Reduced-Complexity Stack Decoder for MIMO Systems

Asma Mejri and Ghaya Rekaya-Ben Othman, Member, IEEE
Telecom ParisTech, 46 Rue Barrault, 75013 Paris, France, {amejri,rekaya}@telecom-paristech.fr

***Abstract*—In this work, we propose a novel sequential decoder for MIMO systems termed the Zigzag Stack decoder. The algorithm combines the search strategy of the Stack decoder with the Schnorr-Euchner zigzagging method. We show that the Zigzag Stack provides ML performance with a reduced complexity compared to the original Stack decoder and a complexity reduction of 40% in average over the commonly used sphere decoder.**

***Index Terms*—MIMO systems, Sequential decoders, Stack decoder, Schnorr-Euchner enumeration.**

I. Introduction

Multiple antenna technologies play a fundamental role in the design of most of the successful wireless communication systems due to their potential to increase the spectral efficiency and the transmission data rates. Several wireless standards such as the LTE and the WiMAX (IEEE 802.16) have incorporated MIMO communications to enhance the network performance and take advantage of the diversity brought by multiple antennas. A main challenge in such systems is the design of efficient and low-complexity receivers.

Linear structure of both coded and uncoded schemes allows to decode these systems using lattice decoders [1]. Optimal performance are obtained when the ML criterion is considered. In literature, several ML detection algorithms exist including the Sphere Decoder (SD) [2] and the Schnorr-Euchner [3]. A main drawback of these decoders is their increasing complexity when the constellation size or the number of antennas increase. Alternatively, suboptimal low-complexity decoders such as the ZF, the ZF-DFE and the MMSE [4] are implemented in practical systems presenting limited computational capabilities.

In this work, we are interested in sequential decoding for MIMO systems. Sequential decoders, mainly the Stack [5] and Fano [6], were originally used to decode convolutional codes transmitted over discrete memoryless channels. Later on, they were rediscovered and adapted for ML detection in MIMO systems [7]. We focus in this work on the Stack decoder having lower complexity than the Fano decoder [5]. This algorithm implements a *Best-First* tree-search strategy to find the closest vector to the received signal according to the ML criterion. At each level of the tree, the algorithm generates all the nodes corresponding to the detected symbols. For an increasing constellation size and high number of antennas, the algorithm requires a high computational complexity. In order to reduce this complexity, the Spherical-Bound Stack decoder (SB-Stack) has been recently proposed in literature [8]. The latter combines the Stack search strategy with the SD search region. The ML solution is sought inside a sphere centered at the received point. Limiting the search space to a spherical region under the SB-Stack decoder allows to reduce the decoding complexity and provide a complexity gain of at least 30% over the SD.

We propose in this work a novel low-complexity Stack decoder we term the Zigzag Stack. The algorithm uses in its essence the best-first tree-search strategy of the Stack decoder and the Schnorr-Euchner zigzaging technique. We show through Monte-Carlo simulations that our novel algorithm offers a complexity gain of at least $\%40$ over the SD while preserving ML performance. Moreover, we provide a parameterized version of the Zigzag Stack decoder offering a wide range of performance/complexity tradeoffs ranging from ML to ZF-DFE performance respectively for a zero-valued bias and for a bias parameter larger than 10.

II. System model and notations

Consider an $n_t \times n_r$ MIMO system with n_t transmit and n_r receive antennas using spatial multiplexing. The complex-valued representation of the channel output is given by:

$$\mathbf{y}_c = \mathbf{H}_c\mathbf{s}_c + \mathbf{w}_c \tag{1}$$

where $\mathbf{H}_c \in \mathbb{C}^{n_r \times n_t}$ denotes the channel matrix of elements drawn i.i.d. according to the distribution $\mathcal{N}(0,1)$ and assumed perfectly known at the receiver. $\mathbf{w}_c \in \mathbb{C}^{n_r}$ is the additive white Gaussian noise of variance $\sigma^2\mathbf{I}_{n_r}$ and $\mathbf{s}_c$ is composed of the complex-valued information symbols. In order to obtain a lattice representation of the channel output, we apply the complex-to-real transformation to get the real-valued system given by:

$$\mathbf{y} = \begin{bmatrix} \Re(\mathbf{H}_c) & -\Im(\mathbf{H}_c) \\ \Im(\mathbf{H}_c) & \Re(\mathbf{H}_c) \end{bmatrix} \begin{bmatrix} \Re(\mathbf{s}_c) \\ \Im(\mathbf{s}_c) \end{bmatrix} + \begin{bmatrix} \Re(\mathbf{w}_c) \\ \Im(\mathbf{w}_c) \end{bmatrix} \tag{2}$$

where $\Re(.)$ and $\Im(.)$ denote respectively the real and imaginary parts of a complex-valued vector. The equivalent channel output can then be written as:

