Improved Maximum Likelihood Detection through Sphere Decoding combined with Box Optimization

Victor M. Garcia-Molla[a,*], Antonio M. Vidal[a], Alberto Gonzalez[b,1], Sandra Roger[b]

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

Abstract

Sphere Decoding is a popular Maximum Likelihood algorithm that can be used to detect signals coming from multiple–input, multiple–output digital communication systems. It is well known that the complexity required to detect each signal with the Sphere Decoding algorithm may become unacceptable, especially for low signal–to–noise ratios. In this paper, we describe an auxiliary technique that drastically decreases the computation required to decode a signal. This technique was proposed by Stojnic, Hassibi and Vikalo in 2008, and is based on using continuous box-bounded minimization in combination with Sphere Decoding. Their implementation is, however, not competitive due to the box minimization algorithm selected. In this paper we prove that by judiciously selecting of the box minimization algorithm and tailoring it to the Sphere Decoding environment, the computational complexity of the resulting algorithm for low signal–to–noise ratios is better (by orders of magnitude) than standard Sphere Decoding implementations.

Keywords: MIMO communication systems, Sphere decoding, Box minimization

1. Introduction

This paper addresses the solution of the discrete linear least squares problem

$$\mathbf{s}_{opt} = \underset{\mathbf{s}\in\mathcal{D}^m\subset\mathbb{R}^m}{\arg\min} \|\mathbf{H}\cdot\mathbf{s}-\mathbf{y}\|^2 \ , \qquad (1)$$

*Corresponding author. Phone Number +34 963879723
Email addresses: vmgarcia@dsic.upv.es (Victor M. Garcia-Molla), avidal@dsic.upv.es (Antonio M. Vidal), agonzal@dcom.upv.es (Alberto Gonzalez), sanrova@iteam.upv.es (Sandra Roger)
[1]EURASIP member

where $\mathbf{H} \in \mathbb{R}^{n,m}$, $\mathbf{s} \in \mathbb{R}^m$, $\mathbf{y} \in \mathbb{R}^n$, $n \geq m$, and $\mathcal{D} \subset \mathbb{R}$. $\mathcal{D}$ is a finite constellation or alphabet, with cardinality L.

This problem has applications in cryptography [1], global positioning systems [2], etc. However, its main application is the detection of signals that are sent through a multiple-input multiple-output (MIMO) communication system. In this case, $\mathbf{H}$ is called the channel matrix, $\mathbf{s}$ is the sent signal, $\mathbf{y}$ is the received signal, and $\mathbf{s}_{opt}$ is the exact solution to this problem; $\mathbf{s}_{opt}$ is called the Maximum Likelihood (ML) solution.

Signal detection algorithms that can solve (1) exactly (i.e., obtain the ML solution) are known as ML algorithms. Considering only the accuracy of the signal detected, these are optimal algorithms since they compute the exact solution of the problem. However, all the algorithms in this category are computationally very expensive, especially in the low signal-to-noise ratio (SNR) regime.

There are many other detection algorithms (such as Zero Forcing (ZF) or MMSE [3]) that are faster, but the solution obtained might not be the ML solution; therefore these algorithms are known as suboptimal. Among the ML algorithms, the Sphere Decoding (SD) family of algorithms is the most popular [4, 5, 6, 7, 8, 9]. These algorithms perform a search in a tree of partial solutions; the size of the tree becomes the critical factor for the computational complexity. Many research papers have been written about this subject, describing all sorts of optimizations to enhance SD algorithms: geometrically based [10], increasing the radius [11], acceleration through ordering the components of the signal [12, 13], and many more. One of the best known versions of the algorithm, and one with a better performance, is the one that uses adaptive radius search and Schnorr-Euchner ordering of the elements of the constellation [6, 7].

