REDUCED-COMPLEXITY DEMODULATION FOR MIMO-BICM-IDD USING MODIFIED STACK ALGORITHMS

Mehran Nekuii and Timothy N. Davidson

Department of Electrical and Computer Engineering
McMaster University, Hamilton, Ontario, Canada

ABSTRACT

Bit-interleaved coded-modulation (BICM) with iterative demodulation and decoding (IDD) is a popular architecture for the development of practical communication schemes that operate at rates close to capacity. In multiple-input multiple-output (MIMO) BICM-IDD schemes, a key computational bottleneck is the demodulation step; that is, the extraction of "soft" information about the transmitted bits from the channel output. The concept of list-based demodulation provides a convenient framework for managing the trade-off between accuracy and computational cost in the extraction of this soft information, especially when tree-search techniques are used to construct the list. In this paper, we will propose several list-based demodulators based on modifications of the stack algorithm for searching a tree. The modifications partition the stack in ways that enable efficient and effective searching of the tree from the perspective of list-based demodulation. Simulation results show that the proposed demodulators achieve desirable trade-offs between complexity and performance.

Index Terms— MIMO demodulation, Iterative demodulation and decoding, Stack algorithm.

1. INTRODUCTION

Wireless communication systems with multiple antennas at the transmitter and receiver are a key component in the development of wireless communication standards as they provide the potential for reliable communication at data rates that are substantially larger than the corresponding single antenna system [1, 2]. However, the computational effort required for optimal detection of multiple-input, multiple-output (MIMO) schemes that operate at such high spectral efficiencies is often beyond the capabilities of the envisioned communication devices, and hence there has been considerable interest in the development of transceivers that balance the competing demands of rate and computational efficiency. A popular transmission strategy that enables considerable flexibility in the selection of an appropriate balance is a MIMO version (e.g., [3]) of the bit interleaved coded modulation strategy (BICM, e.g., [4]); see Fig. 1. Given the complexity of optimal decoding, the standard reception strategy is to adopt a bit-wise iterative "soft" demodulation and decoding (IDD) approach (often called the "Turbo principle") that attempts to maximize the *a posteriori* probability of each bit in the message. Although this iterative demodulation and decoding strategy offers a substantial reduction in computational cost, the demodulation step, which involves the extraction of a sufficiently accurate approximation of the likelihood of each bit from the output of the MIMO channel, remains a substantial computational burden. The use of list-based techniques to manage this burden is the core topic of this paper. (Other approaches to managing this burden include those based on linear equalization, e.g., [5], and those based on soft interference cancellation techniques, e.g, [6].)

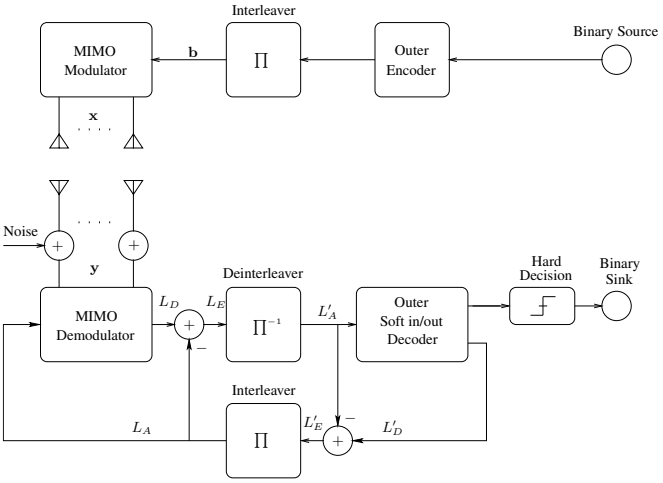

Fig. 1. MIMO BICM-IDD transceiver.

