Maximum Likelihood Soft-Output Detection through Sphere Decoding combined with Box Optimization

Victor M. Garcia-Molla[a,*], M. Angeles Simarro[b], F. J. Martínez-Zaldívar[b], Alberto González[b,1], Antonio M. Vidal[a]

[a] *Department of Information Systems and Computing, Universitat Politècnica de València, Camino de Vera s/n 46022 Valencia SPAIN.*
[b] *Department of Communications, Universitat Politècnica de València, Camino de Vera s/n 46022 Valencia SPAIN.*

Abstract

This paper focuses on the improvement of known algorithms for maximum likelihood soft-output detection. These algorithms usually have large computational complexity, that can be reduced by using clipping. Taking two well-known soft-output maximum likelihood algorithms (Repeated Tree Search and Single Tree Search) as a starting point, a number of modifications (based mainly on box optimization techniques) are proposed to improve the efficiency of the search. As a result, two new algorithms are proposed for soft-output maximum likelihood detection. One of them is based on Repeated Tree Search (which can be applied with and without clipping). The other one is based on Single Tree Search, which can only be applied to the case with clipping. The proposed algorithms are compared with the Single Tree Search algorithm, and their efficiency is evaluated in standard detection problems (4x4 16-QAM and 4x4 64-QAM) with and without clipping. The results show that the efficiency of the proposed algorithms is similar to that of the Single Tree Search algorithm in the case 4x4 16-QAM; however, in the case 4x4 64-QAM, the new algorithms are far more

*Corresponding author
Email addresses: vmgarcia@dsic.upv.es (Victor M. Garcia-Molla), mdesiha@teleco.upv.es (M. Angeles Simarro), fjmartin@dcom.upv.es (F. J. Martínez-Zaldívar), agonzal@dcom.upv.es (Alberto González), avidal@dsic.upv.es (Antonio M. Vidal)
[1] EURASIP member

efficient than the Single Tree Search algorithm.

Keywords: MIMO; Soft-Output Maximum Likelihood Detection.

1. Introduction

Digital communications using Multiple-Input Multiple-Output (MIMO) systems have nowadays been receiving considerable attention. These systems are included in current and future wireless communication standards, such as IEEE 802.11ac [1], Wimax [2] and 3GPP Long Term Evolution Advanced [3].

In MIMO systems, the use of soft-output detectors that are concatenated with a soft-input channel decoder can significantly improve the performance of wireless communications. A soft-output detector provides the reliability information of the received coded bits expressed as log-likelihood ratios (LLRs). These soft values are used by the channel decoder to carry out the final decision on the values of the received coded bits. However, the use of soft detection techniques involves a considerable increase in the computational cost compared with hard detection techniques, especially at low signal-to-noise ratios (SNR). This is so because soft detection methods require many more metric computations than hard detections methods. Practical applications of this technology will only be possible if efficient algorithms are developed.

The MIMO detection algorithms that compute the maximum likelihood solution of the problem are known as maximum likelihood (ML) algorithms. In hard-output detection, demodulators based on the tree search strategy show a lower complexity than those based on exhaustive search, with the Sphere Decoding (SD) variants being the family of algorithms that is most commonly used [4, 5, 6, 7, 8, 9]. Recently, a new hard-output SD ML algorithm was proposed in [10], where the SD algorithm was combined with box optimization. The results obtained were remarkably faster than other known hard-output ML detectors.

There exist several soft-output detection algorithms that use hard-output SD (or variations of it based on tree search) to compute the LLRs. Some of these soft-output algorithms are *Repeated Tree Search* (RTS) [11], a modified

RTS algorithm [12], *Single Tree Search* (STS) [13, 14], the *List-based SD* (LSD) scheme [15], *Soft-output Fixed-complexity SD* (SFSD) [16], the *Smart Ordering and Candidate Adding* (SOCA) algorithms [17], and *Soft-output K-Best* [18, 19]. There are other soft-output detection methods that are not based on tree search, such as the method based on partial marginalization [20], the *SUMIS* method [21], soft-output detection based on *Minimum Mean Square Error–Parallel Interference Cancellation* (MMSE–PIC) [22], soft-ouput based on belief propagation and on factor graphs [23], and a conjugate-gradient method for precoding [24]. Another soft-output ML detector (similar to STS) including several optimizations was proposed in [25]. Some of these algorithms provide exact max-log LLRs (STS and RTS among them), while others (like the LSD or the SFSD algorithms) provide approximations to the max-log LLRs (this entails a certain loss of performance). Since the computational complexity of soft-output algorithms that compute exact max-log LLRs (soft ML algorithms) is too high, in practical applications the complexity must be reduced further through the use of clipping [26].

It must be mentioned that max-log LLRs are approximations to exact LLRs, and some methods may compute LLRs more accurately than with the max-log approximation. However, the max-log approximation is still the most popular form of computing LLRs. In the following we will speak of soft-output ML algorithms as algorithms that compute exactly max-log approximations to LLRs.

The RTS and STS algorithms are the best known soft-output ML algorithms. These algorithms are thoroughly discussed in [13, 14], including the application of clipping to both algorithms. These papers show that STS is more efficient than RTS, thus making it one of the most efficient algorithms for soft-output ML MIMO detection (the version without clipping has been included in the Matlab communications toolbox [27]).

The work described in this paper has as its main goal the improvement in efficiency of soft-output ML detection algorithms, while at the same time preserving the ML property. We have obtained several possibilities for enhancing the RTS and STS algorithms. We propose three alternative implementations:

two based on RTS (for the cases with and without clipping) and another one based on STS which is only valid for the case with clipping. Some of the modifications proposed are based on the hard ML detector described in [10], while others can be implemented using any hard ML detector.

The algorithms obtained will be compared with the RTS and STS algorithms. The comparison of detection algorithms would usually be carried out in terms of efficiency and accuracy. However, since we are comparing soft-output ML algorithms, the accuracy comparison is not needed. This is because any soft-output ML algorithm implemented without clipping (such as STS, RTS or the algorithms proposed in this paper) will obtain the same exact max-log LLRs. The accuracy of MIMO detection methods is usually assessed through plots of Bit Error Rate (BER) against SNR. Therefore, since any two soft-output ML methods obtain the same max-log LLRs, the BER plot of both methods would be exactly the same line.

The same occurs when two soft-output ML methods implemented with clipping are compared (using the same clipping parameter). Since the max-log LLRs obtained are exactly the same, any plot for evaluation of accuracy would produce exactly the same line for both methods; such a plot would not convey any interesting information. The accuracy comparison is relevant when non-ML soft-output methods are compared with ML soft-output methods. However, this would be out of the scope of this paper and has been studied in other papers such as [13] and [17]. In this paper, we concentrate only on comparing different soft-output ML detection methods, and, therefore, we focus on comparing the efficiency of the methods.

In the following, we first describe the problem at hand and the algorithms to be applied or modified, and then we evaluate the resulting algorithms numerically, comparing their efficiency with the STS algorithm.

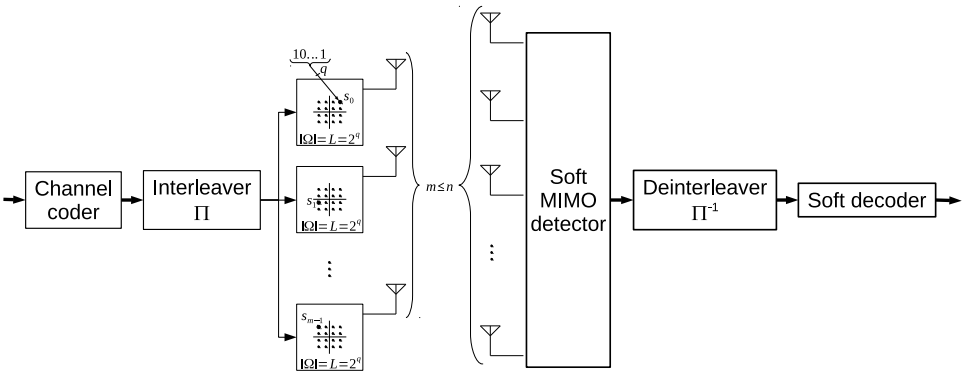


Figure 1: BICM system

2. Problem Description

Let us consider a MIMO-Bit Interleaved Coded Modulation (BICM) system (described graphically in Fig. 1) with m transmit antennas and n receive antennas ($n \geq m$). In this system, the sequence of information bits is encoded using an error-correcting code and is passed through a bitwise interleaver before being demultiplexed into m streams. In each stream, the bits are mapped into a complex symbol s_i, which is taken from a constellation $\Omega \subset \mathbb{C}$ of size $|\Omega| = L$ and hence carrying $q = \log_2 L$ code bits each. The transmit symbol vector is given by $\mathbf{s} = (s_1, \ldots, s_m)^T$, and the associated complex baseband model for the received vector can be written as

$$\mathbf{y} = \mathbf{H} \cdot \mathbf{s} + \mathbf{v}. \tag{1}$$

Here, $\mathbf{H} \in \mathbb{C}^{n \times m}$ is the MIMO channel matrix with independent elements $h_{ij} \sim \mathcal{CN}(\mathbf{0}, \mathbf{1})$ and $\mathbf{v}$ denotes a white-Gaussian noise (AWGN) complex vector with elements $v_i \sim \mathcal{CN}(\mathbf{0}, \frac{\mathbf{N_0}}{\mathbf{2}})$.