One technique to reduce the size of the SD tree, which was described in [14], is based on computing lower bounds for some subproblems that arise during the search; several possible bounds were proposed and compared. One of these bounds was computed using quadratic minimization with box restrictions, also called box minimization in several references (polytope relaxation in [14]). However, the overall computational complexity of that technique was described as being not competitive (compared to the other techniques described in [14]). Similar discrete optimization techniques have been used also to improve the performance of non-ML methods, see for example [15, 16].

This paper, presents our findings after further research in the use of box minimization combined with SD. We have studied the existing box minimization algorithms in depth, and the possibilities to adapt them for joint use with SD. As a result we have adapted a box minimization algorithm that was proposed in [17]. By using the special characteristics of the problem, the cost of each minimization has been reduced to a small fraction of flops. Furthermore the number of nodes of the tree is greatly reduced (as was already discovered in [14]). The overall algorithm has been tested in terms of the number of nodes visited, the execution time and flops. Under any of these criteria, the overall complexity reduction for low SNR is very large, compared to Schnorr-Euchner Sphere Decoding (SD–SE).

The paper is organized as follows: Section 2 describes the problem in hand

and the box optimization technique. Section 3 presents the relation between the problem of box minimization and the Sphere Decoding algorithm. Section 4 discusses the chosen box optimization algorithm, and its adaptation to work with the SD algorithm. Then, Section 5 shows the complete detection algorithm proposed, and in Section 6 numerical results are provided to assess our algorithm compared with other SD algorithms.

2. Problem description

The detection of the signal sent through a MIMO digital system is obtained solving problem (1)(either exactly with a ML method, or approximately with a suboptimal method).

This detection problem is usually described in complex-valued form (i.e., $\mathbf{H} \in \mathbb{C}^{n,m}, \mathbf{s} \in \mathbb{C}^m, \mathbf{y} \in \mathbb{C}^n, \mathcal{D} \subset \mathbb{C}$). However, the procedure to transform problem (1) from complex-valued to real-valued form is easy and well known. On the other hand, some of the algorithms described in this paper require a real-valued formulation. Therefore, we will use the real-valued formulation throughout this paper.

The simplest suboptimal algorithm to solve (1) starts by removing the constraint over the components of $\mathbf{s}$ (that they must belong to $\mathcal{D}$ a finite subset of $\mathbb{R}$) and solving the continuous least squares problem:

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s} \in \mathbb{R}^m} \|\mathbf{H} \cdot \mathbf{s} - \mathbf{y}\|^2 \ . \tag{2}$$

Then, all the components of $\hat{\mathbf{s}}$ are rounded to the nearest element of the constellation $\mathcal{D}$ (this process is called quantization). The vector obtained after this process is $\hat{\mathbf{s}}_q$, known as the Zero-Forcing estimator. This estimator may be a good approximation to $\mathbf{s}_{opt}$ when the SNR is high, but it is known to give bad results if the noise increases. The computation of the $\hat{\mathbf{s}}$ vector requires the QR decomposition of the channel matrix, premultiplication of the received signal $\mathbf{y}$ by the matrix $\mathbf{Q}^T$, and solving a triangular system of equations. The vectors $\hat{\mathbf{s}}$ and $\hat{\mathbf{s}}_q$ will play an important role in the algorithms, described further on.

This paper is concerned mainly with the SD algorithm; which is a Branch–and–Bound algorithm that is adapted to the MIMO Detection problem. There are many versions of this algorithm. To apply this algorithm, it is necessary to transform problem (1) into an equivalent problem using the QR decomposition of the channel matrix:

$$\mathbf{s}_{opt} = \arg\min_{\mathbf{s} \in \mathcal{D}^m \subset \mathbb{R}^m} \|\mathbf{R} \cdot \mathbf{s} - \mathbf{z}\|^2 \ , \tag{3}$$

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

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 signals, 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

$$\|\mathbf{R} \cdot \mathbf{s} - \mathbf{z}\|^2 \leq r^2 \tag{4}$$

are discarded.