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{w} \tag{3}$$

This system is to be considered in the decoding process. When a length-T Space-Time code is used, the channel output can be written in the same form of (1) with the equivalent channel matrix $\mathbf{H}_{eq}$ given by:

$$\mathbf{H}_{eq} = \mathbf{H}_c\mathbf{\Phi} \tag{4}$$

978-1-4799-8088-8/15/$31.00 ©2015 IEEE

where $\mathbf{\Phi} \in \mathbb{C}^{n_t T \times n_t T}$ corresponds to the coding matrix of the underlying code [9]. For ease of presentation and given that both uncoded and coded schemes result in a same real-valued lattice representation, we consider in the remaining of this work the spatial multiplexing and symmetric case with $n_t = n_r$ and let $n = 2n_t$.

A. ML Detection

According to the equivalent system obtained in (3), the received signal can be viewed as a point of the lattice generated by $\mathbf{H}$ and perturbed by the noise vector $\mathbf{w}$. Optimal ML detection remains therefore to solve for the closest vector in the $n-$dimensional lattice generated by $\mathbf{H}$ according to the minimization problem:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}_c \in \mathrm{QAM}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{H}\mathbf{s}\|^2 \tag{5}$$

In this work we are interested in sequential ML decoders implementing tree-search algorithms. For this reason, we need to move to the tree structure of MIMO systems. To do so, we first perform a QR decomposition of the channel matrix such that $\mathbf{H} = \mathbf{Q}\mathbf{R}$ where $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{R}$ is upper triangular. Given the orthogonality of $\mathbf{Q}$, (3) is equivalent to:

$$\tilde{\mathbf{y}} = \mathbf{Q}^t\mathbf{y} = \mathbf{R}\mathbf{s} + \mathbf{Q}^t\mathbf{w} \tag{6}$$

And the decoding problem remains to solve the equivalent system given by:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}_c \in \mathrm{QAM}}{\operatorname{argmin}} \|\tilde{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2 \tag{7}$$

The triangular structure of $\mathbf{R}$ reduces the search of the closest point to a sequential tree-search.

Nodes in the tree represent the different possible values of the symbols s_i. We recall that $s_i, i = 1, ..., n$ represent the real and imaginary components of the information vector $\mathbf{s}_c$. A tree branch represents two consecutive nodes (s_{i+1}, s_i). We define the *weight* metric relative to each node s_i as :

$$w_i(s_i) = \mid \tilde{y}_i - \sum_{j=i}^{n} R_{ij} s_j \mid^2 \tag{8}$$

The weight represents a metric for the branch (s_{i+1}, s_i). Due to the triangular structure of $\mathbf{R}$, we begin the search from the component s_n. We call the *child nodes* of s_i the components s_{i-1} and call a *path* of depth i in the tree, the length-$(n-i+1)$ vector defined by $\mathbf{s}^{(i)} = (s_n, s_{n-1}, ..., s_i)$. A node being in depth n is called a *leaf node*. The cumulated weight of a node s_i represents the metric of the path $\mathbf{s}^{(i)}$. It is therefore equal to the sum over all weights for different nodes forming the path according to:

$$w(s_i) = \sum_{j=i}^{n} w_j(s_j) \tag{9}$$

For a leaf node, the weight $w(s_1)$ corresponds to the Euclidean distance between the received signal $\tilde{\mathbf{y}}$ and $\mathbf{s}^{(1)}$ and is equal to $\mid \tilde{\mathbf{y}} - \mathbf{R}\mathbf{s}^{(1)} \mid^2$. Using this definition, the ML metric minimization of (7) remains to search for the path in the tree having the least cumulated weight.

III. Sequential Decoders

Sequential decoders were initially introduced for decoding convolutional codes. They are based on search algorithms in binary trees where tree'branches represent the binary values of the code. Two principal algorithms were introduced: the Fano decoder [6] and the Stack decoder [5]. These decoders were afterwards generalized to the case of MIMO systems. The search tree is no longer binary, it contains the different possible values of the information symbols. It was shown in [5] that the Stack decoder offers a lower complexity than the Fano decoder. Before describing our algorithm, we review in this section the main tree-search strategies existing in literature and recall the principle of the Stack decoder.

