Soft-Output Sphere Decoding: Performance and Implementation Aspects

C. Studer*, M. Wenk*, A. Burg*, and H. Bölcskei†

*Integrated Systems Laboratory
ETH Zurich, Switzerland
email: {studer, mawenk, apburg}@iis.ee.ethz.ch

†Communication Technology Laboratory
ETH Zurich, Switzerland
email: boelcskei@nari.ee.ethz.ch

Abstract— **Multiple-input multiple-output (MIMO) detection algorithms providing soft information for a subsequent channel decoder pose significant implementation challenges due to their high computational complexity. In this paper, we show how sphere decoding can be used as an efficient tool to implement soft-output MIMO detection with flexible trade-offs between computational complexity and (error rate) performance. In particular, we demonstrate that single tree search, ordered QR decomposition, channel matrix regularization, and log-likelihood ratio clipping are the key ingredients for realizing soft-output MIMO detectors with near max-log performance at a computational complexity that is reasonably close to that of hard-output sphere decoding.**

I. Introduction

Multiple-input multiple-output (MIMO) wireless systems employ multiple antennas on both sides of the wireless link and offer increased spectral efficiency (compared to single-antenna systems) by transmitting multiple data streams concurrently and in the same frequency band (spatial multiplexing). MIMO technology constitutes the basis for upcoming wireless communication standards, such as IEEE 802.11n and IEEE 802.16e.

The main challenge in the practical realization of MIMO wireless systems lies in the efficient implementation of the detector which needs to separate the spatially multiplexed data streams. To this end, a wide range of algorithms offering various trade-offs between performance and computational complexity have been developed [1]. Linear detection producing hard-decision outputs constitutes one extreme of the complexity/performance trade-off region, while computationally demanding a posteriori probability (APP) detection algorithms result in the opposite extreme. The computational complexity of a MIMO detection algorithm depends on the symbol constellation size and the number of spatially multiplexed data streams, but often also on the instantaneous MIMO channel realization and the signal-to-noise ratio (SNR). On the other hand, the overall decoding effort is typically constrained by system bandwidth, latency requirements, and limitations on power consumption. Implementing different algorithms, each optimized for a maximum allowed decoding effort and/or a particular system configuration, would entail a considerable hardware overhead and in addition be highly inefficient since large portions of the chip would remain idle most of the time. A practical MIMO receiver design must therefore be able to cover a wide range of complexity/performance trade-offs using a single tunable detection algorithm.

This work was supported by the STREP project No. IST-026905 (MASCOT) within the sixth framework programme (FP6) of the European Commission.

Contributions: In this (predominantly tutorial) paper, we provide a formulation of the sphere decoder [2], [3] as a tunable MIMO detector with performance ranging from that of successive interference cancellation (SIC) to that of max-log APP detection. Tuning of the detector is achieved through log-likelihood ratio (LLR) clipping, preprocessing, and imposing constraints on the maximum computational complexity of the decoder. We formulate a framework for systematically characterizing the resulting complexity/performance trade-offs. Finally, we elaborate on, and provide some refinements of, the tree-search algorithm introduced in [4] and the LLR clipping approach proposed in [5].

Outline: The remainder of this paper is organized as follows. Section II reviews the transformation of the MIMO detection and LLR computation problems into a tree-search problem. Section III reviews max-log APP sphere decoding and proposes some refinements of existing algorithms. In Section IV, we describe methods for reducing the tree-search complexity. A framework for evaluating the complexity/performance trade-offs of the resulting class of detectors is introduced in Section V. We conclude in Section VI.

II. Soft-Output Sphere Decoding

Consider a MIMO system with M_T transmit and $M_R \geq M_T$ receive antennas. The coded bit-stream is mapped to M_T-dimensional transmit vector symbols $\mathbf{s} \in \mathcal{O}^{M_T}$, where $\mathcal{O}$ stands for the underlying complex-valued scalar constellation of cardinality 2^Q. The individual coded bits are denoted by $x_{j,b}$, where the indices j and b refer to the bth bit in the binary label of the jth entry of $\mathbf{s}$, respectively. The resulting complex baseband input-output relation is given by

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

where $\mathbf{H}$ denotes the $M_R \times M_T$ channel matrix and $\mathbf{n}$ is an i.i.d. proper complex Gaussian distributed M_R-dimensional noise vector with variance N_o per complex entry.

