A Unified Framework for Tree Search Decoding: Rediscovering the Sequential Decoder

Arul D. Murugan, Hesham El Gamal, *Senior Member, IEEE*, Mohamed Oussama Damen, *Senior Member, IEEE*, and Giuseppe Caire, *Fellow, IEEE*

***Abstract*—We consider receiver design for coded transmission over linear Gaussian channels. We restrict ourselves to the class of lattice codes and formulate the joint detection and decoding problem as a closest lattice point search (CLPS). Here, a tree search framework for solving the CLPS is adopted. In our framework, the CLPS algorithm is decomposed into the preprocessing and tree search stages. The role of the preprocessing stage is to expose the tree structure in a form *matched* to the search stage. We argue that the forward and feedback (matrix) filters of the minimum mean-square error decision feedback equalizer (MMSE-DFE) are instrumental for solving the joint detection and decoding problem in a single search stage. It is further shown that MMSE-DFE filtering allows for solving underdetermined linear systems and using lattice reduction methods to diminish complexity, at the expense of a marginal performance loss. For the search stage, we present a generic method, based on the branch and bound (BB) algorithm, and show that it encompasses all existing sphere decoders as special cases. The proposed generic algorithm further allows for an interesting classification of tree search decoders, sheds more light on the structural properties of all known sphere decoders, and inspires the design of more efficient decoders. In particular, an efficient decoding algorithm that resembles the well-known Fano sequential decoder is identified. The excellent performance–complexity tradeoff achieved by the proposed MMSE-DFE Fano decoder is established via simulation results and analytical arguments in several multiple-input multiple-output (MIMO) and intersymbol interference (ISI) scenarios.**

***Index Terms*—Closest lattice point search (CLPS), Fano decoder, lattice codes, sequential decoding, sphere decoding, tree search.**

I. Introduction

RECENT years have witnessed a growing interest in the closest lattice point search (CLPS) problem. This interest was increased by the connection between CLPS and maximum-likelihood (ML) decoding in multiple-input multiple-output (MIMO) channels [1]. MIMO channels offer significant advantages in terms of increased throughput and reliability at the price of a more challenging decoding task for the receiver. For example, the exhaustive search implementation of ML decoding has a complexity that grows exponentially with the degrees of freedom available in the channel. This observation inspired several approaches for suboptimal decoding that offer different performance–complexity tradeoffs (e.g., [2], [3]).

Reduced complexity decoders are typically obtained by exploiting the codebook structure. The scenario considered in our work is no exception. In principle, the decoders considered here exploit the underlying lattice structure of the received signal to cast the decoding problem as a CLPS. Some variants of such decoders are known in the literature as *sphere decoders* (e.g., [4]–[7]). These decoders typically exploit number-theoretic ideas to efficiently span the space of allowed codewords (e.g., [8], [9]). The complexity of such decoders was shown, via simulation and numerical analysis, to be significantly smaller than the exhaustive ML decoder in many scenarios of practical interest (e.g., [4], [5]). The complexity of the state of the art sphere decoder, however, remains prohibitive for problems characterized by a large dimensionality [10]. This observation is one of the main motivations for our work.

The overriding goal of our work is to establish a general framework for the design and analysis of tree search algorithms for joint detection[1] and decoding. Toward this goal, we first divide the decoding task into two interrelated stages; namely, 1) preprocessing and 2) tree search. The preprocessing stage is primarily concerned with exposing the underlying tree structure from the noisy received signal. Here, we discuss the integral roles of minimum mean-square error decision feedback equalizer (MMSE-DFE) filters, lattice reduction techniques, and relaxing the boundary control (i.e., lattice decoding) in tree search decoding. We then proceed to the search stage where a general framework based on the branch and bound (BB) algorithm is presented. This framework establishes, rigorously, the equivalence in terms of performance and complexity between different sphere and sequential decoders. We further use the proposed framework to classify the different search algorithms and identify their advantages/disadvantages. The MMSE-DFE Fano decoder emerges as a special case of our general framework that enjoys a favorable performance–complexity tradeoff. We establish the superiority of the proposed decoder via numerical results and analytical arguments in several relevant scenarios corresponding to coded as well as uncoded

Manuscript received May 13, 2005; revised November 8, 2005. The work of A. D. Murugan and H. El Gamal was supported in part by the National Science Foundation under CAREER Grant 0346887 and by a gift from Texas Instruments. The material in this paper was presented in part at the IEEE Workshop on Signal Processing Advances in Wireless Communication, New York, NY, June 2005.

A. D. Murugan and H. El Gamal are with the Electrical and Computer Engineering Department, the Ohio State University, Columbus, OH 43210-1272 USA (e-mail: palaniva@ece.osu.edu; helgamal@ece.osu.edu).

M. O. Damen is with the Electrical and Computer Engineering Department, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: modamen@ece.uwaterloo.ca).

G. Caire is with the Electrical Engineering Department, University of Southern California, Los Angeles, CA 90080 USA (e-mail: caire@usc.edu).

Communicated by M. P. Fossorier, Associate Editor for Coding Techniques.

Digital Object Identifier 10.1109/TIT.2005.864418

[1]In the sequel, the term detection refers to receiver processing that assumes uncoded transmission. Joint detection and decoding refers to receivers that handle the linear channel and the channel code jointly (possibly in a suboptimal way).

transmission over MIMO and intersymbol-interference (ISI) channels. More specifically, in our simulation experiments, we apply the tree search decoding framework to uncoded vertical Bell Labs layered space–time architecture (V-BLAST) [11], linear dispersion space–time codes [12], algebraic space–time codes [13]–[15], and trellis codes over ISI channels [16]. In all these cases, our results show that the MMSE-DFE Fano decoder achieves near-ML performance with a much smaller complexity than the state-of-the-art decoders. Note that another classification of tree search algorithms, based on a different definition of complexity appeared in [17].

The rest of the paper is organized as follows. Section II introduces our system model and notation. In Section III, we consider the design of the preprocessing stage and discuss the interplay between this stage and the tree search stage. In Section IV, we present a general framework for designing tree search decoders based on the BB algorithm. In Section V, we establish the superior performance–complexity tradeoff achieved by the proposed MMSE-DFE Fano decoder, using analytical arguments and numerical results, in several interesting scenarios. Finally, we offer some concluding remarks in Section VI.

A brief comment about notation is now in order. Throughout the sequel, vectors are denoted by bold lowercase characters (e.g., $\boldsymbol{x}$), and matrices are denoted by bold uppercase characters (e.g., $\boldsymbol{H}$). $\mathbb{Z}, \mathbb{R}, \mathbb{C}$ refer to the ring of integers, field of real numbers, and field of complex numbers, respectively.

II. System Model

We consider the transmission of lattice codes over linear channels with additive white Gaussian noise (AWGN). The importance of this problem stems from the fact that several very relevant applications arising in digital communications fall in this class, as it will be illustrated by some examples at the end of this section. Let $\Lambda \subset \mathbb{R}^m$ be an m-dimensional lattice, i.e., the set of points

$$\Lambda = \{\boldsymbol{\lambda} = \boldsymbol{G}\boldsymbol{x} : \boldsymbol{x} \in \mathbb{Z}^m\} \tag{1}$$

where $\boldsymbol{G} \in \mathbb{R}^{m\times m}$ is the lattice generator matrix. Let $\boldsymbol{v} \in \mathbb{R}^m$ be a vector and $\mathcal{R}$ a measurable region in $\mathbb{R}^m$. A lattice code $\mathcal{C}(\Lambda, \boldsymbol{v}, \mathcal{R})$ is defined [18]–[20] as the set of points of the lattice translate $\Lambda + \boldsymbol{v}$ inside the *shaping region* $\mathcal{R}$, i.e.,

$$\mathcal{C}(\Lambda, \boldsymbol{v}, \mathcal{R}) = \{\Lambda + \boldsymbol{v}\} \cap \mathcal{R}. \tag{2}$$

Typically, the translate vector $\boldsymbol{v}$ is used to maximize the number of lattice points inside $\mathcal{R}$ and/or randomize the distribution of the codebook over $\mathcal{R}$ [18], [21]. Without loss of generality, we can also see $\mathcal{C}(\Lambda, \boldsymbol{v}, \mathcal{R})$ as the set of points $\boldsymbol{c} + \boldsymbol{v}$, such that the *codewords* $\boldsymbol{c}$ are given by

$$\boldsymbol{c} = \boldsymbol{G}\boldsymbol{x}, \qquad \text{for } \boldsymbol{x} \in \mathcal{U} \tag{3}$$

where $\mathcal{U} \subset \mathbb{Z}^m$ is the code *information set*.

The linear additive noise channel is described, in general, by the input–output relation

$$\boldsymbol{r} = \boldsymbol{H}(\boldsymbol{c} + \boldsymbol{v}) + \boldsymbol{z} \tag{4}$$

where $\boldsymbol{r} \in \mathbb{R}^n$ denotes the received signal vector, $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is the AWGN vector, and $\boldsymbol{H} \in \mathbb{R}^{n\times m}$ is a matrix that defines the channel linear mapping between the input and the output.

Consider the following communication problem: a vector of information symbols $\boldsymbol{x}$ is generated with uniform probability over $\mathcal{U}$, the corresponding codeword $\boldsymbol{c} = \boldsymbol{G}\boldsymbol{x}$ is produced by the encoder, and the signal $\boldsymbol{c} + \boldsymbol{v}$ is transmitted over the channel (4). Assuming $\boldsymbol{H}$ and $\boldsymbol{v}$ known to the receiver, the ML decoding rule is given by

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in\mathcal{U}} |\boldsymbol{r} - \boldsymbol{H}\boldsymbol{v} - \boldsymbol{H}\boldsymbol{G}\boldsymbol{x}|^2. \tag{5}$$

The constraint $\mathcal{U} \subset \mathbb{Z}^m$ implies that the optimization problem in (5) can be viewed as a *constrained* version of the CLPS with lattice generator matrix given by $\boldsymbol{H}\boldsymbol{G}$ and constraint set $\mathcal{U}$.

A few remarkable examples of the above framework are as follows.

1. *MIMO flat-fading channels:* One of the simplest and most widely studied examples is a MIMO V-BLAST[2] system with squared quadrature amplitude modulation (QAM) constellations, M transmit, and N receive antennas, operating over a flat Rayleigh-fading channel. The baseband complex received signal[3] in this case can be expressed as

$$\boldsymbol{r}^c = \sqrt{\frac{\rho}{M}}\boldsymbol{H}^c\boldsymbol{c}^c + \boldsymbol{z}^c \tag{6}$$

where the complex channel matrix $\boldsymbol{H}^c \in \mathbb{C}^{N\times M}$ is composed of independent and identically distributed (i.i.d.) elements $h^c_{i,j} \sim \mathcal{N}_{\mathcal{C}}(0,1)$, the input complex signal $\boldsymbol{c}^c$ has components c^c_i chosen from a unit-energy Q^2-QAM constellation, the noise has i.i.d. components $z^c_i \sim \mathcal{N}_{\mathcal{C}}(0,1)$, and ρ denotes the signal-to-noise ratio (SNR) observed at any receive antenna. The system model in (6) can be expressed in the form of (4) by appropriate scaling and by separating the real and imaginary parts using the vector and the matrix transformations defined by

$$\boldsymbol{u}^c \mapsto \boldsymbol{u} = [\mathrm{Re}\{\boldsymbol{u}^c\}^T, \quad \mathrm{Im}\{\boldsymbol{u}^c\}^T]^T$$
$$\boldsymbol{H}^c \mapsto \boldsymbol{H} = \begin{bmatrix} \mathrm{Re}\{\boldsymbol{H}^c\} & -\mathrm{Im}\{\boldsymbol{H}^c\} \\ \mathrm{Im}\{\boldsymbol{H}^c\} & \mathrm{Re}\{\boldsymbol{H}^c\} \end{bmatrix}.$$

The resulting real model is given by (4) where $n = 2N$, $m = 2M$, and the constraint set is given by $\mathcal{U} = \mathbb{Z}_Q^m$, with $\mathbb{Z}_Q = \{0, \ldots, Q-1\}$ denoting the set of integers residues modulo Q.

In the case of V-BLAST, the lattice code generator matrix $\boldsymbol{G} = \kappa\boldsymbol{I}$, where κ is a normalizing constant, function of Q, that makes the (complex) transmitted signal of unit energy per symbol. This formulation extends naturally to MIMO channels with more general lattice-coded inputs [20]. In general, a space–time code of block length T is defined by a set of matrices $\boldsymbol{C}^c = [\boldsymbol{c}^c_1, \ldots, \boldsymbol{c}^c_T]$ in $\mathbb{C}^{M\times T}$.

[2]In the sequel, V-BLAST refers to the transmission of independent uncoded data streams from the different transmit antennas.

[3]We use the superscript^c to denote complex variables.

The columns of the codeword $\boldsymbol{C}^c$ are transmitted in parallel on the M transmit antennas in T channel uses. The received signal is given by the sequence of vectors

$$\boldsymbol{r}_t^c = \sqrt{\frac{\rho}{M}}\boldsymbol{H}^c\boldsymbol{c}_t^c + \boldsymbol{z}_t^c, \qquad t = 1, \ldots, T. \tag{7}$$

Lattice space–time codes are obtained by taking a lattice code $\mathcal{C}(\Lambda, \boldsymbol{v}, \mathcal{R})$ in $\mathbb{R}^{2MT}$, and mapping each codeword $\boldsymbol{c}$ into a complex matrix $\boldsymbol{C}^c$ according to some *linear* one-to-one mapping $\mathbb{R}^{2MT} \rightarrow \mathbb{C}^{M\times T}$. It is easy to see that a lattice-coded MIMO system can be again expressed by (4) where the channel matrix $\bar{\boldsymbol{H}}$ is proportional (through an appropriate scaling factor) to the block-diagonal matrix

$$\boldsymbol{I}_T \otimes \begin{bmatrix} \mathrm{Re}\{\boldsymbol{H}^c\} & -\mathrm{Im}\{\boldsymbol{H}^c\} \\ \mathrm{Im}\{\boldsymbol{H}^c\} & \mathrm{Re}\{\boldsymbol{H}^c\} \end{bmatrix} \tag{8}$$

where $\otimes$ denotes the Kronecker product. In this case, we have $n = 2NT$ and $m = 2MT$. Interestingly, for a wide class of *linear dispersion* (LD) codes [12], [22]–[25], the information set $\mathcal{U}$ is still given by $\mathbb{Z}_Q^m$, as in the simple V-BLAST case, although the generator matrix $\boldsymbol{G}$ is generally not proportional to $\boldsymbol{I}$. For other classes of lattice codes [20], with more involved shaping regions $\mathcal{R}$, the information set $\mathcal{U}$ does not take on the simple form of an "hypercube." For example, consider Λ obtained by construction A [26], i.e., $\Lambda = C + Q\mathbb{Z}^m$, where $C \subseteq \mathbb{Z}_Q^m$ is a linear code over $\mathbb{Z}_Q$ with generator matrix in systematic form $[\boldsymbol{I}, \boldsymbol{P}^T]^T$. A generator matrix of Λ is given by [26]

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{P} & Q\boldsymbol{I} \end{bmatrix}. \tag{9}$$

Typically, the shaping region $\mathcal{R}$ of the lattice code $\mathcal{C}(\Lambda, \boldsymbol{v}, \mathcal{R})$ can be an m-dimensional sphere, the fundamental Voronoi region of a sublattice $\Lambda' \subset \Lambda$, or the m-dimensional hypercube. In these cases, incorporating the information set constraint in the search for the ML codeword might be difficult.

2. *ISI Channels*: For simplicity, we consider a baseband *real* single-input single-output (SISO) ISI channel with the input and output sequences related by

$$r_i = \sum_{\ell=0}^{L} h_\ell c_{i-\ell} + z_i$$

where $(h_0, \ldots, h_L)$ denotes the discrete-time channel impulse response, assumed of finite length $L+1$. The extension to the complex baseband model is immediate. Assuming that the transmitted signal is padded by L zeroes, the channel can be written in the form (4) where $\boldsymbol{c} = [c_1, \ldots, c_m]^T$ is the transmitted sequence, $\boldsymbol{r} \in \mathbb{R}^{m+L}$ is the received sequence, and the channel matrix $\boldsymbol{H} \in \mathbb{R}^{(m+L)\times m}$ takes on the banded Toeplitz form

$$\boldsymbol{H} = \begin{bmatrix} h_0 & & & \\ h_1 & h_0 & & \\ \vdots & \ddots & \ddots & \\ h_L & \ddots & \ddots & h_0 \\ & h_L & \ddots & h_1 \\ & & \ddots & \vdots \\ & & & h_L \end{bmatrix}.$$

A wide family of trellis codes obtained as coset-codes [18], [19], including binary linear codes, can be formulated as lattice codes where Λ is a Construction A lattice and the shaping region $\mathcal{R}$ is chosen appropriately. In particular, coded modulation schemes based on the Q-PAM constellation obtained by mapping group codes over $\mathbb{Z}_Q$ onto the Q-PAM constellation can be seen as lattice codes with hypercubic shaping $\mathcal{R}$. The important case of binary convolutional codes falls in this class for $Q = 2$.

III. The Preprocessing Stage

In our framework, we divide the CLPS into two stages; namely, 1) preprocessing and 2) tree search. The complexity and performance of CLPS algorithms depend critically on the efficiency of the preprocessing stage. Loosely, the goal of preprocessing is to transform the original constrained CLPS problem, described by the lattice generator matrix $\boldsymbol{HG}$ and by the constraint set $\mathcal{U}$, into a form which is *friendly* to the search algorithm used in the subsequent stage. In the following, we discuss the different tasks performed in the preprocessing stage. In general, a *friendly* tree structure can be exposed through three steps: left preprocessing, right preprocessing, and forming the tree.

