\mathcal{L}}$.
Preparatory computations:

1) Using $\mathbf{y}$, $\mathbf{H}$, and $p(b_k)$, select $\mathbf{E}$ and $\mathbf{t}$. Perform the QR decomposition of $\mathbf{HE}$.

Preliminary step: Greedy depth first search

2) Generate the child nodes of the root node and place them in $\mathcal{S}_1$. Set $k = 1$.
3) While $k \leq K-1$, remove from $\mathcal{S}_k$ the node with the smallest metric. Generate all that node's child nodes and place them in $\mathcal{S}_{k+1}$. Increment k.
4) ($k = K$) Select the node from $\mathcal{S}_K$ with the smallest metric, and place it in the list $\hat{\mathcal{L}}$. Set B to the path metric of this node. Clear $\mathcal{S}_K$.

Bounded best first search per level

5) Examine the stacks in the order imposed by $\mathbf{t}$ and select the first non-empty stack. If all stacks are empty, terminate. Otherwise, set k to the index of the first non-empty stack.
6) Select the node in $\mathcal{S}_k$ with the smallest path metric. If that metric is greater than B, clear $\mathcal{S}_k$, and return to 5. Otherwise, remove this node from $\mathcal{S}_k$, and generate all its child nodes.
   a. If $k < K$, place the child nodes into $\mathcal{S}_{k+1}$. If the number of nodes visited is $< N$, increment k and return to 6, otherwise, terminate.
   b. If $k = K$, put those child nodes with metrics $\leq B$ into $\hat{\mathcal{L}}$. If the number of nodes visited is $< N$ and $|\hat{\mathcal{L}}| < L$, return to 5, otherwise, terminate.

and the best-first search per level processing is restarted from the first nonempty stack in that ordering. The variables $\mathbf{E}$ and $\mathbf{t}$ are set prior to each demodulation iteration and enable the designer to exert some control over the way in which the tree is explored. Some candidate orderings are described below.

1) Vertical Bell Labs Layered Space-Time (V-BLAST) Symbol and Restart Orderings: In the first iteration, no *a priori* information is available, and a natural choice for symbol ordering $\mathbf{E}$ is the V-BLAST ordering [15], in which the symbol to be expanded at the next level of the tree is the one with the largest signal-to-interference-plus-noise ratio (SINR). In the generation of the list, it may be fruitful to examine those symbols with low SINR in the greatest detail. This suggests a choice of $\mathbf{t} = [K, K-1, \ldots, 1]$. In the subsequent demodulation iterations, we will retain the same orderings, which means that in this case the ordering of the search is determined by the channel and noise realization, and that the decoder exerts no influence over the ordering.

2) Symbol and Restart Orderings Based on A Priori Information: The principle of the V-BLAST ordering is to place the symbols about which we are most confident at the top of the tree. When we have *a priori* information (i.e., after the first demodulation iteration), we can choose to use the likelihoods provided by the decoder as the measure of confidence instead of the SINR. In particular, if we let $P(s_j^*)$ denote the largest of the prior probabilities for symbols at level j, we can arrange the symbols in descending order of $P(s_j^*)$. (We will use the V-BLAST ordering for the first iteration.) As the deep nodes in the tree represent the symbols about which we are least confident, we will use the restart ordering $\mathbf{t} = [K, K-1, \ldots, 1]$.

3) Restart Ordering Based on A Priori Information: A weakness of the previous ordering is that, at each demodulation–decoding iteration, the *a priori* information is updated, and hence, $\mathbf{E}$ may change. If $\mathbf{E}$ does change, then the QR decomposition of $\mathbf{HE}$ that is implicit in (4) will have to be repeated, and this adds to the computational cost of the algorithm. An alternative is to sort the stacks instead of the symbols. That is, we retain the V-BLAST symbol ordering, and upon restart, we examine the stacks in increasing order of $P(s_j^*)$.

4) Natural Restart Order: In this approach, we order the symbols according to the V-BLAST ordering, and upon restart, we examine the stacks in their natural order. That is, $\mathbf{t} = [1, 2, \ldots, K]$, and we examine the stacks starting from the top of the tree. The motivation for doing so is to provide a diverse collection of candidate paths in the stacks at each level of the tree.