A. Max-Log Soft-Output Computation

Soft-output MIMO detection requires the computation of LLRs for all coded bits. In order to reduce the corresponding computational complexity, we employ the *max-log approximation* [6]

$$L(x_{j,b}) = \min_{\mathbf{s}\in\mathcal{X}_{j,b}^{(0)}} \|\mathbf{y}-\mathbf{H}\mathbf{s}\|^2 - \min_{\mathbf{s}\in\mathcal{X}_{j,b}^{(1)}} \|\mathbf{y}-\mathbf{H}\mathbf{s}\|^2 \tag{2}$$

where $\mathcal{X}_{j,b}^{(0)}$ and $\mathcal{X}_{j,b}^{(1)}$ are the disjoint sets of vector symbols that have the bth bit in the label of the jth scalar symbol equal to 0 and 1, respectively, and the LLRs in (2) are normalized to avoid dependence on the noise variance. For each bit, one of the two minima in (2) is given by $\lambda^{\mathrm{ML}} = \|\mathbf{y}-\mathbf{H}\mathbf{s}^{\mathrm{ML}}\|^2$, where

$$\mathbf{s}^{\mathrm{ML}} = \arg\min_{\mathbf{s}\in\mathcal{O}^{M_T}} \|\mathbf{y}-\mathbf{H}\mathbf{s}\|^2 \tag{3}$$

is the maximum likelihood (ML) solution. The other minimum in (2) is given by

$$\lambda_{j,b}^{\overline{\mathrm{ML}}} = \min_{\mathbf{s}\in\mathcal{X}_{j,b}^{\left(\overline{x_{j,b}^{\mathrm{ML}}}\right)}} \|\mathbf{y}-\mathbf{H}\mathbf{s}\|^2 \tag{4}$$

where the *counter-hypothesis* $\overline{x_{j,b}^{\mathrm{ML}}}$ denotes the binary complement of the bth bit in the binary label of the jth entry of $\mathbf{s}^{\mathrm{ML}}$. With (3) and (4) the max-log LLRs can be written as

$$L(x_{j,b}) = \begin{cases} \lambda^{\mathrm{ML}} - \lambda_{j,b}^{\overline{\mathrm{ML}}} & , \quad x_{j,b}^{\mathrm{ML}} = 0 \\ \lambda_{j,b}^{\overline{\mathrm{ML}}} - \lambda^{\mathrm{ML}} & , \quad x_{j,b}^{\mathrm{ML}} = 1 \,. \end{cases} \tag{5}$$

From (5) we can conclude that efficient max-log APP MIMO detection reduces to efficiently identifying $\mathbf{s}^{\mathrm{ML}}$, λ^{ML}, and $\lambda_{j,b}^{\overline{\mathrm{ML}}}$ for $j = 1, 2, \ldots, M_T$ and $b = 1, 2, \ldots, Q$ [7].

B. Max-Log APP MIMO Detection as a Tree Search

Transforming (3) and (4) into tree-search problems and using the sphere decoding algorithm [2], [3] allows to efficiently compute the LLRs (5). To this end, the channel matrix $\mathbf{H}$ is first QR-decomposed according to $\mathbf{H} = \mathbf{QR}$, where $\mathbf{Q}$ is unitary of dimension $M_R \times M_T$ and the upper-triangular $M_T \times M_T$ matrix $\mathbf{R}$ has real-valued positive entries on its main diagonal. Left-multiplying (1) by[1] $\mathbf{Q}^H$ leads to the modified input-output relation

$$\tilde{\mathbf{y}} = \mathbf{R}\mathbf{s} + \mathbf{Q}^H\mathbf{n} \quad \text{with} \quad \tilde{\mathbf{y}} = \mathbf{Q}^H\mathbf{y}$$

and hence, noting that $\mathbf{Q}^H\mathbf{n}$ has the same statistics as $\mathbf{n}$, to the equivalent formulation of λ^{ML} and $\lambda_{j,b}^{\overline{\mathrm{ML}}}$ as

$$\lambda^{\mathrm{ML}} = \min_{\mathbf{s}\in\mathcal{O}^{M_T}} \|\tilde{\mathbf{y}}-\mathbf{R}\mathbf{s}\|^2 \tag{6}$$

$$\lambda_{j,b}^{\overline{\mathrm{ML}}} = \min_{\mathbf{s}\in\mathcal{X}_{j,b}^{\left(\overline{x_{j,b}^{\mathrm{ML}}}\right)}} \|\tilde{\mathbf{y}}-\mathbf{R}\mathbf{s}\|^2. \tag{7}$$

We next define the partial symbol vectors (PSVs) $\mathbf{s}^{(i)} = [\, s_i \; s_{i+1} \; \cdots \; s_{M_T} \,]^T$ and note that the $\mathbf{s}^{(i)}$ can be arranged in a tree that has its root just above level $i = M_T$ and leaves, which correspond to possible candidate symbol vectors, on level $i = 1$. After initializing $d_{M_T+1} = 0$, the Euclidean distances $d(\mathbf{s}) = \|\tilde{\mathbf{y}} - \mathbf{R}\mathbf{s}\|^2$ in (6) and (7) can be computed recursively as $d(\mathbf{s}) = d_1$ with the partial Euclidean distances (PEDs)

$$d_i = d_{i+1} + |e_i|^2 \quad , \; i = M_T, M_T-1, \ldots, 1 \tag{8}$$

and the distance increments (DIs)

$$|e_i|^2 = \left| \tilde{y}_i - \sum_{j=i}^{M_T} R_{i,j} s_j \right|^2. \tag{9}$$

