Technical Report

UCAM-CL-TR-695
ISSN 1476-2986

Number 695

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Efficient maximum-likelihood decoding of spherical lattice codes

Karen Su, Inaki Berenguer, Ian J. Wassell, Xiaodong Wang

July 2007

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Efficient maximum-likelihood decoding of spherical lattice codes

Karen Su, Inaki Berenguer, Ian J. Wassell and Xiaodong Wang

Abstract

A new framework for efficient and exact Maximum-Likelihood (ML) decoding of spherical lattice codes is developed. It employs a double-tree structure: The first is that which underlies established tree-search decoders; the second plays the crucial role of guiding the primary search by specifying admissible candidates and is our focus in this report. Lattice codes have long been of interest due to their rich structure, leading to numerous decoding algorithms for unbounded lattices, as well as those with axis-aligned rectangular shaping regions. Recently, spherical Lattice Space-Time (LAST) codes were proposed to realize the optimal diversity-multiplexing tradeoff of MIMO channels. We address the so-called boundary control problem arising from the spherical shaping region defining these codes. This problem is complicated because of the varying number of candidates potentially under consideration at each search stage; it is not obvious how to address it effectively within the frameworks of existing schemes. Our proposed strategy is compatible with all sequential tree-search detectors, as well as auxiliary processing such as the MMSE-GDFE and lattice reduction. We demonstrate the superior performance and complexity profiles achieved when applying the proposed boundary control in conjunction with two current efficient ML detectors and show an improvement of 1dB over the state-of-the-art at a comparable complexity.

1 Introduction

The codebook of a lattice code can be described as the intersection of a (infinite) lattice with a bounded shaping region. One of the critical advantages offered by lattice codes is that the algebraic structure of the lattice lends itself to the use of efficient decoding techniques, e.g., *lattice decoding*. Given the received signal vector, a naive lattice decoder returns the closest point of the underlying lattice to that vector, ignoring the shaping region entirely. Therefore its performance can be quite sub-optimal, since it must declare a decoding failure if the closest point that it finds does not lie within the boundaries of the codebook. This loss of optimality becomes increasingly more pronounced as the dimensionality of the problem space grows, since the surface area of the shaping region boundary grows exponentially with dimension, likewise the number of invalid points lying just outside of it.

In the literature, the problem of ensuring that the decoder only considers feasible lattice points, i.e., those lying within the shaping region, is referred to as *boundary control* [1]. It is already well-known how to enforce so-called axis-aligned, or justified, rectangular

boundary control within the context of a tree-based lattice decoder. One of the main contributions of our work is an efficient means of achieving boundary control within this familiar decoding framework when the shaping region is spherical. Spherical lattice codes are an important class of lattice code because their spherical boundaries ensure optimal energy efficiency, i.e., given a fixed lattice, the average energy of a collection of K points is minimized by selecting those contained in a sphere centered at the origin. Specifically, our proposal extends the scope of established tree-based detectors to the Maximum Likelihood (ML) decoding of spherical Lattice Space-Time (LAST) codes at a computational cost comparable to that of naive lattice decoding.

This class of codes is of particular interest because in their seminal work [2], Zheng and Tse characterize the fundamental tradeoff between the diversity and multiplexing gains that can be simultaneously obtained over a given multiple-antenna channel. The authors also pose the then open problem of explicitly constructing coding schemes (for channels more sophisticated than the 2×1 scenario considered in [3]) that achieve the optimal tradeoff curve for any positive multiplexing gain. A solution to their challenge has recently been presented by El Gamal *et al.* in the form of LAST codes [4], which are shown to achieve the optimal diversity-multiplexing tradeoff under generalized minimum Euclidean distance lattice decoding. LAST codes are a recent example of a spherically shaped lattice code and so provide an appropriate setting for demonstrating the utility of the current work.

We begin in Section 2 with an outline of the mathematical structure of the LAST decoding problem. Next we present a generic lattice decoding framework that facilitates the application of existing lattice decoding algorithms to new problems. Specifically we are interested in tackling the ML decoding of spherical LAST codes, which requires the specification of an efficient tree-based boundary control mechanism, as detailed in Section 4. This innovation leads naturally to the development of two new ML LAST decoding schemes, based respectively on the Schnorr-Euchner Adaptive (SEA) sphere decoder and a priority-first tree search (PFTS) approach. Section 5 compares their ML performance and competitive complexities to the profiles of current sub-optimal proposals. Finally, concluding remarks are offered in Section 6.

2 Problem formulation and preliminaries

In this report we consider problems that can be modelled as the minimization of the squared Euclidean distance metric to a *target vector* $\mathbf{v}$ over an m-dimensional *discrete search set* $\mathcal{C} \subset \mathbb{R}^m$:

$$\mathbf{s}_* = \underset{\mathbf{s} \in \mathcal{C}}{\operatorname{argmin}} \, |\mathbf{v} - \mathbf{H}\mathbf{s}|^2, \tag{1}$$

where $\mathbf{v} \in \mathbb{R}^n$, $\mathbf{H} \in \mathbb{R}^{n \times m}$ and the search set is *carved* from an m-dimensional *infinite real lattice* comprising all integer linear combinations of the columns of *generator matrix* $\mathbf{G} \in \mathbb{R}^{m \times m}$

$$\Lambda(\mathbf{G}) \triangleq \{\xi \, : \, \xi = \mathbf{G}\mathbf{z}, \, \mathbf{z} \in \mathbb{Z}^m\} \tag{2}$$

by means of a *translation vector* $\mathbf{u} \in \mathbb{R}^m$ and a *shaping region* $\mathcal{S} \subseteq \mathbb{R}^m$. Note that the shaping region may be bounded or unbounded and is typically convex. The search set is

then given by the intersection of a *translate* of the lattice with the shaping region:

$$\mathcal{C} = (\Lambda(\mathbf{G}) + \mathbf{u}) \cap \mathcal{S}. \tag{3}$$

Thus the decoding problem can be viewed as a constrained closest lattice point search with lattice generator $\mathbf{HG}$, translation vector $\mathbf{Hu}$ and an ellipsoidal shaping region $\mathbf{H}\mathcal{S}$.

To apply integer-based discrete search techniques, we are often interested in centering the lattice $\Lambda(\mathbf{G})$ underlying the search set at the origin. The subscript 0 notation is used to denote entities defined with respect to this frame of reference. For instance, instead of translating the lattice by $\mathbf{u}$ as in (3), we may translate the shaping region by $\mathbf{u}_0 \triangleq -\mathbf{u}$ and make the following alternate definition of the search set:

$$\mathcal{C} \triangleq \mathcal{C}_0 + \mathbf{u} \tag{4}$$

$$\mathcal{C}_0 \triangleq \Lambda(\mathbf{G}) \cap (\mathcal{S} + \mathbf{u}_0)\,. \tag{5}$$

Then minimization problem (1) can be written equivalently as

$$\mathbf{s}_* = \mathbf{u} + \mathbf{G} \operatorname*{argmin}_{\mathbf{z}\in\mathbb{Z}^m} \left\{ \left| \underbrace{\mathbf{v} - \mathbf{Hu}}_{\triangleq \mathbf{v}_0} - \mathbf{\Xi z} \right|^2 \,:\, \mathbf{Gz} \in \mathcal{C}_0 \right\}, \tag{6}$$

where we call $\mathbf{\Xi} \triangleq \mathbf{HG}$ the effective generator matrix of the transformed lattice and search set, seen from the perspective of the received signal space, and borrowing terminology from the optimization literature we call the elements of $\mathbf{z}$ *optimization variables* and $|\mathbf{v}_0 - \mathbf{\Xi z}|^2$ the *cost function*. It is advantageous to consider the minimization problem from the perspective of (6) because then the search set has an underlying Cartesian product structure $\mathbb{Z}^m$ that lends itself easily to *divide and conquer* solution techniques.

We emphasize that the preceding ML decoding formulation is not restricted to space-time systems. Some examples of important and topical communications problems that can be modelled in this manner by appropriately defining parameters $\mathbf{H}$, $\mathbf{G}$, $\mathbf{u}$ and $\mathcal{S}$ include the ML detection of QAM-modulated signals transmitted over MIMO fading channels or in multi-user CDMA systems, *lattice coded* signals transmitted over AWGN channels [5, Ch. 14], over SISO or MIMO fading channels, as well as LAST coded signals transmitted over MIMO fading channels. The fading channels may be flat or frequency selective, and may or may not be quasi-static; problem formulation (1) can be equally applied to these cases.

We assume in this work an over-determined problem, i.e., that $m \leq n$, and that $\mathbf{H}$ is of full rank m. For communication over MIMO fading channels, this assumption means that there are at least as many receive as transmit antennas. In the case where MMSE regularization is being used at the receiver, it has previously been shown that the full (receive) rank tree search techniques considered here are equally applicable to under-determined problems where $m > n$ [6].

We also make use of the following notational conveniences: Given a square $M \times M$ matrix $\mathbf{A}$, let $\mathbf{a}_i$ denote the i^{th} column vector, a_{ii} the element in the i^{th} row and column position, and $\mathbf{A}_{\backslash ii}$ the square submatrix formed by removing the i^{th} row and column. We also denote by $\mathbf{A}^{-T}$ the inverse transpose of matrix $\mathbf{A}$. Note that in the discussions to follow, the inverse transpose operator takes precedence over selection by index, i.e., $\mathbf{A}_i^{-T}$ denotes the i^{th} column of matrix $\mathbf{A}^{-T}$. Given a vector $\mathbf{x}$, let x_i denote the i^{th}

element and $\mathbf{x}_i^j$ the vector formed by extracting elements i to j. Let $\mathbf{0}$, $\mathbf{1}$ and $\mathbf{e}_i$ denote the all-zeros, all-ones and elementary vectors of appropriate length, and $\mathbf{I}_M$ the $M \times M$ identity matrix.

Finally, we introduce some geometric notions that will be important in the discussions to follow. For more convenient visualization, these entites are illustrated in Fig. 1. First we define the *affine sets*

$$\mathcal{F}_j^x(\boldsymbol{\Xi}) \triangleq \left\{\xi \,:\, \left\langle \xi - \boldsymbol{\Xi}_j x, \boldsymbol{\Xi}_j^{-T} \right\rangle = 0\right\}, \qquad x \in \mathbb{Z}, \tag{7}$$

in which the points of lattice $\Lambda(\boldsymbol{\Xi})$ are embedded. Geometrically, $\mathcal{F}_j^x(\boldsymbol{\Xi})$ is a hyperplane defined with respect to normal vector $\boldsymbol{\Xi}_j^{-T}$ and offset x. Algebraically, it contains the subset of lattice points where optimization variable z_j takes a particular value $x \in \mathbb{Z}$. These *affine sub-lattices* may be defined as

$$\Lambda_j^x(\boldsymbol{\Xi}) \triangleq \Lambda(\boldsymbol{\Xi}_{\setminus j}) + \boldsymbol{\Xi}_j x, \qquad x \in \mathbb{Z}, \tag{8}$$

where we refer to $\boldsymbol{\Xi}_j$ and x as the *offset vector* and *offset coefficient*, respectively, of affine sub-lattice $\Lambda_j^x(\boldsymbol{\Xi})$ and affine set $\mathcal{F}_j^x(\boldsymbol{\Xi})$. In particular, observe that $\Lambda_j^x(\boldsymbol{\Xi}) - \boldsymbol{\Xi}_j x$ is nothing more than a finite lattice having one less dimension than $\Lambda(\boldsymbol{\Xi})$.