Expression (4) is used in the SD algorithm to detect components of the signal $\mathbf{s}$ one by one, starting with the last one, taking advantage of the structure of $\mathbf{R}$. For the first level of the tree (level m), this is done by rewriting $\mathbf{R}$, $\mathbf{z}$, and $\mathbf{s}$ as follows:

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{1:m-1,1:m-1} & \mathbf{R}_{1:m-1,m} \\ 0 & \mathbf{R}_{m,m} \end{bmatrix}, \mathbf{z} = \begin{bmatrix} \mathbf{z}_{1:m-1} \\ \mathrm{z}_m \end{bmatrix}, \mathbf{s} = \begin{bmatrix} \mathbf{s}_{1:m-1} \\ \mathrm{s}_m \end{bmatrix}, \tag{5}$$

where $\mathbf{R}_{m,m} \in \mathbb{R}^{1,1}$, $\mathbf{R}_{1:m-1,1:m-1} \in \mathbb{R}^{m-1,m-1}$, and $\mathbf{R}_{1:m-1,m} \in \mathbb{R}^{m-1,1}$.
Using this partition of the problem, inequality (4) is rewritten as:

$$\begin{gathered}\|\mathbf{R} \cdot \mathbf{s} - \mathbf{z}\|^2 = \|\mathbf{R}_{1:m-1,1:m-1} \cdot \mathbf{s}_{1:m-1} + \mathbf{R}_{1:m-1,m} \cdot \mathrm{s}_m - \mathbf{z}_{1:m-1}\|^2 + \\ \|\mathbf{R}_{m,m} \cdot \mathrm{s}_m - \mathrm{z}_m\|^2 \leq r^2 \ .\end{gathered} \tag{6}$$

From here, the pruning condition used in standard SD algorithms is obtained:

$$\|\mathbf{R} \cdot \mathbf{s} - \mathbf{z}\|^2 \leq r^2 \Rightarrow \|\mathbf{R}_{m,m} \cdot \mathrm{s}_m - \mathrm{z}_m\|^2 \leq r^2 \ . \tag{7}$$

Only the values of $s_m \in \mathcal{D}$ verifying (7) are feasible values, so only these values become nodes to be expanded.

In the level k of the tree $(1 < k < m)$, expression (4) is rewritten as:

$$\begin{gathered}\|\mathbf{R} \cdot \mathbf{s} - \mathbf{z}\|^2 = \|\mathbf{R}_{1:k-1,1:k-1} \cdot \mathbf{s}_{1:k-1} + \mathbf{R}_{1:k-1,k:m} \cdot \mathbf{s}_{k:m} - \mathbf{z}_{1:k-1}\|^2 + \\ \|\mathbf{R}_{k:m,k:m} \cdot \mathbf{s}_{k:m} - \mathbf{z}_{k:m}\|^2 \leq r^2 \ .\end{gathered} \tag{8}$$

Taking into account that the components $\mathbf{s}_{k+1:m}$ have already been assigned values, the general pruning condition for the component s_k is:

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

The selection of the initial radius r is an interesting problem. If the initial radius is too small, then there may be no signal verifying (4). If the radius is too large, then the number of signals verifying (4) may be very high, and the cost of the detection will increase. The radius should be as small as possible, but should also guarantee that there is at least one signal that verifies (4). The first solution that an adaptive SD–SE decoder should find is the zero–forcing decision–feedback–equalization (ZF–DFE) estimator or Babai point [9]. Therefore, the effective radius for an SD–SE decoder would be the radius computed using the ZF–DFE estimator. Another standard technique is to compute an initial radius using the ZF estimator $\hat{\mathbf{s}}_q$:

$$r_{zf} = \|\mathbf{H} \cdot \hat{\mathbf{s}}_q - \mathbf{y}\| \ . \tag{10}$$

The radius estimate computed withe the ZF–DFE is typically better than the computed with the ZF estimator. However, it shall be shown later that the ZF estimator is more relevant for the techniques described in this paper.