Since the dependence of the PEDs d_i on the symbol vector $\mathbf{s}$ is only through $\mathbf{s}^{(i)}$, we have transformed ML detection and the computation of the max-log LLRs into a weighted tree-search problem: PEDs and PSVs are associated with *nodes*, *branches* correspond to DIs. Each path from the root down to a leaf corresponds to a symbol vector $\mathbf{s} \in \mathcal{O}^{M_T}$. The leaf associated with the smallest metric in $\mathcal{O}^{M_T}$ and $\mathcal{X}_{j,b}^{\left(\overline{x_{j,b}^{\mathrm{ML}}}\right)}$ corresponds to the solution of (6) and (7), respectively. The basic building block underlying the two tree traversal strategies described in the next section is the Schnorr-Euchner sphere decoder (SESD) with radius reduction [8], briefly summarized as follows: The SESD constrains the search to nodes which lie within a radius r around $\tilde{\mathbf{y}}$ and traverses the tree depth-first, visiting the children of a given node in ascending order of their PEDs. The basic idea of radius reduction is to start the algorithm with $r = \infty$ and to update the radius according to $r^2 \leftarrow d(\mathbf{s})$ whenever a leaf $\mathbf{s}$ has been reached. This avoids the problem of selecting a suitable (initial) radius and leads to efficient pruning of the tree.

Throughout this paper, computational complexity is defined as the number of visited nodes. This complexity measure is directly related to the throughput of corresponding VLSI implementations [9].

III. Tree-Traversal Strategies

Computing the LLRs as in (5) requires determining the metric $\lambda_{j,b}^{\overline{\mathrm{ML}}}$, which is achieved by traversing only those parts of the tree that have leaves in $\mathcal{X}_{j,b}^{\left(\overline{x_{j,b}^{\mathrm{ML}}}\right)}$. Since this computation has to be carried out for every coded bit, it is immediately obvious that the resulting need for repeated tree traversals can lead to a major computational burden. In the following, we review two alternative tree-traversal strategies, proposed in [7] and [4], respectively, for solving (6) and (7). In addition, we propose some minor refinements of the tree-search algorithm introduced in [4].

A. Repeated Tree Search

An algorithm for computing the LLRs based on repeated tree search (RTS) was described in [7]. The basic idea is to start by solving (6) (using the SESD) and to rerun the SESD to solve (7) for each coded bit (i.e., QM_T times) in the vector symbol. When rerunning the SESD to determine $\lambda_{j,b}^{\overline{\mathrm{ML}}}$, the search tree is prepruned by forcing the decoder to exclude all nodes (and the corresponding subtrees) from the search for which $x_{j,b} = x_{j,b}^{\mathrm{ML}}$. This prepruning procedure is illustrated in Fig. 1. Initializing the SESD with $r = \infty$ in

[1]The superscript ^{H} stands for conjugate transposition.

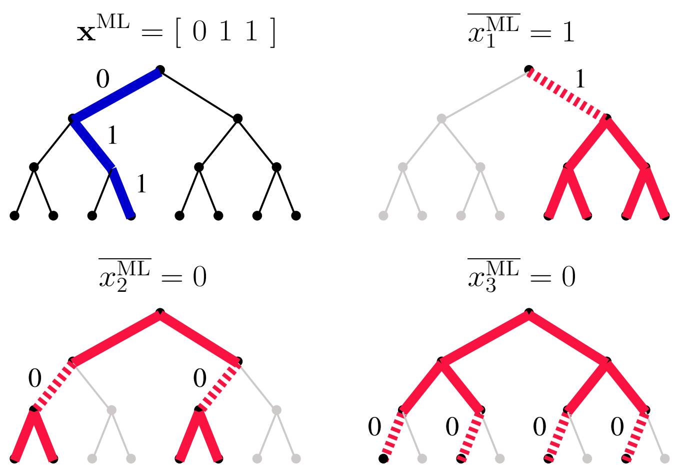

$\mathbf{x}^{\mathrm{ML}} = [\,0\ 1\ 1\,]$
0
1
1
$\overline{x_1^{\mathrm{ML}}} = 1$
1
$\overline{x_2^{\mathrm{ML}}} = 0$
0
0
$\overline{x_3^{\mathrm{ML}}} = 0$
0
0
0
0

Fig. 1. Example of the prepruning procedure in the RTS approach. Counter-hypotheses to the ML solution are found by forcing the algorithm through the dashed branches.

each of the QM_T runs required to obtain $\lambda_{j,b}^{\overline{\mathrm{ML}}}$ will lead to high computational complexity. It is therefore important to realize that, without compromising max-log optimality, we can initialize the search radius $r_{j,b}$ by setting it equal to the minimum value of $\|\tilde{\mathbf{y}} - \mathbf{R}\mathbf{s}\|$ over all $\mathbf{s} \in \mathcal{X}_{j,b}^{\left(\overline{x_{j,b}^{\mathrm{ML}}}\right)}$ found during preceding tree traversals.

The main advantage of the RTS strategy lies in the fact that each traversal of the tree can be performed using a hard-decision SESD with minimal modifications to account for the search being carried out on a prepruned tree. The main disadvantage is the repeated traversal of large parts of the tree. As noted in [10], this problem can be mitigated somewhat by changing the detection order in each run. Unfortunately, the resulting need for multiple QR-decompositions typically leads to prohibitive overall computational complexity.