A. Literature Overview on Tree-search Strategies

1) Breadth-first strategy: Using this strategy, the algorithm starts with the root node and extends all its child nodes existing at the following level. For each explored node, the algorithm extends successively all its child nodes until reaching leaf nodes. Using this approach, the search over the tree is made in the width sense, i.e the algorithm explores all nodes s_i before moving to the level $i-1$ of the tree in such a way that all linear combinations $\mathbf{s}^{(i)} = (s_n, s_{n-1}, ..., s_i)$ are computed. This method implements then an exhaustive search over the tree.

2) Depth-first strategy: Using this strategy, starting from the root node, the algorithm explores the first child node s_n, then one of its child nodes s_{n-1}, and so on until reaching a leaf node s_1. Given this first path found, the algorithm goes back to the level 2 in the tree and explores the neighbors of the already explored node s_2. After finding all the possible paths and computing their corresponding cumulated weights, the algorithm outputs the shortest path. This strategy is used in the Sphere Decoder and Schnorr-Euchner decoder.

3) Best-first strategy: This method is an optimized version of the Breadth-first strategy. The algorithm in this case explores the paths corresponding to the most promising nodes based on a predefined cost function. Starting from the root node, the algorithm explores all child nodes s_n and stores them in a stack in a decreasing order of their cost functions. The child nodes of the top node in the stack are then generated and their cost functions are computed. The algorithm stores then the explored nodes in the stack such that the top node has the least cost function. The search is continued until a leaf node reaches the top of the stack.

B. Stack Decoder

The Stack decoder uses a best-first tree-search strategy. Starting from the root node s_{root}, the algorithm generates all child nodes at level n and computes their respective weights. Nodes are then stored in the stack in an increasing order of their weights such that the node in the top of the stack has the least metric. In addition to the nodes, the algorithm stores for each saved node its weight, its path and its level in the tree. Given this initial stack, the algorithm generates the child nodes of the top node in the stack, computes their weights and stores

them in the stack after removing their parent node. The stack is then reordered and the same processing is performed until a leaf node reaches the top of the stack. The path corresponding to this node represents the ML solution.

C. SB-Stack Decoder

When a QAM constellation is considered, which is the case in this work, the extended nodes at each tree level correspond to the real and imaginary parts of the constellation points. For an increasing constellation size, the traditional Stack decoder requires a high computational complexity since the algorithm extends all the nodes. In order to reduce this complexity, the SB-Stack decoder [8] has been recently proposed in literature. The latter combines the Stack decoding Best-First strategy with the search region of the Sphere Decoder (SD). The decoder searches for the closest vector inside a sphere centered at the received point implementing the stack decoding strategy. The spherical search region imposes a search interval for each detected symbol s_i at level i. Only nodes belonging to these intervals at each tree level are visited and expanded. Limiting the search space to a spherical region under the SB-Stack decoder provides a complexity gain of at least 30% over the Sphere Decoder. In this work, we propose to reduce the complexity of the stack decoder by limiting the number of visited nodes while preserving ML performance.

IV. A Novel low-complexity Zigzag Stack Decoder

The SE decoder uses a depth-first tree-search strategy. It seeks the ML solution inside a spherical region but in a different way as the SD does. Initially, the sphere radius is set to infinity. In a first turn, the decoder finds a first point known as the Babai point corresponding to the ZF-DFE solution. Then, the sphere radius is adapted to the distance from the received point and the Babai point and the decoder continues its search. Taking the Babai point as a starting point, the decoder explores the nodes inside the sphere in a zigzag way between the components of the Babai point. If a point is found inside the sphere, the radius is updated and the point is saved.

A. Zigzag-Stack Decoding

Inspired from the SE decoder, we propose in this work a novel best-first tree-search Stack decoding algorithm implementing the zigzag technique of the SE algorithm based on the orthogonal distance from the received vector to the hyperplans defined by the generating matrix of the decoding lattice. Instead of storing at each tree level $i, i = n, ..., 1$ all the constellation points (tree-nodes) as it is done in the traditional Stack decoder, we propose to store only the nodes corresponding to the projection of the received vector on the corresponding hyperplan of dimension $i-1$ as well as its neighbor nodes counting 2 zigzaging steps, i.e. generating only at most 3 nodes at each level. The limitation of the number of generated nodes is validated numerically since the SE, providing ML performance, does not perform more than 2 zigzags around the ZF-DFE point.