The MIMO detection problem can then be stated as:

$$\mathbf{s}^{\mathrm{ML}} = \underset{\mathbf{s} \in \Omega^m \subset \mathbb{C}^m}{\arg\min} \left\| \mathbf{H} \cdot \mathbf{s} - \mathbf{y} \right\|^2 . \tag{2}$$

The hard ML solution to the MIMO detection problem is the vector $\mathbf{s}^{\mathrm{ML}}$. Throughout this paper, given a possible transmit symbol vector $\mathbf{s}$, we will denote its associated Euclidean distance as:

$$d(\mathbf{s}) = \|\mathbf{y} - \mathbf{H} \cdot \mathbf{s}\|^2. \tag{3}$$

2.1. Hard-Output Sphere Decoding

The ML solution is usually computed using tree search techniques such as the Sphere Decoding algorithms. To apply an algorithm of this type, it is necessary to transform problem (2) into an equivalent problem using the QR decomposition of the channel matrix:

$$\mathbf{s}^{\mathrm{ML}} = \underset{\mathbf{s} \in \Omega^m \subset \mathbb{C}^m}{\arg\min} \left\|\mathbf{R} \cdot \mathbf{s} - \mathbf{z}\right\|^2 , \tag{4}$$

where $\mathbf{H} = \mathbf{Q} \cdot \mathbf{R}$, $\mathbf{Q}$ is a unitary matrix, $\mathbf{R}$ is upper triangular and $\mathbf{z} = \mathbf{Q}^H \cdot \mathbf{y}$.

The solution is obtained by traversing a tree of partial solutions, where the maximum depth of the tree is m and each node can have at most L descendants. A full search of the tree would generate all the possible codewords, which would be very inefficient.

The number of solutions to be visited in the tree can be reduced by selecting a radius r so that the solutions that do not fulfill the condition

$$\left\|\mathbf{R} \cdot \mathbf{s} - \mathbf{z}\right\|^2 \leq r^2 \tag{5}$$

are discarded. The selection of an appropriate initial radius is a difficult problem in SD detection. The optimal radius would be the distance given by ML solution $\mathbf{s}^{\mathrm{ML}}$, computed as:

$$d^{\mathrm{ML}} = \|\mathbf{y} - \mathbf{R} \cdot \mathbf{s}^{\mathrm{ML}}\|^2. \tag{6}$$

However, d^{ML} is known only when $\mathbf{s}^{\mathrm{ML}}$ has been computed. If the selected initial radius is smaller than d^{ML}, there will be no solutions fulfilling (5). On the other hand, a large radius may cause that there are too many solutions fulfilling (5)

and the computational cost may be too large. This problem is most influential in low SNR scenarios.

(In the following we will use the standard Matlab notation to denote an integer vector, $k:m$, where this denotes the vector of integers $(k, k+1, ..., m)$. Accordingly, $\mathbf{s}_{k:m}$ denotes the subvector $(\mathbf{s}_k, \mathbf{s}_{k+1}, \cdots, \mathbf{s}_m)$; and $\mathbf{R}_{a:b,c:d}$ denotes the submatrix of $\mathbf{R}$ obtained by selecting rows $a, a+1, ..., b$ and columns $c, c+1, ..., d$.)

All of the SD detectors use the upper triangular structure of the matrix $\mathbf{R}$ to detect the symbols starting from the level (or antenna) m up to the level 1. In the level k of the tree ($1 < k < m$), a partial transmit vector will have been obtained, which implies that the components $k+1, ..., m$ have already been assigned values belonging to the constellation. Components $1, .., k-1$ do not have values assigned yet, and a decision must be taken regarding component k. Therefore, in level k, expression (5) is rewritten as:

$$\begin{gathered} \|\mathbf{R}\cdot\mathbf{s}-\mathbf{z}\|^2 = \\ \|\mathbf{R}_{1:k-1,1:k-1}\cdot\mathbf{s}_{1:k-1}+\mathbf{R}_{1:k-1,k:m}\cdot\mathbf{s}_{k:m}-\mathbf{z}_{1:k-1}\|^2 + \\ \|\mathbf{R}_{k:m,k:m}\cdot\mathbf{s}_{k:m}-\mathbf{z}_{k:m}\|^2 \leq r^2 \ . \end{gathered} \tag{7}$$

Given a partial transmit vector $\mathbf{s}_{k:m}$, we will denote its partial Euclidean distance (PED) as:

$$d(\mathbf{s}_{k:m}) = \|\mathbf{R}_{k:m,k:m}\cdot\mathbf{s}_{k:m}-\mathbf{z}_{k:m}\|^2 \ . \tag{8}$$

Recall that components $\mathbf{s}_{1:k-1}$ do not have values assigned yet, while, on the other hand, the components $\mathbf{s}_{k+1:m}$ have already been assigned values. The standard practice in SD detection is to neglect the first term in (7), and to use as pruning condition for the component s_k the following expression:

$$d(\mathbf{s}_{k:m}) = \|\mathbf{R}_{k:m,k:m}\cdot\mathbf{s}_{k:m}-\mathbf{z}_{k:m}\|^2 \leq r^2 \ . \tag{9}$$

The ordering in which the symbols are tested severely affects the performance

of the Sphere Decoding algorithms. The best known symbol ordering is the one proposed by Schnorr and Euchner [6]. The different proposals discussed in this paper are based on the Schnorr-Euchner Sphere Decoder (SESD).

2.2. Box Optimization for MIMO Detection

The main proposal in this work and in paper [10] is to use continuous constrained optimization techniques to help SD-based detection algorithms in hard detection (described in [10]) and in soft-output detection (the target of this work). The auxiliary problem to be solved is:

$$\begin{array}{c} \hat{\mathbf{sr}} = \arg\min_{\mathbf{s}\in\mathbb{C}^m} \|\mathbf{R}\cdot\mathbf{s}-\mathbf{z}\|^2, \\ \min(Re(\Omega))\leq Re(s_i)\leq \max(Re(\Omega)),1\leq i\leq m \\ \min(Im(\Omega))\leq Im(s_i)\leq \max(Im(\Omega)),1\leq i\leq m \end{array} \tag{10}$$

where $s_i, 1 \leq i \leq m$ are the components of the vector $\mathbf{s}$. This problem is derived from (2), discarding the condition that the components of the solution belong to the constellation Ω.

Compared to problem (2), this is a continuous problem. The components of the solution vector do not need to belong to Ω; the only restriction is that the search zone be bounded. The search zone has the form of a box, hence the name of box optimization.

The actual algorithm used to solve (10) was fully described in [10], assuming a real-valued formulation. Here, we will describe in less detail (directly over the complex-valued formulation) how this problem was used to speed up hard-output sphere decoding, but still providing enough detail so that the modifications to soft-output algorithms can be easily understood. Throughout the paper, we will assume that there exists a set $\Omega_{\mathbb{R}}$ such that the constellation Ω can be obtained as a cartesian product $\Omega_{\mathbb{R}} \times i\cdot\Omega_{\mathbb{R}}$. This is the most habitual case and simplifies the notation; however, if the constellation cannot be expressed as cartesian product (such as 8-PSK), or if different constellations were used by different antennas, it would not be a serious problem because the algorithms can be easily adapted to such cases.

2.2.1. Box Optimization to obtain an initial point and an initial radius for SD

To start the search, some versions of SD require an initial feasible point, an initial radius, or both. It is quite common to solve the continuous unconstrained least squares problem:

$$\hat{\mathbf{s}} = \underset{\mathbf{s}\in\mathbb{C}^m}{\arg\min} \|\mathbf{R}\cdot\mathbf{s}-\mathbf{z}\|^2 \ . \tag{11}$$

All the components of $\hat{\mathbf{s}}$ are then rounded to the nearest element of the constellation Ω (this process is called quantization). The vector obtained after this process is $\hat{\mathbf{s}}_q$, which is known as the Zero-Forcing (ZF) estimator. This estimator may be a good approximation to $\mathbf{s}^{\mathrm{ML}}$ when the SNR is high, but it is known to give poor results if the SNR decreases.

When one or more of the components of the vector $\hat{\mathbf{s}}$ have real or imaginary parts outside of the interval $[\min(\Omega_{\mathbb{R}}), \max(\Omega_{\mathbb{R}})]$, we say that the vector $\hat{\mathbf{s}}$ is "out" of the constellation. Accordingly, we say that $\hat{\mathbf{s}}$ is "in" the constellation when all its components have their real and imaginary parts inside the interval $[\min(\Omega_{\mathbb{R}}), \max(\Omega_{\mathbb{R}})]$.