B. Single Tree Search

The key to a more efficient (compared to RTS) tree-search strategy is to ensure that every node in the tree is visited at most once. This can be accomplished by searching for the ML solution *and* all counter-hypotheses concurrently. The basic idea behind such a single tree search (STS) approach has been outlined in [4]. In the following, we shall elaborate on the idea presented in [4] and describe some minor refinements. Specifically, we formulate update rules and a pruning criterion based on a list containing the metrics λ^{ML} and $\lambda_{j,b}^{\overline{\mathrm{ML}}}$.

The main concept is to have a list containing the metric λ^{ML} along with the corresponding bit sequence $\mathbf{x}^{\mathrm{ML}}$ and the metrics $\lambda_{j,b}^{\overline{\mathrm{ML}}}$ of all counter-hypotheses and to search the subtree originating from a given node only if the result can lead to an update of either λ^{ML} or one of the $\lambda_{j,b}^{\overline{\mathrm{ML}}}$.

List administration: The algorithm is initialized with $\lambda^{\mathrm{ML}} = \lambda_{j,b}^{\overline{\mathrm{ML}}} = \infty$ $(\forall\, j, b)$. Whenever a leaf with corresponding binary label $\mathbf{x}$ has been reached, the decoder distinguishes between two cases:

1) If a new ML hypothesis is found, i.e., $d(\mathbf{x}) < \lambda^{\mathrm{ML}}$, all $\lambda_{j,b}^{\overline{\mathrm{ML}}}$ for which $x_{j,b} = \overline{x_{j,b}^{\mathrm{ML}}}$ are set to λ^{ML} followed by the updates $\lambda^{\mathrm{ML}} \leftarrow d(\mathbf{x})$ and $\mathbf{x}^{\mathrm{ML}} \leftarrow \mathbf{x}$. In other words, for each bit in the ML hypothesis that is changed in the process of the update, the metric of the *former* ML hypothesis becomes the metric of the new counter-hypothesis, followed by an update of the ML hypothesis. This procedure ensures that all $\lambda_{j,b}^{\overline{\mathrm{ML}}}$ always contain the metric associated with a valid counter-hypothesis to the current ML hypothesis.
2) In the case where $d(\mathbf{x}) \geq \lambda^{\mathrm{ML}}$, only the counter-hypotheses have to be checked. For all j and b for which $d(\mathbf{x}) < \lambda_{j,b}^{\overline{\mathrm{ML}}}$ and $x_{j,b} = \overline{x_{j,b}^{\mathrm{ML}}}$, the decoder updates $\lambda_{j,b}^{\overline{\mathrm{ML}}} \leftarrow d(\mathbf{x})$.

Pruning criterion: The key aspect of this algorithm is the following pruning criterion. A given node $\mathbf{s}^{(i)}$ on level i and the subtree originating from that node have the partial binary label $\mathbf{x}^{(i)}$ consisting of the bits $x_{j,b}$ $(b = 1, 2, \ldots, Q$ and $j = i, i+1, \ldots, M_T)$. The remaining bits $x_{j,b}$ $(j = 1, 2, \ldots, i-1)$ corresponding to the subtree are unknown at this point. The pruning criterion for $\mathbf{s}^{(i)}$ along with its subtree is compiled from two conditions. First, the bits in the partial binary label $\mathbf{x}^{(i)}$ are compared with the corresponding bits in the binary label of the current ML hypothesis. In this comparison, for all j, b with $x_{j,b} = \overline{x_{j,b}^{\mathrm{ML}}}$, the corresponding counter-hypotheses $\lambda_{j,b}^{\overline{\mathrm{ML}}}$ might be affected when further searching the node's subtree. Second, *all* counter-hypotheses corresponding to the subtree of $\mathbf{s}^{(i)}$ with the associated metrics $\lambda_{j,b}^{\overline{\mathrm{ML}}}$ $(j = 1, 2, \ldots, i-1)$ may also be updated since the corresponding bits are not yet known. In summary, the metrics which may be affected during further search in the subtree emanating from a node $\mathbf{s}^{(i)}$ are given by the set

$$\mathcal{A} = \{a_l\} = \left\{\lambda_{j,b}^{\overline{\mathrm{ML}}} \,\middle|\, x_{j,b} = \overline{x_{j,b}^{\mathrm{ML}}},\ j \geq i\right\} \cup \left\{\lambda_{j,b}^{\overline{\mathrm{ML}}} \,\middle|\, j < i\right\}.$$