Starting from the root node, the algorithm generates a first component s_n at level n by performing the projection of the vector $\tilde{\mathbf{y}}$ on the n^{th} layer of the upper triangular matrix $\mathbf{R}$ such that:

$$s_n = \left[\frac{\tilde{y}_n}{R_{nn}}\right] \tag{10}$$

where $[x]$ returns the nearest integer to x. Notice that at this level, the generated component corresponds to the n^{th} component of the ZF-DFE point. Afterwards, the algorithm zigzags around this component and generates its two neighbors s_n-1, s_n+1. The nodes s_n, s_n-1, s_n+1 are then stored in the stack together with their paths, depths in the tree, orthogonal distances and weights computed according to (8). They are then ordered in an increasing order of their weights. The top node in the stack is then selected in order to generate its child nodes at level $n-1$. To do so, the algorithm uses the orthogonal distance of $\tilde{\mathbf{y}}$ on the $(n-1)^{th}$ hyperplan generated by the $(n-1)^{th}$ layer of the matrix $\mathbf{R}$ to compute the component s_{n-1} such that:

$$s_{n-1} = \left[\frac{\tilde{y}_{n-1} - s_n R_{n-1,n}}{R_{n-1,n-1}}\right] \tag{11}$$

Notice that in order to compute this component, the value of s_n corresponds to the parent node, which is the current top node in the stack. And given that the nodes are ordered according to their metrics, the selected top node may not correspond to the n^{th} component of the ZF-DFE point, therefore, the generated symbol s_{n-1} does not necessarily correspond to the $(n-1)^{th}$ component of the ZF-DFE point. The algorithm zigzags then around this first node to generate its neighbor nodes $s_{n-1} - 1, s_{n-1} + 1$. The top node in the stack is removed, the stack is reordered and the algorithm continues in a similar way. For nodes at levels $i = n-2, ..., 1$, the algorithm generates the component s_i as well as its neighbor nodes $s_i - 1, s_i + 1$ such that:

$$s_i = \left[\frac{1}{R_{ii}}\left(\tilde{y}_i - \sum_{j=i+1}^{n} R_{i,j} s_j\right)\right] \tag{12}$$

The algorithm stops when a leaf node reaches the top of the stack.

B. Decoding Finite Constellation

Our algorithm Zigzag Stack is applicable to decode infinite lattices in $\mathbb{Z}^n$. We provide in the following a modification of the algorithm to adapt it to finite constellations, particularly to QAM modulations considered in this work. When information symbols $\mathbf{s}_c$ are carved from a finite alphabet, their real and imaginary parts, which correspond to the detected symbols over the tree, belong to a finite interval $I = [c_{min}, c_{max}]$. For example, in the case of $q-$QAM modulations, symbols $\mathbf{s}_c$ belong to $I_c = \left[\pm 1, \pm 3, ..., \pm(\sqrt{q}-1)\right]$. After performing a change of variables, the nodes in the search-tree that correspond to the used constellation belong to the finite set $I = \left[0, 1, 2, ..., \sqrt{q}-1\right]$ where $c_{min} = 0, c_{max} = \sqrt{q}-1$. In order to guarantee that the estimated vector belongs to the considered constellation, the bound constraint should be

satisfied each time a node and its neighbors are generated. To take this constraint into consideration, we introduce the function in(x) such that:

$$in\,(x) = \begin{cases} [x]\,, & \text{if } [x] \in [c_{min}, c_{max}] \\ c_{min}, & \text{if } [x] < c_{min} \\ c_{max}, & \text{if } [x] > c_{max} \end{cases} \quad (13)$$

C. Parameterized Zigzag Stack

The original Stack decoder offers ML performance with high decoding complexity. However, in its parameterized version, it brings a flexibility in obtaining several performance/complexity tradeoffs. This is possible thanks to a bias parameter $b \in \mathbb{R}^+$ which is deduced from the weight function, such that the novel parameterized weight for a node s_i at level i in the tree is given by [7]:

$$w_i(s_i) = \mid \tilde{\mathbf{y}}_i - \sum_{j=i}^{n} R_{ij} s_j \mid^2 - b.i \quad (14)$$

For a bias $b = 0$, the algorithm returns ML solution. Otherwise, the algorithm favorizes the nodes the most advanced in the tree having large values of i then the smallest weights. Therefore, the larger the value of b, the faster the decoding process is and the lower the complexity is. Nevertheless, the returned solution is no longer ML and depends on the choice of the bias parameter. Since our algorithm is based on Stack decoding, we propose also in this work a parameterized version of the Zigzag Stack allowing to have a wide range of performance/complexity tradeoffs just by varying the bias parameter in the weight function.