B. Bounding the Path Metrics

One of the difficulties encountered in the direct application of best-first search strategies to MIMO soft demodulation is the breadth of the nodes that are visited and the consequent computational cost and memory requirement. We address this issue by identifying those nodes in the tree that have a path metric greater than some prespecified bound B, and cutting the subtrees below such nodes from the tree. To ensure that this bound is adapted to the channel realization and the *a priori* information, it is determined by first performing a preliminary greedy depth-first search[5] that generates a single leaf node, and then selecting that node's path metric as the threshold B. (This preliminary search also populates the stacks at each level of the tree.) In the first demodulation iteration, when there is no *a priori* information, the resulting leaf node corresponds to the output of a zero-forcing decision feedback detector with the ordering prescribed by $\mathbf{E}$ [9], [10].

C. Bounding the Complexity

If we were to run the proposed algorithm until all the leaf nodes with a path metric less than B were found, then our approach would be reminiscent of some adaptations of the sphere decoding (SD) algorithm to list-based soft demodulation [1], [4], [5].[6] However, the goal of the proposed algorithm is to generate a sizeable collection of good leaf nodes in the early stages, so that a good performance can be obtained even if the algorithm is terminated before all the leaf nodes with metrics less than B are found. Since the dominant operations in the tree search are those that are repeated at each node, and since the size of the list determines the complexity of computing the list approximation of the LLRs, the key factors in the computational cost of the algorithm are the number of nodes visited in the tree search and the size of the list. The proposed algorithm provides an explicit control over both of these terms, and we will show in Section V that these controls provide a convenient way to explore the performance–complexity tradeoff.

IV. Likelihood Computation

The goal of the multistack algorithm in Section III (and that of the related algorithms [1]–[7]) is to efficiently construct a reduced-sized list $\hat{\mathcal{L}}$ with which the LLRs in (2) can be approximated. However, as in the related algorithms, after generating $\hat{\mathcal{L}}$ using the algorithm in Table I, there may be bit positions for which $\hat{\mathcal{L}}_{i,0}$ or $\hat{\mathcal{L}}_{i,1}$ is empty, and such cases make the list approximation of the LLRs problematic. Therefore, we will generate an "enriched" list $\hat{\mathcal{L}}'$ by adding all those bit vectors that are within a Hamming distance of at least one member of the subset of $\hat{\mathcal{L}}$ containing the bit vectors with the J smallest metrics [8]. This enriched list can be generated by simply flipping one bit at a time of each list member.[7]

[5]In a greedy depth-first search, the levels of the tree are sequentially expanded, and the child node with the smallest branch metric is selected at each step.

[6]That said, our simple choice for the threshold B avoids the "trial-and-error" methods in some approaches to list sphere decoding [1].

[7]If $\hat{\mathcal{L}}$ has L members, then the enriched list has at most $L + JMK$ members. However, the simulation results indicate that many of the bit-flipped vectors are already members of $\hat{\mathcal{L}}$, and hence, $L' = |\hat{\mathcal{L}}'|$ is typically much smaller than $L + JMK$.

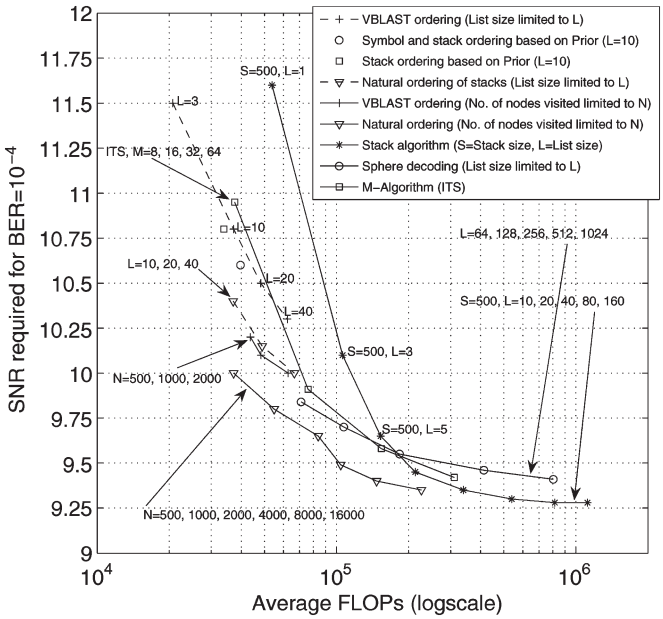

Fig. 4. Tradeoff between the SNR required for a BER of 10^{-4} and the average FLOPs per channel use for different algorithms for a 4×4 MIMO-BICM transmission scheme with 16-QAM symbols.