The node $\mathbf{s}^{(i)}$ along with its subtree is pruned if its PED $d\left(\mathbf{s}^{(i)}\right)$ satisfies

$$d\left(\mathbf{s}^{(i)}\right) > \max_{a_l \in \mathcal{A}} a_l\,. \tag{10}$$

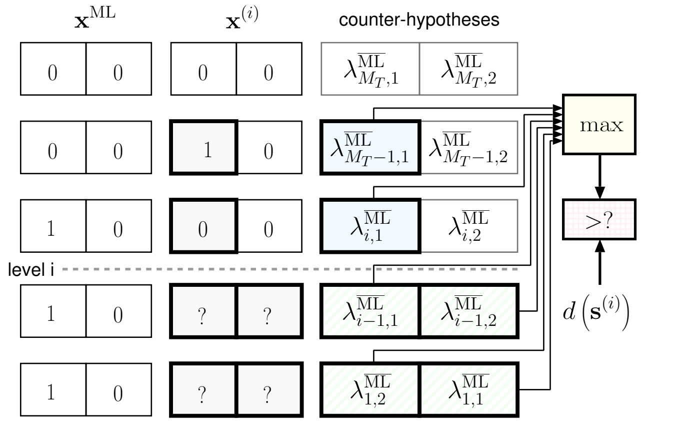
This pruning criterion (illustrated in Fig. 2) ensures that the subtree of a given node is explored only if it can lead to an update of either the ML hypothesis or of at least one of the counter-hypotheses. Note that λ^{ML} does not appear in (10) as $\lambda^{\mathrm{ML}} \leq \lambda_{j,b}^{\overline{\mathrm{ML}}}$ $(\forall\, j, b)$.

IV. Methods for Complexity Reduction

So far we have discussed tree-search strategies which solve (2) exactly and hence do not compromise the performance of the max-log APP decoder. The goal of this section is to describe methods that allow to trade-off decoder complexity with (error rate) performance.

A. LLR Clipping

The dynamic range of LLRs is typically not bounded. However, practical systems need to constrain the maximum LLR value to enable fixed-point implementations. Evidently this will lead to a performance degradation. A straightforward



$\mathbf{x}^{\mathrm{ML}}$
$\mathbf{x}^{(i)}$
counter-hypotheses
$\lambda^{\overline{\mathrm{ML}}}_{M_T,1}$
$\lambda^{\overline{\mathrm{ML}}}_{M_T,2}$
$\lambda^{\overline{\mathrm{ML}}}_{M_T-1,1}$
$\lambda^{\overline{\mathrm{ML}}}_{M_T-1,2}$
$\lambda^{\overline{\mathrm{ML}}}_{i,1}$
$\lambda^{\overline{\mathrm{ML}}}_{i,2}$
$\lambda^{\overline{\mathrm{ML}}}_{i-1,1}$
$\lambda^{\overline{\mathrm{ML}}}_{i-1,2}$
$\lambda^{\overline{\mathrm{ML}}}_{1,2}$
$\lambda^{\overline{\mathrm{ML}}}_{1,1}$
level i
max
>?
$d\left(\mathbf{s}^{(i)}\right)$

Fig. 2. Example of the STS pruning criterion ($M_T = 5$ and two bits per symbol): The partial binary label $\mathbf{x}^{(i)}$ determines which counter-hypotheses may be affected during the search of the subtree emanating from the current node.

way of ensuring that LLR values are bounded is to clip them *after* the detection stage so that

$$\left|L(x_{j,b})\right| \leq L_{\max} \quad \forall\, j,b\ . \tag{11}$$

We emphasize that the constraint in (11) refers to the normalized LLRs $L(x_{j,b})$ as defined in (2). It has been noted in [5] that (11) can be built into the tree-search algorithm such that it leads to a reduction in search complexity. In the following, we briefly describe the application of the idea proposed in [5] to the RTS and the STS tree-traversal strategies.

a) LLR Clipping for RTS: Whenever the RTS algorithm starts to search for a counter-hypothesis, with the search radius $r_{j,b}$ initialized as described in Section III-A, we first update

$$r_{j,b} \leftarrow \min\left\{r_{j,b}, \lambda^{\mathrm{ML}} + L_{\max}\right\} \tag{12}$$

which ensures that (11) is satisfied. Metrics associated with counter-hypotheses for which no valid lattice point can be found are set to $\lambda^{\mathrm{ML}} + L_{\max}$.

b) LLR Clipping for STS: Whenever a leaf has been reached and a new ML hypothesis has been found after carrying out the steps in Case 1 in Section III-B, the counter-hypotheses have to be updated according to

$$\lambda^{\overline{\mathrm{ML}}}_{j,b} \leftarrow \min\left\{\lambda^{\overline{\mathrm{ML}}}_{j,b}, \lambda^{\mathrm{ML}} + L_{\max}\right\} \quad \forall\, j,b\ . \tag{13}$$

For $L_{\max} = \infty$, we obviously get the exact max-log solution, whereas for $L_{\max} \to 0$, the decoder performance approaches that of a hard-output ML detector. On the other hand smaller $L_{\max}$ leads to a reduction in complexity, as more aggressive pruning is performed. The parameter $L_{\max}$ can therefore be used to adjust the complexity/performance trade-off (cf. Section V).