D. Zigzag Stack for Soft-Output Decoding

In addition to hard detection, our proposed Zigzag Stack is applicable to generate soft outputs. Soft-output decoding in MIMO systems has been studied before in literatre. Soft information is delivered using A Posteriori Probability techniques in the form of Log-Likelihood Ratio (LLR) values. The main existing methods to generate these values include two modified versions of the Sphere Decoder to generate a list of limited size known respectively as the List Sphere Decoder (LSD) and the Shifted Spherical List Decoder proposed in [10], [11]. For the Zigzag Stack decoder, the algorithm generates the ML solution which corresponds to the first leaf node reaching the top of the stack while browsing the tree-search. In order to adapt our Zigzag Stack to soft-output detection, in addition to the existing stack, a second fixed-size stack is used. The Zigzag Stack runs in a normal way. When a leaf node reaches the top of the original stack, the algorithm returns the ML solution and continues its processing. Each time a leaf node reaches the top of the stack, this node is stored in the secondary stack. The obtained *near*-ML solutions are then used, together with the ML solution to generate the LLR values for soft-output detection.

V. Simulation Results

We provide in this section numerical results obtained through Monte-Carlo simulations and evaluating the performance and complexity of our proposed Zigzag Stack decoder and its comparison to existing decoders by averaging over 10^5 channel realizations. Through simulations, we validated that the Zigzag Stack provides ML performance as the SD, the Stack decoder and the SB-Stack. Then, we present numerical results related to the computational complexity of these decoders where the total number of multiplications for the predecoding and the search phase is counted.

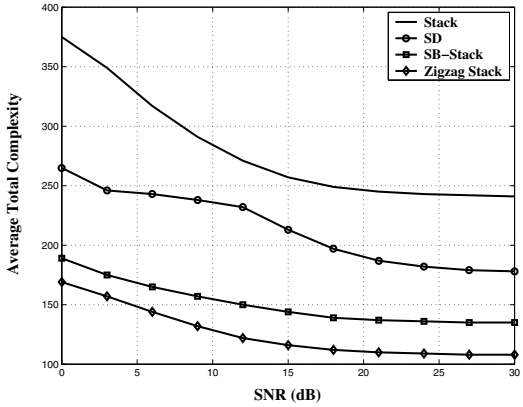

Fig. 1. Total Complexity for 2×2 MIMO system using $16-$QAM.

We consider in our simulations $16-$QAM modulation and study the 2×2 and 4×4 MIMO schemes. Our numerical results depicted in Fig.1 and Fig.2 show the considerable complexity gain offered by our decoder over the existing ones. The Zigzag Stack provides an average complexity gain of 46% over the SD for $n_t = n_r = 2$ and at least 50% when $n_t = n_r = 4$.

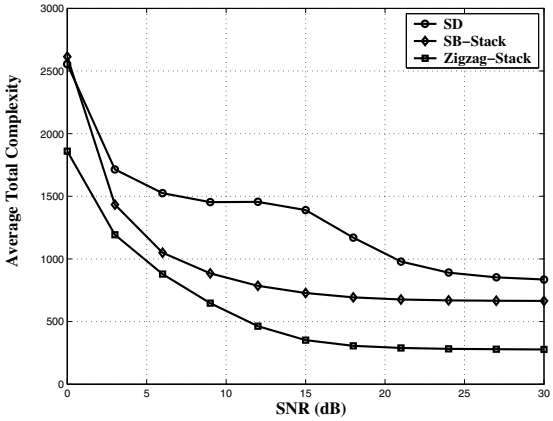

Fig. 2. Total Complexity for 4×4 MIMO system using $16-$QAM.

In addition, we provide numerical results evaluating the Symbol Error Rate and the total computational complexity for different values of the bias parameter b. As illustrated in Fig.3 and Fig.4, small values of b allow to obtain near-ML performance which is reached for $b = 0$. When the bias

increases, the complexity is reduced at the cost of having worse SER performance. For a bias $b \geq 10$, the Zigzag Stack returns ZF-DFE performance with a very reduced complexity.