With large SNR, the estimator $\hat{\mathbf{s}}$ should be "in" the constellation, or at least very close to it. In this case, the ZF estimator $\hat{\mathbf{s}}_q$ should be reasonably close to the ML solution. However, for small SNR, the estimator $\hat{\mathbf{s}}$ will usually be "out" of the constellation, and the $\hat{\mathbf{s}}_q$ estimator may no longer be a good approximation to the ML solution. In that case, the estimator $\hat{\mathbf{sr}}_{\mathbf{q}}$, which is computed by quantizing the result of the box optimization $\hat{\mathbf{sr}}$, will surely be a better approximation to the ML solution $\mathbf{s}^{\mathrm{ML}}$. Therefore, $\hat{\mathbf{sr}}_{\mathbf{q}}$ may be used as an initial point for sphere decoder or even as a non-ML estimator of $\mathbf{s}^{\mathrm{ML}}$.

Another possibility proposed in [28] is the use of $\hat{\mathbf{sr}}_{\mathbf{q}}$ to compute an initial SD radius:

$$r_{\hat{\mathbf{sr}}_{\mathbf{q}}} = \|\mathbf{H}\cdot\hat{\mathbf{sr}}_{\mathbf{q}}-\mathbf{y}\| \ . \tag{12}$$

As reported in [28], in large noise situations, the radius estimate $r_{\hat{\mathbf{sr}}_{\mathbf{q}}}$ is usually a closer estimation to d^{ML} than the standard radius estimate computed using

the ZF estimator:

$$r_{\hat{\mathbf{s}}_\mathbf{q}} = \|\mathbf{H} \cdot \hat{\mathbf{s}}_\mathbf{q} - \mathbf{y}\| \ . \tag{13}$$

Therefore, as a conclusion for this section, box optimization can be used to obtain a better starting point for the search as well as a initial radius closer to d^{ML}.

2.2.2. Radius Bound for SD Search using box optimization

The second technique where box optimization is involved tries to obtain a tighter radius estimation before the expansion of each node. This technique was first proposed and described in [29]; the proposal was to obtain a bound that is tighter than (9) by also using the remaining term in inequality (7),

$$\|\mathbf{R}_{1:k-1,1:k-1} \cdot \mathbf{s}_{1:k-1} + \mathbf{R}_{1:k-1,k:m} \cdot \mathbf{s}_{k:m} - \mathbf{z}_{1:k-1}\|^2 \ . \tag{14}$$

This can be done by obtaining a lower bound c of this term, so inequality (7) can be written as:

$$\|\mathbf{R}_{k:m,k:m} \cdot \mathbf{s}_{k:m} - \mathbf{z}_{k:m}\|^2 \leq r^2 - c \ , \tag{15}$$

which is a tighter pruning condition than (9). This should provide a reduction in the number of feasible values of s_k, and, consequently, a reduction in the number of visited nodes. If c is indeed a lower bound of (14), equation (5) holds. Then, if the initial radius is selected so that there is at least a solution fulfilling (5), the resulting method will still be ML.

In [29], several methods to compute lower bounds of (14) were proposed, discussed, and evaluated. One of the proposals in [29] was to use box optimization to compute a lower bound of (14). This can be done considering (14) as a deflated MIMO detection problem. If the continuous least squares problem is solved:

$$\begin{aligned} \hat{\mathbf{s}}^{k-1} = \underset{\mathbf{s}_{1:k-1} \in \mathbb{C}^{k-1}}{\arg\min} & \|\mathbf{R}_{1:k-1,1:k-1} \cdot \mathbf{s}_{1:k-1} + \\ & \mathbf{R}_{1:k-1,k:m} \cdot \mathbf{s}_{k:m} - \mathbf{z}_{1:k-1}\|^2 \ , \end{aligned} \tag{16}$$

the estimator $\hat{\mathbf{s}}^{k-1}$ is obtained, which is analogous to the estimator $\hat{\mathbf{s}}$ computed as in (11) but for the deflated problem. It must be noted that problem (16) is actually a standard triangular system of linear equations, whose solution $\hat{\mathbf{s}}^{k-1}$ is computed exactly and fulfills:

$$\left\|\mathbf{R}_{1:k-1,1:k-1} \cdot \hat{\mathbf{s}}^{k-1} + \mathbf{R}_{1:k-1,k:m} \cdot \mathbf{s}_{k:m} - \mathbf{z}_{1:k-1}\right\| = 0 \ . \tag{17}$$

If $\hat{\mathbf{s}}^{k-1}$ is out of the constellation, then the estimator $\hat{\mathbf{sr}}^{k-1}$ (which is analogous to $\hat{\mathbf{sr}}$ for the deflated problem) is computed solving the box optimization problem for the deflated problem:

$$\begin{aligned}\hat{\mathbf{sr}}^{k-1} = \underset{\mathbf{s}_{1:k-1}}{\arg\min} \, & \left\|\mathbf{R}_{1:k-1,1:k-1} \cdot \mathbf{s}_{1:k-1} + \mathbf{R}_{1:k-1,k:m} \cdot \mathbf{s}_{k:m} - \mathbf{z}_{1:k-1}\right\|^2 \\ & \min(\Omega_{\mathbb{R}}) \leq Re(\mathrm{s}_i) \leq \max(\Omega_{\mathbb{R}}); 1 \leq i \leq k-1 \\ & \min(\Omega_{\mathbb{R}}) \leq Im(\mathrm{s}_i) \leq \max(\Omega_{\mathbb{R}}); 1 \leq i \leq k-1 \ .\end{aligned} \tag{18}$$

Problem (18) is analogous to (10) and can also be solved using box optimization techniques. For all $\mathbf{s}_{1:k-1} \in \Omega^{k-1}$, the solution $\hat{\mathbf{sr}}^{k-1}$ fulfills that:

$$\begin{aligned}&\left\|\mathbf{R}_{1:k-1,1:k-1} \cdot \hat{\mathbf{sr}}^{k-1} + \mathbf{R}_{1:k-1,k:m} \cdot \mathbf{s}_{k:m} - \mathbf{z}_{1:k-1}\right\|^2 \\ &\leq \left\|\mathbf{R}_{1:k-1,1:k-1} \cdot \mathbf{s}_{1:k-1} + \mathbf{R}_{1:k-1,k:m} \cdot \mathbf{s}_{k:m} - \mathbf{z}_{1:k-1}\right\|^2 \ .\end{aligned} \tag{19}$$

Hence, the proposal is to use the lower bound c:

$$c = \left\|\mathbf{R}_{1:k-1,1:k-1} \cdot \hat{\mathbf{sr}}^{k-1} + \mathbf{R}_{1:k-1,k:m} \cdot \mathbf{s}_{k:m} - \mathbf{z}_{1:k-1}\right\|^2 \tag{20}$$

in inequality (15).

Of course, if $\hat{\mathbf{s}}^{k-1}$ is in the constellation, $\hat{\mathbf{s}}^{k-1} = \hat{\mathbf{sr}}^{k-1}$. Thus, as in (17), the bound would be useless since

$$\left\|\mathbf{R}_{1:k-1,1:k-1} \cdot \hat{\mathbf{sr}}^{k-1} + \mathbf{R}_{1:k-1,k:m} \cdot \mathbf{s}_{k:m} - \mathbf{z}_{1:k-1}\right\|^2 = 0 \ . \tag{21}$$

In this case, it would be better to use other bounding techniques, (e.g., the

technique based on the minimum singular value described in [29]) or simply not to use any additional bound, since the standard SESD algorithm performs quite well in this case.

Paper [10] presents the implementation of a SESD hard detector including the techniques described above, plus a number of improvements and algorithmic optimizations. We will refer to the hard ML detector described in [10] as the Box Optimization Hard Detector (BOHD). The BOHD algorithm is orders of magnitude faster than standard ML SD detectors when applied to large problems (large modulation or large number of antennas), especially in the low SNR range. Furthermore, the practical results show that the performance of the BOHD algorithm is virtually constant across any SNR range (even for impractical SNRs). The key for the performance of this algorithm is that the box optimization (proposed in [29] and later improved in [10]) provides an extremely tight bound on the search radius. This causes a drastic decrease both in the number of nodes that must be explored to obtain the ML solution and in the required execution time.

2.3. Soft-Output Detection

In soft-output detection schemes, the demodulator computes the soft information about the bits in terms of LLRs at the receiver side. Given the received signal $\mathbf{y}$ and the channel matrix $\mathbf{H}$, the LLR of the b-th bit of the j-th entry $s_{j,b}$ is defined as

$$LLR_{j,b} = \ln \frac{P(s_{j,b} = 1|\mathbf{y}, \mathbf{H})}{P(s_{j,b} = 0|\mathbf{y}, \mathbf{H})}, \tag{22}$$

where $P(s_{j,b} = c|\mathbf{y}, \mathbf{H})$ is the probability of the bit $s_{j,b}$ having the value c, given the actual values of the received signal $\mathbf{y}$ and the channel matrix $\mathbf{H}$. The implementation of this formula would require an exhaustive search. In order to reduce the computational complexity, we apply the max-log approximation [15]. Therefore, the max-log LLR of the b-th bit of the symbol of the j-th entry

is denoted $L_{j,b}$ and is computed as:

$$L_{j,b} = \frac{1}{\sigma^2}\left[\min_{\mathbf{s}\in\mathcal{X}_{j,b}^{(0)}} \|\mathbf{y}-\mathbf{H}\cdot\mathbf{s}\|^2 - \min_{\mathbf{s}\in\mathcal{X}_{j,b}^{(1)}} \|\mathbf{y}-\mathbf{H}\cdot\mathbf{s}\|^2\right], \tag{23}$$

where $\mathcal{X}_{j,b}^{(c)}$ denotes the set of symbol vectors for which the b-th bit in entry j equals c.