B. Ordering and Regularization

Ordering: A common approach to reduce complexity in sphere decoding without compromising the decoder's performance is to adapt the detection ordering of the spatial streams to the geometry of the instantaneous channel realization by performing a QR-decomposition on $\mathbf{HP}$ (rather than $\mathbf{H}$), where $\mathbf{P}$ is a suitably chosen permutation matrix. More efficient pruning of the search tree closer to the root is obtained if "stronger streams" correspond to the levels closer to the root, i.e., $\mathbf{P}$ is chosen such that the main diagonal entries of $\mathbf{R}$ in $\mathbf{HP} = \mathbf{QR}$ are sorted in ascending order. In the following, this approach is termed sorted QR-decomposition (SQRD) [11].

Regularization: Poorly conditioned channel realizations $\mathbf{H}$ lead to significant search complexity due to the low effective SNR on one or multiple of the effective spatial streams. An efficient way to counter this problem is to perform the tree-search on a regularized channel matrix by computing

$$\begin{bmatrix} \mathbf{H} \\ \alpha\mathbf{I} \end{bmatrix}\mathbf{P} = \begin{bmatrix} \mathbf{Q}_1 \\ \mathbf{Q}_2 \end{bmatrix}\mathbf{R}$$

where $\mathbf{I}$ is the $M_T \times M_T$-identity matrix, $\mathbf{Q}_1$ is of dimension $M_R \times M_T$, $\mathbf{Q}_2$ and $\mathbf{R}$ are of dimension $M_T \times M_T$, and $\alpha > 0$ is a suitably chosen regularization parameter. Note that $\mathbf{Q}_1$ is, in general, not unitary. LLRs are then computed according to

$$L(x_{j,b}) = \min_{\tilde{\mathbf{s}} \in \mathcal{X}^{(0)}_{j,b}} \|\tilde{\mathbf{y}} - \mathbf{R}\tilde{\mathbf{s}}\|^2 - \min_{\tilde{\mathbf{s}} \in \mathcal{X}^{(1)}_{j,b}} \|\tilde{\mathbf{y}} - \mathbf{R}\tilde{\mathbf{s}}\|^2 \tag{14}$$

where $\tilde{\mathbf{y}} = \mathbf{Q}_1^H\mathbf{y}$ and $\tilde{\mathbf{s}} = \mathbf{Ps}$. Note that the LLRs in (14) need to be reordered at the end of the decoding process to account for the permutation induced by $\mathbf{P}$. Operating on a regularized version of the channel matrix clearly entails an (error rate) performance loss. However, we shall see in Section V that choosing α according to the minimum mean squared error (MMSE) criterion (resulting in MMSE-SQRD) as outlined in [12], degrades the performance only slightly while leading to considerable savings in terms of search complexity.

C. Run-Time Constraints

A disadvantage of all SDs is that the computational complexity required to find the ML solution (and the LLR values) depends on the realization of the channel matrix and the noise; the worst-case complexity corresponds to an exhaustive search. On the other hand, in order to meet the practically important requirement of a fixed throughput, the algorithm run-time must be constrained, which leads to a constraint on the maximum detection effort. This, in turn, generally prevents the detector from achieving ML or max-log APP performance.

A straightforward way of enforcing a run-time constraint is to terminate the search, on a symbol vector by symbol vector basis, after a maximum number of visited nodes. The detector then returns the best solution found so far, i.e., the current ML and counter-hypotheses. A better solution is to impose an aggregate run-time constraint of ND_{avg} visited nodes for an entire block of N vector symbols[2]. The maximum complexity allocated to the detection of the kth vector symbol can, for example, be chosen according to the *maximum-first* (MF) scheduling strategy [13] as

$$D_{\max}(k) = ND_{\mathrm{avg}} - \sum_{i=1}^{k-1} D(i) - (N-k)M_T \tag{15}$$

[2]In an OFDM-based MIMO system, N would, for example, be the number of OFDM tones.

where $D(i)$ denotes the actual number of visited nodes for the ith vector symbol. The concept behind (15) is that a vector symbol is allowed to use up all of the remaining run-time within the block up to a safety margin of $(N-k)M_T$ visited nodes, which allows to find at least the decision feedback solution for the remaining vector symbols. Setting $D_{\text{avg}} = M_T$ maximizes the throughput but reduces the performance to that of hard-decision SIC.

V. Performance/Complexity Trade-offs

In practice, system engineers are typically faced with the problem of designing a receiver that achieves a given target frame error rate (FER) at a given throughput. The quality of the receiver implementation can then be measured by the minimum SNR required to achieve this target FER. In the following, we assess the complexity/performance trade-offs of the concepts described in Sections III and IV by plotting the average (over independent channel and noise realizations) number of visited nodes as a function of this minimum SNR. Since the number of visited nodes translates directly to the required chip area per throughput [9], the corresponding charts allow to associate an SNR penalty with a reduction in hardware complexity.

All simulation results are for a rate $1/2$ (generator polynomials $[133_o\ 171_o]$ and constraint length 7) convolutionally encoded 4×4 MIMO-OFDM system with 16-QAM constellation (using Gray mapping) and $N = 64$ tones. A soft-in Viterbi decoder [14] is employed. One frame consists of 1024 randomly interleaved (across space and frequency) bits and a TGn type C channel model [15] is used.