2.1. Distance of $\hat{\mathbf{s}}$ to the constellation space

Given a finite, m-dimensional constellation $\mathcal{D}^m$, we say that a vector $\mathbf{s} \in \mathbb{R}^m$ is in the constellation if

$$\min(\mathcal{D}) \leq \mathrm{s}_i \leq \max(\mathcal{D})\,; 1 \leq i \leq m\ . \tag{11}$$

Obviously, if any (or several) components of $\mathbf{s}$ are out of the interval $[\min(\mathcal{D})\ ,\ \max(\mathcal{D})]$ then we say that the vector is out of the constellation.

To define the distance of a vector $\mathbf{s}$ to the constellation, we previously define the vector $dist(\mathbf{s}, \mathcal{D})$ as an $\mathbb{R}^m$–vector whose i–th component is defined as

$$dist(\mathbf{s}, \mathcal{D})_i = \left\{ \begin{array}{ccc} 0 & if & \min(\mathcal{D}) \leq \mathrm{s}_i \leq \max(\mathcal{D}) \\ \mathrm{s}_i - \max(\mathcal{D}) & if & s_i \geq \max(\mathcal{D}) \\ \min(\mathcal{D}) - \mathrm{s}_i & if & \mathrm{s}_i \leq \min(\mathcal{D}) \end{array} \right. \ . \tag{12}$$

Then, the distance of the vector $\mathbf{s}$ to the constellation can be simply defined as the Euclidean norm of the vector $dist(\mathbf{s}, \mathcal{D})$.

It is well known that, given a channel matrix $\mathbf{H}$ and a received signal $\mathbf{y}$, the efficiency of the SD algorithm in decoding the signal depends mainly on two factors: the noise affecting the received signal, and the conditioning of the channel matrix. However, in a given problem, it may be difficult to determine whether both factors are present, or just one of them. We have observed that the *distance of the $\hat{\mathbf{s}}$ vector (the solution of the unconstrained least squares problem (2)) to the constellation space* usually gives more information about the performance of the SD. If the $\hat{\mathbf{s}}$ vector is in the constellation, the SD algorithm usually obtains the solution by expanding few nodes. On the other hand, if the $\hat{\mathbf{s}}$ vector is out of the constellation, then the number of nodes explored by the SD algorithm will be larger, and the number of nodes will grow (even exponentially) with the distance of $\hat{\mathbf{s}}$ to the constellation.

The cost of computing $\hat{\mathbf{s}}$ is quite small in a SD environment, since two of the operations needed should have already been carried out, i.e. the QR decomposition and the premultiplication of the received signal $\mathbf{y}$ by $\mathbf{Q}^T$.

2.2. Box minimization as an auxiliary technique

The main proposal in this work is to use continuous constrained minimization techniques to help SD algorithm, in the cases where the $\hat{\mathbf{s}}$ vector is out of the constellation. The auxiliary problem to be solved is:

$$\hat{\mathbf{sr}} = \arg\min_{\mathbf{s} \in \mathbb{R}^m} \|\mathbf{H} \cdot \mathbf{s} - \mathbf{y}\|^2\,, \min(\mathcal{D}) \leq s_i \leq \max(\mathcal{D})\ . \tag{13}$$

Compared to problem (1), this is a continuous problem: the components of the solution vector do not need to belong to $\mathcal{D}$; the only restriction is that the search zone be bounded.

The example in Figure 1 illustrates the situation. A simple 2×2 real-valued MIMO system, with constellation $\mathcal{D} = \{-3/2, -1/2, 1/2, 3/2\}$ is considered. The channel matrix is a 2×2 random real matrix.

The possible sent signals are chosen from the set $\mathcal{D}^2$, depicted as blue circles in subfigure 1a). The solution of the detection problem must be chosen from the set of possible sent signals. Subfigure 1b) depicts the possible received signals in absence of noise (i.e., the images of the possible sent signals) as black circles. The lattice formed by the possible sent signals is deformed through the effect of the channel matrix.