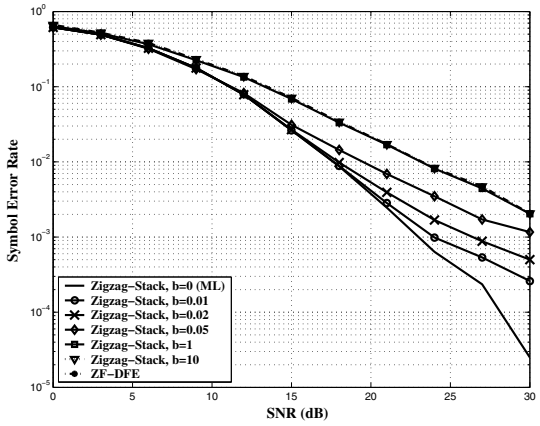

Fig. 3. Symbol Error Rate 2×2 MIMO system using $16-$QAM.

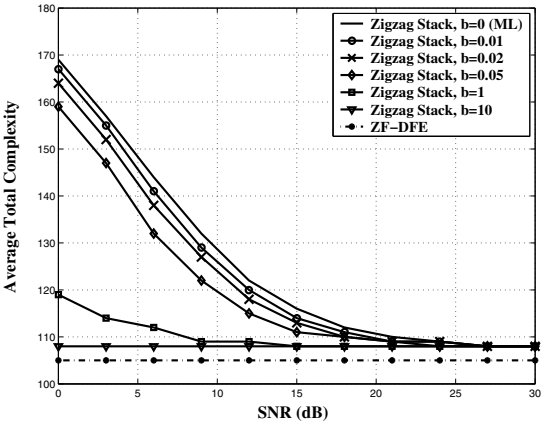

Fig. 4. Total complexity for different bias values for the Zigzag Stack decoder in a 2×2 MIMO system using $16-$QAM.

VI. Conclusion

In this work, a novel low-complexity MIMO decoder based on sequential decoding is proposed. The algorithm, termed Zigzag Stack combines the best-first tree-search strategy of the original Stack decoder with the enumeration technique of the SE decoder. This combination allows to smartlly cross the tree while visiting a reduced number of nodes. The decoder achieves ML performance with a reduced complexity offering an average complexity gain of at least 46% over the traditionally used SD. In addition, a parameterized version of the Zigzag Stack is proposed. By varying the bias parameter, it is possible to obtain a spectrum of performance/complexity tradeoffs ranging from ML to ZF-DFE.

References

[1] O. Damen, A. Chkeif, and J. CBelfiore. Lattice code decoder for space-time codes. *IEEE Communications Letters*, 4(5):161–163, May 2000.

[2] E. Viterbo and J. Boutros. A universal lattice code decoder for fading channels. *IEEE Transactions on Information Theory*, 45(5):1639–1642, July 1999.

[3] C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. In *Math. Programming*, pages 181–191, 1993.

[4] G. Caire K.R. Kumar and A.L. Moustakas. Asymptotic performance of linear receivers in mimo fading channels. *IEEE Transactions on Information Theory*, 55(10):4398–4418, October 2009.

[5] F. Jelinek. Fast sequential decoding algorithm using a stack. *IBM J. Res. Dev.*, 13(6):675–685, November 1969.

[6] R. Fano. A heuristic discussion of probabilistic decoding. *IEEE Transactions on Information Theory*, 9(2):64–74, April 1963.

[7] A.R. Murugan, H. El-Gamal, M.-O. Damen, and G. Caire. A unified framework for tree search decoding: Rediscovering the sequential decoder. *IEEE Transactions on Information Theory*, 52(3):933–953, March 2006.

[8] G.R. Ben-Othman, R. Ouertani, and A. Salah. The spherical bound stack decoder. In *Proceedings of International Conference on Wireless and Mobile Computing*, pages 322–327, October 2008.

[9] G. Rekaya and J.-C Belfiore. On the complexity of ml lattice decoders for decoding linear full rate space-time codes. In *IEEE International Symposium on Information Theory*, pages 206–206, June 2003.

[10] J. Boutros, N. Gresset, L. Brunel, and M. Fossorier. Soft-input soft-output lattice sphere decoder for linear channels. In *Global Telecommunications Conferences*, volume 3, pages 1583–1587, December 2003.

[11] B.M. Hochwald and S. Ten Brink. Achieving near-capacity on a multiple-antenna channel. *IEEE Transactions on Communications*, 51(3):389–399, March 2003.