In addition, the *orthogonal projection* of a vector $\mathbf{y}$ onto affine set $\mathcal{F}_j^x(\boldsymbol{\Xi})$ is defined as

$$\mathrm{proj}_{\mathcal{F}_j^x(\boldsymbol{\Xi})}(\mathbf{y}) \triangleq \mathbf{y} - \frac{\left\langle \mathbf{y}, \boldsymbol{\Xi}_j^{-T} \right\rangle - x}{\left|\boldsymbol{\Xi}_j^{-T}\right|^2} \boldsymbol{\Xi}_j^{-T}, \tag{9}$$

and the corresponding *squared orthogonal distance* as

$$\left\|\mathbf{y} - \mathcal{F}_j^x(\boldsymbol{\Xi})\right\|_\perp^2 \triangleq \min_{\xi \in \mathcal{F}_j^x(\boldsymbol{\Xi})} |\mathbf{y} - \xi|^2 \tag{10}$$

$$= \left|\mathbf{y} - \mathrm{proj}_{\mathcal{F}_j^x(\boldsymbol{\Xi})}(\mathbf{y})\right|^2. \tag{11}$$

It should be clear that $\mathrm{proj}_{\mathcal{F}_j^x(\boldsymbol{\Xi})}(\mathbf{y})$ is the point in the affine set that is closest in Euclidean distance to $\mathbf{y}$.

2.1 Detection of QAM-modulated signals transmitted over fading channels

Next, we briefly illustrate how the (uncoded) MIMO detection problem can be framed in the previously described problem structure. In the flat fading case, the channel can be modelled by an $N \times M$ complex matrix $\mathbf{H}_c$ of fading coefficients, where N and M are the numbers of receive and transmit antennas, respectively. The effective real channel matrix can be expressed as

$$\mathbf{H}_{\mathsf{QAM}} \triangleq \begin{bmatrix} \mathrm{Re}\{\mathbf{H}_c\} & -\mathrm{Im}\{\mathbf{H}_c\} \\ \mathrm{Im}\{\mathbf{H}_c\} & \mathrm{Re}\{\mathbf{H}_c\} \end{bmatrix}, \qquad \mathbf{H}_{\mathsf{QAM}} \in \mathbb{R}^{n_{\mathsf{QAM}} \times m_{\mathsf{QAM}}}, \tag{12}$$

where $n_{\mathsf{QAM}} \triangleq 2N$ and $m_{\mathsf{QAM}} \triangleq 2M$ are the numbers of real received and real transmitted symbols per channel use, respectively. We employ complex B_j^2-QAM modulation on the

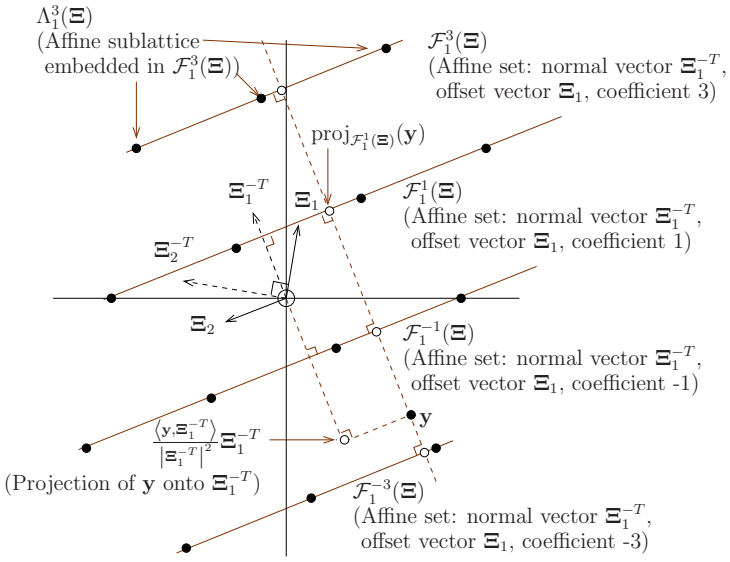


Figure 1: Illustration of some relevant geometric entities with respect to the lattice generated by $\boldsymbol{\Xi}$ and an arbitrary vector $\mathbf{y}$.

j^{th} antenna, i.e., the transmitted symbols are each elements of $B_j \times B_j$ finite rectangular complex-plane lattice-constellations

$$\Lambda_j = \{-B+1, \ldots, -1, +1, +B-1\} \oplus j\{-B+1, \ldots, -1, +1, +B-1\}, \tag{13}$$

where $\oplus$ denotes Minkowski (set) summation. Then the search set from which transmitted signal vectors are assumed to be drawn with equal probability can be written as the M-fold Cartesian product of these rectangular lattices: $\mathcal{C}_{\mathsf{QAM}} = \Lambda_1 \times \cdots \times \Lambda_M$. Equivalently, we can express it in the form of (3) as follows:

$$\mathcal{C}_{\mathsf{QAM}} \triangleq \left(\Lambda(\mathbf{I}_m) + \frac{1}{2}\mathbf{1} \right) \cap \mathcal{S}_{\mathsf{QAM}} \left(\begin{bmatrix} \mathbf{I}_M \\ -\mathbf{I}_M \end{bmatrix}, \begin{bmatrix} \mathbf{b} \\ \mathbf{b} \end{bmatrix} \right), \tag{14}$$

where $\mathbf{b} = \frac{1}{2}[B_1 \ \cdots \ B_M \ B_1 \ \cdots \ B_M]^T \in \mathbb{R}^m$ and the shaping region takes the form of a *(closed) rectangle*

$$\overline{\mathcal{S}}_{\mathsf{QAM}} \left(\begin{bmatrix} \mathbf{I}_M \\ -\mathbf{I}_M \end{bmatrix}, \begin{bmatrix} \mathbf{b} \\ \mathbf{b} \end{bmatrix} \right) \triangleq \left\{ \xi \ : \ \begin{bmatrix} \mathbf{I}_M \\ -\mathbf{I}_M \end{bmatrix} \xi \leq \begin{bmatrix} \mathbf{b} \\ \mathbf{b} \end{bmatrix}, \xi \in \mathbb{R}^m \right\} \tag{15}$$

having sides of length B_j. For the QAM-modulated MIMO detection problem, the generator matrix is $\mathbf{G}_{\mathsf{QAM}} \triangleq \mathbf{I}_m$ and the translation vector is $\mathbf{u}_{\mathsf{QAM}} \triangleq \frac{1}{2}\mathbf{1}$.

In the presence of circularly symmetric additive white (complex) Gaussian noise, the resulting ML detection rule can be written in the form of minimization problem (1), where we denote by $\mathbf{v}_{\mathsf{QAM}}$ the (real) received vector

$$\mathbf{v}_{\mathsf{QAM}} \triangleq \left[\text{Re}\{\mathbf{v}_c\}^T \ \ \text{Im}\{\mathbf{v}_c\}^T \right]^T \tag{16}$$

formed by stacking the real and imaginary components of the complex received signal vector $\mathbf{v}_c$. It can also be written in the alternate form of (6) with the effective generator matrix of the transformed lattice and search set given by $\boldsymbol{\Xi}_{\mathsf{QAM}} = \mathbf{H}_{\mathsf{QAM}}$.

We say that the shaping region is *axis-aligned*, or *justified*, with respect to the code generator matrix $\mathbf{G}_{\mathsf{QAM}}$ because the rows of its normal matrix are multiples of the columns of the dual of the code's generator matrix $\mathbf{G}^{*}_{\mathsf{QAM}} \triangleq \mathbf{G}^{-T}_{\mathsf{QAM}}$. Therefore each facet of shaping region $\overline{\mathcal{S}}_{\mathsf{QAM}}$ is associated with a single optimization variable z_i and containment within its boundary can be verified by considering the particular value taken by each variable independently of the others. As we shall see, a simple decoupling of this form is not possible when the shaping region is spherical, which is the essential cause of increased complexity in the latter case.

2.2 Decoding of spherical LAST codes

The lattice decoding framework presented in this report is primarily demonstrated with reference to the decoding of spherical LAST codes used in a MIMO fading environment. Following the seminal paper on this work [4], next we detail the specific parameterization of this problem.

Lattice Space-Time (LAST) codes are designed for the MIMO fading channel, which can be modelled by an $N \times M$ complex matrix $\mathbf{H}_c$ of fading coefficients and a block length of T channel uses, where N and M are the numbers of receive and transmit antennas, respectively. In the case of quasi-static fading, where the fading coefficients remain unchanged over the duration of the transmission block, the effective real channel matrix can be expressed as a Kronecker product

$$\mathbf{H}_{\mathsf{LAST}} \triangleq \mathbf{I}_T \otimes \begin{bmatrix} \mathrm{Re}\{\mathbf{H}_c\} & -\mathrm{Im}\{\mathbf{H}_c\} \\ \mathrm{Im}\{\mathbf{H}_c\} & \mathrm{Re}\{\mathbf{H}_c\} \end{bmatrix}, \qquad \mathbf{H}_{\mathsf{LAST}} \in \mathbb{R}^{n_{\mathsf{LAST}} \times m_{\mathsf{LAST}}}, \tag{17}$$

where $n_{\mathsf{LAST}} \triangleq 2NT$ and $m_{\mathsf{LAST}} \triangleq 2MT$ are the numbers of real received and real transmitted signals per codeword, respectively. The search set or *codebook* from which transmitted *codewords* $\mathbf{s}$ are drawn with equal probability is given by

$$\mathcal{C}_{\mathsf{LAST}} \triangleq (\Lambda(\mathbf{G}_{\mathsf{LAST}}) + \mathbf{u}_{\mathsf{LAST}}) \cap \overline{\mathcal{S}}_{\mathsf{LAST}}(\mathbf{0}, D), \tag{18}$$

where as suggested by the name given to the codes, the shaping region takes the form of a *(closed) sphere*

$$\overline{\mathcal{S}}_{\mathsf{LAST}}(\mathbf{0}, D) \triangleq \left\{\xi \,:\, |\xi - \mathbf{0}|^2 \leq D,\, \xi \in \mathbb{R}^m\right\} \tag{19}$$

of squared radius D centered at the origin. When it is clear from the context, parameters will be omitted from sets such as Λ and $\overline{\mathcal{S}}$, as well as from entities such as n and m. The specification of the lattice generator matrix $\mathbf{G}_{\mathsf{LAST}} \in \mathbb{R}^{m \times m}$, translation vector $\mathbf{u}_{\mathsf{LAST}} \in \mathbb{R}^m$ and sphere squared radius D comprises the design of a spherical LAST code. Given a selected generator matrix and translation vector, the choice of code radius governs the code rate (i.e., logarithm of the number of codewords in the codebook), as well as the (unnormalized) average power per codeword.

Again assuming circularly symmetric additive white (complex) Gaussian noise, the resulting ML detection rule can be written in the form of minimization problem (1), where we denote by $\mathbf{v}_{\mathsf{LAST}}$ the (real) received vector

$$\mathbf{v}_{\mathsf{LAST}} \triangleq \left[\mathrm{Re}\{\mathbf{v}_c[1]\}^T \;\; \mathrm{Im}\{\mathbf{v}_c[1]\}^T \;\; \cdots \;\; \mathrm{Re}\{\mathbf{v}_c[T]\}^T \;\; \mathrm{Im}\{\mathbf{v}_c[T]\}^T\right]^T \tag{20}$$

formed by stacking the real and imaginary components of the complex signal vectors $\mathbf{v}_c[1], \ldots \mathbf{v}_c[T]$ received during the designated fading block. It can also be written in the alternate form of (6) with the effective generator matrix of the transformed lattice and search set given by $\mathbf{\Xi}_{\mathsf{LAST}} = \mathbf{H}_{\mathsf{LAST}}\mathbf{G}_{\mathsf{LAST}}$.

Observe that in this case, to verify or to disprove membership in the spherical shaping region, a particular vector of the values taken by all of the variables must be evaluated. From a sequential decoding perspective, this characteristic of the problem means that we cannot simply restrict the set of nodes under consideration at each level of the search tree based on the known modulation orders. Thus leading to the challenge that we have addressed in this work of minimizing the number of nodes expanded at each level of the tree while still preserving the desired optimality of the solution.