In (23), one of the minima is the ML distance (d^{ML}) corresponding to the hard ML solution $\mathbf{s}^{\mathrm{ML}}$, computed as in equation (6).

The other minimum in (23) has to be calculated for all coded bits ($\forall j, b$). All these minima are called counter-hypothesis distances, and are computed as

$$\bar{d}_{j,b} = \min_{\mathbf{s}\in\mathcal{X}_{j,b}^{\left(\overline{s_{j,b}^{\mathrm{ML}}}\right)}} \|\mathbf{y}-\mathbf{H}\cdot\mathbf{s}\|^2, \tag{24}$$

where $s_{j,b}^{\mathrm{ML}}$ denotes the b-th bit of the symbol s_j^{ML} and $\overline{s_{j,b}^{\mathrm{ML}}}$ denotes the complement of bit $s_{j,b}^{\mathrm{ML}}$.

Using (23) and (24), the max-log LLRs are now calculated as

$$L_{j,b} = \frac{1}{\sigma^2}\left(d^{\mathrm{ML}} - \bar{d}_{j,b}\right)\left(1-2s_{j,b}^{\mathrm{ML}}\right). \tag{25}$$

Therefore, the soft-output ML algorithms must compute the hard ML solution $\mathbf{s}^{\mathrm{ML}}$, its associated ML distance d^{ML}, and the counter-hypothesis distances $\bar{d}_{j,b}$ for all $j = 1,\ldots,m$, $b = 1,\ldots,q$. (In the following we will use the term LLRs meaning max-log LLRs).

Clipping [26] can be applied to reduce the complexity of the search. Given a clipping parameter L_{clip}, it is assumed that any counter-hypothesis distance larger than $d^{\mathrm{ML}} + L_{clip}$ does not need to be computed exactly and can be set to the value $d^{\mathrm{ML}} + L_{clip}$. When clipping is applied to a soft-output ML method, the resulting method cannot strictly be called ML because the LLRs are no longer exact. However, it is important to note that any ML soft-output algorithm that is applied with a given clipping parameter L_{clip}, must compute exactly the same LLRs as any other ML soft-output method that is applied with the same clipping

parameter. In other words, all of the counter-hypothesis distances that are larger than $d^{\mathrm{ML}} + L_{clip}$ are set to $d^{\mathrm{ML}} + L_{clip}$, and all of the counter-hypothesis distances that are smaller than $d^{\mathrm{ML}} + L_{clip}$ are computed exactly. Therefore, the accuracies obtained by any two ML soft-output algorithms that use the same clipping parameter are the same, and, as mentioned in the introduction, the BER obtained by these algorithms would be the same. Therefore, in this sense, we can speak of "clipped" ML soft-output algorithms.

As stated above, there are other soft-output methods (LSD, SFSD) that do not guarantee finding the exact distances. (Some accuracy is usually lost in order to obtain better computational complexity.) The LSD and the STS algorithms are compared in [13], while SOCA, LSD, and STS are compared in [17].

2.4. Soft-Output Detection Algorithms

In this section, we will describe the soft-output ML algorithms that we would like to improve, that is, the RTS and STS algorithms in their original versions with and without clipping. All the descriptions given in this section are based on [13].

2.4.1. RTS

The RTS algorithm starts by computing the hard ML solution ($\mathbf{s}^{\mathrm{ML}}$ and d^{ML}) through a ML SD algorithm. The adaptive radius SESD is usually selected for this purpose. Then, the LLRs are obtained by computing the counter-hypothesis distances (24). These are obtained by running a ML SESD for each bit in the symbol vector, as described in [11, 13]. Therefore, the SESD algorithm must be executed $m \cdot q + 1$ times. The drawback of this procedure is clearly the increased complexity, especially for low SNR. However, it must be mentioned that once the hard ML solution has been obtained, the computation of each LLR is independent from the others, so the computation of the LLRs can be parallelized.

2.4.2. RTS with Clipping

Clipping is easily included in the RTS strategy. Since $\mathbf{s}^{\mathrm{ML}}$ and d^{ML} have been computed previously, the SD runs needed for each distance $\bar{d}_{j,b}$ are started by using the clipping distance $d^{\mathrm{ML}} + L_{clip}$ as the initial maximum radius. This reduces the number of nodes explored and the computation time considerably.

2.4.3. STS

The STS algorithm proposed in [13] is a sophisticated method that is designed to compute the hard ML solution and the soft information at the same time, traversing the tree of possible solutions only once. In [13], the STS algorithm was proved to be more efficient than the RTS algorithm. STS has the standard SESD structure. However, in order to detect d^{ML} and all the distances $\bar{d}_{j,b}$ simultaneously, the radius must be larger (it must be kept at least as large as $max\left(\bar{d}_{j,b}\right)$), and the radius is recomputed previously to the computation of any node or leaf.

Here we will give a brief overview of the distinguishing features of the STS algorithm, which are the update rules and the method for recalculating the radius. This overview is based on the description given in [13]. Variables $\mathbf{s_{opt}}$ and d_{opt} will be used to store the best signal and distance found at the present moment. Before starting the algorithm, the variables d_{opt} and $\bar{d}_{j,b}$ are initialized: $d_{opt} = \bar{d}_{j,b} = \infty,\ \forall j, b$.

Algorithm 1. *The update rules to be applied when a feasible leaf is found*

1. *If a leaf* $\mathbf{s}$ *is found such that* $d(\mathbf{s}) < d_{opt}$*, Then*
2. $\quad \forall j, b$ *such that the bit* $s_{j,b} = \overline{s_{j,b}^{\mathrm{ML}}}$
3. $\quad\quad$ *Set* $\bar{d}_{j,b}$ *to the value* d_{opt}*.*
4. $\quad$ *Set* $d_{opt} = d(\mathbf{s})$*.*
5. $\quad$ *Set* $\mathbf{s_{opt}} = s$*.*
6. *End If*
7. *If a leaf* $\mathbf{s}$ *is found such that* $d(\mathbf{s}) \geq d_{opt}$*, Then*
8. $\quad \forall j, b$ *such that the bit* $s_{j,b} = \overline{s_{j,b}^{\mathrm{ML}}}$ *and* $d(\mathbf{s}) < \bar{d}_{j,b}$*,*
9. $\quad\quad$ *Set* $\bar{d}_{j,b}$ *to the value* $d(\mathbf{s})$*.*

10. End If

Algorithm 2. *The method for recalculating the radius (which is applied in every explored node) in order to determine whether the node is expanded or pruned. Let* $\mathbf{s}_{k:m}$ *be a partial transmit vector (node) at level k:*

1. *Set* $d_1 = max\left(\bar{d}_{j,b}\right)$, $\forall b$, *for* $j = 1 \ldots k-1$
2. *Set* $d_2 = max\left(\bar{d}_{j,b}\right)$, $\forall b$, *for* $j = k \ldots m$
   and $(\mathbf{s}_{k:m})_{j,b} = \overline{s_{j,b}^{\mathrm{ML}}}$
3. *If* $d(\mathbf{s}_{k:m}) > max(d_1, d_2)$ *Then,* $\mathbf{s}_{k:m}$ *is pruned*
4. *Else* $\mathbf{s}_{k:m}$ *is expanded*
5. *End If*

When the STS concludes, $\mathbf{s_{opt}} = \mathbf{s}^{\mathrm{ML}}$, $d_{opt} = d^{ML}$, and $\bar{d}_{j,b}$ holds the counter-hypothesis distances. All the nodes (and leaves) with a PED in the interval $[d^{\mathrm{ML}}, \max(\bar{d}_{j,b})]$ will have been visited. If the difference between d^{ML} and $\max(\bar{d}_{j,b})$ is large, then the number of visited nodes can become prohibitively large. Since this happens frequently, clipping is needed for any practical implementation.

2.4.4. STS with Clipping

Clipping is included in STS by modifying the updating Algorithm 1, adding the final update: $\bar{d}_{j,b} = min\left(\bar{d}_{j,b}, d_{opt} + L_{clip}\right) \forall j, b$ after line 10. When the search concludes, all of the nodes whose partial Euclidean distance is contained in the interval $[d^{\mathrm{ML}}, d^{\mathrm{ML}} + L_{clip}]$ have been visited.

We investigated several possibilities for improving the RTS and STS algorithms using the box optimization techniques. As a result, we have obtained three alternative methods: two for the case with clipping and one for the case without clipping.