Due to the effect of noise, the received signal $\mathbf{y}$ will not coincide with any of the images of the possible sent signals (the black circles). Under some circumstances (high noise, and/or bad conditioning of the channel matrix), the estimator $\hat{\mathbf{s}}$ (obtained by solving the continuous unconstrained least squares problem (2)) may be out of the constellation, as in the situation portrayed in subfigure 1a).

To obtain a more reliable estimator, it may be better to solve problem (13). In the case of subfigure 1a), this amounts to solving the continuous least squares problem (2), but considering the solution only in the box delimited by the discontinuous red line. In this case the box is $[-3/2, 3/2] \times [-3/2, 3/2]$. Therefore, while $\hat{\mathbf{s}}$ can be located anywhere, the estimator $\hat{\mathbf{sr}}$ is forced to remain in the box.

It should be clear that, if $\hat{\mathbf{s}}$ is in the constellation, then $\hat{\mathbf{s}} = \hat{\mathbf{sr}}$. Thus, the computation of $\hat{\mathbf{sr}}$ only makes sense if $\hat{\mathbf{s}}$ is out of the constellation. Then, $\hat{\mathbf{sr}}$ will necessarily lie on the border of the box, and at least one of its components must take either the value $\max(\mathcal{D})$ or $\min(\mathcal{D})$ (this is an important fact that will be used in Section 4).

Since the bounds are fixed, this is usually described as a box minimization problem. Problem (13) can be solved much faster than (1).

There are several possibilities for using problem (13) to enhance MIMO Detection. A first proposal was described in [18], applied to the code-division multiple access (CDMA) multiuser detection problem. Another possibility is to quantize the estimator $\hat{\mathbf{sr}}$, giving the estimator $\hat{\mathbf{sr}}_{\mathbf{q}}$. This estimator has been proposed in several papers [19, 20] as an independent suboptimal estimator for low SNR detection since it performs quite well.

Another possibility proposed in [20] is the use of $\hat{\mathbf{sr}}_{\mathbf{q}}$ to compute an approximation to the initial SD radius;

$$r_{zfopt} = \|\mathbf{H} \cdot \hat{\mathbf{sr}}_{\mathbf{q}} - \mathbf{y}\| \ . \tag{14}$$

As reported in [20], in large noise situations this radius estimate is usually more accurate than the standard radius estimate using the ZF estimator (10). A similar technique is applied in [21], for the case where the constellation is $\mathcal{D} = \{-1, +1\}$. There, box minimization is combined with quantization in several ways, obtaining good results. However, the technique proposed cannot be applied to larger constellations.

Generally speaking, the solution of problem (13) is quite costly. However, in a SD environment, the cost can be reduced using the special features of the minimization problems that appear in the SD algorithm.

3. Combination of box minimization with SD

The way in which box minimization can be used to improve SD performance was first proposed and described in [14]; the proposal was to obtain a bound that is tighter than (9), by also using the remaining term in inequality (8),

$$\left\|\mathbf{R}_{1:k-1,1:k-1}\cdot\mathbf{s}_{1:k-1}+\mathbf{R}_{1:k-1,k:m}\cdot\mathbf{s}_{k:m}-\mathbf{z}_{1:k-1}\right\|^2\ . \tag{15}$$

This can be done by obtaining a lower bound c of this term, so inequality (8) can be written as:

$$\left\|\mathbf{R}_{k:m,k:m}\cdot\mathbf{s}_{k:m}-\mathbf{z}_{k:m}\right\|^2\le r^2-c\ , \tag{16}$$

which is a tighter pruning condition than (9). This should provide a reduction of the number of feasible values of s_k, and, consequently, in the number of visited nodes.