3 Generic lattice decoding framework

Our goals in this section are twofold: First, we overview a lattice decoding framework that facilitates the application of established decoders to new problems sharing the generic structure of (6). As alluded to in Section 2, it employs a *divide and conquer* approach, recursively decomposing the main minimization into *residual problems* having the same structure, but of decreasing dimension. Secondly, we demonstrate how the so-called *boundary control* problem can be tackled naturally within this framework. We keep our presentation as general as possible, so as not to limit the scope of the framework. In Section 4, we then highlight more specifically its utility in efficiently decoding spherical LAST codes.

We begin by considering the search set from the perspective of the received signal space. In the absence of noise, the observed signals are drawn from a *transformed codebook*, which can be defined as follows with respect to the underlying transformed lattice $\Lambda(\mathbf{\Xi})$ being centered at the origin:

$$\mathcal{T}_0 \triangleq \mathbf{H}\mathcal{C}_0 \tag{21}$$

$$= \Lambda(\mathbf{\Xi}) \cap (\mathbf{H}\mathcal{S} + \underbrace{\mathbf{H}\mathbf{u}_0}_{\triangleq \mathbf{a}_0}). \tag{22}$$

We can then write the optimal cost of (6) as a function having two arguments: the *target* vector $\mathbf{v}_0$ and a *search set* $\mathcal{T}_0$. We remark that both the target and the search set are further embedded in a *search space* $\mathbb{R}^m$ of dimension m

$$g(\mathbf{v}_0, \mathcal{T}_0) = \min_{\upsilon \in \mathcal{T}_0} |\mathbf{v}_0 - \upsilon|^2. \tag{23}$$

Next, we recall that those lattice points $\upsilon \in \mathcal{T}_0$ where variable z_j takes a particular value in $\mathbb{Z}$ are contained in affine set $\mathcal{F}_j^{z_j}(\mathbf{\Xi})$. Therefore we can divide the cost function into two terms: a partial cost incurred by assigning a particular value to z_j and a recursive cost function evaluated over the remaining variables. The first term is precisely the squared distance accumulated by projecting the target onto the affine set $\mathcal{F}_j^{z_j}(\mathbf{\Xi})$ associated with the chosen value of z_j. The second term is a function having the same structure as the original cost function, and so before proceeding we need to specify its target and search set arguments.

To do so, we start with a few observations about the affine set $\mathcal{F}_j^{z_j}(\mathbf{\Xi})$. It is of dimension $m-1$, since it represents the part of the search space that remains after one

variable has been constrained. We may therefore call $\mathcal{F}_1^{z_1}(\boldsymbol{\Xi})$ a *residual search space*. It also follows that the recursive cost function, as well as its arguments, should all be embedded in this residual search space. Hence we define a *residual target* as the projection of the target onto a residual search space

$$\mathbf{v}_0' \triangleq \mathrm{proj}_{\mathcal{F}_1^{z_1}(\boldsymbol{\Xi})}(\mathbf{v}_0), \tag{24}$$

and a *residual search set* as the intersection of the search set with a residual search space

$$\mathcal{T}_0' \triangleq \mathcal{T}_0 \cap \mathcal{F}_1^{z_1}(\boldsymbol{\Xi}). \tag{25}$$

Armed with these notions and definitions, we can then decompose the optimal cost function (23) by decoupling one of the optimization variables from the main problem. Without loss of generality, let $j = 1$, then we can write the following:

$$g(\mathbf{v}_0, \mathcal{T}_0) = \min_{z_1 \in \mathcal{R}_1} \left[\|\mathbf{v}_0 - \mathcal{F}_1^{z_1}(\boldsymbol{\Xi})\|_\perp^2 + g\left(\mathbf{v}_0', \mathcal{T}_0'\right) \right], \tag{26}$$

where $\mathcal{R}_1$ is called the *candidate range* of values for variable z_1 and will be discussed in more detail shortly. This range may also be referred to as the set of *admissible values* in the optimization literature. We say that z_1 has been *decoupled* from the problem because aside from the computation of its arguments, the recursive optimal cost function in the right hand side of (26) is independent of z_1.

In the next stage of the decomposition, another optimization variable is decoupled recursively from the second term of (26) and the dimension of the residual search space is again reduced by one. When all m variables have been decoupled from the problem, the residual search space is of dimension zero and the recursion terminates.

To apply the ideas behind recursive decomposition (26) to lattice decoding, we require efficient means of executing four critical tasks:

1. determining the candidate range for the variable under consideration,
2. finding the orthogonal distance from the residual target to an affine set,
3. computing the projection of the residual target onto an affine set, and
4. constructing the residual search set.

Tasks 2 and 3 can be realized by applying the QR factorization to the effective generator matrix $\boldsymbol{\Xi}$, as is done in many sphere decoders [1, 7–9] and detectors such as V-BLAST [10, 11]. The following sections address Tasks 1 and 4, which may be referred to in the literature collectively as *boundary control*. Please note that to enhance readability, proofs are deferred to the Appendix.

3.1 Determining the candidate range

We determine the candidate range by applying a sort of *relaxation* to the representation of the search set. Instead of considering whether there is at least one point in the *discrete* search set where variable z_j takes a particular value, we consider whether there is at least one point in a *continuous* relaxation of the search set, namely in the shaping region, where variable z_j takes a particular value.

Recall that the affine set $\mathcal{F}_j^x(\boldsymbol{\Xi})$, which contains those translated signal vectors where variable z_j takes a particular value $x \in \mathbb{Z}$, is defined as a hyperplane with normal vector $\boldsymbol{\Xi}_j^{-T}$ and offset x. If the intersection of the shaping region with the affine set is empty for some offset $x \in \mathbb{Z}$, then there are no points in the search set satisfying $z_j = x$ and we say that x is not a *feasible value* for variable z_j.

Note that if, on the other hand, the intersection is non-empty for some offset $x \in \mathbb{Z}$, then there *may or may not* be a point in the search set satisfying $z_j = x$. (A formal proof is provided in the Appendix.) In this case we cannot declare that x is infeasible, and so we call it a *candidate value* and keep it in the search set. Thus we define the *candidate range* for variable z_j as follows:

Definition 1: Given shaping region $\mathcal{S} \subset \mathbb{R}^m$ and generator matrix $\boldsymbol{\Xi} \in \mathbb{R}^{n \times m}$, let the *candidate range* of values for variable z_j be defined as

$$\mathcal{R}_j \triangleq \{x \in \mathbb{Z} : \mathcal{S} \cap \mathcal{F}_j^x(\boldsymbol{\Xi}) \neq \emptyset\}. \tag{27}$$

Because the shaping region is connected, $\mathcal{R}_j$ is a sequence of consecutive integers that can be described by specifying its lower and upper bounds. More precisely, we define the *shadow* of the shaping region on a normal vector:

Definition 2: Given shaping region $\mathcal{S} \subset \mathbb{R}^m$ and normal vector $\mathbf{n} \in \mathbb{R}^m$, let the *(closed) shadow* of $\mathcal{S}$ on $\mathbf{n}$ be defined as the interval

$$\overline{\text{shad}}_{\mathbf{n}}(\mathcal{S}) \triangleq \left[\min_{v \in \mathcal{S}} \langle v, \mathbf{n} \rangle, \max_{v \in \mathcal{S}} \langle v, \mathbf{n} \rangle \right]. \tag{28}$$

The lower and upper bounds of the candidate range $\mathcal{R}_j$ are then given by the ceiling of the lower bound in (28) and the floor of the upper bound in (28), respectively.

3.2 Constructing the residual search set

As before, we approach the task of constructing the residual search set by applying a relaxation to its representation. Instead of trying to obtain a simple concise description of the points in the discrete residual search set where variable z_j takes a particular value, we seek to describe a continuous relaxation of the residual search set, namely a *residual shaping region*, where variable z_j takes a particular value.

Recall from (25) that a residual search set is defined as the intersection of the search set with a residual search space, i.e., with an affine set of the form $\mathcal{F}_j^x(\boldsymbol{\Xi})$. Therefore we can arrive at the desired description by applying the definition directly:

$$\mathcal{T}_0' = \mathcal{T}_0 \cap \mathcal{F}_j^x(\boldsymbol{\Xi}) \tag{29}$$

$$= \Lambda(\boldsymbol{\Xi}) \cap \left[(\mathbf{H}\mathcal{S} + \mathbf{a}_0) \cap \mathcal{F}_j^x(\boldsymbol{\Xi}) \right], \tag{30}$$

where the intersection of the shaping region with the affine set gives the residual shaping region. In this way, the residual search sets can be constructed, and more importantly represented, as the intersections of the transformed lattice $\Lambda(\boldsymbol{\Xi})$ with a residual shaping region, like the definition of the search set itself. This decomposition of the search set into residual sets enables boundary control to be implemented in conjunction with decoding.

3.3 Tree-based lattice decoding

The notion of tree-based lattice decoding arises naturally from the recursive decomposition of (26) and forms the basis for many current detectors, most notably the sphere decoder [1, 8]. We associate with each (residual) problem a node in the tree, starting from the root node, which corresponds to the main search. Next, we select an optimization variable to decouple from the problem, say z_{j_1}. The candidate range $\mathcal{R}_{j_1}$ then provides a superset including all feasible values for z_{j_1}.

Recall from (26) that each candidate value $x \in \mathcal{R}_{j_1}$ generates a partial cost, namely the squared distance from the target to the appropriate affine set, as well as a residual problem having the same structure as the main problem, but of one less dimension. Within the context of the search tree, the size of the candidate range for the next variable to be decoupled $|\mathcal{R}_{j_1}|$ gives the number of children generated by the current node. The weight of the connecting branch to each child is given by the partial cost incurred by assigning a particular value x to variable z_{j_1}, and each child node iteslf is associated with a residual problem.

Continuing in this way, we select subsequent variables to decouple from the residual problems, $z_{j_2}, \ldots, z_{j_m}$, and extend the tree to its full depth of $m+1$ levels. Each leaf node of the tree represents a point in the search set. The corresponding value of the cost function is computed by accumulating the partial costs incurred at each stage of the decomposition, i.e., by computing the sums of the weights of the branches along the path from the root node to the leaf in question. Thus the search tree encapsulates all possible values of the cost function in the weights of its leaf nodes.

We emphasize that although the structure of the tree, i.e., the number of levels and the possibly varying number of children at each node, underlies the decoding operation, only the properties of the root node are known at the outset. Within this context, decomposition (26) becomes a tool for computing the properties of the children of a node, and hence of exploring the tree. We refer to a lattice decoder whose operation is governed by the tree as a *tree-based* lattice decoder. This class includes optimal sphere decoders [1,8] and sub-optimal successive detectors [10, 11], but excludes parallel detection strategies (e.g., linear channel equalization by direct inversion followed by integer quantization).

Because the number of nodes may be exponential in m, it is the task of an efficient Maximum Likelihood (ML) decoder is to find the smallest weight leaf node while exploring as few other nodes as possible. However, we note that there is a problem-dependent minimum set of nodes that must be explored in order to guarantee that the solution returned is optimal. We close our presentation on the generic lattice decoding framework with a brief remark on the node data structure:

The state information maintained at each node must be sufficient to describe the corresponding residual problem. To this end, it must include at least the residual target, its dimension and a description of the residual search set, e.g., for decoding spherical LAST codes, the residual translation vector and squared radius of the shaping region. In addition, redundant computations can be reduced by storing the lower and upper bounds of the candidate range associated with its children, as well as the node's own weight, which is given by the accumulation of the partial costs incurred due to previously constrained variables. A complete description of the proposed data structure is provided in the Appendix.