Some options for these three steps are illustrated in the following subsections. However, before entering the algorithmic details, it is worthwhile to point out some general considerations. The classical sphere decoding approach to the solution of the original constrained CLPS problem (5) consists of applying QR decomposition on the combined channel and code matrix, i.e., letting $\boldsymbol{HG} = \boldsymbol{QR}$ where $\boldsymbol{Q} \in \mathbb{R}^{n\times m}$ has orthonormal columns and $\boldsymbol{R} \in \mathbb{R}^{m\times m}$ is upper triangular. Equation (5) can be written equivalently as

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in\mathcal{U}} |\boldsymbol{y}' - \boldsymbol{R}\boldsymbol{x}|^2 \tag{10}$$

where $\boldsymbol{y}' = \boldsymbol{Q}^T(\boldsymbol{r} - \boldsymbol{H}\boldsymbol{v})$. If $\mathrm{rank}(\boldsymbol{HG}) = m$, $\boldsymbol{R}$ has nonzero diagonal elements and its triangular form can be exploited to search for all the points $\boldsymbol{x} \in \mathcal{U}$ such that $\boldsymbol{R}\boldsymbol{x}$ is in a sphere of a given search radius centered in $\boldsymbol{y}'$. If the sphere is nonempty, the ML solution is guaranteed to be found inside the sphere, otherwise, the search radius is increased and the search is restarted. Different variations on this main theme have been proposed in the literature, and will be reviewed in Section IV as special cases of a general BB algorithm. Nevertheless, it is useful to point out here the two main sources of inefficiency of the above approach. 1) It does not apply to the case $\mathrm{rank}(\boldsymbol{HG}) < m$ and, even when

$\mathrm{rank}(\boldsymbol{HG}) = m$ but $\boldsymbol{HG}$ is ill-conditioned, the spread (or dynamic range) of the diagonal elements of $\boldsymbol{R}$ is large. This entails large complexity of the tree search [27]. Intuitively, when $\boldsymbol{HG}$ is ill-conditioned, the lattice generated by $\boldsymbol{HG}$ has a very skewed fundamental cell such that there are directions in which it is very difficult to distinguish the points $\{\boldsymbol{HGx} : \boldsymbol{x} \in \mathcal{U}\}$; 2) Enforcing the information set constraint $\boldsymbol{x} \in \mathcal{U}$ can be very difficult since checking the condition $\boldsymbol{x} \in \mathcal{U}$ during the search may entail a significant complexity.

Left preprocessing can be seen as an effort to tackle the first problem: it modifies the channel matrix and the noise vector such that the resulting CLPS problem is nonequivalent to ML (therefore, it is suboptimal), but it has a much better conditioned "channel" matrix. The second problem can be tackled by relaxing the constraint set $\mathcal{U}$ to the whole $\mathbb{Z}^m$, i.e., searching over the whole lattice Λ instead of only the lattice code $\mathcal{C}$ (or lattice decoding). In general, lattice decoding is another source of suboptimality. Nevertheless, once the boundary region is removed, we have the freedom of choosing the lattice basis which is more convenient for the search algorithm. This change of lattice basis is accomplished by right preprocessing. Finally, the tree structure is obtained by factorizing the resulting combined channel-lattice matrix in upper triangular form, as in classical sphere decoding. Overall, left and right preprocessing combined with lattice decoding are a way to reduce complexity at the expense of optimality. Fortunately, it turns out that an appropriate combination of these elements yields significant saving in complexity with very small degradation with respect to the ML performance. Thus, it yields a very attractive decoding solution. While the outstanding performance of appropriate preprocessing and lattice decoding can be motivated via rigorous information-theoretic arguments [20], [21], [28], here we are more concerned with the algorithmic aspects of the decoder and we shall give some heuristic motivation based on "signal-processing" arguments.

Finally, we note that the notion of complexity adopted in this work does not capture the complexity of the preprocessing stage (mostly cubic in the lattice dimension). In practice, this assumption is justified in slowly varying channels where the complexity of the preprocessing stage will be shared by many transmission frames (e.g., a wired ISI channel or a wireless channel with stationary terminals). If the number of these frames is large enough, i.e., the channel is slow enough, the preprocessing complexity can be ignored compared to the complexity of the tree-search stage which has to be independently performed in every frame. Optimizing the complexity of the preprocessing stage, however, is an important topic, especially for fast-fading channels.

A. Taming the Channel: Left Preprocessing

In the case of uncoded transmission ($\boldsymbol{G} = \boldsymbol{I}$), QR decomposition of the channel matrix $\boldsymbol{H}$ (assuming $\mathrm{rank}(\boldsymbol{H}) = m$) allows one to employ a simple recursive detection algorithm of the information symbols $\boldsymbol{x}$. Indeed, $\boldsymbol{Q}$ is the feedforward matrix of the zero-forcing decision feedback equalizer (ZF-DFE) [11]. In general, sphere decoders can be seen as ZF-DFEs with some reprocessing capability of their tentative decisions.

It is well known that ZF-DFE is outperformed by the MMSE-DFE in terms of signal-to-interference-plus-noise ratio (SINR) at the decision point, under the assumption of correct decision feedback [29]. This observation motivates the proposed approach for left preprocessing [27]. This new matrix can be obtained through the QR decomposition of the augmented channel matrix

$$\tilde{\boldsymbol{H}} \triangleq \begin{bmatrix} \boldsymbol{H} \\ \boldsymbol{I} \end{bmatrix} = \tilde{\boldsymbol{Q}}\boldsymbol{R}_1 \tag{11}$$

where $\tilde{\boldsymbol{Q}} \in \mathbb{R}^{(n+m)\times m}$ has orthonormal columns and $\boldsymbol{R}_1$ is upper triangular. Let $\boldsymbol{Q}_1$ be the upper $n \times m$ part of $\tilde{\boldsymbol{Q}}$, then $\boldsymbol{Q}_1$ and $\boldsymbol{R}_1$ are the MMSE-DFE forward and backward filters, respectively [20]. The transformed CLPS

$$\min_{\boldsymbol{x}\in\mathcal{U}} |\boldsymbol{y}' - \boldsymbol{R}_1\boldsymbol{Gx}|^2 \tag{12}$$

with $\boldsymbol{y}' = (\boldsymbol{Q}_1^T\boldsymbol{r} - \boldsymbol{R}_1\boldsymbol{v})$ is not equivalent to (5) since, in general, $\boldsymbol{Q}_1$ does not have orthonormal columns. The additive noise $\boldsymbol{w} = \boldsymbol{y}' - \boldsymbol{R}_1\boldsymbol{Gx}$ in (12) contains both a Gaussian component, given by $\boldsymbol{Q}_1^T\boldsymbol{z}$, and a non-Gaussian (signal-dependent) component, given by $(\boldsymbol{Q}_1^T\boldsymbol{H} - \boldsymbol{R}_1)(\boldsymbol{c} + \boldsymbol{v})$. Nevertheless, for lattice codes such that $\mathrm{cov}(\boldsymbol{c}+\boldsymbol{v}) = \boldsymbol{I}$, it can be shown that $\mathrm{cov}(\boldsymbol{w}) = \boldsymbol{I}$ [20]. Hence, the additive noise component $\boldsymbol{w}$ in (12) is still white, although non-Gaussian and data dependent. Therefore, the minimum distance rule (12) is expected to be only slightly suboptimal.[4] On the other hand, the augmented channel matrix $\tilde{\boldsymbol{H}}$ in (11) has always rank equal to m and it is well conditioned, since $\boldsymbol{R}_1^T\boldsymbol{R}_1 = \boldsymbol{I} + \boldsymbol{H}^T\boldsymbol{H}$. Therefore, in some sense we have *tamed* the channel at the (small) price of the non-Gaussianity of the noise. The better conditioning achieved by the MMSE-DFE preprocessing is illustrated later in Fig. 1 (b) and (c).

B. Inducing Sparsity: Right Preprocessing

In order to obtain the tree structure, one needs to put $\boldsymbol{R}_1\boldsymbol{G}$ in upper triangular form $\boldsymbol{R}$ (e.g., via QR decomposition). The sparser the matrix $\boldsymbol{R}$, the smaller the complexity of the tree search algorithm. For example, a diagonal $\boldsymbol{R}$ means that symbol-by-symbol detection is optimal, i.e., the tree search reduces to exploring a single path in the tree. Loosely, if one adopts a depth-first search strategy, then a sparse $\boldsymbol{R}$ will lead to a better *quality* of the first leaf node found by the algorithm.[5] Consequently, the algorithm finds the closest point in a shorter time [4].

While we have no rigorous method for relating the "sparsity" of $\boldsymbol{R}$ to the complexity of the tree search, inspired by decision feedback equalization in ISI channel, we define the sparsity index of the upper triangular matrix $\boldsymbol{R}$ as follows:

$$\mathsf{S}(\boldsymbol{R}) \triangleq \max_{i\in\{1,\ldots,m\}} \frac{\sum_{j=i+1}^{m} r_{i,j}^2}{r_{i,i}^2} \tag{13}$$

where $r_{i,j}$ denotes the (i,j)th element of $\boldsymbol{R}$. One can argue that the smaller $\mathsf{S}(\boldsymbol{R})$ the sparser $\boldsymbol{R}$ (e.g., $\mathsf{S}(\boldsymbol{R}) = 0$ for $\boldsymbol{R}$ diagonal).The goal of right preprocessing is to find a change of basis

[4]This argument can be made rigorous by considering certain classes of lattices of increasing dimension, Voronoi shaping and random uniformly distributed dithering common to both the transmitter and the receiver, as shown in [20], [21].

[5]More details on the different search strategies are reported in Section IV.

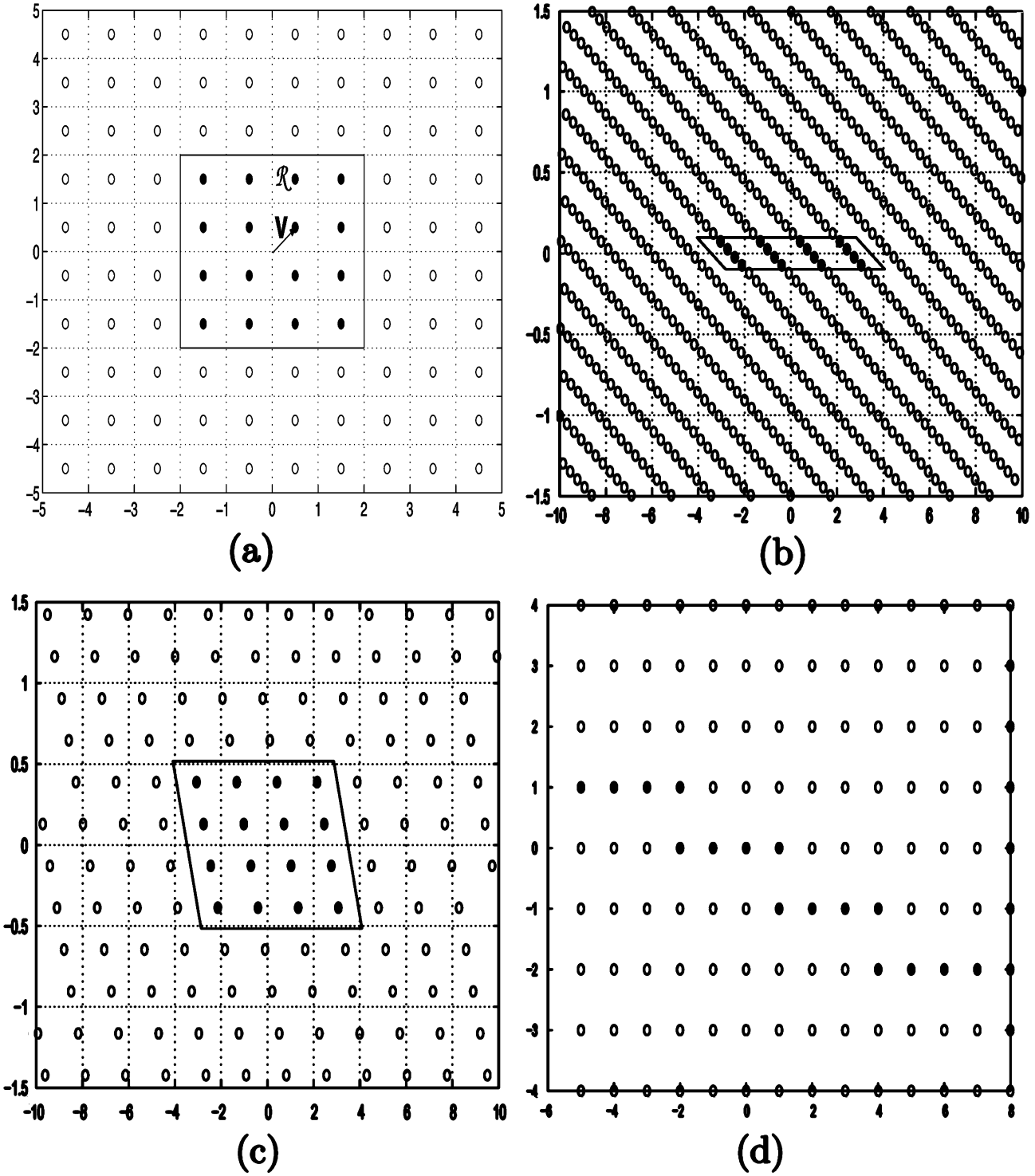

Fig. 1. The effect of left preprocessing on the lattice (parts (b) and (c)) and the right preprocessing on the information set (parts (a) and (d)). (a) The translated $\mathbb{Z}^2$ lattice and the lattice code $(\mathbb{Z}^2 + \boldsymbol{v}) \cap \mathcal{R}$. (b) The received lattice after channel distortion(multiplication by $\boldsymbol{H}$) ($\boldsymbol{R}_{\mathrm{ZF}}(\mathbb{Z}^2 + \boldsymbol{v})$ and $\boldsymbol{R}_{\mathrm{ZF}}\mathcal{R}$). (c) The received lattice after MMSE-DFE left preprocessing ($\boldsymbol{R}_{\mathrm{MMSE}}(\mathbb{Z}^2 + \boldsymbol{v})$ and $\boldsymbol{R}_{\mathrm{MMSE}}\mathcal{R}$). (d) Boundary control after right preprocessing. ($\boldsymbol{U}^{-1}(\mathbb{Z}^2 \cap (\mathcal{R} - \boldsymbol{v}))$).

of the lattice $\{\boldsymbol{R}_1\boldsymbol{G}\boldsymbol{x} : \boldsymbol{x} \in \mathbb{Z}^m\}$, such that the new lattice generator matrix $\boldsymbol{S}$ satisfies $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{R}$ with $\mathsf{S}(\boldsymbol{R})$ as small as possible. This amounts to finding a unimodular matrix $\boldsymbol{T}$ (i.e., the entries of $\boldsymbol{T}$ and $\boldsymbol{T}^{-1}$ are integers) such that $\boldsymbol{R}_1\boldsymbol{G} = \boldsymbol{Q}\boldsymbol{R}\boldsymbol{T}$ with $\boldsymbol{Q}$ unitary and $\mathsf{S}(\boldsymbol{R})$ minimized over the group of unimodular matrices. This optimization problem appears very difficult to solve; however, there exist many heuristic approaches to find unimodular matrices that give small values of $\mathsf{S}(\boldsymbol{R})$. Examples of such methods, considered here, are lattice reduction, column permutation, and a combination thereof.

Lattice reduction finds a reduced lattice basis, i.e., the columns of the reduced generator matrix $\boldsymbol{S}$ have small norms and are as orthogonal as possible.[6] The most widely used reduc-

[6]For more details on the different notions and methods of lattice reduction, the reader is referred to [30].

tion algorithm is due to Lenstra, Lenstra, and Lovász (LLL) [31] and has a polynomial complexity in the lattice dimension. An enhanced version of the LLL algorithm, namely, the deep insertion modification, was later proposed by Schnorr and Euchner [9]. LLL with deep insertion gives a reduced basis with significantly shorter vectors [30]. In practice, the complexity of the LLL with deep insertion is similar to the original one even though it is an exponential time algorithm in the worst case sense [30].

Another method for decreasing $\mathsf{S}(\boldsymbol{R})$ consists of ordering the columns of $\boldsymbol{R}_1\boldsymbol{G}$, i.e., by right-multiplication by a permutation matrix $\boldsymbol{\Sigma}$. In the sequel, we shall use the V-BLAST greedy ordering strategy proposed in [11], [32]. This algorithm finds a permutation matrix $\boldsymbol{\Sigma}$ such that $\boldsymbol{R}_1\boldsymbol{G} = \boldsymbol{Q}\boldsymbol{R}\boldsymbol{\Sigma}$ maximizes $\min_i r_{i,i}^2$. Since

$$\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{\Sigma}^{-T}\boldsymbol{G}^T\boldsymbol{R}_1^T\boldsymbol{R}_1\boldsymbol{G}\boldsymbol{\Sigma}^{-1}$$

i.e., the set $\{\sum_j r_{i,j}^2 : i = 1,\ldots,m\}$ depends only on $\boldsymbol{R}_1\boldsymbol{G}$ and not on $\boldsymbol{\Sigma}$, by maximizing the minimum $r_{i,i}^2$ this algorithm miminizes $\mathsf{S}(\boldsymbol{R})$ over the group of permutation matrices (a subgroup of the unimodular matrices).

Lattice reduction and column permutation can be combined. This yields a unimodular matrix $\boldsymbol{T} = \boldsymbol{\Sigma}\boldsymbol{T}_1$, where $\boldsymbol{T}_1$ is obtained by lattice-reducing $\boldsymbol{R}_1\boldsymbol{G}$ and $\boldsymbol{\Sigma}$ by applying the V-BLAST greedy algorithm on the resulting reduced matrix $\boldsymbol{R}_1\boldsymbol{G}\boldsymbol{T}_1^{-1}$.

As observed before, the unimodular right multiplication does not change the lattice but may significantly complicate the boundary control. In fact, we have

$$\begin{aligned}\min_{\boldsymbol{x}\in\mathcal{U}} |\boldsymbol{y}' - \boldsymbol{R}_1\boldsymbol{G}\boldsymbol{x}|^2 &= \min_{\boldsymbol{x}\in\mathcal{U}} |\boldsymbol{y}' - \boldsymbol{Q}\boldsymbol{R}\boldsymbol{T}\boldsymbol{x}|^2 \\ &= \min_{\boldsymbol{x}\in\boldsymbol{T}\mathcal{U}} |\boldsymbol{Q}^T\boldsymbol{y}' - \boldsymbol{R}\boldsymbol{x}|^2. \qquad (14)\end{aligned}$$

The new constraint set $\boldsymbol{T}\mathcal{U}$ might be even more complicated to enforce than the original information set $\mathcal{U}$ (see Fig. 1(d)). However, it is clear that although modifying the boundary control may result in a significant complexity increase for ML decoding, lattice decoding is not affected at all, since $\boldsymbol{T}\mathbb{Z}^m = \mathbb{Z}^m$.