The principle of list-based demodulation (e.g., [3, 7]) is to efficiently obtain a list of the candidate bit-vectors that generate the dominant components of the soft information, and then to compensate the soft information for components that may be missing. In this paper, we emphasize the tree-search interpretation of MIMO demodulation; e.g., [8, 9]. In particular, we provide modified versions of the stack algorithm (e.g., [9]) for searching the tree. In addition to the use of a modified path metric that includes prior information (c.f., [10]), the modifications involve the partitioning of the stack so that there is one stack per layer of the tree. Hence, our search algorithms have an additional degree of freedom — the order in which the stacks are explored. As we will show, the combination of this extra degree of freedom and some simple termination criteria offers considerable control over the trade-off between performance and complexity in MIMO demodulation.

2. SYSTEM MODEL

We will consider the standard MIMO BICM-IDD transceiver structure [3] illustrated in Fig. 1. Since the focus of the paper is on the MIMO demodulation task, we will choose conventional compo-

This work was supported in part by a Premier's Research Excellence Award from the Government of Ontario. The work of the second author is also supported in part by the Canada Research Chairs program.

nents for the rest of the system. In particular, we consider a generic MIMO channel in which the received signal vector can be written as $\mathbf{y} = \mathcal{H}\mathbf{x} + \mathbf{v}$, where $\mathcal{H}$ is an $N_r \times N_t$ matrix of channel gains, $\mathbf{x}$ is the transmitted signal vector, and $\mathbf{v}$ is a vector of noise samples. We consider communication schemes in which $\mathbf{x}$ is a linear function of a vector $\mathbf{s}$ of $K \leq N_t$ channel symbols, each of which is selected from a scalar constellation $\mathcal{C}$ of size 2^M by mapping a sub-block of M bits from the vector of interleaved encoded bits to $\mathcal{C}$. Therefore, the received signal vector can be written as

$$\mathbf{y} = \mathbf{H}\mathbf{s}(\mathbf{b}) + \mathbf{v}, \tag{1}$$

where $\mathbf{H}$ incorporates the linear mapping of $\mathbf{s}$ to $\mathbf{x}$ and $\mathbf{b}$ is a block of MK interleaved encoded bits. The encoded bits are generated using a standard binary encoder, such as a convolutional, turbo or LDPC encoder.

At the receiver, the role of the (coherent) soft demodulator is to extract information regarding the elements of $\mathbf{b}$, b_i, from the channel measurement $\mathbf{y}$, given knowledge of $\mathbf{H}$ and, at the second and subsequent iterations, prior information on the bit probabilities provided by the decoder. Since b_i is binary, the information to be extracted by the demodulator can be succinctly captured in the log-likelihood ratio (LLR):

$$\begin{aligned} L_{Di} \triangleq L(b_i|\mathbf{y},\mathbf{H}) &= \log \frac{p(b_i=1|\mathbf{y},\mathbf{H})}{p(b_i=0|\mathbf{y},\mathbf{H})} \\ &= \log \frac{\Sigma_{\mathcal{L}_{i,1}} p(\mathbf{y}|\mathbf{b},\mathbf{H})p(\mathbf{b})}{\Sigma_{\mathcal{L}_{i,0}} p(\mathbf{y}|\mathbf{b},\mathbf{H})p(\mathbf{b})}, \end{aligned} \tag{2}$$

where $\mathcal{L}$ is the list of all the binary vectors $\mathbf{b}$, $\mathcal{L}_{i,1}$ is the sub-list of vectors that have 1 at the ith bit position, and $\mathcal{L}_{i,0}$ is defined analogously. If we model each element of the noise vector $\mathbf{v}$ in (1) by an i.i.d. zero-mean circular complex Gaussian random variable of variance σ^2, per real dimension, we have that:

$$p(\mathbf{y}|\mathbf{b},\mathbf{H}) \propto e^{-\|\mathbf{y}-\mathbf{H}\mathbf{s}(\mathbf{b})\|_2^2/(2\sigma^2)}.$$

Furthermore, if the interleaver in the transmitter portion of Fig. 1 is designed well enough, the simplifying assumption that the elements of $\mathbf{b}$ are independent is quite mild; i.e.,

$$p(\mathbf{b}) \approx \textstyle\prod_{k=1}^{MK} p(b_k) = e^{\sum_{k=1}^{MK} \log p(b_k)}.$$