4 Maximum-likelihood decoding for spherical lattice codes

To see how the previously described framework can be applied to the decoding of spherical LAST codes, let us first consider the graphical view of the LAST decoding problem as shown in Fig. 2. The codebook is illustrated in the form of transformed lattice $\Lambda(\boldsymbol{\Xi})$ and codebook $\mathcal{T}_0$ with ellipsoid shaping region $\overline{\mathcal{E}}(\mathbf{a}_0, \mathbf{H}^{-1}, D)$.[1]

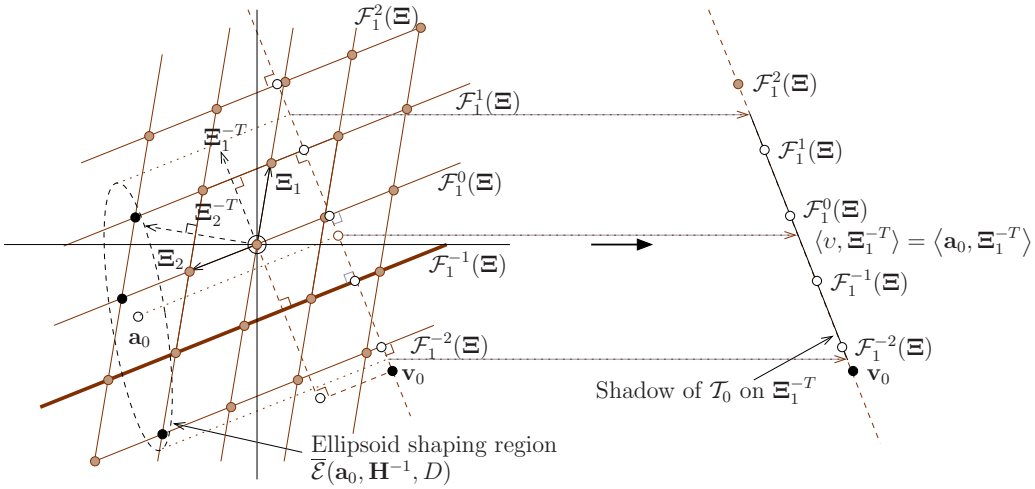


Figure 2: Transformed lattice $\Lambda(\boldsymbol{\Xi})$ and codebook $\mathcal{T}_0$ for LAST decoding problem with $m = 2$ and ellipsoid shaping region $\overline{\mathcal{E}}(\mathbf{a}_0, \mathbf{H}^{-1}, D)$, along with its shadow on $\boldsymbol{\Xi}_1^{-T}$. The affine set $\mathcal{F}_1^{-1}(\boldsymbol{\Xi})$ is highlighted for further commentary.

The corresponding search tree that arises from a decomposition of the cost function for this decoding problem is provided in Fig. 3.

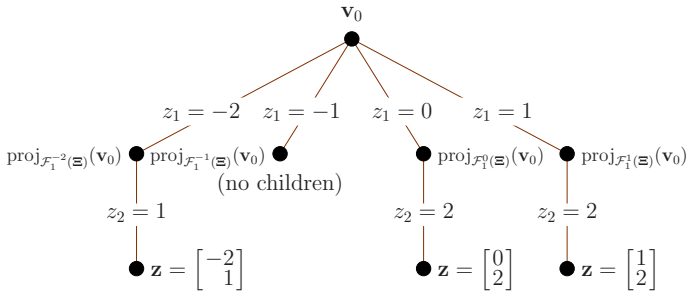


Figure 3: A tree-based decomposition for computing values of the cost function given transformed LAST codebook $\mathcal{T}_0$ shown in Fig. 2.

We note in particular that for a spherical LAST code *each node, even those at the same level, may have a different number of children, or no children at all.* For instance, the node associated with the projection of the target $\mathbf{v}_0$ onto affine set $\mathcal{F}_1^{-1}(\boldsymbol{\Xi})$ illustrates such a case. To see how this sort of scenario can arise from a geometric perspective, consider

[1]The spherical shaping region, transformed by the channel matrix $\mathbf{H}$, becomes an ellipsoidal shaping region from the perspective of the received signal space.

again the codebook depicted in Fig. 2. Observe that although the offset coefficient of the associated affine set, i.e., -1, is in the shadow of the shaping region $\overline{\text{shad}}_{\boldsymbol{\Xi}_1^{-T}}(\overline{\mathcal{E}})$, there are in fact no elements of the codebook $\mathbf{s} = \mathbf{G}\mathbf{z}$ such that $z_1 = -1$. Recall that in this case we call -1 a candidate value for variable z_1, but it is not in fact a feasible value.

To find the desired candidate ranges, we find it computationally simpler to work in the codeword domain, where the shaping region is spherical. A similar result for the case of an ellipsoidal shaping region can also be easily derived from the following:

Proposition 1: Given (closed) sphere $\overline{\mathcal{S}}(\mathbf{u}_0, D) \subset \mathbb{R}^m$ with centre $\mathbf{u}_0 \in \mathbb{R}^m$ and squared radius D, and normal vector $\mathbf{n} \in \mathbb{R}^m$, the shadow of $\overline{\mathcal{S}}$ on $\mathbf{n}$ is given by

$$\overline{\text{shad}}_{\mathbf{n}}\left(\overline{\mathcal{S}}\right) = \left[\langle \mathbf{u}_0, \mathbf{n}\rangle - \sqrt{D}\,|\mathbf{n}|\,,\, \langle \mathbf{u}_0, \mathbf{n}\rangle + \sqrt{D}\,|\mathbf{n}|\right]. \tag{31}$$

Having determined the candidate range for the variable under consideration, application of the tree-based lattice decoding framework involves specifying rules for computing the properties of a child node from those of its parent, or equivalently, the parameters of a residual problem from those of its parent. As discussed previously, a key ingredient in these derivations is the affine set associated with the variable under consideration and the value to which it is being constrained. Given this affine set, the residual target is then the projection of the target onto it, the partial cost (or branch weight) is the orthogonal distance from the target to the affine set, and again working in the codeword domain, the following result enables us to easily compute the parameters of the residual search set:

Proposition 2: Given (closed) sphere $\overline{\mathcal{S}}(\mathbf{u}_0, D) \subset \mathbb{R}^m$ with centre $\mathbf{u}_0 \in \mathbb{R}^m$ and squared radius D, normal vector $\mathbf{n} \in \mathbb{R}^m$ and offset $b \in \mathbb{R}$, the intersection of $\overline{\mathcal{S}}$ with hyperplane

$$\mathcal{P}(\mathbf{n}, b) \triangleq \{\upsilon \,:\, \langle \upsilon, \mathbf{n}\rangle = b\} \tag{32}$$

is an $(m-1)$-dimensional sphere that can be written as $\overline{\mathcal{S}}(\mathbf{u}_0', D') \cap \mathcal{P}(\mathbf{n}, b)$ where centre

$$\mathbf{u}_0' = \text{proj}_{\mathcal{P}(\mathbf{n},b)}(\mathbf{u}_0) \tag{33}$$

and squared radius

$$D' = D - \|\mathbf{u}_0 - \mathcal{P}(\mathbf{n}, b)\|_{\perp}^2 \tag{34}$$

Proposition 2 allows us to construct the residual search set or *residual codebook* when decoding spherical LAST codes by means of the same two parameters used in the definition of the codebook itself, namely the *residual translation vector* $\mathbf{u}_0'$ of the shaping region and its *residual squared radius* D'.

4.1 Priority First Tree Search LAST Decoder

The Priority First Tree Search LAST (PFTS-L) decoder maintains an ordered list of nodes $\mathcal{N}_b$ defining the border between the explored and unexplored parts of the tree, which initially contains only the root node. In each iteration, it selects and *expands* the border node with the smallest weight. The expanded node is then deleted from $\mathcal{N}_b$, since it is no longer on the border, and replaced by two new ones: its *first child* and its *next sibling*. Traversal of the tree continues in this *priority-first* fashion until a leaf node is

encountered. Spherical boundary control as detailed in this report is employed by the PFTS-L decoder.

By definition and as suggested by its name, the PFTS-L explores the nodes of the search tree in order of increasing node weight. Therefore when the smallest weight border node is a leaf, it returns the corresponding point in the search set $\mathbf{z}$, along with its weight σ, and terminates. Pseudocode for the PFTS LAST decoder is provided for the interested reader in the Appendix and a Matlab implementation of the decoder can be found at the Matlab Central File Exchange [12].

To isolate and establish the benefits afforded by spherical boundary control, the behaviour of the PFTS-L decoder may be benchmarked against its *naive* predecessor called PFTS-N. This standard priority-first lattice decoder does not implement any boundary control. It simply returns the closest lattice point to the target, which may or may not be a valid codeword.

Because of its ordered traversal of the search tree, the priority-first approach requires the maintenance of a priority queue that is able to return the smallest weight leaf node quickly. It turns out that such a queue can be efficiently implemented using a systolic hardware architecture [13]; this design is able to fetch the smallest weight leaf node in constant time. In addition, because of its node expansion philosophy, the PFTS-L is able to return not only the closest lattice point, but also the next closest and subsequent lattice points, again in order of increasing distance from the target. Thus making it particularly suitable for dealing with punctured codebooks as discussed in Remark 2.

4.2 Schnorr-Euchner Adaptive LAST sphere decoder

The SEA LAST (SEA-L) sphere decoder applies the same *depth-first* approach as its counterpart for uncoded MIMO fading channels, which is studied in [1]. Pseudocode for the LAST SEA decoder is provided for the interested reader in the Appendix and a Matlab implementation of the decoder can be found at the Matlab Central File Exchange [14].

A primary benefit of the depth-first approach is that only a fixed amount of memory needs to be allocated for the decoding stage. However, because its underlying philosophy is one of enumeration of the lattice points lying within some distance of the target, it is only suitable for certifying the optimality of the closest lattice point. It may or may not be possible to certify that of the next closest or subsequent points. Like the priority-first approach, the SEA-L requires a variable amount of runtime.

We close our discussion on these proposed lattice decoders for spherical LAST codes with some important remarks on the optimality properties of the algorithms and on their efficient implementation through the QR factorization of the code generator matrix $\mathbf{G}$.

Remark 1: (On axis-aligned sphere decoding) We can contrast the search tree of Fig. 3 to the *complete* $(m+1)$-level B-ary tree that underlies the operation of a standard sphere decoder. Then we find that the sphere decoder can be described as a lattice decoder employing *axis-aligned rectangular* boundary control, which results in ML decoding for lattice codes whose shaping regions share this structure. The axis-aligned property means that each variable z_j takes values in some *fixed* alphabet, independently of the values taken by variables z_i, $i \neq j$. Equivalently, from a tree-based decoding perspective, each node has the same number of children, corresponding to the cardinality B of the alphabet. Although the standard algorithm can easily be extended to allow the alphabet associated

with each variable to vary on a global scale, it cannot be trivially modified to incorporate dependence on the values taken by the other variables, as is necessary to perform efficient decoding over more general search spaces.

Remark 2: (On decoder optimality) To achieve the desired code rate in the LAST code design procedure, it is sometimes the case that the codebook $\mathcal{C}$ is actually a proper subset of $(\Lambda(\mathbf{G})+\mathbf{u})\cap\overline{\mathcal{S}}(\mathbf{0}, D)$. More specifically, the code may be punctured by removing some so-called *outer shell* codewords, i.e., those whose 2-norms satisfy $|\mathbf{s}|^2 > D - \epsilon$, from the codebook. In this case the closest feasible lattice point returned by the proposed detectors may not actually be an element of the punctured codebook. For such codes, the performance of the SEA LAST decoder is slightly sub-optimal. However, the decoding error can be detected by appending a test for codebook membership to the end of the decoder. Such a test can be implemented far more efficiently than a brute force exhaustive membership check, since only those codewords in the outer shell need to be considered.