A. Comparison of Tree-Search Strategies

Fig. 3 compares the performance of RTS and STS max-log APP decoders, and the list sphere decoder (LSD) [6] for different target FERs, different values of $L_{\max}$ and in the case of the LSD for different list sizes. Changing the list size allows to adjust the complexity/performance trade-off.

The STS approach is seen to clearly outperform the RTS strategy in terms of average complexity. We can furthermore see that for this setup max-log APP performance is achieved for $L_{\max} = 0.2$. Increasing the LLR clipping level beyond this value only increases complexity without improving performance.

The implementation of the LSD requires additional memory and logic for the administration of the candidate list, which is not accounted for in this comparison. Fig. 3 shows that even when this additional complexity is ignored, the LSD is still inferior to the STS algorithm.

B. Impact of Preprocessing and Regularization

Fig. 4 compares the impact of SQRD, MMSE-SQRD, and standard (unordered) QRD-based preprocessing on the complexity/performance trade-off of the STS algorithm at a target FER of 0.01. It can be seen that the improvement resulting from SQRD compared to unordered QRD becomes significant in the low (but realistic) complexity region. Further (minor)

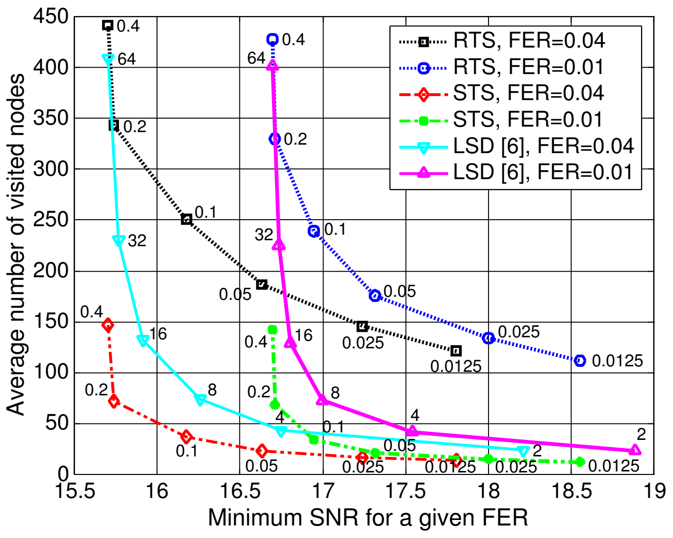


Fig. 3. Comparison of repeated tree search (RTS), single tree search (STS) and the list sphere decoder (LSD) as proposed in [6]. The numbers next to the curves correspond to $L_{\max}$ for RTS and STS and to the list size in the case of the LSD.

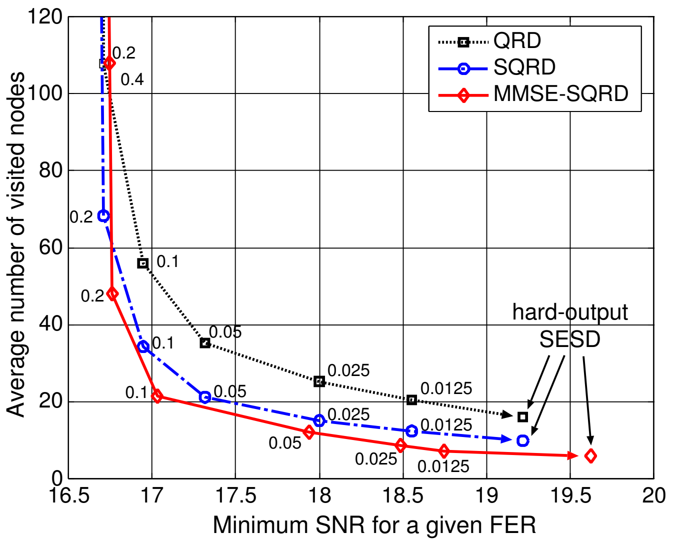


Fig. 4. Comparison of unordered QRD, SQRD and MMSE-SQRD preprocessing applied to STS at a target FER of 0.01. The numbers next to the curves correspond to $L_{\max}$. For $L_{\max} \to 0$, the performance approaches that of hard-output SESD.

improvements are obtained from regularization using MMSE-SQRD. In the region where the average complexity is very high, the performance penalty resulting from regularization eventually renders MMSE-SQRD inferior to SQRD.

C. LLR Clipping

Both Fig. 3 and Fig. 4 show that adjusting the LLR clipping level $L_{\max}$ allows to sweep an entire family of sphere decoders ranging from the exact max-log APP SESD (obtained, in our setup, for $L_{\max} \geq 0.2$) to hard-output SESD ($L_{\max} = 0$). The LLR clipping level is therefore an important design parameter which can be used to conveniently adjust the decoder *at runtime* to a given complexity constraint.