C. Forming the Tree

The final step in preprocessing is to expose the tree structure of the problem. In this step, QR decomposition is applied on the transformed combined channel and lattice matrix $\boldsymbol{Q}_1^T\boldsymbol{H}\boldsymbol{G}\boldsymbol{T}^{-1}$, after left and right preprocessing. The upper triangular nature of $\boldsymbol{R}$ means that a tree search can now be used to solve the CLPS problem. Fig. 2 illustrates an example of such a tree.

Here, we wish to stress that our approach for exposing the tree is fundamentally different from the one traditionally used for codes over finite alphabets (e.g., linear block codes, convolutional codes, trellis coset codes in AWGN channels). Here, we operate over the field of real numbers and consider the lattice corresponding to the joint effect of encoding and channel distortion. In the conventional approach, the tree is generated from the trellis structure of the code alone, and hence, does not allow for a natural tree search that handles jointly detection (the linear channel) and decoding. In fact, joint detection and decoding is achieved at the expenses of an increase of the overall system memory (joint trellis), or by neglecting some paths in the search

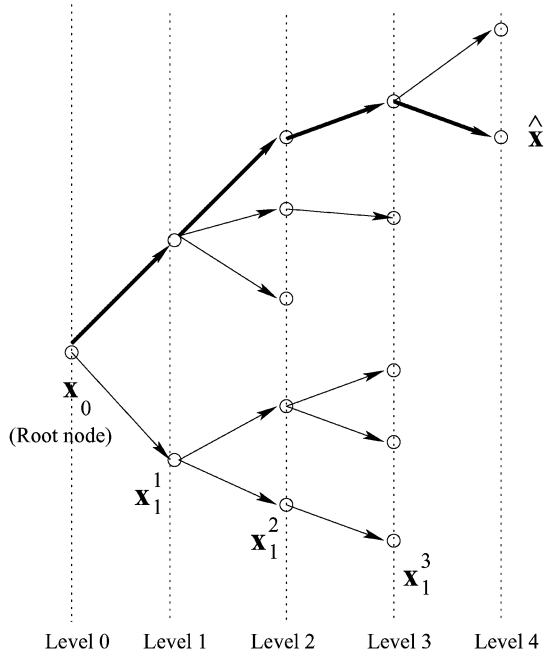

Fig. 2. Tree representation of the paths searched by sequential decoding algorithms in the case $m = 4$

(e.g., by per-survivor reduced state processing). Since operating on the full joint trellis is usually too complex, both the proposed and the conventional per-survivor (reduced state) approach are suboptimal, and the matter is to see which one achieves the best performance–complexity tradeoff.

For the sake of convenience, in the following we shall denote again by $\boldsymbol{y}$ the channel output after all transformations, i.e., the tree search is applied to the CLPS problem $\min_{\boldsymbol{x}\in\mathbb{Z}^m} |\boldsymbol{y} - \boldsymbol{R}\boldsymbol{x}|^2$ with $\boldsymbol{R}$ in upper triangular form. The components of vectors and matrices are numbered in reverse order, so that the preprocessed received signal can finally be written as

$$\begin{pmatrix} y_m \\ \vdots \\ y_1 \end{pmatrix} = \begin{pmatrix} r_{m,m} & \cdots & \cdots & r_{m,1} \\ 0 & r_{m-1,m-1} & \cdots & r_{m-1,1} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & r_{1,1} \end{pmatrix} \begin{pmatrix} x_m \\ \vdots \\ x_1 \end{pmatrix} + \begin{pmatrix} w_m \\ \vdots \\ w_1 \end{pmatrix}. \qquad (15)$$

Notice that after preprocessing the problem is always squared, of dimension m, even though the original problem has arbitrary m and n. Throughout the paper, we consider a tree rooted at a fixed dummy node x_0. The node at level k is denoted by the label $\boldsymbol{x}_1^k = (x_1, x_2, \ldots, x_k)$. Moreover, every node $\boldsymbol{x}_1^k$ is associated with the squared distance $\sum_{i=1}^k w_i(\boldsymbol{x}_1^i)$, where

$$w_i\left(\boldsymbol{x}_1^i\right) = \left| y_i - \sum_{j=1}^{i} r_{i,j}x_j \right|^2. \qquad (16)$$

The difference between the transmitted codeword $\hat{\boldsymbol{x}}$ and any valid codeword $\boldsymbol{x}$ is denoted by $\tilde{\boldsymbol{x}}$, i.e., $\tilde{\boldsymbol{x}} = \hat{\boldsymbol{x}} - \boldsymbol{x}$.

We hasten to stress that the preprocessing steps highlighted in Sections III-A–C are for a general setting. In some special

cases, some steps can be eliminated or alternative options can be used. Some of these cases are listed hereafter.

1. *Upper Triangular Code Generator Matrix*:
   In this case, after taming the channel, $\boldsymbol{H} \longrightarrow \boldsymbol{R}_1$, the new combined matrix $\boldsymbol{R}_1\boldsymbol{G}$ is also upper triangular and can be directly used to form the tree without any further preprocessing (if one decides against right preprocessing).
2. *Uncoded V-BLAST*:
   For the uncoded V-BLAST systems (i.e., $\boldsymbol{G} = \boldsymbol{I}$), applying the MMSE-DFE greedy ordering of [33], [32] may achieve better complexity of the tree search stage than applying MMSE-DFE left preprocessing, lattice reduction, and greedy ordering of the final QR decomposition. This is especially true for large dimensions, where lattice reduction is less effective [4].
3. *The Hermite Normal Form Transformation*:
   Ultimately, any hardware implementation of the decoder requires finite arithmetics. In this case, all quantities are scaled and quantized such that they take on integer values. While all the preprocessing steps in Sections III-A–C can be easily adapted to finite arithmetics, there exist other efficient transformations for integral matrices that may yield smaller complexity over the ones mentioned above. For example, one can apply the Hermite normal form (HNF) [30] directly on the scaled (and quantized) matrix $\tau\boldsymbol{H}\boldsymbol{G}$, such $\tau\boldsymbol{H}\boldsymbol{G} = \boldsymbol{R}\boldsymbol{T}$, with $\boldsymbol{T}$ unimodular and $\boldsymbol{R}$ upper triangular with the property that each diagonal element dominates the rest of the entries on the same row (i.e., $r_{i,i} > r_{i,j} \geq 0$, $i = 1, \ldots, m$, $j = i+1, \ldots, m$). Interestingly, the HNF transformation improves the sparsity index and reduces the preprocessing to a single step.

IV. The Tree Search Stage

After proper preprocessing, the second stage of the CLPS corresponds to an instance of searching for the best path in a tree. In this setting, the tree has a maximum depth m, and the goal is to find the node(s) at level m that has (have) the least squared distance $\sum_{i=1}^{m} w_i(\boldsymbol{x}_1^i)$. Visiting all leaf nodes to find the one with the least distance is either prohibitively complex (exponential in m), or not possible, as with lattice decoding. The complexity of tree search can be reduced by the BB algorithm [34], [35] which determines if an intermediate node $\boldsymbol{x}_1^k$, on extending, has any chance of yielding the desired leaf node. This decision is taken by comparing the *cost function* assigned to the node by the search algorithm, against a *bounding function*. In the following subsection, we propose a generic tree search stage, inspired by the BB algorithm, that encompasses many known algorithms for CLPS as its special cases. We further use this algorithm to classify various tree search algorithms and elucidate some of their structural properties.

A. Generic Branch and Bound Algorithm

Before describing the proposed algorithm, we first need to introduce some more notation.

- ACTIVE is an ordered list of nodes.
- $f(\boldsymbol{x}_1^k) \in \mathbb{R}$ is the *cost function* of any node $\boldsymbol{x}_1^k$ in the tree, and $\boldsymbol{t} \in \mathbb{R}^{m \times 1}$ is the bounding function.
- Any node $\boldsymbol{x}_1^k$ in the search space of the search algorithm is a *valid* node, if $f(\boldsymbol{x}_1^k) < t_k$.
- A node is *generated* by the search algorithm if the node occupies any position in ACTIVE at some instant during the search.
- "*sort*" is a rule for ordering the nodes in the list ACTIVE.
- "*gen*" is a rule defining the order for generating the child nodes of the node being extended.
- g_1 and g_2 are rules for tightening the bounding function.
- At any instant, the leaf node with the least distance generated by the search algorithm so far in the search process is stored in $\hat{\boldsymbol{x}}$.
- We define the search *complexity* of a tree search algorithm as the number of nodes generated by the algorithm.
- Two search algorithms are said to be equivalent if they generate the same set of nodes.
- A BB algorithm whose solution is guaranteed to be (one of) leaf node(s) with least distance is called an *optimal* BB algorithm. If the solution is not guaranteed to have the least distance to $\boldsymbol{y}$ among all leaf nodes, then the BB algorithm is a *heuristic* BB algorithm.

We are now ready to present our generic branch and bound (GBB) search algorithm.

GBB$(f, \boldsymbol{t}, sort, gen, g_1, g_2)$:

1. Create the empty list ACTIVE, and place the root node in . ACTIVE. Set $n_c \leftarrow 1$
2. Let $\boldsymbol{x}_1^k$ be the top node of ACTIVE.
   If $\boldsymbol{x}_1^k$ is a leaf node ($k = m$), then
   $\boldsymbol{t} \leftarrow g_1(\boldsymbol{t}, f(\boldsymbol{x}_1^m))$ and
   $$\hat{\boldsymbol{x}} \leftarrow \arg\min \left(\sum_{i=1}^{m} w_i(\boldsymbol{x}_1^i), \sum_{i=1}^{m} w_i(\hat{\boldsymbol{x}}_1^i) \right).$$
   Remove $\boldsymbol{x}_1^m$ from ACTIVE.
   Go to step 4.
   If $\boldsymbol{x}_1^k$ is not a valid node, then remove $\boldsymbol{x}_1^k$ from ACTIVE. *Go to* step 4.
   If all valid child nodes of $\boldsymbol{x}_1^k$ have already been generated, then remove $\boldsymbol{x}_1^k$ from ACTIVE. *Go to* step 4.
   Generate a valid child node $\boldsymbol{x}_1^{k+1}$ of $\boldsymbol{x}_1^k$, not generated before, according to the order *gen*, and place it in ACTIVE. Set $n_c \leftarrow n_c + 1$. Set $\boldsymbol{t} \leftarrow g_2(\boldsymbol{t}, n_c, \text{ACTIVE})$. Update $f(\boldsymbol{x}_1^k), f(\boldsymbol{x}_1^{k+1})$.
3. Sort the nodes in ACTIVE according to *sort*.
4. *If* ACTIVE is empty, then exit. *Else, Go to* step 2.

In **GBB**, g_1 allows one to tighten the bounding function when a leaf node reaches the top of ACTIVE, whereas g_2 allows for restricting the search space in heuristic BB algorithms. For example, setting

$$g_2(\boldsymbol{t}, n_{c,t}, \text{ACTIVE}) = [-\infty, -\infty, \ldots, -\infty]^T$$

will force the search algorithm to terminate when the number of nodes generated increases beyond a tolerable limit on the com-

plexity given by $n_{c,t}$. Whenever a leaf node reaches the top of ACTIVE, $\hat{\boldsymbol{x}}$ is updated if appropriate. Now, we use **GBB** to classify various tree search algorithms in three broad categories. This classification highlights the structural properties and advantages/disadvantages of the different search algorithms.

1) Breadth-First Search: **GBB** becomes a breadth-first search (BrFS) if $g_1(\boldsymbol{t}, f(\boldsymbol{x}_1^m)) = \boldsymbol{t}$, and the cost function f of any node, once determined, is never updated. Ultimately, all nodes $\boldsymbol{x}_1^k$ whose cost function along the path $\boldsymbol{x}_1^k$ does not rise above the bounding function, are generated before the algorithm terminates, unless the g_2 function removes their parent nodes from ACTIVE. Now, we can establish the equivalence between various sphere/sequential decoders and BrFS.

The first algorithm is the Pohst enumeration strategy reported in [8]. In this strategy, the bounding function $\boldsymbol{t}$ consists of equal components C_0, where C_0 is a constant chosen before the start of search,[7] and the cost function of a node $\boldsymbol{x}_1^k$ is $f(\boldsymbol{x}_1^k) = \sum_{i=1}^{k} w_i(\boldsymbol{x}_1^i)$. Therefore, all nodes $\boldsymbol{x}_1^k$ in the search space that satisfy

$$\sum_{i=1}^{k} w_i\left(\boldsymbol{x}_1^i\right) < C_0 \tag{17}$$

are generated before termination. Generating the child nodes in this strategy is simplified by the following observation. For any parent node $\boldsymbol{x}_1^k$, the condition $\sum_{i=1}^{k+1} w_i(\boldsymbol{x}_1^i) < C_0$ for the set of generated child nodes implies that the $(k+1)$th component of the generated child nodes lies in some interval $[a_0, a_1]$. The second example is the statistical pruning (SP) decoder which is equivalent to a heuristic BrFS decoder. Two variations of SP are proposed in [36], the increasing radii (IR) and elliptical pruning (EP) algorithms. The IR algorithm is a BrFS with the bounding function $\boldsymbol{t} = \{t_1, \ldots, t_m\}$, where $t_k, 1 \le k \le m$ are constants chosen before the start of the search. The cost function for any node in IR is the same as in Pohst enumeration. The EP algorithm is given by the bounding function $\boldsymbol{t} = \{1, \ldots, 1\}$, and the cost function for the node $\boldsymbol{x}_1^k$ given by

$$f(\boldsymbol{x}_1^k) = \sum_{i=1}^{k} \frac{w_i(\boldsymbol{x}_1^i)}{e_k}$$

where $e_k, 1 \le k \le m$ are constants. More generally, when $g_2(\cdot) = \boldsymbol{t}$, i.e., g_2 is not used, the resulting BrFS algorithm is equivalent to the Wozencraft sequential decoder [37] where, depending on the cost function, the decoder can be heuristic or optimal.

The M-algorithm [17] and T-algorithm [38] are also examples of heuristic BrFS. Here, however, g_2 serves an important role in restricting the search space. In both algorithms, *sort* is defined as follows. Any node in ACTIVE at level k is placed above any node at level $k+1$, and nodes in the same level are sorted in ascending order of their cost functions. In the M-algorithm, after the first node at level $k+1$ is generated (indicating that all valid nodes at level k have already been generated), g_2 sets t_k to the cost function of the Mth node at level k (where M is an initial parameter of the M-algorithm). In the T-algorithm, g_2 sets t_k to $(f(\bar{\boldsymbol{x}}_1^k) + T)$, where $\bar{\boldsymbol{x}}_1^k$ is the top node at level k in ACTIVE, and T is a parameter of the T-algorithm. After t_k is tightened in this manner, all nodes in ACTIVE at level k that satisfy $f(\boldsymbol{x}_1^k) > t_k$ are rendered invalid, and are subsequently removed from ACTIVE.

In general, BrFS algorithms are naturally suited for applications that require soft outputs, as opposed to a hard decision on the transmitted frame. The reason is that such algorithms output an ordered[8] list of candidate codewords. One can then compute the soft outputs from this list using standard techniques (e.g., [39], [40]). Here, we note that in the proposed joint detection and decoding framework, soft outputs are generally not needed. Another advantage of BrFS is that the complexity of certain decoders inspired by this strategy is robust against variations in the SNR and channel conditions. For example, the M-algorithm has a constant complexity independent of the channel conditions. This property is appealing for some applications, especially those with hard limits on the maximum, rather than average, complexity. On the other hand, decoders inspired by the BrFS strategy usually offer poor results in terms of the average complexity, especially at high SNR. One would expect a reduced average complexity if the bounding function is varied during the search to exploit the additional information gained as we go on. This observation motivates the following category of tree search algorithms.

2) Depth-First Search: **GBB** becomes a depth-first search (DFS) when the following conditions are satisfied. The sorting rule *sort* orders the nodes in ACTIVE in reverse order of generation, i.e., the last generated node occupies the top of ACTIVE, and

$$g_1\left(\boldsymbol{t}, f\left(\boldsymbol{x}_1^m\right)\right) = \left[\min\left(t_1, f\left(\boldsymbol{x}_1^m\right)\right), \ldots, \min\left(t_m, f\left(\boldsymbol{x}_1^m\right)\right)\right]^T.$$

As in BrFS, the cost function of any node, once generated, remains constant. Even among algorithms within the class of DFS algorithms, other parameters, like *gen* and g_2, can significantly alter the search behavior. To illustrate this point, we contrast in the following several sphere decoders which are equivalent to DFS strategies.

The first example of such decoders is the modified Viterbo–Boutros (VB) decoder reported in [4]. In this decoder, $g_2(\cdot) = \boldsymbol{t}$, and the cost function for any node $\boldsymbol{x}_1^k$ is

$$f(\boldsymbol{x}_1^k) = \sum_{i=1}^{k} w_i(\boldsymbol{x}_1^i).$$

For any node $\boldsymbol{x}_1^k$ and its corresponding interval $[a_0, a_1]$ for valid child nodes, the function *gen* generates the child node with a_0 as its $(k+1)$th component first. Our second example is the Schnorr–Euchner (SE) search strategy first reported in [6]. This decoder shares the same cost functions and g_2 with the modified VB decoder, but differs from it in the order of generating the child nodes. For any node $\boldsymbol{x}_1^k$ and its corresponding interval $[a_0, a_1]$ for the valid child nodes, let

$$a_m \triangleq \left\lfloor \frac{a_0 + a_1}{2} \right\rceil \quad \text{and} \quad \delta \triangleq \operatorname{sign}(w_{k+1}(\boldsymbol{x}_1^{k+1})).$$

[7]For the sake of simplicity, we assumed in the above classification that the bounding function is chosen such that at least one leaf node is found before the search terminates. If, however, no leaf node is found before the search terminates, the bounding function is relaxed and the search is started afresh.

[8]The list is ordered based on the cost function of the different candidates.

Then, the function *gen* in the SE decoder generates nodes according to the order $\{a_m, a_m+\delta, a_m-\delta, a_m+2\delta, \ldots\}$.