On the other hand, because of its ordered traversal approach, the PFTS is able to achieve true ML performance. If the codebook membership test fails, then it can continue searching for the next closest lattice point(s), until it finally returns a point that is in the punctured codebook.

Remark 3: (On efficient implementation) In an efficient implementation, we pre-process the code generator matrix $\mathbf{G}$ by applying a QR factorization to obtain orthogonal matrix $\mathbf{Q_G}$ and upper triangular equivalent transform matrix $\mathbf{P}$. The translation vector of the shaping region is orthogonally transformed to $\widetilde{\mathbf{u}} \triangleq \mathbf{Q}_{\mathbf{G}}^T\mathbf{u}_0$ and the candidate range for the first variable to be constrained z_m is then given by

$$\mathcal{R}_m = \left\{ \left\lceil \frac{\widetilde{u}_m}{p_{mm}} - \frac{\sqrt{D}}{|p_{mm}|} \right\rceil, \ldots, \left\lfloor \frac{\widetilde{u}_m}{p_{mm}} + \frac{\sqrt{D}}{|p_{mm}|} \right\rfloor \right\}, \tag{35}$$

Equation (35) takes such a simple form because $\mathbf{P}$ and $\mathbf{P}^{-1}$ are upper triangular, and so $\mathbf{P}^{-T}$ is lower triangular and $\mathbf{P}_m^{-T} = \frac{1}{p_{mm}}\mathbf{e}_m$.

It is important to observe that it is entirely possible for the lower bound of the integer set in (35) to be greater than its upper bound. In this case we make the logical interpretation that $\mathcal{R}_j = \emptyset$. From an algebraic perspective, this scenario corresponds precisely to the case where a node, having $\mathcal{R}_j$ as the candidate range of values for its child nodes, does not generate any children.

The simplification offered by the upper triangular form of equivalent generator matrix $\mathbf{P}$ also extends to the computation of the parameters of the residual search sets. For the first variable to be constrained z_m the residual translation vector and residual squared radius are as follows:

$$\widetilde{\mathbf{u}}' = \begin{bmatrix} \widetilde{\mathbf{u}}_{\backslash m} \\ p_{mm}x \end{bmatrix} \tag{36}$$

$$D' = D - (\widetilde{u}_m - p_{mm}x)^2, \tag{37}$$

where $x \in \mathcal{R}_m$ is the value under consideration.

5 Performance and complexity results

In this section we discuss the performance and complexity profiles arising from two spherical LAST codes that have been designed for the $M = N = T = 2$ quasi-static Rayleigh

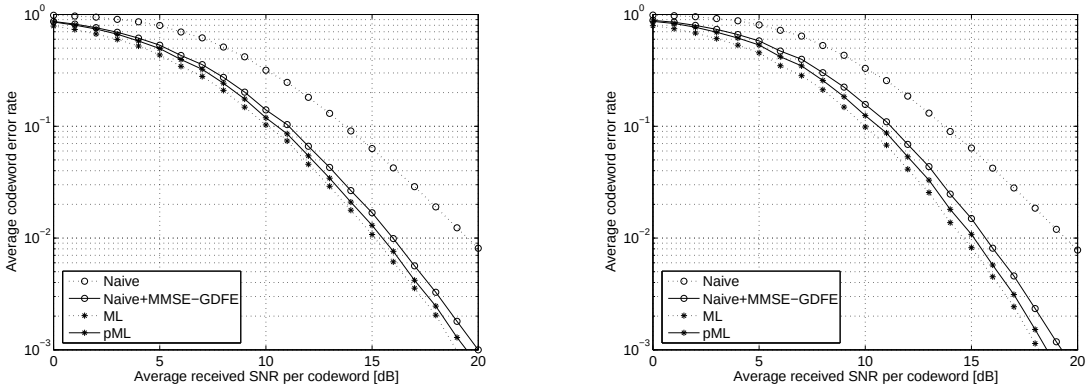
flat fading channel scenario. Their underlying lattices are generated by the (A) randomly chosen [4] and (B) minimum error rate [15] generator matrices reported in the literature and their codebooks contain 256 codewords each, for a communication rate of $4\frac{\text{bits}}{\text{channel use}}$.

We present selected performance results, given by average codeword error rates, and complexity results, given by average complexity and node exploration exponents, obtained using a number of decoding strategies derived from the ideas detailed in this report. In particular, we consider the behaviours of naive lattice decoders operating without any boundary control and LAST decoders using spherical boundary control, based on both the SEA and PFTS tree search strategies, all with and without the well-known complexity and/or performance benefits afforded by the Minimum Mean Squared Error Generalized Decision Feedback Equalizer (MMSE-GDFE) and Lattice-Reduction Aided (LRA) detection. The specific configurations of the decoders are summarized in Table 1.

Decoder label	**MMSE-GDFE front end**	**LRA detection**	**Boundary control**	**Memory [nodes]**	**Decoder optimality**
SEA-N (Naive)	no	no	none	$m = 2MT$	sub-optimal
SEA-N+MMSE-GDFE	yes	no	none	m	sub-optimal
PFTS-N	no	no	none	∞	sub-optimal
SEA-L	no	yes/no	spherical	m	ML
SEA-L+MMSE-GDFE	yes	yes/no	spherical	m	pseudo-ML (pML)
PFTS-L	no	yes/no	spherical	∞	ML
PFTS-L+MMSE-GDFE	yes	yes/no	spherical	∞	pseudo-ML (pML)
PFTS-L+MMSE-GDFE+LRA	yes	yes	spherical	τ	sub-optimal

Table 1: Summary of principal characteristics of lattice decoders under consideration.

First we consider in Fig. 4 the average codeword error rates attained under various decoding strategies:



(a) LAST code from [4].

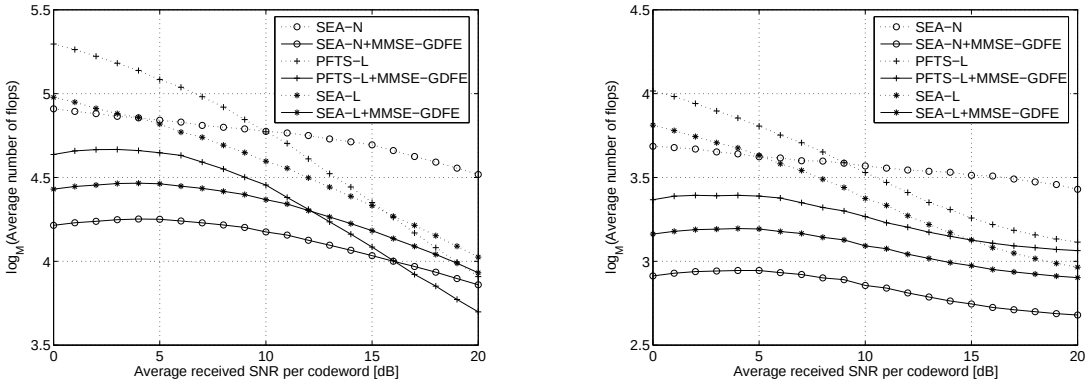
(b) LAST code designed for minimum error rate [15].

Figure 4: Average codeword error rate vs. average received SNR per codeword for the (sub-optimal) naive lattice and ML LAST decoders, shown both with and without the MMSE-GDFE front end.

We may immediately observe a large gap, e.g., of almost 5dB at a target codeword error rate of 10^{-2}, between the performance of the naive lattice decoder without boundary control and the ML curve. Although a part of this gap is recovered by the efficient naive decoder using MMSE-GDFE pre-processing (but no boundary control) proposed in [4], we see that there remains about 1dB of room for improvement. Also shown is the gap between the various error rates and the outage curve, i.e., the probability that the channel is too singular to support communication at the design rate. A gain of 1dB when performance is so close to outage is particularly difficult to obtain.

This gap can be completely closed by the proposed ML detectors, or roughly halved by applying the proposed ML strategies, i.e., with boundary control, to the problem following MMSE-GDFE pre-processing. The resulting *pseudo-ML* performance curve is labelled "p-ML" in the plots. By carefully comparing Figs. 4(a) and 4(b), we also see that the minimum error rate spherical LAST code achieves a slightly better performance profile than the unoptimized code. This benefit persists for all of the detection algorithms.

Next, Fig. 5 compares the average complexity exponents of the detectors under consideration. Observe that the complexities of the SEA-L decoders, both with and without MMSE-GDFE pre-processing, are slightly higher but comparable to that of the naive lattice decoder with the MMSE-GDFE front end. This increase in computational cost is consistent with the improved performance profile that is offered by the SEA-L decoders.



(a) LAST code from [4].

(b) LAST code designed for minimum error rate [15].

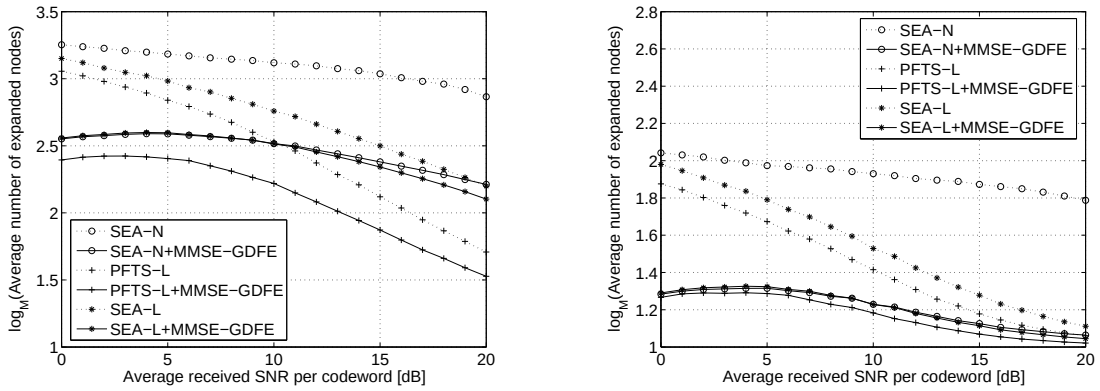
Figure 5: Average complexity exponent vs. average received SNR per codeword for the (sub-optimal) naive lattice, and ML SEA-L and PFTS-L decoders, shown both with and without the MMSE-GDFE front end.

More interestingly, our simulations reveal an almost full unit gap between the complexity exponents attained when using the two different LAST codes, a phenomenon that appears to persist broadly over all detection strategies.

In addition to the complexity exponents of the various decoders, it is also insightful to investigate their node exploration exponents. By comparing the curves shown in Fig. 6 obtained using the SEA-N, SEA-L and SEA-N+MMSE-GDFE decoders, we observe that the naive lattice decoders actually expand the largest numbers of nodes. This effect can be mitigated at lower SNRs by MMSE regularization alone via the MMSE-GDFE

front end, or at relatively high SNRs by boundary control alone. Recall that there is a 1dB performance penalty incurred when using MMSE regularization in this scenario. Considering the curves labeled SEA-N+MMSE-GDFE and SEA-L+MMSE-GDFE, we see that combining boundary control and MMSE regularization at higher SNRs yields additional benefit.

While the curves presented in Fig. 6 have the same shapes as their complexity exponent counterparts in Fig. 5, we also note that a significant further reduction in node exploration requirements can be obtained through the use of a priority-first PFTS-L strategy compared to the depth-first SEA-L. The priority-first schemes enjoy an almost constant reduction in their exploration exponents, across all SNRs, due to the exclusion of the operations relating to the maintenance of an ordered nodelist. These curves are indicative of the complexity behaviour of hardware implementations of the decoders, since the operations associated with expanding a single node in either case (including partially sorting the nodelist and fetching the smallest weight node) can be executed by specialized hardware in a single clock-cycle [13, 16].