In that case, the negative of the logarithm of the summand in the numerator and denominator of (2) is (c.f., [10])

$$D(\mathbf{b}) \triangleq \|\mathbf{y} - \mathbf{H}\mathbf{s}(\mathbf{b})\|_2^2 - 2\sigma^2 \textstyle\sum_{k=1}^{MK} \log p(b_k). \tag{3}$$

Therefore, the summations in (2) can be written as summations of $e^{-D(\mathbf{b})}$, with $\mathbf{b}$ being selected from $\mathcal{L}_{i,1}$ or $\mathcal{L}_{i,0}$, respectively. However, as each list contains 2^{MK-1} terms, there has been considerable interest in schemes that enable the approximation of (2) with lower computational cost.

3. LIST-BASED MIMO DEMODULATION

Given the rapid growth of the computational cost of (2) with MK, exact MIMO demodulation is feasible only in rather low rate scenarios. The principle behind list-based demodulation schemes is to reduce the computational cost of (2) by replacing the list $\mathcal{L}$ with a reduced-size list $\hat{\mathcal{L}}$ containing the dominant components in the summations. The problem of finding the dominant components corresponds to finding binary vectors $\mathbf{b}$ that yield small values for (3).

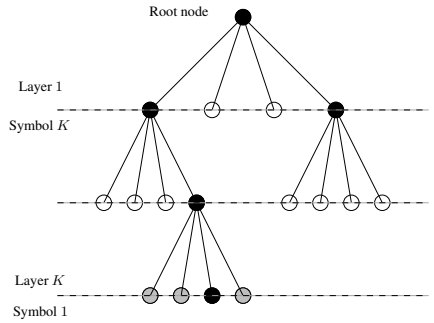

Fig. 2. An example of a 2^M-ary tree search with $M = 2$ and $K = 3$.

The implementation of that search is significantly simplified when the QR decomposition is used to make the implicit tree structure of the MIMO demodulation problem explicit; e.g., [3, 7–10]. In particular, if we let $\mathbf{HE} = \mathbf{QR}$ denote the QR decomposition of $\mathbf{HE}$, where $\mathbf{E}$ is a column permutation matrix that determines the arrangement of the symbols in the tree, and define $\tilde{\mathbf{y}} = \mathbf{Q}^T\mathbf{y}$, $\tilde{\mathbf{v}} = \mathbf{Q}^T\mathbf{v}$, $\tilde{\mathbf{s}} = \mathbf{E}^T\mathbf{s}$, and $\mathbf{R}_j$ to be the j'th row of $\mathbf{R}$, then (3) can be rewritten as:

$$\begin{aligned} D(\mathbf{b}) &= \|\tilde{\mathbf{y}} - \mathbf{R}\tilde{\mathbf{s}}\|_2^2 - 2\sigma^2 \textstyle\sum_{k=1}^{MK} \log p(b_k) \\ &= \textstyle\sum_{j=0}^{K-1} (\tilde{y}_{K-j} - \mathbf{R}_{K-j}\tilde{\mathbf{s}})^2 - 2\sigma^2 \log p(\tilde{s}_{K-j}), \end{aligned} \tag{4}$$

where $p(\tilde{s}_i) = \prod_k p(b_k)$ with the product being over only those bits that index the symbol $\tilde{s}_i$, and we have dropped the explicit dependence of $\mathbf{s}$ on $\mathbf{b}$ for simplicity. In (4), the j'th summand depends only on symbols j to K, and hence the inherent tree structure is exposed. Indeed, if the root node of the tree is called layer zero, the child nodes of the nodes at layer j are indexed by the bits that are mapped to $\tilde{s}_{K-j}$. Therefore, the tree is a 2^M-ary tree, where M is the number of bits per (scalar) symbol. See Fig. 2 for an example. For later convenience, we observe that the path metric for a node at level J is the sum of the first J terms in (4).