Once the enriched list has been constructed, the LLR in (2) can be approximated by performing the "max-log" approximation [1] over the sublists $\hat{\mathcal{L}}'_{i,1}$ and $\hat{\mathcal{L}}'_{i,0}$, respectively. That is, we will approximate the LLR only using the dominant vector in each sublist. To guard against severe overestimation or underestimation of the soft information caused by the list and max-log approximations [6], we will employ the common practice of clipping the approximated LLRs to a certain range [6]; in our case, to the interval $[-5, 5]$.

V. Simulation Results

We consider a narrowband MIMO transmission over a Rayleigh block-fading channel with channel gains that are independent and identically distributed (i.i.d.) zero-mean circular complex Gaussian random variables of unit variance. To facilitate comparisons of our results with those in [1] and [2], we employ the same transmitter and receiver components and parameters. That is, at the transmitter, we use a rate-1/2 punctured parallel concatenated turbo code with block length 8192 and (5, 7) recursive systematic convolutional codes as the component codes, and the V-BLAST transmission scheme [15].[8] At the receiver, we use the conventional Bahl, Cocke, Jelinek, and Raviv (BCJR) algorithm to decode the constituent codes of the turbo code. We perform eight turbo decoding iterations before we pass the soft information back to the demodulator, and four demodulation–decoding iterations. We will consider two MIMO systems; a system with four transmit and four receive antennas with Gray-mapped 16 quadratic-amplitude modulation (16-QAM) symbols, and an 8×8 MIMO system with Gray-mapped QPSK symbols. The size of the complete list for both of these systems is $|\mathcal{L}| = 65\,536$.

In Figs. 4 and 5, we have provided performance-versus-complexity tradeoff curves for a variety of soft MIMO demodulators, each with their LLRs clipped to $[-5, 5]$.[9] The performance is measured in terms of the SNR required to achieve a bit error rate (BER) of 10^{-4} after four

[8]The (different) interleavers in the turbo code and in the BICM transmitter are selected from randomly generated candidates in each Monte Carlo iteration.

[9]Using these tradeoff curves, one can develop a notion of an "efficient frontier" for soft MIMO demodulation that is similar in spirit to the efficient frontier for hard demodulation in [16].

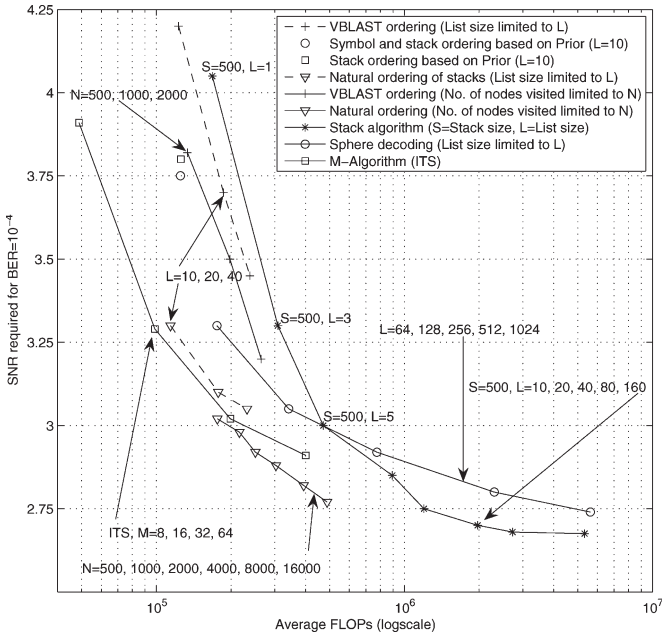

Fig. 5. Tradeoff between the SNR required for a BER of 10^{-4} and the average FLOPs per channel use for different algorithms for a 8×8 MIMO-BICM transmission scheme with QPSK symbols.

demodulation–decoding iterations, and the complexity is measured by explicitly computing the average number of floating-point operations (FLOPs)[10] that are required to generate the (enriched) list $\hat{\mathcal{L}}'$ and to compute the approximate LLRs for one channel use. Qualitatively similar results are obtained if one considers other measures of complexity, such as the peak FLOPs per channel use or the tail probability of the empirical complexity distribution (cf. [17]). However, the advantages of the proposed demodulator tend to be greater in those cases because it was designed to provide good performance with low complexity.