(a) LAST code from [4].

(b) LAST code designed for minimum error rate [15].

Figure 6: Average node exploration exponent vs. average received SNR per codeword for the (sub-optimal) naive lattice, and ML SEA-L and PFTS-L decoders, shown both with and without the MMSE-GDFE front end.

In the implementation whose exploration exponents are shown in Fig. 6, we use a naive lattice decoder based on the Schnorr-Euchner enumeration with adaptive radius reduction. By definition it must expand at least as many nodes as an equivalent naive implementation based on priority-first search, i.e., $\nu_{\mathsf{SEA-N}} \geq \nu_{\mathsf{PFTS-N}}$. However, we observe that the number of nodes expanded by the naive SEA-based decoder is often *larger* than that explored by the PFTS-L decoder. In fact, we can show that its lower bound, $\nu_{\mathsf{PFTS-N}}$, may also be larger than the number of nodes explored by the PFTS-L strategy.

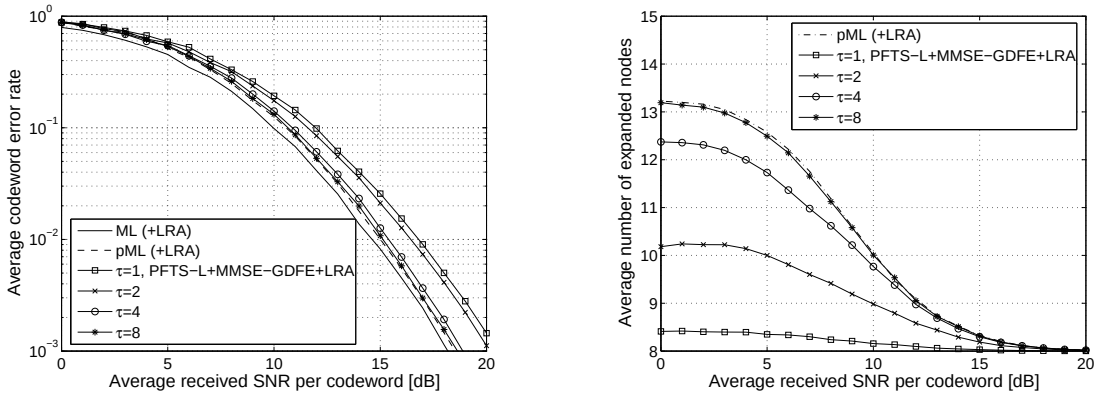
Proposition 3: Given target $\mathbf{v}$ and lattice decoding problem parameters $\mathbf{H}$, $\mathbf{G}$, $\mathbf{u}$ and $\mathcal{S}$, as well as naive and ML lattice decoders PFTS-N and PFTS-L employing ordered traversal, the following statements hold:

1. The squared search radii C^2 of the two decoders satisfy $C^2_{\mathsf{PFTS-N}} \leq C^2_{\mathsf{PFTS-L}}$.

2. The numbers of nodes ν expanded by the two decoders do not necessarily satisfy $\nu_{\mathsf{PFTS-N}} \leq \nu_{\mathsf{PFTS-L}}$.

The result, which may seem counter-intuitive at first, arises because the PFTS-L decoder only expands those nodes whose associated residual targets lie both inside the search sphere, as well as on affine sets that correspond to candidate values of the optimization variables. Therefore, in many (but clearly not all) cases it actually expands fewer nodes than the naive PFTS-N.

As is to be expected, any reduction in the number of nodes explored by the PFTS-L decoder comes at the cost of additional space complexity. Because of the recursive nature of the SEA-L, the number of nodes that it maintains at any time is upper bounded by the height of the search tree, i.e., its space complexity (in node data structures) is upper bounded by m. On the other hand, the true space requirements of the PFTS-L depend on number of nodes expanded at runtime. In a practical implementation, a finite priority queue of fixed size τ would be used, the size of which may be traded-off against some cost to performance as illustrated in Fig. 7. We observe that as $\tau \to m$, both the performance and complexity characteristics of the PFTS-L decoder approach that of the SEA-L.



(a) Average codeword error rate vs. average received SNR per codeword.

(b) Average number of expanded nodes vs. average received SNR per codeword.

Figure 7: Performance and complexity profile attained by practical fixed memory PFTS-L decoders, which make use of both the MMSE-GDFE front end and LRA detection, shown here for a minimum error rate LAST code [15].

Finally, it is interesting to highlight the role that LRA detection can play alongside the decoding schemes considered in this section. For the algorithms achieving ML or pML performance, spherical boundary control is necessarily being enforced through the parallel tree decoder structure. Therefore, applying lattice reduction does not affect their performance at all; it only has an impact on their complexity, the nature and extent of which is illustrated in Fig. 8.

Observe that the MMSE-GDFE front end plays a very significant role in the lower SNR regime, whereas the complexity reduction supported through LRA detection becomes relatively more significant in the higher SNR regime. Debatably, at such point we may no longer be so interested in this reduction, as even the unassisted detector's complexity

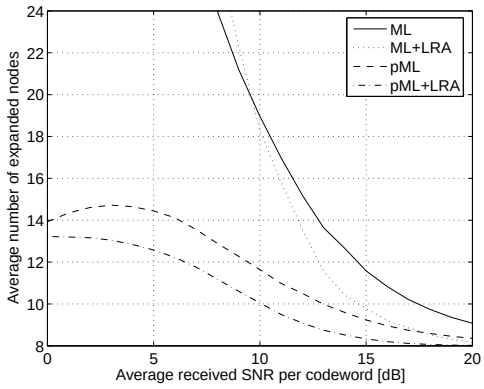


Figure 8: Average number of expanded nodes vs. average received SNR per codeword for PFTS-L decoder, both with and without the MMSE-GDFE front end and with and without LRA detection, shown here for a minimum error rate LAST code [15].

will have nearly converged to its minimum, on average. What we do note is that the combination of the two pre-processing techniques affords the greatest benefit, i.e., their effects are complementary and cumulative. We also remark that LRA detection has been observed to play a more significant role in the performance of sub-optimal schemes, such as the fixed memory version of the PFTS-L decoder. Thus we may conjecture that LRA detection is a more critical consideration for performance enhancement of sub-optimal schemes than it is for complexity reduction of optimal decoding strategies.

6 Conclusions

In this report we have presented a generic framework for the efficient ML decoding of spherical lattice codes. Specifically we apply it to the spherical LAST codes pioneered in [4] and demonstrate that a 1dB improvement in performance over the naive decoder with MMSE-GDFE pre-processing is available at a comparable complexity. Within our framework, the problem of boundary control is handled naturally, alongside the decoding process, by means of a parallel tree-based structure. In fact, for delay-limited rather than processing-limited applications, traversal of the two trees can be done in parallel, as there is no data-dependence between them. Such a strategy would further reduce the required computation time. We have demonstrated the performance and complexity profiles of various flavours of decoders making use of the proposed spherical boundary control mechanism through extensive simulation, including consideration of the isolated and joint effects of MMSE-GDFE pre-processing, LRA detection and fixed space complexity constraints.

Appendix

Proof of Proposition 1

Proposition 1: Given (closed) sphere $\overline{\mathcal{S}}(\mathbf{u}_0, D) \subset \mathbb{R}^m$ with centre $\mathbf{u}_0 \in \mathbb{R}^m$ and squared radius D, and normal vector $\mathbf{n} \in \mathbb{R}^m$, the shadow of $\overline{\mathcal{S}}$ on $\mathbf{n}$ is given by

$$\overline{\text{shad}}_{\mathbf{n}}\left(\overline{\mathcal{S}}\right) = \left[\langle \mathbf{u}_0, \mathbf{n}\rangle - \sqrt{D}\,|\mathbf{n}|\,,\, \langle \mathbf{u}_0, \mathbf{n}\rangle + \sqrt{D}\,|\mathbf{n}|\right]. \tag{38}$$

Proof We consider first the upper bound of the shadow:

$$\max_{\upsilon \in \overline{\mathcal{S}}} \langle \upsilon, \mathbf{n}\rangle = \max_{\upsilon \in \mathbb{R}^m} \left\{ \left\langle \mathbf{u}_0 + \sqrt{D}\upsilon, \mathbf{n} \right\rangle \,:\, |\upsilon| = 1 \right\} \tag{39}$$

$$= \langle \mathbf{u}_0, \mathbf{n}\rangle + \sqrt{D}\left\langle \frac{\mathbf{n}}{|\mathbf{n}|}, \mathbf{n} \right\rangle \text{ (by Cauchy-Schwarz inequality)} \tag{40}$$

$$= \langle \mathbf{u}_0, \mathbf{n}\rangle + \sqrt{D}|\mathbf{n}|. \tag{41}$$

The analogous result for the lower bound can be shown in a similar manner. ∎

Proof of Proposition 2

Proposition 2: Given (closed) sphere $\overline{\mathcal{S}}(\mathbf{u}_0, D) \subset \mathbb{R}^m$ with centre $\mathbf{u}_0 \in \mathbb{R}^m$ and squared radius D, normal vector $\mathbf{n} \in \mathbb{R}^m$ and offset $b \in \mathbb{R}$, the intersection of $\overline{\mathcal{S}}$ with hyperplane

$$\mathcal{P}(\mathbf{n}, b) \triangleq \{\upsilon \,:\, \langle \upsilon, \mathbf{n}\rangle = b\} \tag{42}$$

is an $(m-1)$-dimensional sphere that can be written as $\overline{\mathcal{S}}(\mathbf{u}_0', D') \cap \mathcal{P}(\mathbf{n}, b)$ where centre

$$\mathbf{u}_0' = \text{proj}_{\mathcal{P}(\mathbf{n},b)}(\mathbf{u}_0) \tag{43}$$

and squared radius

$$D' = D - \|\mathbf{u}_0 - \mathcal{P}(\mathbf{n}, b)\|_\perp^2 \tag{44}$$

Proof We begin by recalling the definition of a sphere:

$$\overline{\mathcal{S}} = \{\xi \,:\, |\xi - \mathbf{u}_0|^2 \leq D,\, \xi \in \mathbb{R}^m\}. \tag{45}$$

The intersection of a sphere and a hyperplane is clearly a lower-dimensional sphere. It remains to determine the precise values of the solution sphere's centre and squared radius parameters. By symmetry, the centre of the solution sphere must be of the form $\mathbf{u}_0 + \alpha\mathbf{n}$ for some scaling factor α. In particular, let $\upsilon = \xi - \mathbf{u}_0 = \upsilon_\parallel + \upsilon_\perp$ such that $\langle \upsilon_\perp, \mathbf{n}\rangle = 0$. Then it follows that $\langle \xi, \mathbf{n}\rangle = \langle \mathbf{u}_0, \mathbf{n}\rangle + \langle \upsilon_\parallel, \mathbf{n}\rangle = b$. Therefore the component of υ that is parallel to normal vector $\mathbf{n}$ has a fixed 2-norm of

$$|\upsilon_\parallel| = \frac{b - \langle \mathbf{u}_0, \mathbf{n}\rangle}{|\mathbf{n}|}. \tag{46}$$