To obtain bounds for (15), it is useful to consider that this norm is equivalent to the one appearing in (4) but for a deflated MIMO detection problem where the channel matrix is $\mathbf{R}_{1:k-1,1:k-1}$, the received signal is $\mathbf{z}_{1:k-1}-\mathbf{R}_{1:k-1,k:m}\cdot\mathbf{s}_{k:m}$, and the signal to be detected is $\mathbf{s}_{1:k-1}$.

In [14] several methods to compute lower bounds of (15) were proposed, discussed and evaluated. The technique that gave the best results (in terms of overall flop reduction) was based on the minimum singular value of the submatrix $\mathbf{R}_{1:k-1,1:k-1}$. That proposal was also used and tested in [22].

One of the proposals in [14] was to use box minimization to compute a lower bound of (15). As mentioned above, this can be done considering (15) as a deflated MIMO detection problem. If the continuous least squares problem is solved:

$$\hat{\mathbf{s}}^{k-1}=\underset{\mathbf{s}_{1:k-1}\in\mathbb{R}^{k-1}}{\arg\min}\left\|\mathbf{R}_{1:k-1,1:k-1}\cdot\mathbf{s}_{1:k-1}+\mathbf{R}_{1:k-1,k:m}\cdot\mathbf{s}_{k:m}-\mathbf{z}_{1:k-1}\right\|^2 \tag{17}$$

the estimator $\hat{\mathbf{s}}^{k-1}$ is obtained, which is analogous to the estimator $\hat{\mathbf{s}}$ computed as in (2) but for the deflated problem. It must be noted that the problem (17) is actually a standard triangular system of linear equations, whose solution $\hat{\mathbf{s}}^{k-1}$ is computed exactly and fulfills:

$$\left\|\mathbf{R}_{1:k-1,1:k-1}\cdot\hat{\mathbf{s}}^{k-1}+\mathbf{R}_{1:k-1,k:m}\cdot\mathbf{s}_{k:m}-\mathbf{z}_{1:k-1}\right\|=0\ . \tag{18}$$

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

$$\begin{aligned}\widehat{\mathbf{sr}}^{k-1}=\underset{\mathbf{s}_{1:k-1}}{\arg\min}&\left\|\mathbf{R}_{1:k-1,1:k-1}\cdot\mathbf{s}_{1:k-1}+\mathbf{R}_{1:k-1,k:m}\cdot\mathbf{s}_{k:m}-\mathbf{z}_{1:k-1}\right\|^2\\ &\min(\mathcal{D})\le\mathrm{s}_i\le\max(\mathcal{D})\,;1\le i\le k-1\ .\end{aligned} \tag{19}$$

Problem (19) is analogous to (13) and can also be solved using box minimization techniques. The solution $\widehat{\mathbf{sr}}^{k-1}$ fulfills that, for all $\mathbf{s}_{1:k-1}\in\mathcal{D}^{k-1}$:

$$\begin{aligned} &\left\| \mathbf{R}_{1:k-1,1:k-1} \cdot \hat{\mathbf{sr}}^{k-1} + \mathbf{R}_{1:k-1,k:m} \cdot \mathbf{s}_{k:m} - \mathbf{z}_{1:k-1} \right\|^2 \\ \le &\left\| \mathbf{R}_{1:k-1,1:k-1} \cdot \mathbf{s}_{1:k-1} + \mathbf{R}_{1:k-1,k:m} \cdot \mathbf{s}_{k:m} - \mathbf{z}_{1:k-1} \right\|^2 . \end{aligned} \tag{20}$$

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

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

in inequality (16).

Of course, if $\hat{\mathbf{s}}^{k-1}$ is in the constellation, $\hat{\mathbf{s}}^{k-1} = \hat{\mathbf{sr}}^{k-1}$, and thus the bound would be useless since, like in (18),

$$\left\| \mathbf{R}_{1:k-1,1:k-1} \cdot \hat{\mathbf{sr}}^{k-1} + \mathbf{R}_{1:k-1,k:m} \cdot \mathbf{s}_{k:m} - \mathbf{z}_{1:k-1} \right\|^2 = 0 . \tag{22}$$

In this case, it would be better to use other bounding techniques, (such as the technique based on the minimum singular value described in [14]) or, simply not to use any additional bound, since the standard SD–SE algorithm performs quite well in this case.