In Figs. 4 and 5, we consider the multistack algorithm with the four different symbol and restart orderings mentioned in Section III-A and limits placed on either the list size (using L) or the number of nodes visited (using N). (The bit-flipping scheme in Section IV was applied to the best $J = 20$ members of the original list.) We also consider the stack algorithm, the list SD algorithm, and the M-algorithm. For the stack algorithm, we chose the list sequential (LISS) method in [2]. In that method, once an initial list of size L has been found, the LLRs are approximated using an augmented list that incorporates information from the incomplete paths of the tree. In our implementation of the LISS algorithm, we used a stack of size 500 and different initial list sizes L (mentioned on the figures), and we augmented the list to a size of 100. For the list SD algorithm, we used the method in [1], in which the list is generated only once per channel use. While that removes the computational load of list generation for the subsequent demodulation iterations, the list does not adapt to the updated *a priori* information from the decoder, and hence rather long lists are required for good performance. To obtain a target list size, the list sphere decoder in [1] employs a trial-and-error method to determine the appropriate search radius. However, we have excluded the FLOPs allocated to this task

[10]Although the demodulators compared here are all based on tree-search methods, there are significant differences in the computational effort required to process a given node, and counting the FLOPs enables us to take these differences into account. Counting the FLOPs also enables us to incorporate the computational cost of the QR decompositions that are performed and the impact of the different list sizes on the cost of the list approximation of the LLRs.

TABLE II
AVERAGE SIZE OF THE PRELIMINARY LIST $\hat{\mathcal{L}}$ AND THE ENRICHED LIST $\hat{\mathcal{L}}'$, AND THE AVERAGE NUMBER OF NODES VISITED N', FOR THE MULTISTACK ALGORITHM WITH NATURAL OR V-BLAST RESTART ORDERINGS, AND A LIMIT L ON THE PRELIMINARY LIST SIZE OR A LIMIT N ON THE NUMBER OF NODES VISITED. THE SCENARIO CONSIDERED IS THAT IN FIG. 5

System\Iteration	1st			2nd			4th		
	$\lvert\hat{\mathcal{L}}\rvert$	$\lvert\hat{\mathcal{L}}'\rvert$	N'	$\lvert\hat{\mathcal{L}}\rvert$	$\lvert\hat{\mathcal{L}}'\rvert$	N'	$\lvert\hat{\mathcal{L}}\rvert$	$\lvert\hat{\mathcal{L}}'\rvert$	N'
Natural, $N = 500$	27.2	264	498	18.1	187	461	1.6	23	60
Natural, $L = 20$	14.3	191	987	11.6	152	705	2.4	34	122
VBLAST, $N = 1000$	38.3	237	959	25	191	792	1.5	22	58
VBLAST, $L = 20$	13.8	180	1132	12.2	158	836	4.1	56	235

in the curves in Figs. 4 and 5. For the M-algorithm, we have used the efficient iterative tree search (ITS) implementation in [6].

One way to gauge the significance of the SNR gains in Figs. 4 and 5 is to compute the SNR threshold for the systems under consideration (cf. [1]). For the 4×4 system with 16-QAM symbols, that threshold is about 6.9 dB, and for the 8×8 system with QPSK symbols, it is about 1.6 dB. Since our focus is on the soft demodulator rather than the whole transceiver, an arguably more relevant benchmark is the performance of the given MIMO-BICM-IDD system with precise soft demodulation [cf. (2)]. Although that demodulator is computationally infeasible, the performance of the LISS method [2] for large list sizes provides an indication of its performance, because the stack algorithm generates the leaf nodes of the tree in increasing order of their path metrics, and the list augmentation process incorporates information from the incomplete paths in the tree.