The centre of the solution sphere is then

$$\mathbf{u}_0' = \mathbf{u}_0 + |\upsilon_\parallel|\frac{\mathbf{n}}{|\mathbf{n}|} \tag{47}$$

and since we also have that $|v|^2 = |v_\parallel|^2 + |v_\perp|^2 \leq D$, its squared radius is given by the upper bound on the squared 2-norm of the component of v that is orthogonal to normal vector $\mathbf{n}$, i.e., that lies within hyperplane $\mathcal{P}(\mathbf{n}, b)$:

$$|v_\perp|^2 \leq D - |v_\parallel|^2. \tag{48}$$

■

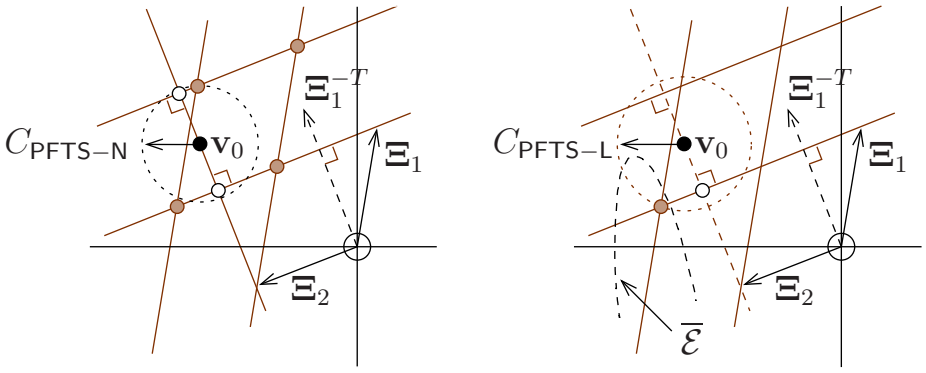
Proof of Proposition 3

Proposition 3: Given target $\mathbf{v}$ and lattice decoding problem parameters $\mathbf{H}$, $\mathbf{G}$, $\mathbf{u}$ and $\mathcal{S}$, as well as naive and ML lattice decoders PFTS-N and PFTS-L employing ordered traversal, the following statements hold:

1. The squared search radii C^2 of the two decoders satisfy $C^2_{\mathsf{PFTS-N}} \leq C^2_{\mathsf{PFTS-L}}$.
2. The numbers of nodes ν expanded by the two decoders do not necessarily satisfy $\nu_{\mathsf{PFTS-N}} \leq \nu_{\mathsf{PFTS-L}}$.

Proof Because the search set underlying the operation of the PFTS-L is a subset of that of the PFTS-N, it follows that the optimal cost returned by the PFTS-N is less than or equal to that returned by the PFTS-L. This optimal cost is precisely the squared search radius C^2.

Intuition might suggest at this point that the number of nodes expanded by the two decoders should also obey the same inequality relationship. However, whereas the PFTS-N expands all nodes associated with residual targets located within its (open) search sphere $\mathcal{S}(\mathbf{v}_0, C^2_{\mathsf{PFTS-N}})$, the PFTS-L only expands nodes associated with admissible candidates, or equivalently those lying within the intersection of its (open) search sphere $\mathcal{S}(\mathbf{v}_0, C^2_{\mathsf{PFTS-L}})$ and the collection of infinite strips defined by the shadows of the code shaping region $\mathcal{S}$ on the problem-dependent normal vectors.



(a) By the PFTS-N, $\nu_{\mathsf{PFTS-N}} = 3$.

(b) By the PFTS-L, $\nu_{\mathsf{PFTS-L}} = 2$.

Figure 9: Nodes expanded by PFTS naive and ML lattice decoders for LAST decoding problem shown in Fig. 2, where the target is shown as a filled dot, residual targets as empty dots and feasible points in the search set as gray dots.

A simple counterexample where $\nu_{\mathsf{PFTS-N}} > \nu_{\mathsf{PFTS-L}}$ is provided in Fig. 9: The smaller search sphere of the PFTS-N shown in Fig. 9(a) contains three (residual) targets, while the intersection of the larger search sphere of the PFTS-L and the appropriate shaping region shadows shown in Fig. 9(b) contains only two. ■

Pseudocode for the SEA and PFTS LAST decoders

Finally, we present complete detailed pseudocode descriptions of two LAST decoders based on the SEA and PFTS approaches. We assume that the (square) upper triangular transform matrices $\underline{\mathbf{R}}$ and $\mathbf{P}$, arising from the QR factorizations of transformed lattice generator $\boldsymbol{\Xi}$ and of the code lattice generator $\mathbf{G}$, respectively, and the border nodelist $\mathcal{N}_b$ are available throughout as global variables. The node data structure is defined as an 11-tuple comprising

- its weight σ in $\mathbb{R}_{\geq 0}$,
- its parent dimension d' in $\mathcal{I}_m$,
- its parent residual target vector $\widetilde{\mathbf{y}}'$ in $\mathbb{R}^{d'}$,
- its associated vector of applied constraint values $\check{\mathbf{z}}$ in $\mathbb{Z}^{m-d'+1}$,
- its position with respect to its siblings q in $\mathbb{Z}_{>0}$,
- the weight of its parent node σ' in $\mathbb{R}_{\geq 0}$,
- the constraint value of its previous sibling x^- in $\mathbb{R}$,
- its parent residual translation vector $\widetilde{\mathbf{u}}'$ in $\mathbb{R}^{d'}$,
- its parent residual squared radius D' in $\mathbb{R}_{\geq 0}$,
- the lower bound of its candidate range $x_{\min}$ in $\mathbb{Z}$, and
- the upper bound of its candidate range $x_{\max}$ in $\mathbb{Z}$.

The SEA LAST decoder makes use of the same FirstChild-L and NextSibling-L functions as the PFTS LAST decoder. However, instead of maintaining the nodes in a heap (or other data structure of choice), the SEA expands them recursively. We assume that the functions are appropriately modified to return the computed child and sibling node data structures to RecursiveExpand-L. Fixed memory implementations of the PFTS-L and SEA-L lattice decoders with spherical boundary control can be found at the Matlab Central File Exchange (please see [12] and [14], respectively).

Algorithm 1 Priority-First Tree Search LAST Decoder PFTS-L$(\mathbf{v}, \mathbf{H}, \mathbf{G}, \mathbf{u}, D)$

Input: The target vector $\mathbf{v}$, the channel matrix $\mathbf{H}$, the lattice generator matrix $\mathbf{G}$, its translation vector $\mathbf{u}$, and the squared radius D of the spherical code shaping region.

Output: A vector $\mathbf{z}_*$ such that $\mathbf{s}_* = \mathbf{G}\mathbf{z}_* + \mathbf{u} \in \mathcal{C}$ and $|\mathbf{v} - \mathbf{H}\mathbf{s}_*|^2 \leq |\mathbf{v} - \mathbf{H}\mathbf{s}|^2 \ \forall\, \mathbf{s} \in \mathcal{C}$, where $\mathcal{C} = (\Lambda(\mathbf{G}) + \mathbf{u}) \cap \overline{\mathcal{S}}(\mathbf{0}, D)$, and the radius C of the optimal search sphere.

Pre-compute (once per LAST code):

$(\mathbf{Q}_\mathbf{G}, \mathbf{P}) \leftarrow \mathsf{QR}(\mathbf{G})$ — Factor code lattice generator matrix

$\widetilde{\mathbf{u}} \leftarrow -\mathbf{Q}_\mathbf{G}^T\mathbf{u}$ — Project translation vector onto codeword space

$x_{\min} \leftarrow \left\lceil \frac{\widetilde{u}_m}{p_{mm}} - \frac{\sqrt{D}}{|p_{mm}|} \right\rceil$ — Compute lower bound of candidate range

$x_{\max} \leftarrow \left\lfloor \frac{\widetilde{u}_m}{p_{mm}} + \frac{\sqrt{D}}{|p_{mm}|} \right\rfloor$ — Compute upper bound of candidate range

Pre-process (once per fading block):

$\mathbf{\Xi} \leftarrow \mathbf{H}\mathbf{G}$ — Compute effective generator matrix

$(\mathbf{Q}_\mathbf{\Xi}, \underline{\mathbf{R}}) \leftarrow \mathsf{QR}(\mathbf{\Xi})$ — Factor effective generator matrix

Decode (once per received word):

$\widetilde{\mathbf{v}} \leftarrow \mathbf{Q}_\mathbf{\Xi}^T(\mathbf{v} - \mathbf{H}\mathbf{u})$ — Offset and project target onto search space

$x_{[0]} \leftarrow \frac{\widetilde{v}_m}{\underline{r}_{mm}}$ — Compute root unconstrained value

$x_{[1]} \leftarrow \mathsf{FirstValue\text{-}L}(x_{[0]}, x_{\min}, x_{\max})$ — Determine first candidate value

$\sigma \leftarrow d^2(\widetilde{v}_m, \underline{r}_{mm}x_{[1]})$ — Compute weight of first child of root node

$\mathcal{N}_b \leftarrow \{(\sigma, m, \widetilde{\mathbf{v}}, x_{[1]}, 1, 0, x_{[0]}, \widetilde{\mathbf{u}}, D, x_{\min}, x_{\max})\}$ — Put first child of root on border

repeat

  $(\mathbf{z}, \sigma) \leftarrow \mathsf{SmallestWeightLeaf\text{-}L}(\mathcal{N}_b)$ — Find (next) smallest weight leaf node

until $\mathbf{G}\mathbf{z} + \mathbf{u} \in \mathcal{C}$

Return $\mathbf{z}_* = \mathbf{z}$ and $C_* = \sqrt{\sigma}$ — Report optimal solution and search radius

Function 2 SmallestWeightLeaf-L$(\mathcal{N}_b)$

Input: The border nodelist $\mathcal{N}_b$ (global variable).

Output: The constraint value vector $\check{\mathbf{z}}$ and weight σ of the smallest weight unexplored leaf node, i.e., the smallest weight leaf node lying outside input border nodelist $\mathcal{N}_b$. The updated border nodelist reflecting any newly explored nodes is also returned (global variable).

$(\sigma, d', \widetilde{\mathbf{y}}', \check{\mathbf{z}}, q, \sigma', x^-, \widetilde{\mathbf{u}}', D', x_{\min}, x_{\max}) \leftarrow \mathsf{Get\&DeleteMin}(\mathcal{N}_b)$ — Smallest weight node

while $d' > 1$ **do** — Until leaf node selected

  Call $\mathsf{FirstChild\text{-}L}(\sigma, d', \widetilde{\mathbf{y}}', \check{\mathbf{z}}, \widetilde{\mathbf{u}}', D')$ — Expand node

  Call $\mathsf{NextSibling\text{-}L}(d', \widetilde{\mathbf{y}}', \check{\mathbf{z}}, q, \sigma', x^-, \widetilde{\mathbf{u}}', D', x_{\min}, x_{\max})$

  $(\sigma, d', \widetilde{\mathbf{y}}', \check{\mathbf{z}}, q, \sigma', x^-, \widetilde{\mathbf{u}}', D', x_{\min}, x_{\max}) \leftarrow \mathsf{Get\&DeleteMin}(\mathcal{N}_b)$ — Smallest weight node

end while

Return $\check{\mathbf{z}}$ and σ — Return constraints and weight of smallest weight leaf

Function 3 FirstChild-L$(\sigma, d', \widetilde{\mathbf{y}}', \check{\mathbf{z}}, \widetilde{\mathbf{u}}', D')$

Input: The weight σ, the parent dimension d', the parent residual target $\widetilde{\mathbf{y}}'$ and the constraint values $\check{\mathbf{z}}$ of the node, as well as the residual translation vector $\widetilde{\mathbf{u}}'$ and residual squared radius D' of its parent.

Output: The updated border nodelist $\mathcal{N}_b$ (global variable).