3. Proposal of Soft-Output Decoding Algorithms without Clipping

A simple and effective proposal for the case without clipping is to perform a straightforward replacement in the RTS algorithm, replacing the standard SESD hard detector by the BOHD algorithm. The large reduction in time and in visited nodes shown in [10] for hard detection is immediately reflected in a large reduction in complexity for the new RTS algorithm, which we will denote as Box Optimization Repeated Tree Search (BORTS).

The STS algorithm without clipping cannot be easily combined with the box optimization techniques. The reason is that the box optimization obtains extremely tight bounds for the radius, while the STS must keep a radius that is large enough to obtain all of the counter-hypothesis distances in a single tree traversal.

4. Proposals for Soft-output Detection with Clipping

The case with clipping is more relevant from a practical point of view, because the complexity of the algorithms without clipping is still too high for practical implementations. The BORTS algorithm described above is easily adapted for the case with clipping (exactly as described in 2.4.2), and it is possible to refine and improve it further in different ways (for example, through parallel computing, like the RTS algorithm).

As mentioned above, the STS algorithm without clipping does not fit very well with the box optimization aids. However, the situation is different when clipping is applied; there are several techniques that can be applied. We have found the following modifications to the STS algorithm to be quite influential.

4.1. Precomputation of $d^{\mathrm{ML}}, s^{\mathrm{ML}}$

This modification is an attempt to take advantage of the availability of the fast BOHD algorithm. Note that when the STS with clipping ends, all of the nodes with a PED contained in the interval $[d^{\mathrm{ML}}, d^{\mathrm{ML}} + L_{clip}]$ have been visited. The minimum number of nodes to be visited should clearly be the number of

nodes with a PED contained in this interval. However, the STS proceeds like the Schnorr-Euchner detector, that is, it starts with the initial best distance as $+\infty$ and updates it whenever STS finds a feasible leaf. When the STS finds a leaf with a smaller PED, it updates the best distance. However, these first leaves may have a PED that is larger than d^{ML}. As long as the STS does not find the ML solution, it must expand nodes with PED that are larger than $d^{\mathrm{ML}} + L_{clip}$. This means that some (possibly many) extra nodes may have to be expanded.

A technique that can be applied to reduce the number of nodes is simply to first compute $\mathbf{s}^{\mathrm{ML}}$ and d^{ML} using BOHD (or any other hard detector). Then the STS is modified so that it does not search $\mathbf{s}^{\mathrm{ML}}$ since it has already been computed, and so that the maximum value for the counter-hypothesis distances $\bar{d}_{j,b}$ is set to the value $d^{\mathrm{ML}} + L_{clip}$.

This is quite easy to implement. Since $\mathbf{s}^{\mathrm{ML}}$ and d^{ML} have already been computed, the update rules no longer have to consider updates of $\mathbf{s}^{\mathrm{ML}}$ or d^{ML}. Therefore the first update rule (Algorithm 1, lines 1–6) is no longer needed and all the counterhypothesis distances $\bar{d}_{j,b}$ can be initialized with the value $d^{\mathrm{ML}} + L_{clip}$. This will avoid the expansion of any node with a PED that is larger than $d^{\mathrm{ML}} + L_{clip}$. With this modification, the number of visited nodes should decrease.

4.2. Avoiding Radius Recalculations

In our experiments, we have observed that the radius recalculation (Algorithm 2) is quite an expensive process, especially because it is carried out on every visited node. In terms of computing time, we have found it very beneficial to avoid this recalculation. However, if no recalculation is made, the number of visited nodes can be too high. To alleviate this problem, as in the previous proposal, we try to take advantage of the fast BOHD algorithm by computing the hard ML information in a previous step, then we can use the following as a pruning condition: Given $\mathbf{s}_{k:m}$ a partial transmit vector (node) at level k, if $d(\mathbf{s}_{k:m}) > d^{\mathrm{ML}} + L_{clip}$, this node is pruned; otherwise the node is expanded.

The outcome of this modification (compared with the original STS algorithm) should be that the number of visited nodes increases and the average time complexity decreases, because a large number of radius recalculations is avoided.

Since this algorithm first uses the BOHD to obtain $\mathbf{s}^{\mathrm{ML}}$ and d^{ML} and then carries out a second tree search to obtain the counterhypothesis distances $\bar{d}_{j,b}$, we will refer to this algorithm (including the two modifications proposed) as the Double Tree Search algorithm (DTS).

Actually, these two modifications to the STS algorithm could be applied using any ML hard-output detector for the first search. However, the speed of the BOHD makes the whole method competitive.

5. Numerical Experiments and Discussion

In order to evaluate our proposals, we have compared the proposed algorithms with the STS algorithm through numerical experimentation. The Matlab implementation of our proposed algorithms can be found at `http://www.inco2.upv.es/box-optimization.php`. For the comparison, we used the code made available by Dr C.Studer (`http://www.nari.ee.ethz.ch/commth/research/downloads/siso_sts-sd.html`), which implements the soft-input soft-output STS algorithm described in [14]. This code can easily be used as just a soft-output STS algorithm by setting the a-priori LLRs to zero, and it can also be used to perform soft-output detection without clipping by setting L_{clip} to $+\infty$, or (as we have done in our experiments) skipping the sections of the code where clipping is performed.

In Figure 2, we reproduce the BER obtained by all the methods for all the configurations considered. As mentioned in the introduction, the BER of two ML soft-output algorithms ($\mathbf{s}^{\mathrm{ML}}$, d^{ML} and $\bar{d}_{j,b}$) without clipping is the same. The same occurs when two ML soft-output algorithms with clipping (using the same clipping parameter) are compared. This has always been verified in the simulations performed. As mentioned above, since the accuracy of the

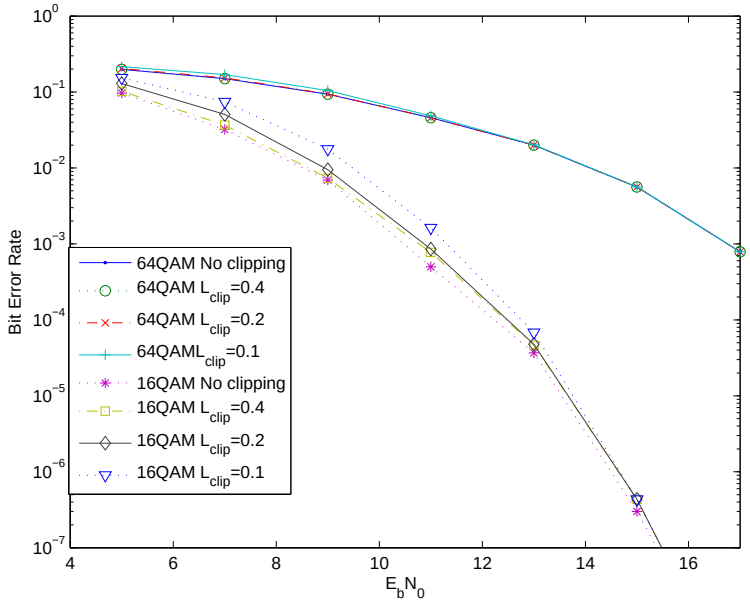


Figure 2: BER obtained with soft-output ML detectors, with 16 and 64-QAM modulations. Results obtained without clipping and with clipping parameter L_{clip} equal to 0.1, 0.2 and 0.4.

methods considered is the same, we will concentrate our efforts on comparing the complexity of these algorithms.

The computational complexity of MIMO detectors can be evaluated through different metrics: number of nodes expanded, computing time, and number of floating point operations (flops) are the metrics that are most commonly used. For most tree search MIMO detectors, the number of expanded nodes would be chosen as the main metric because it is independent of the computing platform. However, our experiments show that the algorithms that we are comparing can have large variations in the cost of the expansion of a single node. Therefore, even though the number of expanded nodes is an important factor, it cannot be used alone to evaluate the efficiency of the methods.

The number of flops is another metric that is often used, however, in this case, it can be somewhat misleading. The reason is that these algorithms perform a large number of comparisons, which in some cases is larger than the

Table 1: Average time (seconds), expanded nodes, flops, and comparisons for soft-output detection of a signal with the STS and BORTS methods in a 4×4 complex MIMO system with constellation 16-QAM without clipping.

	E_b/N_0 (dB)	5	7	9	11	13	15	17
Time	STS	1.0e-1	9.5e-2	1.0e-1	8.5e-2	8.8e-2	8.5e-2	8.4e-2
	BORTS	1.2e-1	1.2e-1	1.2e-1	1.1e-1	1.1e-1	1.1e-1	1.1e-2
Nodes	STS	5.2e2	5.6e2	5.6e2	4.7e2	4.4e2	4.2e2	4.2e2
	BORTS	4.1e2	4.1e2	4.2e2	3.8e2	3.8e2	3.5e2	3.5e2
Flops	STS	1.5e5	1.4e5	1.5e5	1.2e5	1.3e5	1.2e5	1.2e5
	BORTS	2.2e5	2.1e5	2.1e5	2.1e5	2.0e5	2.0e5	1.9e5
Comps.	STS	4.0e5	3.7e5	3.9e5	3.2e5	3.4e5	3.2e5	3.2e5
	BORTS	4.2e4	4.1e4	4.2e4	3.9e4	3.9e4	3.8e4	3.8e4

number of flops (as will be shown below). We modified the codes in order to record both the number of flops and the number of comparisons.