Let us first make some observations regarding the performance of the different instances of the proposed algorithm. By comparing the dashed and solid curves with the same symbol, it is apparent that for a given computational cost, limiting the number of nodes visited provides better performance than limiting the size of the preliminary list, at least in the low computational cost region that we have examined. In addition, by comparing curves with different symbols, it is apparent that the natural restart ordering provides better performance for a given computational cost. To provide insight into these comparisons, we selected four instances of the proposed algorithm that result in about the same computational cost in Fig. 5; two that employ the natural restart ordering and two that employ the V-BLAST ordering. For these demodulators, Table II provides the average number of nodes visited, the average size of the preliminary list, and the average size of the enriched list for the first, second, and fourth demodulation–decoding iterations. For each ordering, the scheme with the limit on the number of nodes visited provides, on average, a larger enriched list in the first two iterations than the scheme with the bound on the size of the preliminary list, and it visits fewer nodes in generating these lists. The richness of these lists enables the demodulator to more effectively leverage the power of the outer decoder. As shown in Table II, this results in a significant reduction in the number of nodes visited in the fourth iteration (and hence a reduction in the computational cost of that iteration), and as shown in Fig. 5, it also results in a reduction of the SNR that is required to achieve the desired target error rate.

The tradeoff curves in Figs. 4 and 5 for the stack (LISS) algorithm in [2] demonstrate the improved performance that can be obtained by searching for the best leaf nodes in the sequence. However, these figures also demonstrate the significant computational cost of that search. In particular, by employing the proposed multistack algorithm with the natural restart ordering and a bound on the number of nodes visited, we obtain a performance–complexity tradeoff that dominates that of the stack algorithm in the low-complexity region. Figs. 4 and 5 also show that the proposed method offers similar performance to the list sphere decoder [1] at a significantly lower computational

cost and better performance than the M-algorithm [6] for the same computational cost. The performance–complexity tradeoff of all the demodulators in Figs. 4 and 5 can be improved by employing (unbiased) MMSE preprocessing [10], [18], but the relative tradeoffs remain qualitatively similar. The relative tradeoffs after fewer than four demodulation–decoding iterations are also qualitatively similar.

VI. Conclusion

We have proposed a tree-search algorithm for list-based MIMO soft demodulation. The proposed algorithm is based on the "best-first" search principle used in the stack algorithm, but rather than applying that principle to a single (global) stack, the global stack is partitioned into a stack for each level of the tree, and the algorithm sequentially proceeds by performing one best-first search step in each of these stacks in the natural ordering of the tree. By assigning appropriate priorities to the level at which this best-first search per level processing restarts once a leaf node has been obtained, we have shown that the proposed approach can achieve a performance–complexity tradeoff that dominates those of the stack (LISS) algorithm in [2], the list sphere decoder [1], and the M-algorithm [6] in the low-complexity region.

References

[1] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[2] S. Baro, J. Hagenauer, and M. Witzke, "Iterative detection of MIMO transmission using a list-sequential (LISS) detector," in *Proc. IEEE Int. Conf. Commun.*, Anchorage, AK, May 2003, vol. 4, pp. 2653–2657.

[3] J. Hagenauer and C. Kuhn, "The list-sequential (LISS) algorithm and its application," *IEEE Trans. Commun.*, vol. 55, no. 5, pp. 918–928, May 2007.

[4] J. Boutros, N. Gresset, L. Brunel, and M. Fossorier, "Soft-input soft-output lattice sphere decoder for linear channels," in *Proc. IEEE Global Telecomm. Conf.*, San Francisco, CA, Dec. 2003, vol. 3, pp. 1583–1587.

[5] H. Vikalo, B. Hassibi, and T. Kailath, "Iterative decoding for MIMO channels via modified sphere decoding," *IEEE Trans. Wireless Commun.*, vol. 3, no. 6, pp. 2299–2311, Nov. 2004.

[6] Y. L. C. de Jong and T. J. Willink, "Iterative tree search detection for MIMO wireless systems," *IEEE Trans. Commun.*, vol. 53, no. 6, pp. 930–935, Jun. 2005.

[7] S. Bittner, E. Zimmermann, W. Rave, and G. Fettweis, "List sequential MIMO detection: Noise bias term and partial path augmentation," in *Proc. IEEE Int. Conf. Commun.*, Istanbul, Turkey, Jun. 2006, pp. 1300–1305.

[8] D. J. Love, S. Hosur, A. Batra, and R. W. Heath, Jr., "Space–time Chase decoding," *IEEE Trans. Wireless Commun.*, vol. 4, no. 5, pp. 2035–2039, Sep. 2005.

[9] J. Luo, K. R. Pattipati, P. Willett, and G. M. Levchuk, "Fast optimal and suboptimal any-time algorithms for CDMA multiuser detection based on branch and bound," *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 632–642, Apr. 2004.

[10] A. D. Murugan, H. El Gamal, M. O. Damen, and G. Caire, "A unified framework for tree search decoding: Rediscovering the sequential decoder," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 933–953, Mar. 2006.