$d \leftarrow d' - 1$ — Determine current residual dimension
$\widetilde{\mathbf{u}} \leftarrow (\widetilde{\mathbf{u}}' - \mathbf{p}_{d'}\check{z}_1)_{\setminus d'}$ — Compute current residual translation vector
$D \leftarrow D' - d^2(\widetilde{u}'_{d'}, p_{d'd'}\check{z}_1)$ — Compute current residual squared radius
$x_{\min,\mathrm{c}} \leftarrow \left\lceil \frac{\widetilde{u}_d}{p_{dd}} - \frac{\sqrt{D}}{|p_{dd}|} \right\rceil$ — Compute lower bound of candidate range
$x_{\max,\mathrm{c}} \leftarrow \left\lfloor \frac{\widetilde{u}_d}{p_{dd}} + \frac{\sqrt{D}}{|p_{dd}|} \right\rfloor$ — Compute upper bound of candidate range
if $x_{\max,\mathrm{c}} \geq x_{\min,\mathrm{c}}$ **then** — If there are any children
  $\widetilde{\mathbf{y}} \leftarrow (\widetilde{\mathbf{y}}' - \underline{\mathbf{r}}_{d'}\check{z}_1)_{\setminus d'}$ — Compute current residual target
  $x_{[0]} \leftarrow \frac{\widetilde{y}_d}{\underline{r}_{dd}}$ — Compute current unconstrained value
  $x_{[1]} \leftarrow$ FirstValue-L$(x_{[0]}, x_{\min}, x_{\max})$ — Determine first candidate value
  $\sigma_c \leftarrow \sigma + d^2(\widetilde{y}_d, \underline{r}_{dd}x_{[1]})$ — Compute new child node components
  $\check{\mathbf{z}}_c \leftarrow \begin{bmatrix} x_{[1]} \\ \check{\mathbf{z}} \end{bmatrix}$
  $x_c^- \leftarrow x_{[0]}$
  Insert $(\sigma_c, d, \widetilde{\mathbf{y}}, \check{\mathbf{z}}_c, 1, \sigma, x_c^-, \widetilde{\mathbf{u}}, D, x_{\min,\mathrm{c}}, x_{\max,\mathrm{c}})$ into $\mathcal{N}_b$ — Put child on border
end if

Function 4 FirstValue-L$\big(x_{[0]}, x_{\min}, x_{\max}\big)$

Input: The unconstrained target $x_{[0]}$ and the lower and upper bounds $x_{\min}$ and $x_{\max}$ of the candidate range of the node.

Output: The constraint value $x_{[1]}$ of the first child node.

$x_{[1]} \leftarrow$Round$\big(x_{[0]}\big)$ — Ensure $x_{[1]} \in [x_{\min}, x_{\max}]$
if $x_{[1]} > x_{\max}$ **then**
  $x_{[1]} \leftarrow x_{\max}$
else if $x_{[1]} < x_{\min}$ **then**
  $x_{[1]} \leftarrow x_{\min}$
end if
Return $x_{[1]}$

Function 5 NextSibling-L$(d', \widetilde{\mathbf{y}}', \check{\mathbf{z}}, q, \sigma', x^-, \widetilde{\mathbf{u}}', D', x_{\min}, x_{\max})$

Input: The parent dimension d', the parent residual target $\widetilde{\mathbf{y}}'$, the constraint values $\check{\mathbf{z}}$ and the position q of the node, the weight σ' of its parent, the constraint value x^- of its previous sibling, as well as the residual translation$\widetilde{\mathbf{u}}'$ and residual squared radius D' of its parent, and the lower and upper bounds $x_{\min}$ and $x_{\max}$ of its candidate range.

Output: The updated border nodelist $\mathcal{N}_b$ (global variable).

if $q \leq x_{\max} - x_{\min}$ **then** — If there are more siblings
  $x_{[q]} \leftarrow \check{\mathbf{z}}_1$ — Get current constraint value
  $x_{[q+1]} \leftarrow$ NextValue-L$(q, x_{[q]}, x^-, x_{\min}, x_{\max})$ — Determine next candidate value
  $\sigma_s \leftarrow \sigma' + d^2(\widetilde{y}'_{d'}, \underline{r}_{d'd'}x_{[q+1]})$ — Compute new sibling node components
  $\check{\mathbf{z}}_s \leftarrow \begin{bmatrix} x_{[q+1]} \\ \check{\mathbf{z}}_2^l \end{bmatrix}$
  $q_s \leftarrow q + 1$
  $x_s^- \leftarrow x_{[q]}$
  Insert $(\sigma_s, d', \widetilde{\mathbf{y}}', \check{\mathbf{z}}_s, q_s, \sigma', x_s^-, \widetilde{\mathbf{u}}', D', x_{\min}, x_{\max})$ into $\mathcal{N}_b$ — Put sibling on border
end if

Function 6 $\mathsf{NextValue\text{-}L}\big(q, x_{[q]}, x_{[q-1]}, x_{\min}, x_{\max}\big)$

Input: The position q and the constraint value $x_{[q]}$ of the node, as well as that of its previous sibling x^-, and the lower and upper bounds $x_{\min}$ and $x_{\max}$ of its candidate range.

Output: The constraint value $x_{[q+1]}$ of the next sibling node.

$x_{[q+1]} \leftarrow x_{[q]} - q \cdot \mathsf{Sign}\big(x_{[q]} - x_{[q-1]}\big)$ — Ensure $x_{[q+1]} \in [x_{\min},\, x_{\max}]$
if $x_{[q+1]} > x_{\max}$ **then**
  $x_{[q+1]} \leftarrow x_{\max} - q$
else if $x_{[q+1]} < x_{\min}$ **then**
  $x_{[q+1]} \leftarrow x_{\min} + q$
end if
Return $x_{[q+1]}$

Algorithm 7 Schnorr-Euchner Adaptive LAST Decoder $\mathsf{SEA\text{-}L}(\mathbf{v}, \mathbf{H}, \mathbf{G}, \mathbf{u}, D)$

Input: The target vector $\mathbf{v}$, the channel matrix $\mathbf{H}$, the lattice generator matrix $\mathbf{G}$, its translation vector $\mathbf{u}$, and the squared radius D of the spherical code shaping region.

Output: A vector $\mathbf{z}_*$ such that $\mathbf{s}_* = \mathbf{G}\mathbf{z}_* + \mathbf{u} \in \mathcal{C}$ and $|\mathbf{v} - \mathbf{H}\mathbf{s}_*|^2 \leq |\mathbf{v} - \mathbf{H}\mathbf{s}|^2 \;\; \forall\, \mathbf{s} \in \mathcal{C}$, where $\mathcal{C} = (\Lambda(\mathbf{G}) + \mathbf{u}) \cap \overline{\mathcal{S}}(\mathbf{0}, D)$, and the radius C of the optimal search sphere.

Pre-compute (once per LAST code):
$(\mathbf{Q}_\mathbf{G}, \mathbf{P}) \leftarrow \mathsf{QR}(\mathbf{G})$ — Factor code lattice generator matrix
$\widetilde{\mathbf{u}} \leftarrow -\mathbf{Q}_\mathbf{G}^T \mathbf{u}$ — Project translation vector onto codeword space
$x_{\min} \leftarrow \left\lceil \frac{\widetilde{u}_m}{p_{mm}} - \frac{\sqrt{D}}{|p_{mm}|} \right\rceil$ — Compute lower bound of candidate range
$x_{\max} \leftarrow \left\lfloor \frac{\widetilde{u}_m}{p_{mm}} + \frac{\sqrt{D}}{|p_{mm}|} \right\rfloor$ — Compute upper bound of candidate range

Pre-process (once per fading block):
$\boldsymbol{\Xi} \leftarrow \mathbf{H}\mathbf{G}$ — Compute effective generator matrix
$(\mathbf{Q}_{\boldsymbol{\Xi}}, \underline{\mathbf{R}}) \leftarrow \mathsf{QR}(\boldsymbol{\Xi})$ — Factor effective generator matrix

Decode (once per received word):
$\widetilde{\mathbf{v}} \leftarrow \mathbf{Q}_{\boldsymbol{\Xi}}^T(\mathbf{v} - \mathbf{H}\mathbf{u})$ — Offset and project target onto search space
$x_{[0]} \leftarrow \frac{\widetilde{v}_m}{\underline{r}_{mm}}$ — Compute root unconstrained value
$x_{[1]} \leftarrow \mathsf{FirstValue\text{-}L}(x_{[0]}, x_{\min}, x_{\max})$ — Determine first candidate value
$\sigma \leftarrow d^2(\widetilde{v}_m, \underline{r}_{mm} x_{[1]})$ — Compute weight of first child of root node
$\mathsf{RecursiveExpand\text{-}L}(\sigma, m, \widetilde{\mathbf{v}}, x_{[1]}, 1, 0, x_{[0]}, \widetilde{\mathbf{u}}, D, x_{\min}, x_{\max})$ — Expand first child of root
Return $\mathbf{z}_* = \mathbf{z}$ and $C_* = \sqrt{\sigma}$ — Report optimal solution and search radius

Function 8 RecursiveExpand-L$\left(\sigma, d', \widetilde{\mathbf{y}}', \check{\mathbf{z}}, q, \sigma', x^-, \widetilde{\mathbf{u}}', D', x_{\min}, x_{\max}\right)$

Input: The current search radius C (global variable) and a node data structure.

Output: The current best solution vector $\mathbf{z}$ (global variable) and the current search radius C (global variable).

if $d' > 1$ **then** Expand node
  $n_c \leftarrow \mathsf{FirstChild}(\sigma, d', \widetilde{\mathbf{y}}', \check{\mathbf{z}}, \widetilde{\mathbf{u}}', D')$
  if $\sigma(n_c) < C^2$ **then**
    $\mathsf{RecursiveExpand}(n_c)$
  end if
  $n_s \leftarrow \mathsf{NextSibling}(d', \widetilde{\mathbf{y}}', \check{\mathbf{z}}, q, \sigma', x^-, \widetilde{\mathbf{u}}', D', x_{\min}, x_{\max})$
  if $\sigma(n_s) < C^2$ **then**
    $\mathsf{RecursiveExpand}(n_s)$
  end if
else if $\sigma < C^2$ **then** Smaller weight leaf node found
  $C \leftarrow \mathsf{Sqrt}(\sigma)$
  $\mathbf{z} \leftarrow \check{\mathbf{z}}$
end if

References

[1] M. O. Damen, H. E. Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inform. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[2] L. Zheng and D. N. C. Tse, "Diversity and multiplexing: A fundamental tradeoff in multiple-antenna channels," *IEEE Trans. Inform. Theory*, vol. 49, no. 5, pp. 1073–1096, May 2003.

[3] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 8, pp. 1451–1458, October 1998.

[4] H. E. Gamal, G. Caire, and M. O. Damen, "Lattice coding and decoding achieve the optimal diversity-multiplexing tradeoff in MIMO channels," *IEEE Trans. Inform. Theory*, vol. 50, no. 6, pp. 968–985, June 2004.

[5] E. A. Lee and D. G. Messerschmitt, *Digital communication.* Kluwer Academic Publishers, 1994.

[6] M. O. Damen, H. E. Gamal, and G. Caire, "MMSE-GDFE lattice decoding for solving under-determined linear systems with integer unknowns," in *IEEE Int'l Symposium on Information Theory*, July 2004, p. 538.

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

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

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

[10] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *ISSSE*, September 1998, pp. 295–300.

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

[12] K. Su, "Priority-first spherical lattice space-time decoder with boundary control," August 2006, http://www.mathworks.com/matlabcentral/, File 11952.

[13] P. Lavoie, D. Haccoun, and Y. Savaria, "A systolic architecture for fast stack sequential decoders," *IEEE Transactions on Communications*, vol. 42, no. 2, pp. 324–335, February 1994.

[14] K. Su, "Depth-first spherical lattice space-time decoder with boundary control," August 2006, http://www.mathworks.com/matlabcentral/, File 12034.

[15] I. Berenguer, X. Wang, N. Prasad, J. Wang, and M. Madihian, "Design of minimum error-rate LAST codes via stochastic optimization and gradient estimation," in *IEEE Globecom*, vol. 4, November 2005, pp. 2374–2378.

[16] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bölcskei, "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, July 2005.