Due to the adaptive tightening of the bounding function, DFS algorithms have a lower average complexity than the corresponding BrFS algorithms with the same cost functions, especially at high SNR. Another advantage of the DFS approach is that it allows for greater flexibility in the performance–complexity tradeoff through carefully constructed termination strategy. For example, if we terminate the search after finding the first leaf node, i.e., $n_c = m$, then we have the MMSE-DFE Babai point decoder [27]. This decoder corresponds to the MMSE-DFE solution aided with the right preprocessing stage. It was shown in [27] that the performance of this decoder is within a fraction of a decibel from the ML decoder in systems with small dimensions. The fundamental weakness of DFS algorithms is that the sorting rule is *static* and does not exploit the information gained thus far to speed up the search process.

3) Best-First Search: **GBB** becomes a best-first search (BeFS) when the following conditions are satisfied. The nodes in ACTIVE are sorted in ascending order of their cost functions, and

$$g_1(\boldsymbol{t}, f(\boldsymbol{x}_1^m)) = [\min(t_1, f(\boldsymbol{x}_1^m)), \ldots, \min(t_m, f(\boldsymbol{x}_1^m))]^T.$$

Note that in BeFS, the search can be terminated once a leaf node reaches the top of the list, since this means that all intermediate nodes have cost functions higher than that of this leaf node. Thus, the bounding function is tightened just once in this case. The stack algorithm [41], [42] is an example of BeFS decoder obtained by setting $g_2(\cdot) = \boldsymbol{t}$, and defining the cost function of any node in ACTIVE at any instant as follows: If $\boldsymbol{x}_1^k$ is a leaf node, then $f(\boldsymbol{x}_1^k) = -\infty$. Otherwise, we let $\boldsymbol{x}_{1,g}^{k+1}$ be the best child node of $\boldsymbol{x}_1^k$ not generated yet, and define

$$f(\boldsymbol{x}_1^k) = \sum_{i=1}^{k+1} w_i(\boldsymbol{x}_{1,g}^{k+1}) - b(k+1)$$

where we refer to $b \in \mathbb{R}^+$ as the *bias*. Because of the efficiency of the sorting rule, BeFS algorithms are generally more efficient than the corresponding BrFS and DFS algorithms. This fact is formalized in the following theorems. Theorem 1 establishes the efficiency of the stack decoder with $b = 0$ among all known sphere decoders.

Theorem 1 [43]: The stack algorithm with $b = 0$ generates the least number of nodes among all optimal tree search algorithms.

The following result compares the heuristic stack algorithm, i.e., $b > 0$, with a special case of the IR algorithm [36], where the bounding function takes the form $t_k = bk + \delta$.

Theorem 2: The IR algorithm with cost function $\{\boldsymbol{t} : t_k = bk + \delta\}$ generates at least as many nodes as those generated by the stack algorithm when the same bias b is used.

Proof: Appendix C

At this point, it is worth noting that in our definition of search complexity, we count only the number of generated nodes, i.e., nodes that occupy some position in ACTIVE at some instant. In general, this is a reasonable abstraction of the actual computational complexity involved. However, in the stack algorithm, for each node generated, the cost functions of two nodes are updated instead of one; one for the generated node, and one for the parent node. Thus, the comparisons in Theorems 1 and 2 are not completely fair.

Finally, we report the following two advantages offered by the stack algorithm. First, it offers a natural solution for the problem of choosing the initial radius (or radii), which is commonly encountered in the design of sphere decoders (e.g., [4]). By setting all the components of $\boldsymbol{t}$ to ∞, it is easy to see that we are guaranteed to find the closest lattice point while generating the minimum number of nodes (among all search algorithms that guarantee finding the closest point). Second, it allows for a systematic approach for trading off performance for complexity. To illustrate this point, if we set $b = 0$, we obtain the closest point lattice decoder (i.e., best performance but highest complexity). On the other extreme, when $b \to \infty$, the stack decoder reduces to the MMSE-DFE Babai point decoder discussed in the DFS section (the number of nodes visited is always equal to m). In general, for systems with small m, one can obtain near-optimal performance with a relatively large values of b. As the number of dimensions increases, more complexity must be expended (i.e., smaller values of b) to approach the optimal performance.

B. Iterative Best-First Search

In Section IV-A, our focus was primarily devoted to complexity, defined as the number of nodes visited by the tree search algorithm. Another important aspect is the memory requirement entailed by the search. Straightforward implementation of the **GBB** algorithm requires maintaining the list ACTIVE, which can have prohibitive lengths in certain applications. This motivates the investigation of *modified* implementations of these search strategies that are more efficient in terms of storage requirements. The BrFS and DFS sphere decoders discussed in Sections IV–A1 and IV–A.2 lend themselves naturally to storage efficient implementations. Such implementations have been reported in [8], [4], [27], [6], [36], [44], [45].

In order to exploit the complexity reduction offered by BeFS strategy in practice, it is therefore important to seek modified memory-efficient implementations of such algorithms. This can be realized by storing only one node at a time, and allowing nodes to be visited more than once. The search in this case progresses in contours of increasing bounding functions [46, Fig. 4], thus allowing more and more nodes to be generated at each step, finally terminating once a leaf node is obtained. The Fano decoder [47] is the *iterative* BeFS variation of the stack algorithm. Although the stack algorithm and the Fano decoder, with the same cost functions, generate essentially the same set of nodes [46], the Fano decoder visits some nodes more than once. However, the Fano decoder requires essentially no memory, unlike the stack algorithm. Appendix A provides an algorithmic description of the Fano decoder and a brief description of the relevant parameters. Overall, the proposed decoder consists of left preprocessing (MMSE-DFE), right preprocessing (combined lattice reduction and greedy ordering), and QR decomposition, followed by the Fano (or stack) search stage for lattice, not ML, decoding.

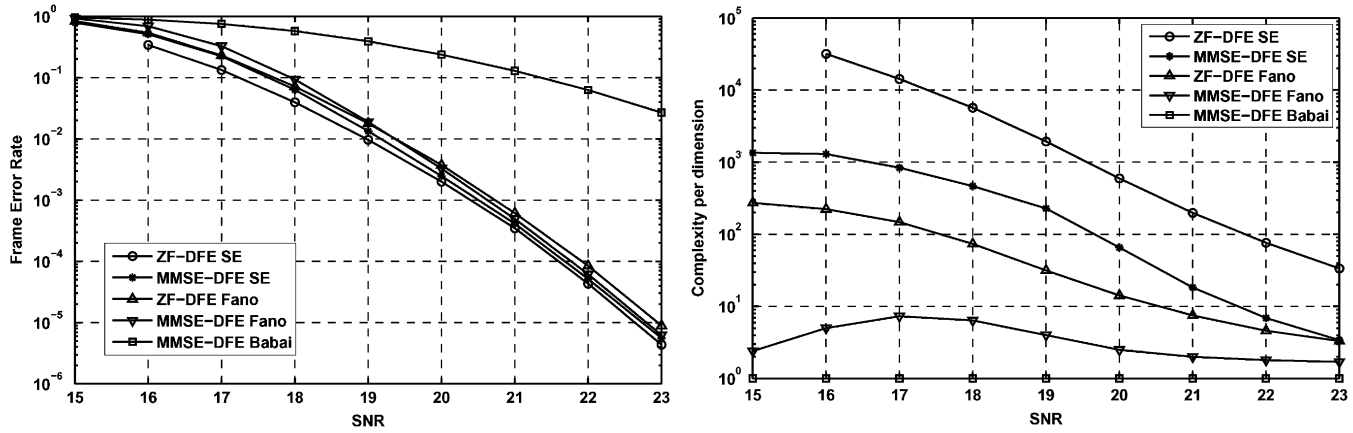

Fig. 3. Performance and complexity of SE enumeration and Fano decoder for a 20×20 16-QAM V-BLAST system.

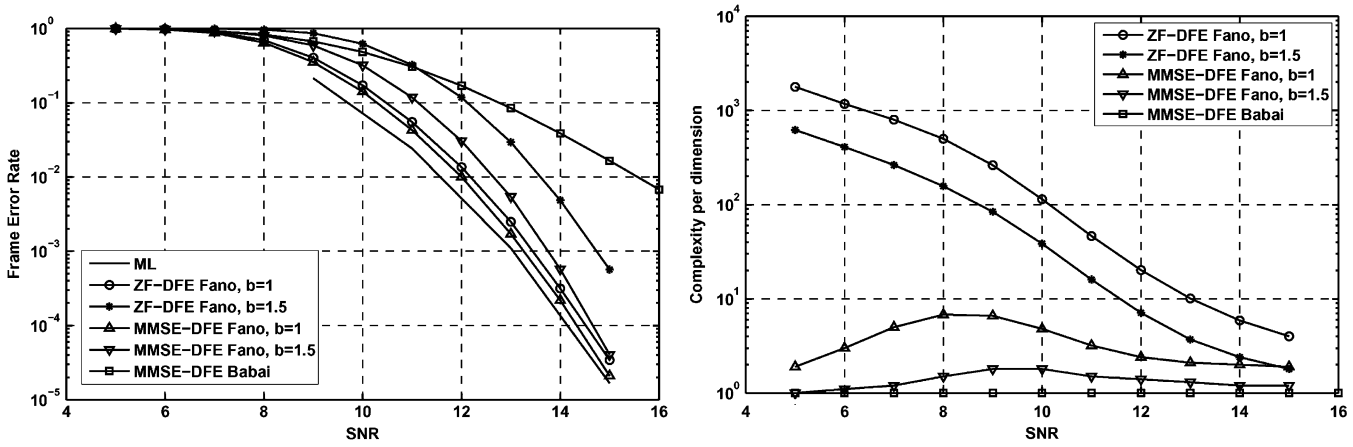

Fig. 4. Performance and complexity of Fano decoder with ZF-DFE and MMSE-DFE based preprocessing for a 30×30 4-QAM V-BLAST system.

V. Analytical and Numerical Results

To illustrate the efficiency and generality of the proposed framework, we utilize it in three distinct scenarios. First, we consider uncoded transmission over MIMO channels (i.e., V-BLAST). Here, we present analytical, as well as simulation, results that demonstrate the excellent performance–complexity tradeoff achieved by the proposed Stack and Fano decoders. Then, we proceed to coded MIMO systems and apply tree search decoding to two different classes of space–time codes. Finally, we conclude with trellis coded transmission over ISI channels.

A. The V-BLAST Configuration

Unfortunately, analytical characterization of the performance–complexity tradeoff for sequential/sphere decoders with arbitrary $\boldsymbol{HG}$ and $\mathcal{U}$ still appears intractable. To avoid this problem, we restrict ourselves in this section to uncoded transmission over flat Rayleigh MIMO channels. In our analysis, we further assume that ZF-DFE preprocessing (i.e., QR decomposition of $\boldsymbol{H}$) is used. The complexity reductions offered by the proposed preprocessing stage are demonstrated by numerical results.

Theorem 3: The Stack algorithm and the Fano decoder with any finite bias b achieve the same diversity as the ML decoder when applied to a V-BLAST configuration with $r = n - m \geq 0$.

Proof: Appendix D.

Theorem 4: In a V-BLAST system with Q^2-QAM, the average complexity per dimension of the stack algorithm for a sufficiently large bias b is linear in m when the SNR ρ grows linearly with m and $r = n - m \geq 0$.

Proof: Appendix E.

Thus, one can achieve linear complexity with the stack algorithm by allowing the SNR to increase linearly with the lattice dimension. To validate our theoretical claims, we further report numerical results in selected scenarios. In our simulations, we assume that the channel matrix is square and choose the SE enumeration as the reference sphere decoder for comparison purposes. In Fig. 3, the average complexity per lattice dimension and frame error rate of the Fano decoder with $b = 1$ and the SE sphere decoder are shown for different values of SNR in a 20×20 16-QAM V-BLAST system. Thus, for $m = 40$, the Fano decoder can offer a reduction in complexity up to a factor of 100. Moreover, the performance of the Fano decoder is seen to be only a fraction of a decibel away from that of the SE decoder, which achieves ML performance. We also see that the frame error rate curves for both the Fano decoder and the SE (ML) decoder have the same slope in the high-SNR region, as expected from our analysis. Fig. 4 compares the complexity and performance of the MMSE-DFE and ZF-DFE Fano decoders in a 30×30 4-QAM V-BLAST system (i.e., $m = 60$). In

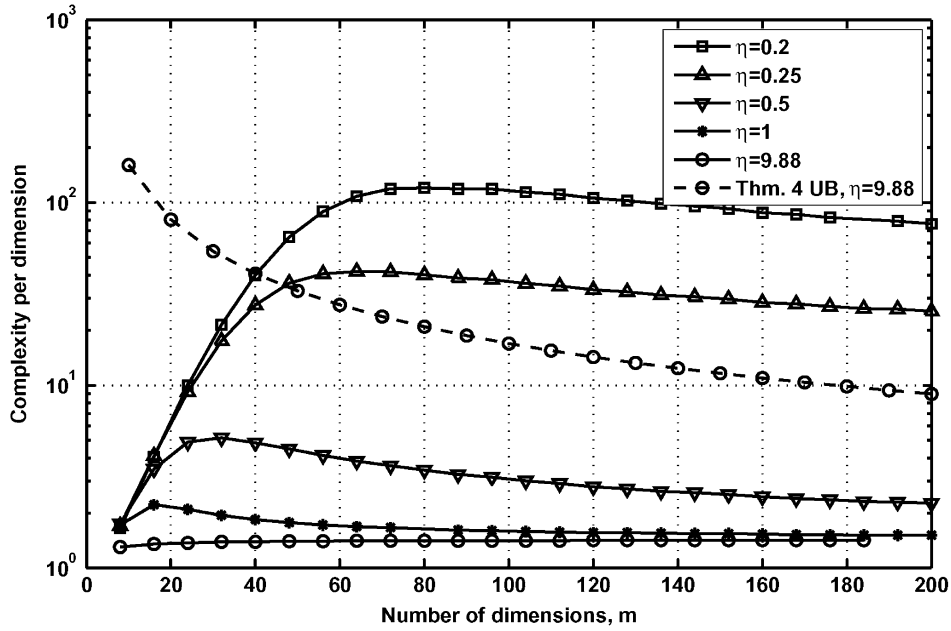

Fig. 5. Complexity per dimension of ZF-DFE Fano *vs.* m for 4-QAM V-BLAST for different values of $\eta = \frac{\rho}{m}$.

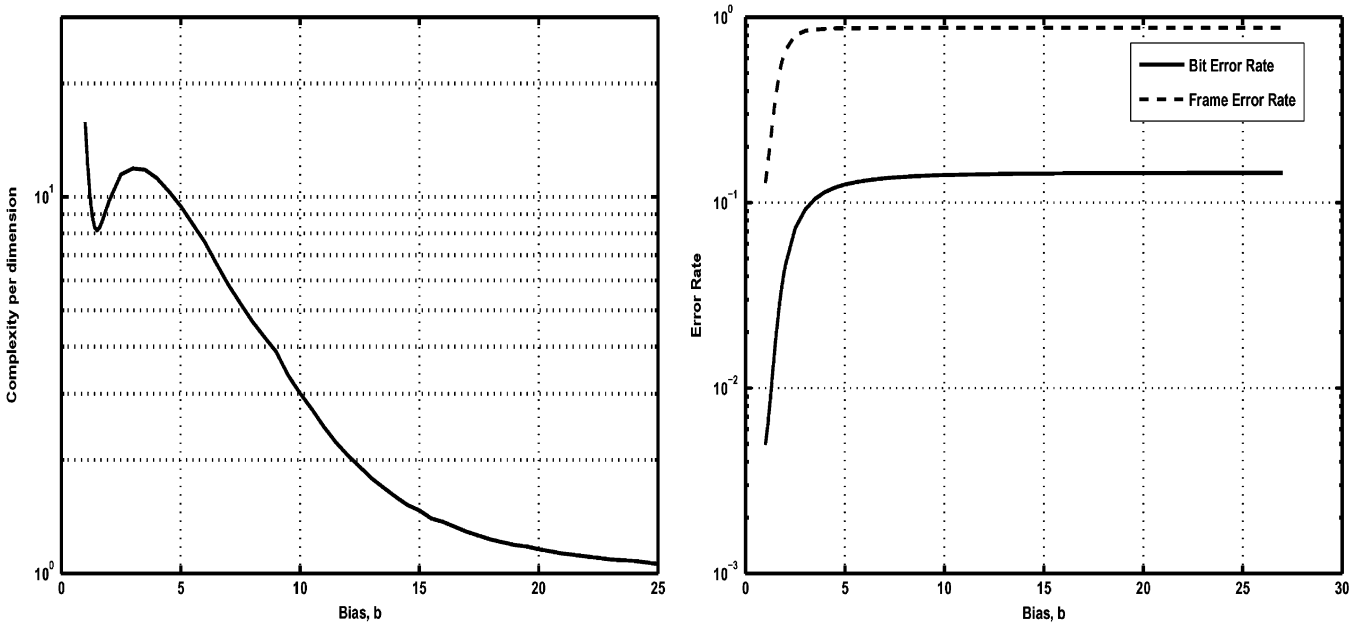

Fig. 6. Complexity and performance of Fano decoder with different bias, for a 20×20 4-QAM V-BLAST system with ZF-DFE preprocessing.

the figure, the complexity of the MMSE-DFE Fano decoder is shown to be orders of magnitude lower than the ZF-DFE Fano decoder. The figure also demonstrates the poor performance of the MMSE-DFE Babai decoder in this setup. We will come back to this decoder in Fig. 8. Fig. 5 shows the complexity per dimension of the ZF-DFE Fano decoder *vs.* number of dimensions for 4-QAM V-BLAST system, when the SNR ρ increases linearly with m. The complexity per dimension of the ZF-DFE Fano decoder for large m does not increase *w.r.t* m, when $\eta = \frac{\rho}{m}$ is held constant. For the parameters chosen $(Q = 2, b = 2)$, the upper bound on the stack decoder complexity in (79) holds true for $\eta \geq 9.87$. Fig. 5 also shows the upper bound in (80) for $\eta = 9.88$. From the figure, we see that the bound is quite loose (since the complexity of the Fano decoder is higher than that of the stack decoder, the complexity curve of the stack algorithm lies below that of the Fano decoder).