Finally, we have also recorded the computing times. The computing times depend on the computing platform, the implementation, and, in some cases, on the operating system. However, since the final goal is to obtain methods that can execute faster, the computing times help to identify the actual complexity.

We estimated the average number of expanded nodes, flops, comparisons, and execution time by means of Monte Carlo simulation. The experiments were carried out varying E_b/N_0 (or bit-normalized SNR; E_b is the transmitted energy per uncoded bit) from 5 to 17 dB. This is equivalent to a SNR variation from 8.01 to 20.01 dBs for 16-QAM and from 9.77 to 21.77 dBs for 64-QAM. We also used a rate 1/2 convolutional encoder of codeword size 2304 bits (generator polynomials $[133_O, 171_O]$ and constraint length 7) and max-log BCJR channel decoder. We simulated 4x4 complex MIMO systems with 16-QAM and 64-QAM constellations. The tests were carried out running Matlab R2014 using a single core of an Intel Xeon CPU X5680 processor with the Ubuntu operating system.

5.1. The Case without Clipping

In the case without clipping, we compare the complexity of the BORTS algorithm with the STS algorithm. The numerical results are summarized in Tables 1 and 2. The results show that the STS is faster for 16-QAM modulation;

Table 2: Average time (seconds), expanded nodes, flops, and comparisons for soft-output detection of a signal with the STS and BORTS methods in a 4×4 complex MIMO system with constellation 64-QAM without clipping.

	E_b/N_0 (dB)	5	7	9	11	13	15	17
Time	STS	6.2e0	6.7e0	8.0e0	6.8e0	7.5e0	6.6e0	4.9e0
	BORTS	3.8e-1	4.3e-1	4.7e-1	4.2e-1	4.9e-1	4.0e-1	3.6e-1
Nodes	STS	1.3e4	1.2e4	1.2e4	1.1e4	1.1e4	1.0e4	1.0e4
	BORTS	2.1e3	2.1e3	2.1e3	2.1e3	2.1e3	2.1e3	2.0e3
Flops	STS	9.6e6	1.0e7	1.2e7	1.0e7	1.1e7	1.0e7	7.3e6
	BORTS	8.9e5	1.0e6	1.1e6	1.1e6	1.2e6	9.9e5	9.2e5
Comps.	STS	6.0e7	6.2e7	7.5e7	6.4e7	7.0e7	6.2e7	4.5e6
	BORTS	3.1e5	3.8e5	4.3e5	3.9e5	4.7e5	3.8e5	3.4e5

however, the BORTS algorithm is around ten times faster in time, and around five times faster in terms of expanded nodes for 64-QAM. It is also clear that there is a substantial difference in the number of comparisons. In the 64-QAM case the difference is of two orders of magnitude.

The BORTS requires a previous run of the BOHD algorithm; the times, flops, comparisons, and number of nodes recorded include the times, flops, comparisons, and nodes expanded in the previous BOHD execution. However, the extra cost of this first BOHD run is quite small; in terms of computing time in the 16-QAM case, around $3-5\%$ of the time (on average) is devoted to the extra BOHD run. In the case of 64-QAM, the percentage of computing time taken by the extra BOHD run is around $1-3\%$.

5.2. The Case with Clipping

In this case, we compare STS (with clipping) with BORTS (with clipping) and DTS. These experiments were repeated with three different clipping parameter values (0.1, 0.2 and 0.4). The results are summarized in Tables 3 and 4. We have also chosen to display the complexity results (computing times and expanded nodes) for the smallest (4x4 16-QAM, $L_{clip} = 0.1$) and the largest (4x4 64-QAM, $L_{clip} = 0.4$) problems in Figs. 3 and 4.

BORTS and DTS require a previous run of the BOHD algorithm, whose computing times and expanded nodes were recorded and added. Again, the

extra cost of this first BOHD run turns out to be smaller than 5% in all the cases.

In the 16-QAM case, the performances of STS and DTS are similar in terms of computing times and flops. STS carries out more comparisons (due to the radius recalculations in algorithm 2) while it expands fewer nodes. Both methods (STS and DTS) exhibit a better performance when compared with BORTS.

There is a clear change in performance when the order of the modulation changes. In the 64-QAM case, DTS is substantially faster than STS in terms of computing times. However, DTS expands more nodes than STS, as shown in Figure 4. Clearly, the computing time per node of DTS is much smaller than for STS. The reason for this behaviour has been traced (through profiling) to the large number of radius recalculations (hence the large number of comparisons) carried out in algorithm 2. The large difference in the number of comparisons between DTS and STS can be seen in table 4.

The behaviour of BORTS is also worth analyzing. Table 4 shows that the complexity of BORTS (under any of the metrics considered) has a small or moderate variation when the clipping parameter changes. On the other hand, STS and DTS have comparatively large complexity variations when the clipping parameter varies. Therefore, BORTS becomes comparatively more efficient when the clipping parameter increases. For the largest clipping parameter in the 64-QAM problem, BORTS is faster than STS in all the metrics, while it is somewhat slower than DTS in computing time. It must be remembered that BORTS can be accelerated further using parallel computing (which was not done in the experiments described in this paper).

Another phenomenon that requires attention is that, in the largest problem considered (4x4 64-QAM, $L_{clip} = 0.4$), even though BORTS is slower than DTS, it uses less flops, comparisons and nodes. This phenomenon is due to the algorithmic structure of BORTS. BORTS needs to perform many calls to the subroutine where the BOHD detector is implemented. In turn, the BOHD detector needs to perform calls to other routines (such as the box optimization subroutine). The algorithmic structure of DTS and STS is quite different from

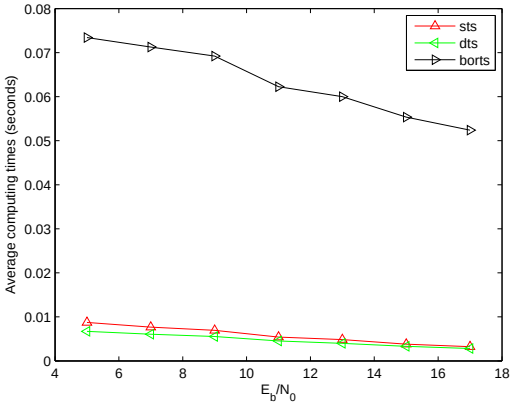
that of BORTS, DTS and STS perform just a few subroutine calls. The extra calls have a significant impact on the computing time of BORTS.

Finally, we would like to comment on some interesting results that indicate possible future research lines. In MIMO detection, it is very common to reorder the columns of the channel matrix to improve the efficiency of the tree search. There are many possible reorderings; those described in [16, 30, 31] are just some of them. As usual, there is a trade-off between the complexity of the reordering and the benefits obtained (reduction of number of expanded nodes). Any of these reorderings can be similarly applied to all of the methods considered in this paper (i.e., to the original RTS and STS and to the proposed methods BORTS and DTS). Since the techniques described in this paper are not linked to any particular reordering, we have chosen to use no reordering to obtain the results shown in this paper. However, some preliminary experiments show similar benefits from applying a given reordering to any of the methods. In other words, the improvement obtained from applying a reordering to STS and RTS is similar to the improvement obtained from applying the same reordering to BORTS and DTS. However, there are many reorderings not yet tested, so that this matter must be explored further.

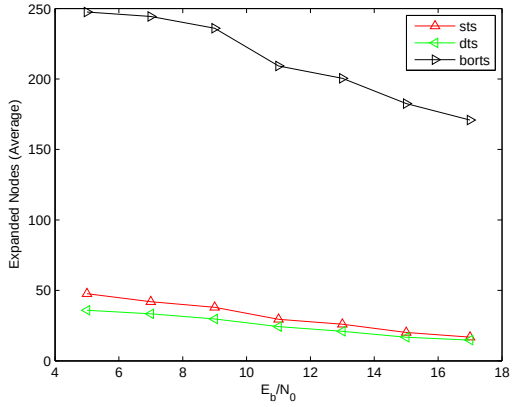
We have also experimented with larger problems (a larger constellation or an increase in the number of antennas). It is not easy to perform Monte Carlo simulations with larger problems because the time needed to complete a meaningful simulation becomes huge. However, we have verified that the new methods proposed increase their efficiency (compared to STS) when the size of the problem increases. This is consistent with the increased efficiency of the proposed methods in the 4x4 64-QAM case compared with the 4x4 16-QAM case.