Although the list construction problem has been reduced to a tree-search problem, direct application of conventional tree-search algorithms is complicated by the fact that we are looking for a number of leaf nodes with small values of (4), rather than being concerned only with the leaf node with the smallest value for (4). The algorithms that we will employ are modifications of the stack algorithm; e.g., [7, 9]. The stack algorithm is a "best first search" algorithm, in the sense that all the "exposed" nodes of the tree are stored in a stack, and at each step the algorithm explores the child nodes of the member of the stack with the smallest path metric. As a result, the first leaf node that the algorithm would select for expansion corresponds to the bit-vector with the smallest value for (4). However, the stack algorithm may partially explore many branches of the tree without generating many leaf nodes with small values for (4). As a result, it may expend a large amount of computational effort to find only a few of the dominant leaf nodes. The goal of the proposed modified stack algorithm is to obtain a larger set of the dominant leaf nodes with lower computational cost. In the proposed algorithm, the (global) stack is partitioned into separate stacks, one for each layer of the tree. At each step of the algorithm a layer is selected and the node in that layer's stack with the smallest path metric is expanded. Therefore, we will call these algorithms Best First Search per Layer (BFSPL) algorithms. In addition to the search strategy, the performance of a tree-search-based list demodulator also depends on how

Table 1. Proposed List Construction Algorithm

Input data: $\mathbf{y}$; $\mathbf{H}$; $p(b_k)$; M; a bound on the list size, L; a bound on the number of nodes visited, N.

Variables: $\mathbf{E}$; one stack per layer, $\mathcal{S}_k$; 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 $p(s_k)$, 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 layer

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.
   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 $|\hat{\mathcal{L}}| < L$, increment k and return to 5.

the search is terminated. We will terminate the search if a prespecified number of leaf nodes has been obtained, or if a prespecified number of nodes in the tree have been visited. The bound on the list size enables us to bound the complexity of computing the list version of (2), while the bound on the number of nodes visited limits the computational cost of the tree search.

In addition to these controls, we may also wish to control the "breadth" of the tree search, as this will help to focus the computational effort on the generation of dominant leaf nodes. One way in which this can be done is by performing a preliminary step that generates one candidate for the list and computes its metric. Since each summand in (4) is non-negative, if a given node in the tree has a path metric that is greater than that of the leaf node generated in the preliminary step, then the contribution to (2) of each of the leaf nodes lying "below" this node will be less than that of the leaf node generated in the preliminary step. This suggests that one can significantly reduce the search complexity without incurring a significant performance loss by cutting the tree at this node, and we will do so in our algorithm. In the preliminary step we will use a greedy depth first search, in which the layers are expanded sequentially, with the child node with the smallest metric being selected at each step.

A formal statement of the proposed algorithm is provided in Tab. 1. Some of the candidate orderings for the symbols, $\mathbf{E}$, and the searching of the stacks, $\mathbf{t}$, are described in the subsections below.

3.1. BFSPL with VBLAST symbol ordering.

In the first iteration, no *a'priori* information is available, and hence the leaf node found in the preliminary step is the same as the first leaf node found in the method in [8]. As suggested in [8], a natural choice for the ordering of the symbols is the V-BLAST ordering [11], in which the symbol to be expanded at each step is the one with the largest SINR. In the generation of the list, it is often fruitful to examine those symbols with low SINR in the greatest detail. Hence, an appropriate ordering of the stacks is $[K, K-1, \ldots, 1]$. In subsequent iterations, when *a'priori* information is available from the decoder, we will retain the same orderings. While this means that we do not have to perform additional QR decompositions, it does mean that, the ordering of the searches is determined by the channel and noise realizations and that the decoder exerts no influence over those orderings.

3.2. Symbol ordering 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., at the second and subsequent iterations of the receiver), we can choose to use the prior probabilities 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 symbol j, we can arrange the symbols in descending order of $P(s_j^*)$. (Since there is no *a'priori* information at the first iteration, we will use the V-BLAST ordering for that iteration.) As the deep nodes in the tree represent the symbols about which we are least confident, we will use the same stack ordering as the first method.