Fig. 6 reports the dependence of the complexity of the Fano decoder on the value of b. The complexity attains a local minimum for some $b^* > 1$, and for large values of b, the complexity of the ZF-DFE Fano decoder decreases as b is increased. The error rate, however, increases monotonically with b and approaches that of the ZF-DFE Babai decoder as $b \to \infty$. Fig. 7 shows the dependence of performance and complexity of the MMSE-DFE Fano decoder on the value of stepsize. As the step size is increased, the complexity of the MMSE-DFE Fano decoder decreases and its error rate increases. Based on the

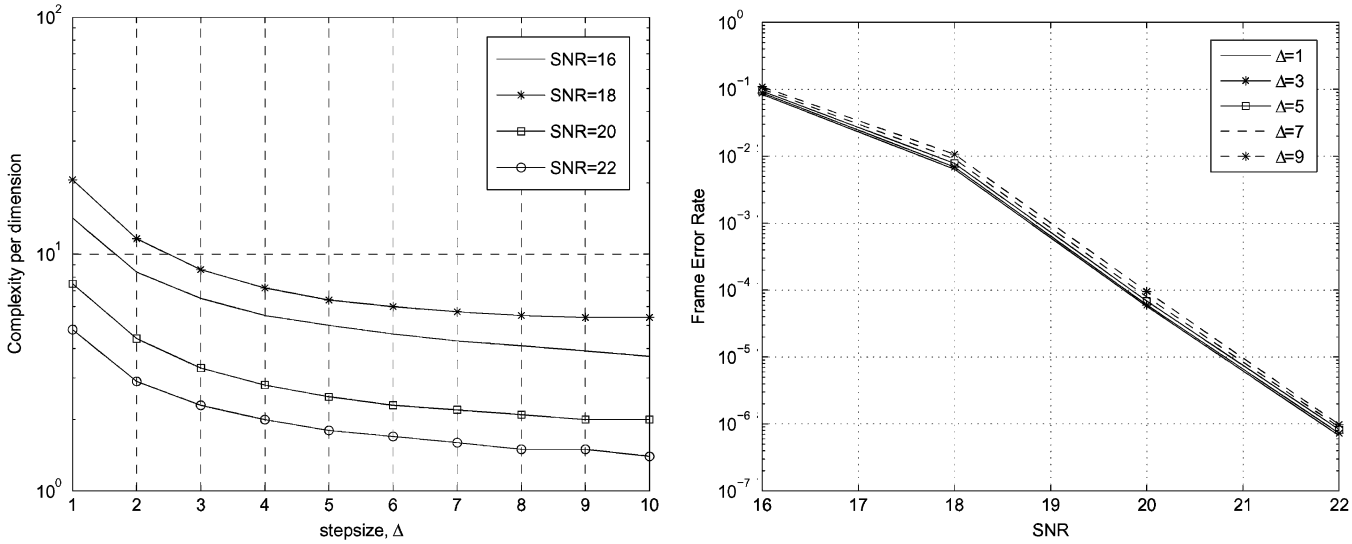

Fig. 7. Complexity and performance of the MMSE-DFE Fano decoder versus step size Δ, for a 20×20 16-QAM V-BLAST system.

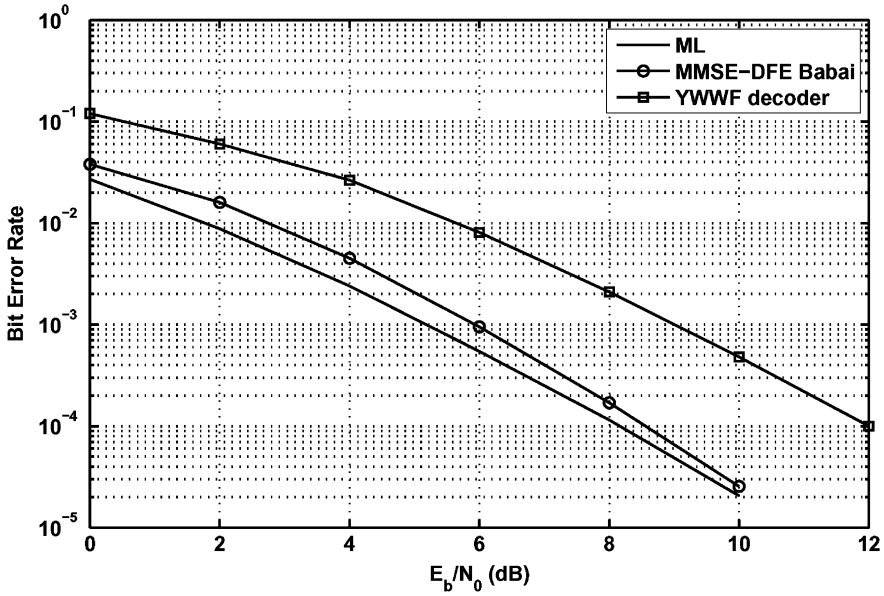

Fig. 8. Performance of MMSE-DFE preprocessing with DFE for a 4×4, 4-QAM V-BLAST system.

performance–complexity tradeoff required, one can choose an appropriate step size for the MMSE-DFE Fano decoder. In all our V-BLAST simulations, the step size was chosen to be 5. For small dimensions, the performance of the MMSE-DFE Babai decoder is remarkable. Fig. 8 compares the performance of this decoder with the ML performance for a 4×4, 4-QAM V-BLAST system. We also report the performance of the Yao–Wornell and Windpassinger–Fischer (YWWF) decoder which has the same complexity as the MMSE-DFE Babai decoder [48], [49]. It is shown that the performance of the proposed decoder is within a fraction of a decibel from that of ML decoder, whereas the algorithm in [48], [49] exhibits a loss of more than 3 dB.

B. Coded MIMO Systems

In this subsection, we consider two classes of space–time codes. The first class is the LD codes which are obtained by applying a linear transformation (over $\mathbb{C}$) to a vector of pulse amplitude modulation (PAM) symbols. For convenience, we follow the setup of Dayal and Varanasi [50], where two variants of the threaded algebraic space–time (TAST) constellations [22] are used in a 3×1 MIMO channel. This setup also allows for demonstrating the efficiency of the MMSE-DFE preprocessing in solving underdetermined systems. In [50], the rate-1 TAST constellation uses 64-QAM inputs at a rate of one symbol per channel use. The rate-3 TAST constellation, on the other hand, uses 4-QAM inputs to obtain the same throughput as the rate-1 constellation. As observed in [50], one obtains a sizable performance gain when using rate-3 TAST constellation under ML decoding. The main disadvantage, however, of the rate-3 code is that it corresponds to an underdetermined system with six-excess unknowns which significantly complicates the decoding problem. Fig. 9 shows that the performance of the proposed MMSE-DFE lattice decoder is less than 0.1 dB away from the

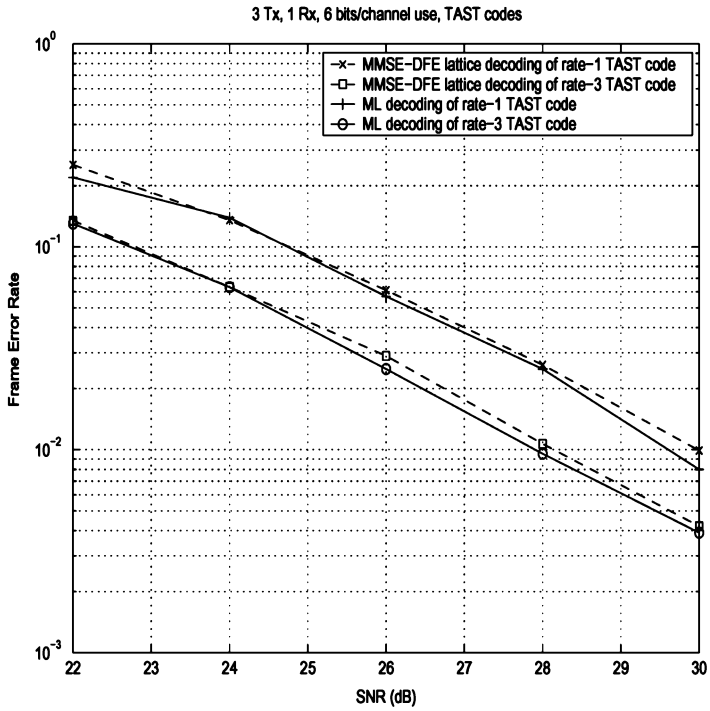

Fig. 9. Performance of TAST codes under MMSE-DFE lattice decoding and ML detection with $M = 3$ and $N = 1$.

TABLE I
COMPLEXITY RATIO OF THE PROPOSED ALGORITHM FOR RATE-3 TAST CONSTELLATION OVER RATE-1 TAST CONSTELLATION IN A 3×1 MIMO SYSTEM

SNR (dB)	22	24	26	28	30
γ	41	31	23	16	12

ML decoder for both cases. In order to quantify the complexity reduction offered by our approach, compared with the generalized sphere decoder (GSD) used in [50], we measure the average complexity increase with the excess dimensions. If we define

$$\gamma \triangleq \frac{\text{Average complexity of decoding rate-3 constellation}}{\text{Average complexity of decoding rate-1 constellation}} \tag{18}$$

then a straightforward implementation of the GSD, as outlined in [51], for example, would result in $\gamma = \mathcal{O}(4^6)$. In fact, even with the modification proposed in [50], Dayal and Varanasi could only bring this number down to $\gamma = 460$ at an SNR of 30 dB. In Table I, we report γ for the proposed algorithm at different SNRs, where one can see the significant reduction in complexity (i.e., from 460 to 12 at an SNR of 30 dB). Based on experimental observations, we also expect this gain in complexity reduction to increase with the excess dimension $m - n$.

The second space–time coding class is the algebraic codes proposed in [13]–[15]. This approach constructs linear codes, over the appropriate finite domain, and then the encoded symbols are mapped into QAM constellations. The QAM symbols are then parsed and appropriately distributed across the transmit antennas to obtain full diversity. It has been shown that the complexity of ML decoding of this class of codes grows exponentially with the number of transmit antennas and data rates. Here, we show that the proposed tree search framework allows for an efficient solution to this problem. Fig. 10 shows the performance of MMSE-DFE lattice decoding for two such constructions of space–time codes, i.e., Golay space–time code for two transmit antennas and the companion matrix code for three transmit antennas [13]. In both cases, the performance of the MMSE-DFE lattice decoder is seen to be essentially the same as the ML performance. In the proposed decoder, we use the lattice $\mathbf{\Lambda}$ obtained from underlying algebraic code through Construction A. The ML performance, obtained via exhaustive search in Fig. 10 is not feasible for higher dimensions due to exponential complexity in the number of dimensions.

C. Coded Transmission Over ISI Channels

In this subsection, we compare the performance of the MMSE-DFE Fano decoder with the Per-Survivor-Processing (PSP) algorithm for convolutionally coded transmission over ISI channels. The MMSE-DFE Fano decoder uses the Construction A lattice obtained from the convolutional code. For this scenario, it is known that PSP achieves near-ML frame error rate performance [52]. Fig. 11 compares the frame and bit error rates for a 4-state, rate-$1/2$ convolutional code with generator polynomials given by $(5, 7)$ and code length 200, over a five-tap ISI channel. The channel impulse response was chosen as $(0.848, -0.424, 0.2545, -0.1696, 0.0848)$. The Fano decoder with $b = 1.2$ and step size 5 is seen to achieve essentially the same performance as the PSP algorithm for this code, with reasonable search complexity over the entire SNR range. We again note that the loss in lattice decoding as opposed to finite search space is negligible, due to MMSE-DFE preprocessing of the channel prior to the search. The complexity (number of nodes generated in the search) of the PSP decoder is the same as that of a Viterbi algorithm for the convolutional code alone. Therefore (neglecting trellis

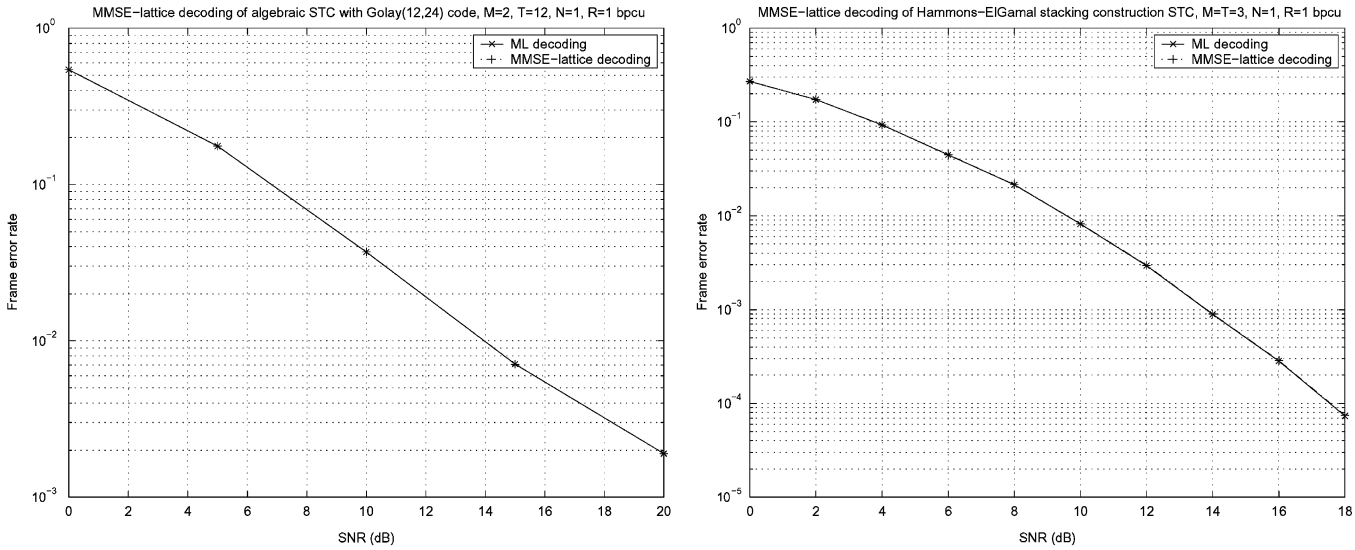

Fig. 10. Performance of MMSE-DFE lattice decoding and ML decoding for algebraic space–time codes.

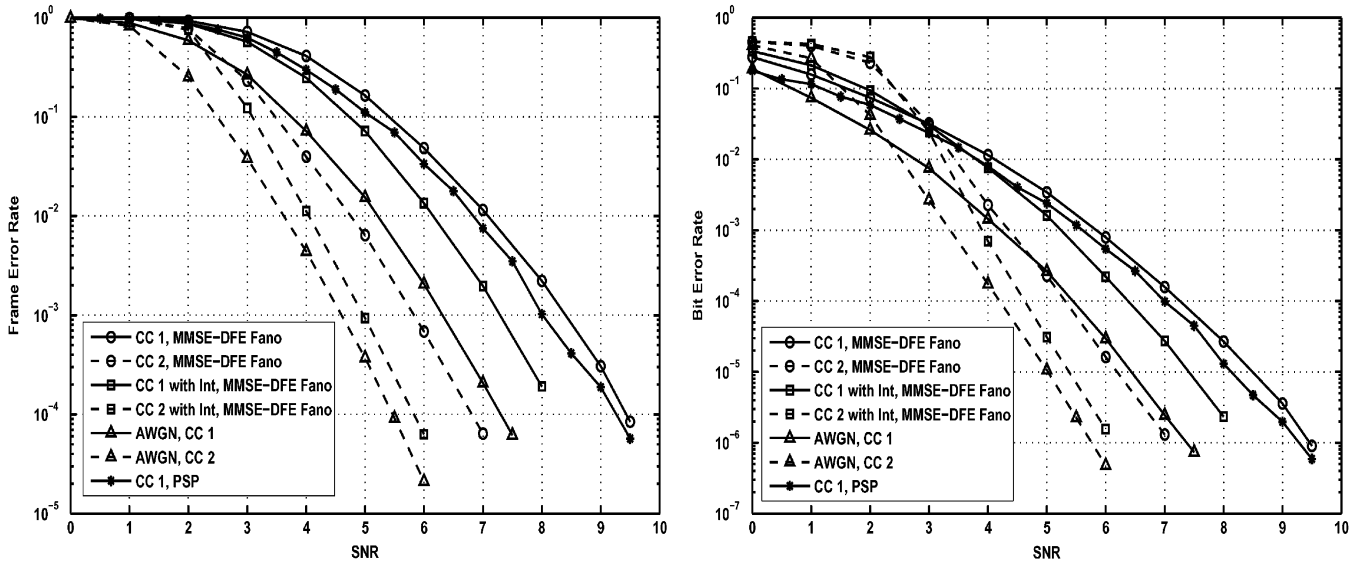

Fig. 11. Frame and bit error rate curves for the MMSE-DFE Fano and PSP algorithms for convolutional codes over an ISI channel. CC 1 and CC 2 denote the 4-state and 1024-state CC, respectively. The curves denoted by "Int." indicate interleaving after convolutional coding.

boundaries), the complexity per dimension of the PSP decoder is $R_c \times S \times B$, where R_c is the rate of the convolutional code, S the number of states, and B the number of branches per state in the code trellis. Thus, for the $(5,7)$ code, the complexity of the PSP decoder is four nodes per dimension. Notice that the complexity of PSP algorithm is constant with SNR but increases exponentially with the code constraint length. On the contrary, the complexity of the MMSE-DFE Fano decoder depends on SNR but it is essentially independent of the constraint length. Fig. 11 also shows the performance of the MMSE-DFE Fano decoder for a rate-$1/2$, 1024-state convolutional code with generator polynomials $(4672, 7542)$, with the same frame size. Due to the increased constraint length, the performance is significantly better (with almost no increase in complexity). The complexity of PSP algorithm, on the other hand, is 1024 nodes per dimension for this code. Fig. 11 also reports the performance of the MMSE-DFE Fano decoder when applied to an interleaved coded system over the ISI channel. This figure shows the performance gain offered by interleaving (i.e., the performance now is within a decibel from the idealized AWGN scenario). The price is an increased complexity of the MMSE-DFE Fano algorithm. The complexity increase due to interleaving is approximately $\times 2$ in this setup. Since interleaving destroys the time-ordering of the symbols input to the ISI channel, the PSP approach is not applicable with interleaving. Alternative schemes such as turbo equalization [53] still require a Bahl–Cocke–Jelinek–Raviv (BCJR) decoder with the same trellis complexity of the convolutional code alone, plus a soft-output detector for the ISI channel alone. Fig. 12 shows the complexities of the MMSE-DFE Fano and the PSP decoders for the cases considered in this section. For the 4-state convolutional code, the MMSE-DFE Fano decoder has an average complexity in the same range as that of the PSP decoder over the same code. However, for the 1024-state CC, the complexity of the MMSE-DFE Fano decoder is orders of magnitude lesser than that of the PSP decoder.