6. Conclusion

Two new algorithms for soft-output ML detection (DTS and BORTS) have been presented. These algorithms were obtained by combining the RTS and



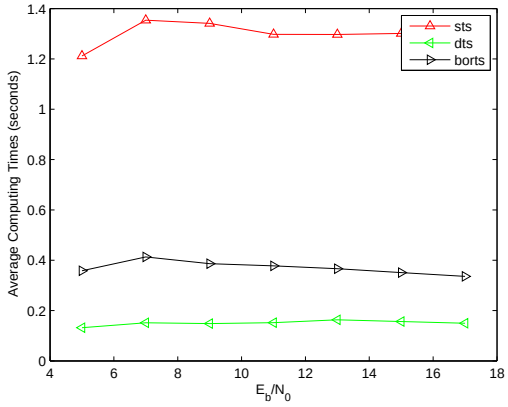
(a)



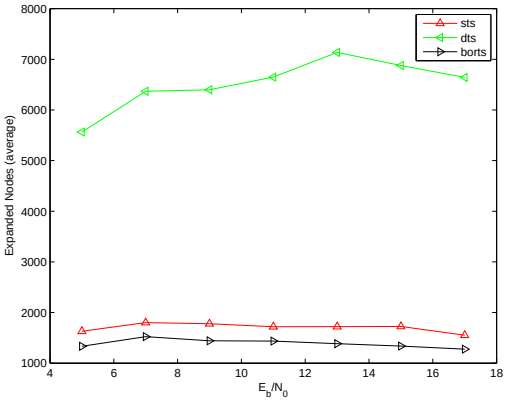
(b)

Figure 3: Average computing time (a) and expanded nodes (b) in 4x4 16-QAM, $L_{clip} = 0.1$.

STS algorithms with the hard detector proposed in [10], in different forms. The algorithms were tested in two relevant configurations, 4x4 16-QAM and 4x4 64-QAM. The results are especially good for the 4x4 64-QAM case. In the case with clipping (which is the most relevant from a practical point of view), DTS exhibits a smaller computing time, while BORTS expands fewer nodes. Even though there is some uncertainty about the results because the metrics show contradictory trends in some cases, the results still clearly show that the



(a)



(b)

Figure 4: Average computing time (a) and expanded nodes (b) in 4x4 64-QAM, $L_{clip} = 0.4$.

proposed algorithms have an excellent performance in large MIMO detection problems.

It is important to note that the results can be further improved by using some techniques that were not considered in this study, such as parallel computing (which can be applied easily to the RTS and BORTS methods) and reorderings (which can be applied to all the methods considered). Preliminary results show that the proposed methods can perform comparatively even better for larger

Table 3: Average time (seconds), expanded nodes, flops, and comparisons for soft detection of a signal with the STS, BORTS, and DTS methods in a 4×4 complex MIMO system with constellation 16-QAM with clipping.

		E_b/N_0 (dB)	5	7	9	11	13	15	17
$L_{clip} = 0.1$	Time	STS	8.7e-3	7.6e-3	6.9e-3	5.4e-3	4.8e-3	3.8e-3	3.2e-3
		BORTS	7.3e-2	7.1e-2	6.9e-2	6.2e-2	6.0e-2	5.5e-2	5.2e-2
		DTS	6.7e-3	6.0e-3	5.5e-3	4.5e-3	4.0e-3	3.3e-3	2.8e-3
	Nodes	STS	4.7e1	4.2e1	3.8e1	3.0e1	2.6e1	2.0e1	1.7e1
		BORTS	2.4e2	2.4e2	2.3e2	2.0e2	2.0e2	1.8e2	1.7e2
		DTS	5.5e1	3.3e1	2.9e1	2.4e1	2.0e1	1.6e1	1.4e1
	Flops	STS	1.3e4	1.1e4	1.0e4	8.0e3	7.1e3	5.6e3	4.7e3
		BORTS	1.3e5	1.3e5	1.3e5	1.2e5	1.1e5	1.1e5	1.0e5
		DTS	1.7e4	1.6e4	1.5e4	1.2e4	1.1e4	9.1e3	7.8e3
	Comps.	STS	3.3e4	2.8e4	2.6e4	2.0e4	1.8e4	1.4e4	1.2e4
		BORTS	2.8e4	2.8e4	2.7e4	2.5e4	2.4e4	2.3e4	2.2e4
		DTS	4.4e3	4.2e3	3.8e3	3.2e3	2.8e3	2.3e3	2.0e3
$L_{clip} = 0.2$	Time	STS	1.2e-2	1.1e-2	1.0e-2	8.4e-3	7.8e-3	6.4e-3	5.8e-3
		BORTS	7.8e-2	7.8e-2	7.6e-2	6.9e-2	6.8e-2	6.4e-2	6.1e-2
		DTS	7.1e-3	6.7e-3	6.1e-3	5.1e-3	4.6e-3	3.8e-3	3.3e-3
	Nodes	STS	6.8e1	6.2e1	5.8e1	4.7e1	4.4e1	3.6e1	3.3e1
		BORTS	2.7e2	2.7e2	2.6e2	2.3e2	2.3e2	2.1e2	2.0e2
		DTS	5.7e1	5.5e1	5.1e1	4.3e1	4.0e1	3.3e1	3.1e1
	Flops	STS	1.8e4	1.6e4	1.5e4	1.2e4	1.2e4	9.7e3	8.8e3
		BORTS	1.4e5	1.4e5	1.4e5	1.3e5	1.3e5	1.2e5	1.2e5
		DTS	2.1e4	2.0e4	1.9e4	1.6e4	1.5e4	1.2e4	1.1e4
	Comps.	STS	4.6e4	4.2e4	3.9e4	3.2e4	2.99e4	2.4e3	2.2e3
		BORTS	3.0e4	3.0e4	2.9e4	2.7e4	2.7e4	2.6e4	2.5e4
		DTS	5.7e3	5.5e3	5.1e3	4.3e3	4.0e3	3.4e3	3.1e3
$L_{clip} = 0.4$	Time	STS	2.0e-2	1.9e-2	1.8e-2	1.6e-2	1.5e-2	1.4e-2	1.3e-2
		BORTS	8.8e-2	8.8e-2	8.6e-2	8.1e-2	8.0e-2	7.7e-2	7.5e-2
		DTS	9.0e-3	8.7e-3	8.0e-3	6.9e-3	6.4e-3	5.5e-3	5.1e-3
	Nodes	STS	1.2e2	1.1e2	1.0e2	9.3e1	9.1e1	8.0e1	7.6e1
		BORTS	3.0e2	3.0e2	3.0e2	2.8e2	2.7e2	2.6e2	2.5e2
		DTS	1.2e2	1.2e2	1.2e2	1.1e2	1.0e2	9.3e1	9.0e1
	Flops	STS	3.0e4	2.9e4	2.8e4	2.4e4	2.4e4	2.1e4	2.0e4
		BORTS	1.6e5	1.6e5	1.6e5	1.5e5	1.5e5	1.4e5	1.4e5
		DTS	3.2e4	3.1e4	2.9e4	2.6e4	2.5e4	2.2e4	2.1e4
	Comps.	STS	7.8e4	7.5e4	7.2e4	6.2e4	6.1e4	5.3e4	5.1e4
		BORTS	3.3e4	3.3e4	3.3e4	3.1e4	3.1e4	3.0e4	2.9e4
		DTS	9.1e3	9.1e3	8.7e3	7.7e3	7.5e3	6.6e3	6.3e3

constellations or for systems with more antennas.

Table 4: Average time (seconds), expanded nodes, flops, and comparisons for soft detection of a signal with the STS, BORTS, and DTS methods in a 4×4 complex MIMO system with constellation 64-QAM with clipping.