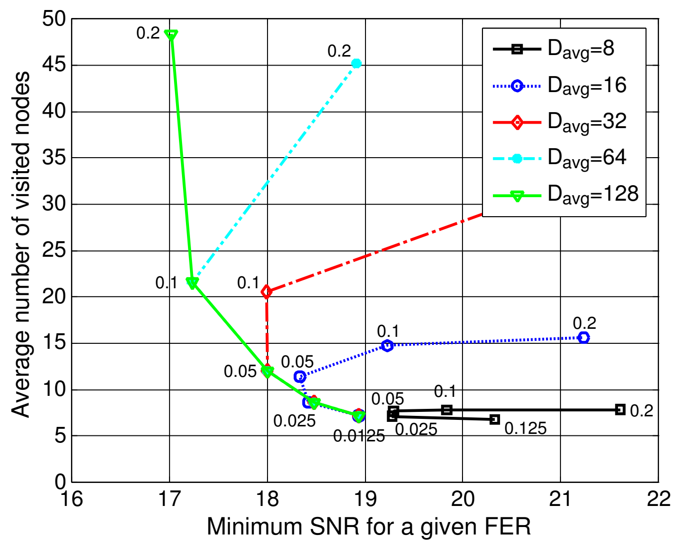


Fig. 5. Impact of run-time constraints with MF scheduling on STS SESD with MMSE-SQRD preprocessing at a FER of 0.01. The performance can be optimized by choosing an appropriate LLR clipping level (shown next to the curves) for a given average run-time constraint D_{avg}.

D. Run-time Constraints

In Fig. 5, we finally demonstrate the impact of imposing a maximum run-time constraint of ND_{avg} visited nodes for a block of $N = 64$ vector symbols using the strategy described in Section IV-C. The resulting curves essentially consist of two regions:

- If the LLR clipping level is large (corresponding to high search complexity), the run-time constrained detector is not able to compute accurate LLR values, which results in (very) poor performance, unless D_{avg} is large. For $D_{\text{avg}} = 128$, the performance is very close to that of the unconstrained max-log APP decoder.
- In the region where $L_{\max}$ is small, the performance is dominated by aggressive LLR clipping rather than by the run-time constraint.

In summary, we can conclude that for a given average run-time constraint there exists an optimum LLR clipping level, which minimizes the SNR required to achieve a certain target FER. It is therefore of paramount importance to choose the LLR clipping level in accordance with the average run-time constraint.

VI. Conclusions

The sphere decoder is a suitable tool to implement MIMO detection with flexible complexity/performance trade-offs. In particular, adjusting the LLR clipping level is an efficient way of realizing an entire family of decoders ranging from exact max-log soft-output SD to hard-output SIC detection. The keys to achieving low complexity are the single tree-search strategy in Section III-B, MMSE-SQRD preprocessing, LLR clipping, and imposing run-time constraints with MF scheduling. Our results demonstrate that MIMO detection with near max-log APP performance can be realized with a complexity that is reasonably close to that of a hard-output sphere decoder.

References

[1] H. Bölcskei, D. Gesbert, C. Papadias, and A. J. van der Veen, Eds., *Space-Time Wireless Systems: From Array Processing to MIMO Communications*. Cambridge Univ. Press, 2006.

[2] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Math. Programming*, vol. 66, no. 2, pp. 181–191, Sept. 1994.

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

[4] J. Jaldén and B. Ottersten, "Parallel implementation of a soft output sphere decoder," in *Proceedings Asilomar Conference on Signals, Systems and Computers*, Nov. 2005, pp. 581–585.

[5] M. S. Yee, "Max-Log-Map sphere decoder," in *Proc. IEEE ICASSP 2005*, vol. 3, Mar. 2005, pp. 1013–1016.

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

[7] R.Wang and G. Giannakis, "Approaching MIMO channel capacity with reduced-complexity soft sphere decoding," in *Proc. of IEEE Wireless Communications and Networking Conf. (WCNC)*, vol. 3, Mar. 2004, pp. 1620–1625.

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

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

[10] P. Marsch, E. Zimmermann, and G. Fettweis, "Smart candidate adding: A new low-complexity approach towards near-capacity MIMO detection," in *Proceedings of 13th European Signal Processing Conference (EUSIPCO)*, Sept. 2005.

[11] D. Wübben, R. Böhnke, J. Rinas, V. Kühn, and K. Kammeyer, "Efficient algorithm for decoding layered space-time codes," *IEE Electronics Letters*, vol. 37, no. 22, pp. 1348–1350, Oct. 2001.

[12] D. Wübben, R. Böhnke, V. Kühn, and K. Kammeyer, "MMSE extension of V-BLAST based on sorted QR decomposition," in *IEEE Proc. Vehicular Technology Conference (Fall)*, vol. 1, Oct. 2003, pp. 508–512.

[13] A. Burg, M. Borgmann, M. Wenk, C. Studer, and H. Bölcskei, "Advanced receiver algorithms for MIMO wireless communications," in *Proceedings of the Design Automation and Test Europe Conf. (DATE)*, vol. 1, May 2006, pp. 593–598.

[14] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

[15] V. Erceg *et al.*, *TGn channel models*, May 2004, IEEE 802.11 document 03/940r4.