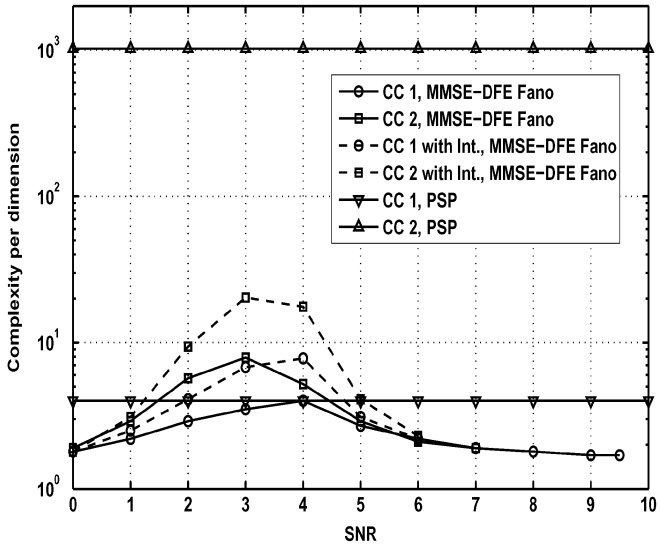

Fig. 12. Complexities of MMSE-DFE Fano and PSP for convolutional codes over an ISI channel. CC 1 and CC 2 denote the 4-state and 1024-state CC, respectively. The curves denoted by "Int." indicate interleaving after convolutional coding.

VI. Conclusion

A central goal of this paper was to introduce a unified framework for tree search decoding in wireless communication applications. Toward this end, we identified the roles of two different, but interrelated, components of the decoder; namely, 1) Preprocessing and 2) Tree Search. We presented a preprocessing stage composed of MMSE-DFE filtering for left preprocessing and lattice reduction with column ordering for right preprocessing. We argued that this preprocessor allows for ignoring the boundary control in the tree search stage while entailing only a marginal loss in performance. By relaxing the boundary control, we were able to build a generic framework for designing tree search strategies for joint detection and decoding. Within this framework, BeFS emerged as a very efficient solution that offers many valuable advantages. To limit the storage requirement of BeFS, we rediscovered the Fano decoder as our proposed tree search algorithm. Finally, we established the superior performance–complexity tradeoff of the Fano decoder analytically in a V-BLAST configuration and demonstrated its excellent performance and complexity in more general scenarios via simulation results.

Appendix A
The Fano Decoder

In this appendix, we obtain the cost function used in the Fano decoder from the Fano metric defined for tree codes over general point-to-point channels, and give a brief description of the Fano decoder and its properties.

A. Generic Cost Function of the Fano Decoder

For the transmitted sequence $\hat{\boldsymbol{x}}$, let

$$\boldsymbol{y} = \boldsymbol{R}\hat{\boldsymbol{x}} + \boldsymbol{w} \tag{19}$$

be the system model, as in Section II. In (19), the noise sequence $\boldsymbol{w}$ is composed of *i.i.d* Gaussian noise components with zero mean and unit variance.

For a general point-to-point channel with continuous output, the Fano metric of the node $\boldsymbol{x}_1^k$ can be written as [54]

$$\mu\left(\boldsymbol{x}_1^k\right) = \log\left(\frac{\Pr(\mathcal{H}(\boldsymbol{x}_1^k))p(\boldsymbol{y}_1^k|\mathcal{H}(\boldsymbol{x}_1^k))}{p(\boldsymbol{y}_1^k)}\right) \tag{20}$$

where $\mathcal{H}(\boldsymbol{x}_1^k)$ is the hypothesis that $\boldsymbol{x}_1^k$ form the first k symbols of the transmitted sequence.

For $1 \leq k \leq m$, if $\Pr(\mathcal{H}(\boldsymbol{x}_1^k))$ is uniform over all nodes $\boldsymbol{x}_1^k$ that consist of the first k components of any valid codeword in $\mathcal{C}$, from (20), the cost function for the Fano decoder for our system model (19) can be simplified as

$$f\left(\boldsymbol{x}_1^k\right) = -\mu\left(\boldsymbol{x}_1^k\right) = \log\left(\sum_{\boldsymbol{x}_1^k} e^{-\frac{\sum_{j=1}^k w_j\left(\boldsymbol{x}_1^j\right)}{2}}\right) + \frac{\sum_{j=1}^k w_j\left(\boldsymbol{x}_1^j\right)}{2}. \tag{21}$$

Since summation over $\boldsymbol{x}_1^k$ in (21) is not feasible, we use the following approximations: first, $\log(\sum a_i) \approx \log(\max(a_i))$, so the sum can be approximated by the largest term. Second, for moderate to high SNRs, the transmitted sequence is actually the closest vector with a high probability, i.e., the largest term corresponds to the transmitted sequence. Thus, (21) can be approximated as

$$\log\left(\sum_{\boldsymbol{x}_1^k} e^{-\frac{\sum_{j=1}^k w_j\left(\boldsymbol{x}_1^j\right)}{2}}\right) \approx -\frac{|\boldsymbol{w}_1^k|^2}{2}. \tag{22}$$

After averaging (22) over noise samples and scaling, we have

$$f\left(\boldsymbol{x}_1^k\right) = \sum_{j=1}^k w_j\left(\boldsymbol{x}_1^j\right) - k.$$

In general, the cost function for the Fano decoder can be written in terms of the parameter b, the bias, as

$$f\left(\boldsymbol{x}_1^k\right) = \sum_{j=1}^k w_j\left(\boldsymbol{x}_1^j\right) - bk.$$

B. The Algorithm

The operation of the Fano decoder with no boundary control (lattice decoding) follows the following steps.

- *Step 1: (Initialize)* Set $k \leftarrow 0, T \leftarrow 0, \boldsymbol{x} \leftarrow x_0$.
- *Step 2: (Look forward)* $\boldsymbol{x}_1^{k+1} \leftarrow (\boldsymbol{x}_1^k, x_{k+1})$, where x_{k+1} is the $(k+1)$th component of the best child node of $\boldsymbol{x}_1^k$.
- *Step 3:*
  If $f(\boldsymbol{x}_1^{k+1}) \leq T$,
    If $k+1 = m$ (*leaf node*), then $\hat{\boldsymbol{x}} = \boldsymbol{x}_1^m$; exit.
    Else (*move forward*), $k \leftarrow k+1$.
      If $f(\boldsymbol{x}_1^{k-1}) > T - \Delta$,
        while $f(\boldsymbol{x}_1^k) \leq T - \Delta, T \leftarrow T - \Delta$ (*tighten threshold*).
    Go to step 2.
  Else
    If ($k = 0$ or $f(\boldsymbol{x}_1^{k-1}) > T$), $T \leftarrow T + \Delta$ (*cannot move back, so relax threshold).*
      Go to step 2.

Else (*move back and look forward to the next best node*)
$\boldsymbol{x}_1^k \leftarrow \{\boldsymbol{x}_1^{k-1}, x_k\}$, where x_k is the last component of the next best child node of $\boldsymbol{x}_1^{k-1}$.
$k \leftarrow k-1$.
Go to Step 3. □

Note that T (i.e., the threshold) is allowed to take values only in multiples of the step size Δ (i.e., $0, \pm\Delta, \pm 2\Delta, \ldots$). When a node is visited by the Fano decoder for the first time, the threshold T is tightened to the least possible value while maintaining the validity of the node. If the current node does not have a valid child node, then the decoder moves back to the parent node (if the parent node is valid) and attempts moving forward to the next best node. However, if the parent node is not valid, the threshold is relaxed and attempt is made to move forward again, proceeding in this way until a leaf node is reached.

The determination of best and next best child nodes is simplified in CLPS problem; the child node generation order *gen* in SE enumeration (Section IV-A2) generates child nodes with cost functions in ascending order, given any node $\boldsymbol{x}_1^k$.

C. Properties of the Fano Decoder

The main properties of the Fano decoder used in our analysis are [54] as follows.

1. A node $\boldsymbol{x}_1^k$ is generated by the Fano decoder only if its cost function is not greater than the bound T.
2. Let f_M be the maximum cost function along the transmitted path. The bound T is always less than $(f_M + \Delta)$, where Δ is the step size of the Fano decoder; that is, $\max\{T\} < T_M \triangleq f_M + \Delta$.

All nodes that are generated by the Fano decoder are necessarily those with cost function less than the bound T, by Property 1. However, even though the cost function of some node $\boldsymbol{x}_1^k$ may be smaller than the bound, the node itself might not be visited when bound takes the value T. If any of the cost functions along the path $\{\boldsymbol{x}_1^r, r < k\}$ increases above T, the node $\boldsymbol{x}_1^r$ is not generated and thus $\boldsymbol{x}_1^k$ is not visited. Hence, this is not a sufficient condition for a node to be generated.

Moreover, in Property 2, the bound T is always lesser than $(f'_M + \Delta)$, where f'_M is the maximum cost function along any path of length m. A tighter bound is obtained only when the maximum cost function corresponding to the path with the least f'_M is chosen. However, f_M along the transmitted path is usually easier to characterize statistically than f'_M.

Appendix B
Properties of the Stack Decoder

For any node $\boldsymbol{x}_1^k$ in the tree, let

$$h(\boldsymbol{x}_1^k) \triangleq \sum_{i=1}^{k} w_i(\boldsymbol{x}_1^k) - bk.$$

For the stack algorithm, the cost function of any node in ACTIVE at any instant is defined as follows: If $\boldsymbol{x}_1^k$ is a leaf node, then $f(\boldsymbol{x}_1^k) = -\infty$. Otherwise, we let $\boldsymbol{x}_{1,g}^{k+1}$ be the best child node of $\boldsymbol{x}_1^k$ not generated yet, and define $f(\boldsymbol{x}_1^k) = h(\boldsymbol{x}_{1,g}^{k+1})$. We note that h of any node, once generated, remains constant throughout the algorithm, and f of any node is nondecreasing as the algorithm progresses.

Proposition 1: Let $\bar{\boldsymbol{x}}_1^m = (\bar{x}_1, \ldots \bar{x}_m)$ be the path chosen by the stack algorithm, and $\boldsymbol{x}_1^m = (x_1, \ldots, x_m)$ be any path in the tree. Then

$$\max_{1 \le j \le m} h\left(\bar{\boldsymbol{x}}_1^j\right) \le \max_{1 \le j \le m} h\left(\boldsymbol{x}_1^j\right). \tag{23}$$

Proof: On the contrary, assume there exists a path $(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_d, \breve{x}_{d+1}, \ldots, \breve{x}_m)$ that does not satisfy (23). Here, the path is assumed to share the same nodes with the chosen path until level d, and diverges from the chosen path $\bar{\boldsymbol{x}}$ from level $d+1$ onwards. Since this path does not satisfy (23)

$$\max_{d+1 \le j \le m} h\left(\bar{\boldsymbol{x}}_1^j\right) > \max_{d+1 \le j \le m} h\left(\breve{\boldsymbol{x}}_1^j\right). \tag{24}$$

Let $\bar{\boldsymbol{x}}_1^k$, $k > d$, be the node for which $\max_{d+1 \le j \le m} h(\bar{\boldsymbol{x}}_1^j)$ occurs. Then, we have

$$h\left(\bar{\boldsymbol{x}}_1^k\right) > h\left(\breve{\boldsymbol{x}}_1^j\right), \qquad d < j \le m. \tag{25}$$

Since $\bar{\boldsymbol{x}}_1^m$ is the chosen path, the node $\bar{\boldsymbol{x}}_1^k$ is generated at some instant before the search terminates. Just before $\bar{\boldsymbol{x}}_1^k$ is generated, $f(\bar{\boldsymbol{x}}_1^{k-1}) = h(\bar{\boldsymbol{x}}_1^k)$, since $\bar{\boldsymbol{x}}_1^k$ is the best child node of $\bar{\boldsymbol{x}}_1^{k-1}$ not generated yet. Moreover, since $h(\bar{\boldsymbol{x}}_1^k) > h(\breve{\boldsymbol{x}}_1^{d+1})$, the node $\bar{\boldsymbol{x}}_1^d$ with cost function $f(\bar{\boldsymbol{x}}_1^d) = h(\breve{\boldsymbol{x}}_1^{d+1})$ appears at the top of the stack at some instant before $\bar{\boldsymbol{x}}_1^k$ is generated. Therefore, $\breve{\boldsymbol{x}}_1^{d+1}$ is generated before $\bar{\boldsymbol{x}}_1^k$ is generated. Since the search does not terminate before $\bar{\boldsymbol{x}}_1^k$ is generated, applying the same argument, one sees that all the nodes $\breve{\boldsymbol{x}}_1^{d+2}, \ldots, \breve{\boldsymbol{x}}_1^m$ are generated before $\bar{\boldsymbol{x}}_1^k$ is generated. However, once $\breve{\boldsymbol{x}}_1^m$ is generated by the stack algorithm, the search terminates, with $(\bar{x}_1, \ldots, \bar{x}_d, \breve{x}_{d+1}, \ldots, \breve{x}_m)$ as the chosen path, thus leading to a contradiction. Since $1 \le d \le m$ can take any value, the inequality in (23) is satisfied by all paths.

Proposition 2: If

$$\max_{1 \le j \le d} h\left(\boldsymbol{x}_1^d\right) > \max_{1 \le j \le m} h\left(\bar{\boldsymbol{x}}_1^j\right) \tag{26}$$

then, the node $\boldsymbol{x}_1^d$ is not generated.

Proof: First, we show that if

$$h\left(\boldsymbol{x}_1^d\right) > \max_{1 \le j \le m} h\left(\bar{\boldsymbol{x}}_1^j\right) \tag{27}$$

then $\boldsymbol{x}_1^d$ is not generated. Let (27) be true, and assume $\boldsymbol{x}_1^d$ is generated. Then, just before $\boldsymbol{x}_1^d$ is generated, its parent node $\boldsymbol{x}_1^{d-1}$ is at the top of ACTIVE, with cost function $f(\boldsymbol{x}_1^{d-1}) = h(\boldsymbol{x}_1^d)$. However, since $h(\boldsymbol{x}_1^d) > h(\bar{\boldsymbol{x}}_1^j)$, $1 \le j \le m$, all nodes along the chosen path are generated before $\boldsymbol{x}_1^d$ is generated, and hence, the search terminates before $\boldsymbol{x}_1^d$ is generated. Noting that $\boldsymbol{x}_1^d$ can be generated only if all the nodes $\boldsymbol{x}_1^1, \ldots, \boldsymbol{x}_1^{d-1}$ are generated, and applying the same argument for $\boldsymbol{x}_1^{d-1}, \ldots, \boldsymbol{x}_1^1$, we have (26).

Appendix C
Proof of Theorem 2

Let $\mathcal{A}_{IR}$ be the set of nodes generated by the IR algorithm, where the bounding function $\boldsymbol{t}$ has components given by $t_k = bk + \delta$. Let $\mathcal{A}_s$ be the set of nodes generated by the stack decoder with the bias b. The IR algorithm in Theorem 2 can be defined with bounding function given by $\{t_k = bk + \delta, 1 \le k \le m\}$, and the cost function for any node $\boldsymbol{x}_1^k$ given by $\sum_{i=1}^{k} w_i(\boldsymbol{x}_1^i)$,

or equivalently, with the bounding function $t_k = \delta$ and cost function $(\sum_{i=1}^{k} w_i(\boldsymbol{x}_1^i) - bk)$. As in Appendix B, let

$$h(\boldsymbol{x}_1^k) \triangleq \left(\sum_{i=1}^{k} w_i(\boldsymbol{x}_1^i) - bk\right)$$

for any path $\boldsymbol{x}_1^k$ in the tree. If δ is the bound of the IR algorithm, then any node $\boldsymbol{x}_1^k$ is generated by the algorithm if and only if all the conditions $\{h(\boldsymbol{x}_1^1) < \delta, h(\boldsymbol{x}_1^2) < \delta, \ldots, h(\boldsymbol{x}_1^k) < \delta\}$, are satisfied (since $\boldsymbol{x}_1^k$ is generated only if $\boldsymbol{x}_1^1, \ldots, \boldsymbol{x}_1^{k-1}$ are also generated). Therefore,

$$\mathcal{A}_{\mathrm{IR}} = \left\{\boldsymbol{x}_1^k : \max_{1\le j\le k} h\left(\boldsymbol{x}_1^j\right) < \delta\right\}. \tag{28}$$

Moreover, δ should be such that at least one sequence $\boldsymbol{x} \in \mathcal{U}$ is included within the search space.[9] Let $\hat{\boldsymbol{x}}_{\mathrm{IR}}$ be a leaf node such that

$$\hat{\boldsymbol{x}}_{\mathrm{IR}} = \arg\min_{\boldsymbol{x}\in\mathcal{U}} \left(\max_{1\le j\le m} h\left(\boldsymbol{x}_1^j\right)\right) \tag{29}$$

i.e., $\hat{\boldsymbol{x}}_{\mathrm{IR}}$ has the least value of maximum cost function among all paths of length m. If

$$\delta \le \max_{1\le j\le m} h(\boldsymbol{x}_{1,\mathrm{IR}}^j)$$

then no $\boldsymbol{x} \in \mathcal{U}$ lies within the search space, and the search space is empty. If lattice decoding is used, then the minimum in (29) is taken over all $\boldsymbol{x} \in \mathbb{Z}^m$. Thus, $\delta > \max_{1\le j\le m} h(\boldsymbol{x}_{1,\mathrm{IR}}^j)$. From Appendix B, Proposition 1, the path chosen by the stack algorithm $\bar{\boldsymbol{x}}_1^m$ satisfies

$$\max_{1\le j\le m} h\left(\bar{\boldsymbol{x}}_1^j\right) \le \max_{1\le j\le m} h\left(\boldsymbol{x}_1^j\right) \tag{30}$$

where $\boldsymbol{x}_1^m$ is any other path.

From (29) and (30)

$$\max_{1\le j\le m} h\left(\bar{\boldsymbol{x}}_1^j\right) = \max_{1\le j\le m} h\left(\boldsymbol{x}_{1,\mathrm{IR}}^j\right) < \delta. \tag{31}$$

From Proposition 2 and (31), $\mathcal{A}_s \subseteq \mathcal{A}_{\mathrm{IR}}$.