		E_b/N_0 (dB)	5	7	9	11	13	15	17
$L_{clip} = 0.1$	Time	STS	2.1e-1	2.1e-1	1.9e-1	1.7e-1	1.8e-1	1.6e-1	1.5e-1
		BORTS	2.8e-1	3.3e-1	3.0e-1	2.9e-1	2.8e-1	2.7e-1	2.5e-1
		DTS	1.7e-2	1.8e-2	1.7e-2	1.6e-2	1.6e-2	1.4e-2	1.3e-2
	Nodes	STS	2.9e2	2.9e2	2.7e2	2.5	2.5e2	2.3e2	2.1e2
		BORTS	1.0e3	1.2e3	1.1e3	1.1e3	1.1e3	1.0e3	9.6e2
		DTS	2.7e2	2.9e2	2.8e2	2.8e2	3.0e2	2.7e2	2.5e2
	Flops	STS	3.1e5	3.1e5	2.9e5	2.5e5	2.6e5	2.4e5	2.2e5
		BORTS	7.4e5	8.6e5	8.1e5	8.0e5	7.9e5	7.5e5	7.2e5
		DTS	2.1e5	2.2e5	2.1e5	2.0e5	2.1e5	2.0e5	1.8e5
	Comps.	STS	1.9e6	1.9e6	1.8e6	1.6e6	1.6e6	1.5e6	1.4e6
		BORTS	2.9e5	3.5e5	3.3e5	3.3e5	3.2e5	3.1e5	2.9e5
		DTS	9.4e4	1.0e5	9.7e4	9.2e4	9.6e4	8.9e4	8.3e4
$L_{clip} = 0.2$	Time	STS	4.5e-1	4.9e-1	4.7e-1	4.5e-1	4.6e-1	4.5e-1	4.0e-1
		BORTS	3.1e-1	3.7e-1	3.4e-1	3.3e-1	3.2e-1	3.1e-1	2.9e-1
		DTS	3.7e-2	4.1e-2	4.0e-2	3.9e-2	4.2e-2	3.9e-2	3.6e-2
	Nodes	STS	6.3e2	6.8e2	6.6e2	6.2e2	6.4e2	5.2e2	5.6e2
		BORTS	1.2e3	1.3e3	1.2e3	1.2e3	1.2e3	1.1e3	1.1e3
		DTS	1.0e3	1.1e3	1.1e3	1.2e3	1.3e3	1.2e3	1.1e3
	Flops	STS	6.9e5	7.4e5	7.2e5	6.8e5	7.0e5	6.8e5	6.1e5
		BORTS	8.1e5	9.3e5	8.9e5	8.9e5	8.7e5	8.4e5	8.1e5
		DTS	5.3e5	6.0e5	5.9e5	5.8e5	6.2e5	5.9e5	5.6e5
	Comps.	STS	4.2e6	4.6e6	4.4e6	4.2e6	4.3e6	4.2e6	3.7e6
		BORTS	3.1e5	3.8e5	3.6e5	3.6e5	3.5e5	3.4e5	3.2e5
		DTS	2.3e5	2.7e5	2.6e5	2.6e5	2.8e5	2.7e5	2.5e5
$L_{clip} = 0.4$	Time	STS	1.2e0	1.3e0	1.3e0	1.3e0	1.3e0	1.3e0	1.1e0
		BORTS	3.5e-1	4.1e-1	3.8e-1	3.7e-1	3.6e-1	3.5e-1	3.3e-1
		DTS	1.3e-1	1.5e-1	1.5e-1	1.5e-1	1.6e-1	1.5e-1	1.5e-1
	Nodes	STS	1.6e3	1.8e3	1.7e3	1.7e3	1.7e3	1.7e3	1.5e3
		BORTS	1.3e3	1.5e3	1.4e3	1.4e3	1.4e3	1.3e3	1.2e3
		DTS	5.6e3	6.3e3	6.2e3	6.6e3	7.1e3	6.8e3	6.6e3
	Flops	STS	1.8e6	2.0e6	2.0e6	1.9e6	1.9e6	1.9e6	1.7e6
		BORTS	8.9e5	1.0e6	9.9e5	9.8e5	9.7e5	9.3e5	9.0e5
		DTS	1.8e6	2.0e6	2.0e6	2.1e6	2.2e6	2.1e6	2.0e6
	Comps.	STS	1.1e7	1.3e7	1.2e7	1.2e7	1.2e7	1.2e7	1.0e7
		BORTS	3.4e5	4.1e5	3.9e5	4.0e5	3.8e5	3.8e5	3.5e5
		DTS	8.2e5	9.5e5	9.4e5	9.3e3	1.0e6	9.8e5	9.3e5

7. Acknowledgments

This work has been partially funded by Generalitat Valenciana through the projects *ISIC/2012/006* and *PROMETEO II/2014/003*, and by Ministerio

Español de Economía y Competitividad through the project *TEC2012-38142-C04* and through the Grant RACHEL TEC2013-47141-C4-4-R.

References

[1] Specification Framework for TGac. IEEE 802.11-09/0992r21, IEEE P802.11ac, Jan. 2011.

[2] IEEE 802.16: Broadband Wireless Metropolitan Area Networks (MANs) [online]. Available: `http://standards.ieee.org/about/get/802/802.16.html`

[3] Overview of 3GPP Release 10 V0.0.8 (2010-09) [Online]. Available: `http://www.3gpp.org/ftp/Information/WORK_PLAN/Description_Releases/Rel-10_description_20100924.zip`

[4] R. Kannan, Improved algorithm for integer programming and related lattice problems, Proc. 15th ACM Symp. on Theory of Computing (1983) 193-206.

[5] U. Fincke, M. Pohst, Improved methods for calculating vectors of short length in a lattice, including a complexity analysis, Math. Comput. 44 (170) (1985) 463–471.

[6] C. Schnorr, M. Euchner, Lattice basis reduction: Improved practical algorithms and solving subset sum problems, Math. Programming 48 (66) (1994) 181–191.

[7] E. Agrell, T. Eriksson, A. Vardy, K. Zeger, Closest point search in lattices, IEEE Trans. Commun. 48 (2002) 2201–2214.

[8] M.O. Damen, H. El Gamal, G. Caire, On Maximum-Likelihood Detection and the Search for the Closest Lattice Point, IEEE Trans. on Inform. Theory 49 (10) (2003) 2389–2402.

[9] B. Hassibi, H. Vikalo, On sphere decoding algorithm. I. Expected complexity, IEEE Trans. Signal Process. 53 (8) (2005) 2806–2818.

[10] V. M. Garcia-Molla, A. Vidal, A. Gonzalez, S. Roger, Improved Maximum Likelihood Detection through Sphere Decoding combined with Box Optimization, Signal Processing, Elsevier 98 (2014) 287–294.

[11] R. Wang, G. Giannakis, Approaching MIMO channel capacity with reduced-complexity soft sphere decoding, Proc. IEEE Wireless Communications and Networking Conf. 3 (2004) 1620–1625.

[12] S. Shieh, R. Chiu, S. Feng, P. Chen, Low-complexity soft-output sphere decoding with modified repeated tree search strategy, IEEE Commun. Lett. 17 (1) (2013) 51–54.

[13] C. Studer, A. Burg, H. Bölcskei, Soft-output sphere decoding: algorithms and VLSI implementation, IEEE J. Sel. Areas Commun. 26 (2) (2008) 290–300.

[14] C. Studer, H. Bölcskei, Soft-input soft-output single tree-search sphere decoding, IEEE Trans. Inf. Theory, 56 (10) (2010) 4827-4842.

[15] B. M. Hochwald, S. ten Brink, Achieving Near-Capacity on a Multiple-Antenna Channel, IEEE Trans. Commun. 51 (2003) 389–399.

[16] L. Barbero, T. Ratnarajah, C. Cowan, A low-complexity soft-MIMO detector based on the fixed-complexity sphere decoder, IEEE International Conference on Acoustics, Speech and Signal Processing (ICCASP)(2008).

[17] D. Milliner, E. Zimmermann, J. Barry, G. Fettweis, A fixed-complexity smart candidate adding algorithm for soft-output MIMO detection, IEEE J. Sel. Topics Signal Process 3 (6) (2009) 1016–1025.

[18] S. Chen, T. Zhang, Y. Xin, Relaxed K-Best MIMO Signal Detector Design and VLSI Implementation, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., 15 (3) (2007) 328-337.

[19] X. Wu, J. Thompson, A fixed-complexity soft-MIMO detector via parallel candidate adding scheme and its FPGA implementation, IEEE Commun. Lett. 15 (2) (2011) 241–243.

[20] E. G. Larsson, J. Jalden, Fixed-complexity soft MIMO detection via partial marginalization, IEEE Trans. Signal Process., 56 (8) (2008) 3397–3407.

[21] M. Cirkic, E. G. Larsson, SUMIS: Near-optimal soft-in soft-out MIMO detection with low and fixed complexity, IEEE Trans. Signal Process. 62 (12) (2014) 3084–3097.

[22] C. Studer, S. Fateh, D. Seethaler, ASIC Implementation of Soft-Input Soft-Output MIMO Detection Using MMSE Parallel Interference Cancellation, IEEE J. Solid-State Circuits, 46 (7) (2011) 1754–1765.

[23] A. Chockalingham, B. Sundar Rajan, Large MIMO Systems, Cambridge University Press, 2014

[24] B. Yin, M. Wu, J.R. Cavallaro, C. Studer, Conjugate Gradient-based Soft-Output Detection and Precoding, Massive MIMO Systems IEEE Global Communications Conference (GLOBECOM), 2014.

[25] M.M. Mansour, S.P. Alex, L.M.A. Jalloul, Reduced Complexity Soft-Output MIMO Sphere Detectors - Part I: Algorithmic Optimizations, IEEE Trans. Signal Process. 62 (21) (2014) 5505–5520.

[26] M. S. Yee, Max-Log-Map sphere Decoder, IEEE International Conference on Acoustics, Speech and Signal Processing (ICCASP) 3 (2005), 1013–1016.

[27] The Mathworks Inc., MATLAB R14 Natick MA, 2004.

[28] X. Wen, Q. Han, Solving Box–Constrained Integer Least Squares Problems, IEEE Trans. on Wireless Communications, 7 (1) (2008) 277–287.

[29] M. Stojnic, H. Vikalo, B. Hassibi, Speeding up the Sphere Decoder with H^∞ and SDP inspired lower bounds, IEEE Transactions on Signal Processing 56 (2) (2008) 712–726.

[30] P. W. Wolniansky, G. J. Foschini, G. D. Golden, R. A. Valenzuela, V-BLAST: an architecture for realizing very high data rates over the rich-scattering wireless channel, Proceedings of the URSI International Symposium on Signals, Systems, and Electronics (ISSSE '98), (1998), 295-300.

[31] K. Su, I. Wassell, A new ordering for efficient sphere decoding, Proc. IEEE Int. Conf. Commun. (ICC) (2005), 1906-1910.