3.3. Stack ordering based on *a'priori* information.

A problem with the previous method is that at each demodulation iteration the *a'priori* information is updated and the tree may be reordered. When that occurs, $\mathbf{E}$ changes and we will have to repeat the QR decomposition. This will increase the complexity for systems with large values for K. An alternative is to sort the stacks, instead of the symbols. That is, we retain the V-BLAST tree ordering, and simply order the search of the stacks in increasing order of $P(s_j^*)$.

3.4. Straight ordering of stacks.

In this approach, we order the tree according to the V-BLAST ordering and re-order the search of the stacks as $[1, 2, \ldots, K]$. In some scenarios, this will increase the richness of the list membership and hence improve the performance of the demodulator.

4. LLR COMPUTATION

In order to reduce the complexity of computing (2), we will use the BFSPL strategies to generate a reduced-size list $\hat{\mathcal{L}}$ and we will use the consequent sub-lists $\hat{\mathcal{L}}_{i,0}$, $\hat{\mathcal{L}}_{i,1}$ to replace the complete sub-lists $\mathcal{L}_{i,0}$, $\mathcal{L}_{i,1}$ in (2), respectively. However, there may be bit positions for which $\hat{\mathcal{L}}_{i,0}$ or $\hat{\mathcal{L}}_{i,1}$ is empty, and in such scenarios the list approximation of (2) fails. In order to avoid those scenarios, one can "enrich" the list by adding all those bit vectors that are within a preset Hamming distance of at least one member of the list $\hat{\mathcal{L}}$; see, e.g., [12]. We will let $\hat{\mathcal{L}}'$ denote the enriched list. In our simulation studies, we will select a Hamming distance of 1, and hence the enriched list is generated by simply flipping one bit at a time of each

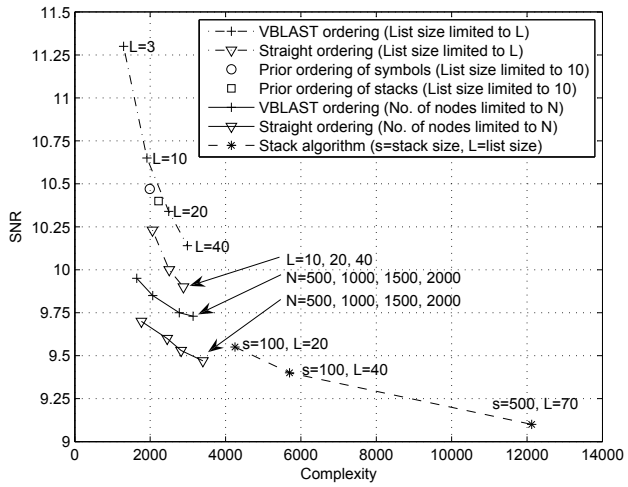

Fig. 3. SNR-Complexity trade-off for different search methods.

list member. In that case, if the original list $\hat{\mathcal{L}}$ has L members, then the enriched list has at most $L(KM+1)$ members. However, our 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 $L(KM+1)$.

Once the enriched list has been formed, the LLR in (2) can be approximated by performing the summations over the sub-lists $\hat{\mathcal{L}}'_{i,1}$ and $\hat{\mathcal{L}}'_{i,0}$, respectively. However, there may be cases in which these sub-lists are of different lengths. That can make the approximation of the LLR unbalanced and can cause under or over estimation. A well known strategy to mitigate that effect is to approximate the LLR using the dominant bit-vector in each sublist — a strategy that is called the max-log approximation [3]. A common final safe-guard against errors in the approximation of the LLR is to clip the approximated LLRs to a certain range [13]. In our simulation studies, the LLRs will be clipped to the interval $[-5, 5]$.