Appendix D
Proof of Theorem 3

In this appendix, we derive an upper bound to the frame error rate for a V-BLAST system with uncoded input (with Q-PAM constellation for the components), for the Fano decoder that visits paths in the regular Q-PAM signal space. The preprocessing assumed here is QR transformation of $\boldsymbol{H}$.

From Appendix A, part C, any sequence $\boldsymbol{x}$ can be generated by the Fano decoder only if its cost function is not greater than $(f_M + \Delta)$, where f_M is the minimum cost function along the transmitted sequence path and Δ is the stepsize of the Fano decoder. One has

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{z} = \boldsymbol{Q}\begin{pmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{pmatrix}\boldsymbol{x} + \boldsymbol{z} \tag{32}$$

[9]Otherwise, δ is increased and search is repeated afresh.

and therefore,

$$\boldsymbol{y} \leftarrow \boldsymbol{Q}^T\boldsymbol{y} = \begin{pmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{pmatrix}\boldsymbol{x} + \begin{pmatrix}\boldsymbol{w}_{r+1}^n\\ \boldsymbol{w}_1^r\end{pmatrix} \tag{33}$$

where $r = n - m$ is the excess degrees of freedom in the V-BLAST system. Since the cost function of a leaf node $\boldsymbol{x}_1^m$ is

$$f(\boldsymbol{x}_1^m) = \sum_{i=1}^{m} w_i(\boldsymbol{x}_1^i) - bm = |\boldsymbol{R}\tilde{\boldsymbol{x}} + \boldsymbol{w}_{r+1}^n|^2 - bm$$

the conditional pairwise error probability of transmitting $\hat{\boldsymbol{x}}$ and decoding $\boldsymbol{x}$ at the receiver can be upper-bounded as

$$P_e(\hat{\boldsymbol{x}} \to \boldsymbol{x}|f_M, \boldsymbol{H}) \le \Pr\left(\sum_{i=1}^{m} w_i\left(\boldsymbol{x}_1^i\right) - bm < f_M + \Delta\right) \tag{34}$$

$$= \Pr\left(\left|\boldsymbol{R}\tilde{\boldsymbol{x}} + \boldsymbol{w}_{r+1}^n\right|^2 < bm + f_M + \Delta\right) \tag{35}$$

where

$$f_M = \max\{0, |\boldsymbol{w}_{r+1}^{r+1}|^2 - b, |\boldsymbol{w}_{r+1}^{r+2}|^2 - 2b, \ldots, |\boldsymbol{w}_{r+1}^n|^2 - mb\}.$$

In (34), the upper bound is due to the fact that in general, $f(\boldsymbol{x}_1^m) < f_M + \Delta$ is only a necessary condition for $\boldsymbol{x}_1^m$ to be decoded by the Fano decoder.

In this section, we obtain an upper bound on the frame error rate P_e of the Fano decoder by union bounding (35) over all possible sequence pairs $\{\hat{\boldsymbol{x}} \in \mathcal{U}, \boldsymbol{x} \in \mathcal{U} : \hat{\boldsymbol{x}} \ne \boldsymbol{x}\}$ and averaging over f_M and $\boldsymbol{H}$

$$P_e \le E_{f_M,\boldsymbol{H},\hat{\boldsymbol{x}}}\left(\sum_{\{\boldsymbol{x}\in\mathcal{U},\hat{\boldsymbol{x}}\ne\boldsymbol{x}\}} P_e(\hat{\boldsymbol{x}} \to \boldsymbol{x}|f_M, \boldsymbol{H})\right). \tag{36}$$

Inequality (35) can be rewritten as

$$P_e(\hat{\boldsymbol{x}} \to \boldsymbol{x}|f_M, \boldsymbol{H}) \le \Pr\left(\left|\begin{pmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{pmatrix}\tilde{\boldsymbol{x}} + \boldsymbol{w}_1^n\right|^2 < bm + f_M + \Delta + |\boldsymbol{w}_1^r|^2\right) \tag{37}$$

$$= \Pr\left(\left|\begin{pmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{pmatrix}\tilde{\boldsymbol{x}} + \boldsymbol{w}_1^n\right|^2 - |\boldsymbol{w}_1^n|^2 < bm + f_M + \Delta - |\boldsymbol{w}_{r+1}^n|^2\right) \tag{38}$$

$$= \Pr\left(|\boldsymbol{H}\tilde{\boldsymbol{x}} + \boldsymbol{z}|^2 - |\boldsymbol{z}|^2 < bm + f_M + \Delta - |\boldsymbol{w}_{r+1}^n|^2\right) \tag{39}$$

$$\le \Pr(|\boldsymbol{H}\tilde{\boldsymbol{x}} + \boldsymbol{z}|^2 - |\boldsymbol{z}|^2 < bm + \Delta) \tag{40}$$

since

$$f_M - |\boldsymbol{w}_{r+1}^n|^2 = \max\{-|\boldsymbol{w}_{r+1}^n|^2, -|\boldsymbol{w}_{r+2}^n|^2 - b, \ldots, -mb\} \le 0.$$

The bound in (40) is now independent of f_M and hence,

$$P_e(\hat{\boldsymbol{x}} \to \boldsymbol{x}|\boldsymbol{H}) = E_{f_M}(P_e(\hat{\boldsymbol{x}} \to \boldsymbol{x}|f_M, \boldsymbol{H})) \le \Pr(|\boldsymbol{H}\tilde{\boldsymbol{x}} + \boldsymbol{z}|^2 - |\boldsymbol{z}|^2 < bm + \Delta). \tag{41}$$

Let $d^2(\hat{\boldsymbol{x}},\boldsymbol{x}) = |\boldsymbol{H}\tilde{\boldsymbol{x}}|^2$ represent the squared Euclidean distance between the lattice points $\boldsymbol{H}\boldsymbol{x}$ and $\boldsymbol{H}\hat{\boldsymbol{x}}$. Then

$$\Pr(|\boldsymbol{H}\tilde{\boldsymbol{x}}+\boldsymbol{z}|^2-|\boldsymbol{z}|^2<mb+\Delta) \le \begin{cases} e^{\left(-\frac{1}{8}(d^2(\hat{\boldsymbol{x}},\boldsymbol{x})-mb-\Delta)^2/d^2(\hat{\boldsymbol{x}},\boldsymbol{x})\right)}, & d^2(\hat{\boldsymbol{x}},\boldsymbol{x})>mb+\Delta \\ 1, & d^2(\hat{\boldsymbol{x}},\boldsymbol{x})\le mb+\Delta \end{cases} \tag{42}$$

by Chernoff bound. For $d^2(\hat{\boldsymbol{x}},\boldsymbol{x}) > mb+\Delta$, (42) can be simplified as

$$\begin{aligned} \Pr(|\boldsymbol{H}\tilde{\boldsymbol{x}}+\boldsymbol{z}|^2-|\boldsymbol{z}|^2 \le mb+\Delta) &< e^{-\frac{1}{8}\left(d^2(\hat{\boldsymbol{x}},\boldsymbol{x})+\frac{(mb+\Delta)^2}{d^2(\hat{\boldsymbol{x}},\boldsymbol{x})}-2(mb+\Delta)\right)} && (43)\\ &\le e^{-\frac{1}{8}\left(d^2(\hat{\boldsymbol{x}},\boldsymbol{x})\right)}e^{(mb+\Delta)/4} && (44)\end{aligned}$$

since $e^{-(mb+\Delta)^2/(8d^2(\hat{\boldsymbol{x}},\boldsymbol{x}))} < 1$, for $d^2(\hat{\boldsymbol{x}},\boldsymbol{x}) > 0$. Let

$$q \triangleq \min_{\boldsymbol{x}_i,\boldsymbol{x}_j\in\mathcal{U},i\neq j}(|\boldsymbol{H}(\boldsymbol{x}_i-\boldsymbol{x}_j)|^2). \tag{45}$$

The bound in (42) can be simplified as

$$\Pr(|\boldsymbol{H}\tilde{\boldsymbol{x}}+\boldsymbol{z}|^2-|\boldsymbol{z}|^2 \le mb+\Delta) \le \begin{cases} e^{-q/8}e^{(mb+\Delta)/4}, & q>mb+\Delta \\ 1, & q\le mb+\Delta. \end{cases} \tag{46}$$

Noting that the bound in (46) is same for all pairs $\hat{\boldsymbol{x}},\boldsymbol{x}\in\mathcal{U}$, $\hat{\boldsymbol{x}}\neq\boldsymbol{x}$, we have from (36)

$$P_e \le \int_0^{mb+\Delta} p(q)dq + Q^m e^{(mb+\Delta)/4}\int_{mb+\Delta}^{\infty} e^{-q/8}p(q)dq \tag{47}$$

where $p(q)$ is the probability density function (pdf) of q. An upper bound on $p(q)$ is given by [55]

$$p(q) < p_\chi(q)\sum_{k=1}^{m}\binom{m}{k}\frac{1}{k} \tag{48}$$

where $p_\chi(q)$ is the pdf of a scaled chi-square random variable with n degrees of freedom and mean $\frac{n\rho}{m\alpha}$, where $\alpha = (Q^2-1)/12$ (i.e., a random variable that is the sum of squares of n i.i.d. zero-mean Gaussian variables with variance $\frac{\rho}{m\alpha}$). Therefore, we have

$$\begin{aligned} P_e &\le AQ^m e^{(mb+\Delta)/4}\int_{mb+\Delta}^{\infty} e^{(-q/8)} \times \frac{q^{(n/2-1)}e^{-q/(2\sigma^2)}}{2^{n/2}\Gamma(\frac{n}{2})\sigma^n}dq + A\gamma\left(\frac{mb+\Delta}{2\sigma^2},\frac{n}{2}\right) && (49)\\ &\le AQ^m e^{(mb+\Delta)/4}\int_{0}^{\infty} e^{(-q/8)} \times \frac{q^{(n/2-1)}e^{-q/(2\sigma^2)}}{2^{n/2}\Gamma(\frac{n}{2})\sigma^n}dq + A\gamma\left(\frac{mb+\Delta}{2\sigma^2},\frac{n}{2}\right) && (50)\\ &= \frac{AQ^m e^{(mb+\Delta)/4}}{\left(1+\frac{\rho}{4m\alpha}\right)^{n/2}} + A\gamma\left(\frac{mb+\Delta}{2\sigma^2},\frac{n}{2}\right) && (51)\end{aligned}$$

where

$$\sigma^2 = \frac{\rho}{m\alpha}, A \triangleq \sum_{k=1}^{m}\frac{1}{k}\binom{m}{k}$$

is a constant independent of q or ρ, and $\gamma(x,a)$ is the incomplete gamma function. If b is bounded (i.e., $b<\mathcal{M}<\infty)\ \forall\rho$, then $e^{(mb+\Delta)/4}$ is also bounded for all ρ and finite m. The error performance of the Fano decoder can now be characterized by the sum of two terms. The dependence of the first error term on ρ is of the form $\rho^{-(n/2)}$ for large values of SNR, and hence has the same diversity as the ML decoder. For $\rho > \frac{m\alpha(mb+\Delta)}{n}$, the second term can be bounded as

$$\begin{aligned} \gamma\left(\frac{mb+\Delta}{2\sigma^2},\frac{n}{2}\right) &\le \left(\frac{m\alpha(mb+\Delta)}{n\rho}\right)^{n/2} e^{\left(\frac{n}{2}-\frac{m\alpha(mb+\Delta)}{2\rho}\right)} && (52)\\ &= \left(\frac{em\alpha(mb+\Delta)}{n\rho}\right)^{n/2} e^{-\frac{m\alpha(mb+\Delta)}{2\rho}} && (53)\\ &\le \left(\frac{em\alpha(mb+\Delta)}{n\rho}\right)^{n/2} && (54)\end{aligned}$$

where (52) follows from inequality (61) on the incomplete gamma function. The second term also has the dependence $\rho^{-(n/2)}$, and hence, the Fano decoder achieves the *same* diversity as that of the ML decoder for this system.

The above derivation also applies to the Stack algorithm, with minor modifications. Since any path $\boldsymbol{x}\neq\hat{\boldsymbol{x}}$ is decoded as the closest point by the stack algorithm only if $h(\boldsymbol{x}) = \sum_{i=1}^{m} w_i(\boldsymbol{x}_1^i) - bm$ is not greater than f_M (Proposition 1, Appendix B), the conditional pairwise error probability for the stack decoder is bounded as

$$\begin{aligned} P_e(\hat{\boldsymbol{x}}\to\boldsymbol{x}|f_M,\boldsymbol{H}) &\le \Pr\left(\sum_{i=1}^{m} w_i\left(\boldsymbol{x}_1^i\right) - bm < f_M\right) && (55)\\ &= \Pr\left(\left|\boldsymbol{R}\tilde{\boldsymbol{x}}+\boldsymbol{w}_{r+1}^n\right|^2 < bm+f_M\right). && (56)\end{aligned}$$

From (35) and (56), the frame error rate of the stack algorithm is also bounded by (51), with $\Delta=0$. Thus, the stack algorithm too achieves the same diversity as the ML decoder for a V-BLAST system, for any finite value of b. □

APPENDIX E
PROOF OF THEOREM 4

The following are required for the proof.

A. Wald's Inequality

Let $S_0=0, S_1, S_2, \ldots$ be a random walk, with

$$S_j = \sum_{i=1}^{j} X_i$$

where X_is are *i.i.d* random variables such that $\Pr(X_i>0)>0$, $\Pr(X_i<0)>0$, and $E(X_i)<0$. Let $g(\lambda) = E(e^{\lambda X_i})$ be

the moment generating function of X_i. Let $\lambda_0 > 0$ be a root of $g(\lambda) = 1$. Then, from Wald's identity [54]

$$\Pr(S_{\max} > u) \leq e^{-\lambda_0 u} \tag{57}$$

where $S_{\max} = \max_j(S_j)$.

For the random walk with $X_i = w_i^2 - b$, where $w_i \sim \mathcal{N}(0,1)$, the above conditions are satisfied if $b > 1$. The moment-generating function for $X_i = w_i^2 - b$ is given by

$$g(\lambda) = \frac{e^{-\lambda b}}{\sqrt{1-2\lambda}}. \tag{58}$$

From (58), $\lambda_0 > 0$ can be found as the positive root of the equation

$$-2\lambda b = \log(1-2\lambda).$$

Notice that since $\log(1-2\lambda)$ decreases from 0 to $-\infty$ as λ increases from 0 to $\frac{1}{2}$, λ_0 satisfies $\lambda_0 \in (0, 0.5)$. Since $\max_{0\leq j\leq m} S_j \leq \max_{j\geq 0} S_j$, the bound in (57) is also valid for any *stopped* random walk.

B. Upper Bound on $\gamma(\beta, k)$

For a scaled chi-square random variable X with k degrees of freedom and mean $k\sigma^2$

$$\Pr(X \leq \beta) = \gamma\left(\frac{\beta}{2\sigma^2}, \frac{k}{2}\right)$$

where γ is known as the incomplete gamma function. From Chernoff bound, we have

$$\gamma\left(\frac{\beta}{2\sigma^2}, \frac{k}{2}\right) = \Pr(X \leq \beta) = \int_0^{\beta} p_X(x)dx \tag{59}$$

$$\leq \int_0^{\infty} \frac{e^{-sx}}{e^{-s\beta}} p_X(x)dx \tag{60}$$

for $s > 0$. Minimizing (60) over $s > 0$, we have

$$\gamma\left(\frac{\beta}{2\sigma^2}, \frac{k}{2}\right) \leq \begin{cases} \left(\frac{\beta}{\sigma^2 k}\right)^{k/2} e^{(\frac{k}{2} - \frac{\beta}{2\sigma^2})}, & \frac{\beta}{2\sigma^2} < \frac{k}{2} \\ 1, & \frac{\beta}{2\sigma^2} \geq \frac{k}{2}. \end{cases} \tag{61}$$

C. Proof

Let $\boldsymbol{x}_1^k$ be any path in the tree, and

$$h(\boldsymbol{x}_1^k) = \sum_{i=1}^{k} w_i(\boldsymbol{x}_1^i) - bk$$

as in Appendix B. Let $f_M \triangleq \max_{1\leq i\leq m} h(\hat{\boldsymbol{x}}_1^i)$, where $\hat{\boldsymbol{x}}$ is the transmitted sequence. From Appendix B, Propositions 1 and 2, any node $\boldsymbol{x}_1^k$ is generated only if $\max_{1\leq j\leq k} h(\boldsymbol{x}_1^j)$ is not above f_M. Let $G(\boldsymbol{x}_1^k) = 1$ if the node $\boldsymbol{x}_1^k$ is generated before the algorithm terminates, and 0 otherwise. Then, $\sum_{k=1}^{m}\sum_{\boldsymbol{x}_1^k} G(\boldsymbol{x}_1^k)$ is the number of nodes generated by the stack algorithm, and

$$C_m = \frac{1}{m} E_{f_M, \boldsymbol{H}, \boldsymbol{z}}\left(\sum_{k=1}^{m}\sum_{\boldsymbol{x}_1^k} G\left(\boldsymbol{x}_1^k\right)\right) \tag{62}$$

is the average complexity per dimension of the stack algorithm. Let $\bar{\boldsymbol{R}}_{k,k}$ be the lower $k \times k$ part of the $\bar{\boldsymbol{R}}$ matrix, i.e.,

$$\boldsymbol{R}_{k,k} = \begin{pmatrix} r_{k,k} & \cdots & r_{k,1} \\ & \ddots & \vdots \\ \boldsymbol{0} & & r_{1,1} \end{pmatrix}.$$

Then, we have

$$\Pr\left(G\left(\boldsymbol{x}_1^k\right) = 1 \,|\, f_M, \boldsymbol{H}\right)$$

$$\leq \Pr\left(\left|\boldsymbol{R}_{k,k}\tilde{\boldsymbol{x}}_1^k + \boldsymbol{w}_{r+1}^{r+k}\right|^2 - bk < f_M\right) \tag{63}$$

$$= \Pr\left(\left|\begin{pmatrix}\boldsymbol{R}_{k,k} \\ \boldsymbol{0}\end{pmatrix}\tilde{\boldsymbol{x}}_1^k + \boldsymbol{w}_1^{r+k}\right|^2 - bk < f_M + |\boldsymbol{w}_1^r|^2\right) \tag{64}$$

where $r = n - m$ is the excess degrees of freedom in the V-BLAST system. From [5], for each $k \leq m$, one can find an $(r+k) \times k$ matrix $\bar{\boldsymbol{H}}_{r+k,k}$ that has the same distribution as the lower $(r+k) \times k$ part of $\boldsymbol{H}$, and an $(r+k) \times (r+k)$ unitary matrix $\boldsymbol{\Theta}^{(r+k)}$ whose distribution is independent of $\boldsymbol{R}_{k,k}$, such that

$$\bar{\boldsymbol{H}}_{r+k,k} = \boldsymbol{\Theta}^{(r+k)}\begin{pmatrix}\boldsymbol{R}_{k,k} \\ \boldsymbol{0}\end{pmatrix}.$$