[11] J. Anderson and S. Mohan, "Sequential coding algorithms: A survey and cost analysis," *IEEE Trans. Commun.*, vol. COM-32, no. 2, pp. 169–176, Feb. 1984.

[12] B. Hassibi and B. Hochwald, "High-rate codes that are linear in space and time," *IEEE Trans. Inf. Theory*, vol. 48, no. 7, pp. 1804–1824, Jul. 2002.

[13] R. A. Horn and C. R. Johnson, *Matrix Analysis*. New York: Cambridge Univ. Press, 1985.

[14] P. Chevillat and D. Costello, Jr., "A multiple stack algorithm for erasurefree decoding of convolutional codes," *IEEE Trans. Commun.*, vol. COM-25, no. 12, pp. 1460–1470, Dec. 1977.

[15] G. J. Foschini, G. Golden, R. Valenzuela, and P. Wolniansky, "Simplified processing for high spectral efficiency wireless communication employing multi-element arrays," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 11, pp. 1841–1852, Nov. 1999.

[16] F. Hasegawa, J. Luo, K. R. Pattipati, P. Willett, and D. Pham, "Speed and accuracy comparison of techniques for multiuser detection in synchronous CDMA," *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 540–545, Apr. 2004.

[17] M. Nekuii, "Soft demodulation schemes for MIMO communication systems," Ph.D. dissertation, McMaster Univ., Hamilton, ON, Canada, Aug. 2008.

[18] E. Zimmermann and G. Fettweis, "Unbiased MMSE tree search detection for multiple antenna systems," in *Proc. Int. Symp. Wireless Pers. Multimedia Commun.*, San Diego, CA, Sep. 2006.

Wireless Access Point Voice Capacity Analysis and Enhancement Based on Clients' Spatial Distribution

Ahmed Shawish, *Student Member, IEEE*, Xiaohong Jiang, *Member, IEEE*, Pin-Han Ho, *Member, IEEE*, and Susumu Horiguchi, *Senior Member, IEEE*

***Abstract*—An efficient voice-over-IP (VoIP) support at the wireless access point (AP) of a wireless LAN (WLAN) remains a challenge for the last-mile wireless coverage of IP networks with mobility support. Due to the limited bandwidth that is available in WLANs, an accurate analysis of the voice capacity in such networks is crucial for the efficient utilization of their resources. The available analytical models only provide the upper and lower bounds on voice capacity, which may significantly overestimate or underestimate the WLAN's capability of supporting VoIP and, thus, are not suitable for the mentioned purpose. In this paper, we focus on the voice capacity analysis of a wireless 802.11(a/b) AP running the distributed coordination function (DCF). In particular, we show that by incorporating the clients' spatial distribution into the analysis, we are able to develop a new analytical model for a much more accurate estimation of the average voice capacity. By properly exploring this spatial information, we further propose a new scheme for AP placement such that the overall voice capacity can be enhanced. The efficiency of the new voice capacity model and the new AP placement scheme is validated through both analytical and simulation studies.**

***Index Terms*—IEEE 802.11 distributed coordination function (DCF), spatial distribution, voice capacity, voice over IP (VoIP), wireless access point (AP).**

I. Introduction

Voice over Internet Protocol (VoIP) is one of the fastest-growing Internet applications due to its cost efficiency and its promising ability to merge voice communication with other multimedia and data applications [1]. Driven by the huge demands for flexible connectivity and portable access at reduced costs, wireless local area networks (WLANs) are increasingly making their way into residential, commercial, industrial, and public areas. While the majority of traffic in WLAN are data, it is expected that the voice application will become

Manuscript received February 12, 2008; revised July 2, 2008. First published September 9, 2008; current version published May 11, 2009. The review of this paper was coordinated by Prof. R. Jäntti.

A. Shawish is with the Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan, and also with the Faculty of Computer and Information Sciences, Ain Shams University, Cairo 11566, Egypt (e-mail: ahmedmg@ecei.tohoku.ac.jp).

X. Jiang and S. Horiguchi are with the Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan (e-mail: jiang@ecei.tohoku.ac.jp; susumu@ecei.tohoku.ac.jp).

P.-H. Ho is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: pinhan@bbcr.uwaterloo.ca).

Digital Object Identifier 10.1109/TVT.2008.2005523