5. SIMULATION RESULTS

In order to evaluate the performance and computational cost of various demodulation strategies, we consider a MIMO system with 4 transmit and 4 receive antennas and a Rayleigh block-fading channel with channel gains that are i.i.d. standard complex Gaussian random variables. We employ a rate $\frac{1}{2}$ parallel concatenated turbo code with block length 8192 as the outer code. The component codes of the turbo code are both $(5, 7)$ recursive systematic convolutional codes. The (different) interleavers in the turbo code and in the BICM transmitter were selected from randomly generated candidates. The simple V-BLAST scheme [11] with 16-QAM symbols was adopted at the transmitter, and the conventional BCJR algorithm was used as the decoder of the constituent convolutional codes of the turbo code. Following [7], we perform 8 turbo decoding iterations before we pass the soft information back to the demodulator.

In Fig. 3, we have provided a performance versus complexity comparison of a variety of our proposed demodulators and that of the stack algorithm of [7]. Performance is measured in terms of the SNR required to achieve a bit error rate of 10^{-4} after four demodulation-decoding iterations and as a proxy for complexity we will add the total of the number of nodes visited in the tree to $K = 4$ times the size of the enriched list, $L' = |\hat{\mathcal{L}}'|$. This proxy accounts for the cost of the tree search and the cost of computing the list version of (2), in a reliable and repeatable way. However, it does not include the additional sorting costs associated with the standard implementation of the stack algorithm that has an active constraint on the stack size.

From Fig. 3, it is apparent that the instances of the proposed tree-search algorithm require substantially less computational effort than the stack algorithm. Furthermore, by controlling the bounds on the list size (dash-dot lines in the figure) or the maximum number of nodes visited (solid lines in the figure) one can adjust the trade-off between performance and complexity of these methods. Of these two bounding schemes, it appears that bounding the number of nodes provides a better trade-off. Also, for each bounding scheme, the straight stack ordering (shown with symbol $\triangledown$) appears to provide a superior trade-off to that of the other ordering methods. In summary, it appears that a combination of the straight ordering of the stacks and a complexity limit based on the number of visited nodes provides the best performance-complexity trade-off among the considered schemes.

6. REFERENCES

[1] G. J. Foschini and M. J. Gans, "On limits of wireless communications in a fading environment when using multiple antennas," *Wireless Personal Com.*, vol. 6, pp. 311–335, Mar. 1998.

[2] I. E. Telatar, "Capacity of multiple antenna Gaussian channels," *Eur. Trans. Telecom.*, vol. 10, pp. 585–595, Nov. 1999.

[3] B. M. Hochwald and S. ten Brink, "Achieving near capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, pp. 389–399, Mar. 2003.

[4] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved coded modulation," *IEEE Trans. Inform. Theory*, vol. 44, pp. 927–946, May 1998.

[5] M. R. McKay and I. B. Collings, "Capacity and performance of MIMO-BICM with zero-forcing receivers," *IEEE Trans. Commun.*, vol. 53, pp. 74–83, Jan. 2005.

[6] J. Choi, "MIMO-BICM iterative receiver with the EM based channel estimation and simplified MMSE combining with soft cancellation," *IEEE Trans. Signal Processing*, vol. 54, pp. 3247–3251, Aug. 2006.

[7] 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, May 2003, vol. 4, pp. 2653–2657.

[8] 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, pp. 632–642, Apr. 2004.

[9] 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. Inform. Theory*, vol. 52, pp. 933–953, Mar. 2006.

[10] H. Vikalo, B. Hassibi, and T. Kailath, "Iterative decoding for MIMO channels via modified sphere decoding," *IEEE Trans. Wireless Commun.*, vol. 3, pp. 2299–2311, Nov. 2004.

[11] G. J. Foschini, G.D. Golden, R.A. Valenzuela, and P.W. Wolniansky, "Simplified processing for high spectral efficiency wireless communication employing multi-element arrays," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 1841–1852, Nov. 1999.

[12] D. J. Love, S. Hosur, A. Batra, and R. W. Heath, Jr., "Space-time Chase decoding," *IEEE Trans. Wireless Commun.*, vol. 4, pp. 2035–2039, Sept. 2005.

[13] Y. L. C. de Jong and T. J. Willink, "Iterative tree search detection for MIMO wireless systems," *IEEE Trans. Commun.*, vol. 53, pp. 930–935, June 2005.