Let $\bar{\boldsymbol{z}}_1^{r+k} = \boldsymbol{\Theta}^{(r+k)}\boldsymbol{w}_1^{r+k}$. The bound in (64) can now be rewritten as

$$\Pr\left(G\left(\boldsymbol{x}_1^k\right) = 1 \,|\, f_M, \boldsymbol{H}\right) \leq \Pr\left(\left|\bar{\boldsymbol{H}}_{r+k,k}\tilde{\boldsymbol{x}}_1^k + \bar{\boldsymbol{z}}_1^{r+k}\right|^2 - \left|\bar{\boldsymbol{z}}_1^{r+k}\right|^2 < f_M + bk - \left|\boldsymbol{w}_{r+1}^{r+k}\right|^2\right) \tag{65}$$

In (65), $f_M + bk - |\boldsymbol{w}_{r+1}^{r+k}|^2$ can be bounded as

$$f_M + bk - |\boldsymbol{w}_{r+1}^{r+k}|^2$$

$$= \max\left\{bk - \left|\boldsymbol{w}_{r+1}^{r+k}\right|^2, b(k-1) - \left|\boldsymbol{w}_{r+2}^{r+k}\right|^2, \ldots, f_M'\right\} \tag{66}$$

$$\leq \max\{bk, f_M'\} \tag{67}$$

where

$$f_M' = \max\{0, |\boldsymbol{w}_{r+k+1}^{r+k+1}|^2 - b, \ldots, |\boldsymbol{w}_{r+k+1}^{n}|^2 - b(m-k)\}.$$

Let $\beta = \max\{bk, f_M'\}$. Equation (65) can be rewritten as

$$\Pr\left(G\left(\boldsymbol{x}_1^k\right) = 1 \,|\, \beta, \bar{\boldsymbol{H}}\right) \leq \Pr\left(\left|\bar{\boldsymbol{H}}_{r+k,k}\tilde{\boldsymbol{x}}_1^k + \bar{\boldsymbol{z}}_1^{r+k}\right|^2 - \left|\bar{\boldsymbol{z}}_1^{r+k}\right|^2 < \beta\right). \tag{68}$$

Using Chernoff bound, (68) can be written as

$$\Pr\left(G\left(\boldsymbol{x}_1^k\right) = 1 \,|\, q_k, \beta\right) \leq \begin{cases} e^{-(q_k-\beta)^2/(8q_k)}, & q_k > \beta \\ 1, & q_k \leq \beta \end{cases} \tag{69}$$

where $q_k = |\bar{\boldsymbol{H}}_{r+k,k}\tilde{\boldsymbol{x}}_1^k|^2$. Since $q_k = 0$ along the transmitted path, the sum in (62) can be expressed as

$$C_m \leq 1 + \frac{1}{m} E_{q_k,\beta}\left(\sum_{k=1}^{m}\sum_{\left\{\boldsymbol{x}_1^k : \left|\tilde{\boldsymbol{x}}_1^k\right|^2 \neq 0\right\}} \Pr\left(G\left(\boldsymbol{x}_1^k\right) = 1 | q_k, \beta\right)\right) \tag{70}$$

Let $\eta = \frac{\rho}{m}$. Then, for $|\tilde{\boldsymbol{x}}_1^k|^2 \neq 0$, q_k in (69) is a scaled chi-square random variable with $(r+k)$ degrees of

freedom and mean $\frac{\eta|\tilde{\boldsymbol{x}}_1^k|^2}{\alpha}$. Since the three random variables, $\bar{\boldsymbol{H}}_{r+k,k}\tilde{\boldsymbol{x}}_1^k, \bar{\boldsymbol{z}}_1^{r+k}$, and β are independent, averaging (69) over q_k and β gives

$$\Pr\left(G\left(\boldsymbol{x}_1^k\right)=1\right) \le E_\beta\left(\int_0^\beta p(q_k)dq_k + \int_\beta^\infty e^{-(q_k-\beta)^2/(8q_k)}p(q_k)dq_k\right) \tag{71}$$

$$\le \Pr(\beta=bk)\left(\int_0^{bk} p(q_k)dq_k + \int_{bk}^\infty e^{-(q_k-bk)^2/(8q_k)}p(q_k)dq_k\right) + \Pr(\beta>bk). \tag{72}$$

In (72), $\Pr(\beta > bk) = \Pr(f'_M > bk) \le e^{-bk\lambda_0}$ for $b > 1$ (see Appendix E, part A). Note that imposing the condition $b > 1$ allows us to use a reasonably tight bound on $\Pr(f'_M > bk)$ that is independent of m. For $b \le 1$, any tight bound on $\Pr(f'_M > bk)$ must depend on m.[10] The bound in (72) amounts to counting all the nodes $\boldsymbol{x}_1^k$ in the search space when $\beta > bk$. Since $\Pr(\beta > bk)$ decreases sufficiently fast as k increases, this upper bound is still tight for our purposes. For any $\tilde{\boldsymbol{x}}_1^k \neq \boldsymbol{0}$, (72) can be further simplified as

$$\Pr(G\left(\boldsymbol{x}_1^k\right)=1)) \le \int_0^{bk} p(q_k)dq_k + \int_{bk}^\infty e^{-(q_k-bk)^2/(8q_k)}p(q_k)dq_k + e^{-bk\lambda_0} \tag{73}$$

$$\le \int_0^{bk} p(q_k)dq_k + \int_0^\infty e^{-(q_k-bk)^2/(8q_k)}p(q_k)dq_k + e^{-bk\lambda_0} \tag{74}$$

$$\le \gamma\left(\frac{bk\alpha}{2\eta|\tilde{\boldsymbol{x}}_1^k|^2}, \frac{r+k}{2}\right) + e^{bk/4}\int_0^\infty e^{-q_k/8}p(q_k)dq_k + e^{-bk\lambda_0} \tag{75}$$

$$\le \gamma\left(\frac{bk\alpha}{2\eta}, \frac{r+k}{2}\right) + \frac{e^{bk/4}}{\left(1+\frac{\eta}{4\alpha}\right)^{(r+k)/2}} + e^{-bk\lambda_0} \tag{76}$$

with $\gamma(\cdot,\cdot)$ as the incomplete gamma function. Assuming $r \ge 0$, (76) can be bounded as

$$\Pr\left(G\left(\boldsymbol{x}_1^k\right)=1\right) \le \left(\frac{b\alpha}{\eta}e^{1-\frac{b\alpha}{\eta}}\right)^{\frac{k}{2}} + \left(\frac{e^{b/2}}{\left(1+\frac{\eta}{4\alpha}\right)}\right)^{\frac{k}{2}} + e^{-bk\lambda_0} \tag{77}$$

for $\eta > b\alpha$ and any $|\tilde{\boldsymbol{x}}_1^k|^2 \neq 0$. The inequality in (77) follows from (61). From (62) and (77)

$$C_m \le 1 + \sum_{k=1}^m \sum_{\{\tilde{\boldsymbol{x}}_1^k \neq \boldsymbol{0}\}} \left(\frac{b\alpha}{\eta}e^{1-\frac{b\alpha}{\eta}}\right)^{\frac{k}{2}} + \left(\frac{e^{b/2}}{\left(1+\frac{\eta}{4\alpha}\right)}\right)^{\frac{k}{2}} + e^{-bk\lambda_0} \tag{78}$$

$$\le 1 + \frac{1}{m}\sum_{k=1}^m Q^k\left(\left(\frac{b\alpha}{\eta}e^{1-\frac{b\alpha}{\eta}}\right)^{\frac{k}{2}} + \left(\frac{e^{b/2}}{\left(1+\frac{\eta}{4\alpha}\right)}\right)^{\frac{k}{2}} + e^{-bk\lambda_0}\right) \tag{79}$$

$$\le 1 + \frac{1}{m}\left(\frac{1}{1-Q^2\frac{b\alpha}{\eta}e^{1-\frac{b\alpha}{\eta}}} + \frac{1}{1-Q^2\frac{e^{b/2}}{(1+\frac{\eta}{4\alpha})}} + \frac{1}{1-Qe^{-b\lambda_0}}\right) \tag{80}$$

when b and η are sufficiently large, so that all the three sums converge. The inequality in (79) is true, since the number of nodes at level k is Q^k. Since the terms inside the parenthesis in (80) are all independent of m, the number of nodes visited by the stack algorithm scales at most linearly, when $\eta > \eta_0$, where η_0 is the minimum $\frac{\rho}{m}$ ratio required for convergence of the sums in (79). □

[10] Later, we will require a stronger condition on b to guarantee the convergence of the sums in (79).

ACKNOWLEDGMENT

The authors are indebted to Prof. G. David Forney, Jr., for his insight and valuable comments on the paper. The authors also wish to thank the Associate Editor for his diligence, and the anonymous reviewers for their comments and suggestions.

REFERENCES

[1] M. O. Damen, A. Chkeif, and J.-C. Belfiore, "Lattice codes decoder for space-time codes," *IEEE Communi. Lett.*, vol. 4, no. 5, pp. 161–163, May 2000.

[2] L. Babai, "On Lovász lattice reduction and the nearest lattice point problem," *Combinatorica*, vol. 6, no. 1, pp. 1–13, 1986.

[3] G. Foschini, "Layered space-time architecture for wireless communication in a fading environment using multi-element antennas," *Bell Labs Tech. J.*, vol. 1, no. 2, pp. 41–59, 1996.

[4] M. O. Damen, H. El Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2389–2401, Oct. 2003.

[5] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm I. Expected complexity," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pp. 2806–2818, Aug. 2005.

[6] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inf. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.

[7] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inf. Theory*, vol. 45, no. 5, pp. 1639–1642, Jul. 1999.

[8] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Math. Comput.*, vol. 44, pp. 463–471, Apr. 1985.

[9] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Math. Programming*, vol. 66, pp. 181–191, 1994.

[10] J. Jalden and B. Ottersten, "On the complexity of sphere decoding in digital communications," *IEEE Trans. Signal Process.*, vol. 53, no. 4, pp. 1474–1484, Apr. 2005.

[11] 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.

[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] A. R. Hammons Jr. and H. El Gamal, "On the theory of space–time codes for PSK modulation," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 524–542, Mar. 2000.

[14] H.-F. Lu and P. Kumar, "A unified construction of space-time codes with optimal rate-diversity tradeoff," *IEEE Trans. Inf. Theory*, vol. 51, no. 5, pp. 1709–1730, May 2005.

[15] Y. Liu, M. Fitz, and O. Takeshita, "A rank criterion for QAM space–time codes," *IEEE Trans. Inf. Theory*, vol. 48, no. 12, pp. 3062–3079, Dec. 2002.

[16] A. Duel-Hallen and C. Heegard, "Delayed decision-feedback sequence estimation," *IEEE Trans. Commun.*, vol. 37, no. 5, pp. 428–436, May 1989.

[17] J. B. 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.

[18] G. D. Forney Jr., "Coset codes. I. Introduction and geometrical classification," *IEEE Trans. Inf. Theory*, vol. 34, no. 5, pp. 1123–1151, Sep. 1988.

[19] ——, "Coset codes. II. Binary lattices and related codes," *IEEE Trans. Inf. Theory*, vol. 34, no. 5, pp. 1152–1187, Sep. 1988.

[20] H. El Gamal, G. Caire, and M. O. Damen, "Lattice coding and decoding achieve the optimal diversity–multiplexing tradeoff of MIMO channels," *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 968–985, Jun. 2004.

[21] U. Erez and R. Zamir, "Achieving $\frac{1}{2}\log(1 + \mathrm{SNR})$ on the AWGN channel with lattice encoding and decoding," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2293–2314, Oct. 2004.

[22] H. El Gamal and M. O. Damen, "Universal space–time coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 5, pp. 1097–1119, May 2003.

[23] M. O. Damen, H. El Gamal, and N. C. Beaulieu, "Linear threaded algebraic space–time constellations," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2372–2388, Oct. 2003.

[24] J.-C. Belfiore, G. Rekaya, and E. Viterbo, "The Golden code: A 2×2 full-rate space–time code with nonvanishing determinants," *IEEE Trans. Inf. Theory*, vol. 51, no. 4, pp. 1432–1436, Apr. 2005.

[25] B. A. Sethuraman, B. S. Rajan, and V. Shashidhar, "Full-diversity, high-rate space–time block codes from division algebras," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2596–2616, Oct. 2003.

[26] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices, and Groups*, 3rd ed. New York: Springer-Verlag, 1999.

[27] M. O. Damen, H. El Gamal, and G. Caire, "MMSE-GDFE lattice decoding for under-determined linear channels," in *Proc. Conf. Information Science and Systems*, Princeton, NJ, Mar. 2004.

[28] U. Erez, S. Litsyn, and R. Zamir, "Lattices which are good for (almost) everything," in *Proc. IEEE Information Theory Workshop, 2003*, Paris, France, Apr. 2003, pp. 271–274.

[29] J. M. Cioffi, G. P. Dudevoir, M. V. Eyuboglu, and G. D. Forney Jr., "MMSE decision-feedback equalizers and coding. I. Equalization results," *IEEE Trans. Commun.*, vol. 43, no. 10, pp. 2582–2594, Oct. 1995.

[30] H. Cohen, *A Course in Computational Algebraic Number Theory*. New York: Springer-Verlag, 1995.

[31] A. K. Lenstra, A. W. Lenstra Jr., and L. Lovász, "On factoring polynomials with rational coefficients," *Math. Annalen*, vol. 261, pp. 515–534, 1982.

[32] J. Benesty, Y. A. Huang, and J. Chen, "A fast recursive algorithm for optimum sequential signal detection in a BLAST system," *IEEE Trans. Signal Process.*, vol. 51, no. 7, pp. 1722–1731, Jul. 2003.

[33] B. Hassibi, "An efficient square-root algorithm for BLAST," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. II, Istanbul, Turkey, Jun. 2000, pp. 737–740.

[34] E. L. Lawler and D. E. Wood, "Branch and bound methods: A survey," *Oper. Res.*, pp. 14:699–14:719, 1966.

[35] 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.

[36] R. Gowaikar and B. Hassibi, "Efficient statistical pruning for maximum likelihood decoding," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. V, Hong Kong, China, Apr. 2003, pp. 49–52.

[37] J. M. Wozencraft and B. Reiffen, *Sequential Decoding*. New York: MIT Press/Wiley, 1961.

[38] M. Kokkonen and K. Kalliojarvi, "Soft-decision decoding of binary linear codes using the t-algorithm," in *Proc. IEEE 8th Int. Symp. Personal, Indoor and Mobile Radio Communications (PIMRC)*, Finland, Sep. 1997, pp. 1181–1185.

[39] B. 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.

[40] S. Baro, J. Hagenauer, and M. Witzke, "Iterative detection of MIMO transmission using a list-sequential (LISS) detector," in *Proc. IEEE Int. Conf. Communication*, Seattle, WA, May 2003, pp. 2653–2657.

[41] K. S. Zigangirov, "Some sequential decoding procedures," *Probl. Pered. Inform.*, pp. 19–35, Nov. 1966.

[42] F. Jelinek, "A fast sequential decoding algorithm using a stack," *IBM J. Res. Devel.*, vol. 13, pp. 675–685, Nov. 1969.

[43] W. Xu, Y. Wang, Z. Zhou, and J. Wang, "A computationally efficient exact ML sphere decoder," in *Proc. IEEE Global Telecommunications Conf., 2004.*, Dallas, TX, Nov. 2004, pp. 2594–2598.

[44] W. Zhao and G. B. Giannakis, "Sphere decoding algorithms with improved radius search," *IEEE Trans. Commun.*, vol. 53, no. 7, pp. 1104–1109, Jul. 2005.

[45] ——, "Reduced complexity closest point algorithms for random lattices," in *Proc. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Oct. 2003.

[46] J. Geist, "Search properties of some sequential decoding algorithms," *IEEE Trans. Inf. Theory*, vol. IT-19, no. 4, pp. 519–526, Jul. 1973.

[47] R. M. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Trans. Inf. Theory*, vol. IT-9, no. 2, pp. 64–74, Apr. 1963.

[48] H. Yao and G. Wornell, "Lattice-reduction-aided detectors for MIMO communication systems," in *Proc. IEEE Global Conf. Communications*, Taipei, Taiwan, Nov. 2002.

[49] C. Windpassinger and R. Fischer, "Low-complexity near-maximum-likelihood detection and precoding for MIMO systems using lattice reduction," in *Proc. IEEE Information Theory Workshop*, Paris, France, Mar. 2003.

[50] P. Dayal and M. K. Varanasi, "A fast generalized sphere decoder for optimum decoding of under-determined MIMO systems," in *Proc. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Oct. 2003.

[51] M. O. Damen, K. Abed-Meraim, and J.-C. Belfiore, "Generalized sphere decoder for asymmetrical space-time communication architecture," *Electron. Lett.*, vol. 36, p. 166, Jan. 2000.

[52] G. Caire and G. Colavolpe, "On low complexity space-time coding for quasistatic channels," *IEEE Trans. Inf. Theory*, vol. 49, no. 6, pp. 1400–1416, June 2003.

[53] M. Tuchler, R. Koetter, and A. C. Singer, "Turbo equalization: Principles and new results," *IEEE Trans. Commun.*, vol. 50, no. 5, pp. 754–767, May 2002.

[54] R. Johannesson and K. Zigangirov, *Fundamentals of Convolutional Coding*. Piscataway, NJ: IEEE Press, 1999.

[55] R. van Nee, A. van Zelst, and G. Awater, "Maximum likelihood decoding in a space division multiplexing system," in *Proc. IEEE Vehicular Technology Conf.*, Boston, MA, May 2000, pp. 6–10.