As shown in section 6, the bound (21) proposed in [14] obtains excellent results in terms of visited nodes. However, in [14], this technique was discarded because the reported overall complexity (in terms of flops and time) of the algorithm was not competitive. In our view, there are two reasons why this proposal was discarded in [14]: first, the box minimization was carried out using a standard Matlab [23] routine (**quadprog**), which cannot take into account the special characteristics of the detection problem and of the SD algorithm; and, second, the constellation used for the tests was a simple constellation $\{-1, 1\}$ with only two elements; as shall be seen in section 6, this technique is really beneficial for larger constellations.

We will show in section 6 that this bound is indeed competitive, no matter if it is examined in terms of visited nodes, computing times, or flops. In section 4 we will study how to tailor an specific box minimization algorithm to be used with the sphere decoding algorithm.

4. Linear least squares problems with inequality constraints

There exist several algorithms for the solution of linear least squares problems with inequality constraints. These are usually common minimization methods (Maximum Descent, Newton's method, Quasi-Newton's methods, ...) adapted to start from a feasible point (i.e., a point that verifies the inequality constraints, or, in our problem, a point in the constellation) and where the minimization proceeds through "descent directions". The length of the steps taken through the descent directions is chosen to guarantee that the new point remain in the constellation. In the following, we use standard minimization terminology, denoting as feasible a point that satisfies the inequality constraints, or, equivalently, a point that is in the constellation.

First, the special characteristics of box minimization combined with SD that can be used to reduce significantly the computational cost should be mentioned. When expanding a node, at most L new nodes will be obtained. To determine which of these L nodes must be expanded, inequality (7) must be evaluated for the candidate nodes. For the nodes satisfying (7), the tighter inequality (16) must be evaluated to check whether this node must be finally expanded. Therefore, at most, L instances of the minimization problem (19) must be solved, although the number will be usually smaller. The cost of these minimizations can be reduced selecting carefully the starting point. Indeed, it is well known that in most iterative minimization methods, a good choice for the starting point (a starting point close to the solution) usually reduces the computing cost. This is especially relevant in this case for the minimizations in each level k, the constrained minimum coming from the previous level $\hat{\mathbf{sr}}^k$ provides an excellent starting point by simply selecting the $k-1$ first components.

We have tested several basic minimization algorithms with relatively good results. Thanks to the selection of the starting point outlined above, we have adapted an algorithm described in the book [17, chap. 2] by Ake Björck, so that the resulting algorithm works very well combined with SD. A summarized version is included in subsection 4.1 (see [17] for the extended version).

4.1. Box minimization algorithm adapted to SD

As described above, in our proposed version of the SD algorithm, a box minimization problem (13) must be solved to decide (using inequality (16)) whether a level k node must be expanded. Recall that problem (13) must be solved only if the local estimator $\hat{\mathbf{s}}^{k-1}$ is out of the constellation, and that, in this situation, there will be several components in $\hat{\mathbf{sr}}^{k-1}$ that take the extreme values of the constellation, $\max(\mathcal{D})$ or $\min(\mathcal{D})$. Intuitively, this means that the solution $\hat{\mathbf{sr}}^{k-1}$ is located in one "side" of the box.

For any feasible point $\mathbf{x} \in \mathbb{R}^{k-1}$, its components can be divided into two sets: those that take the extreme values of the constellation, (i.e., those that place $\mathbf{x}$ be on a side) and those that take other internal values. Expressed more rigorously, this means that for any feasible point $\mathbf{x}$, there will exist a set of components $B = \{i_1, i_2, ...\}$ such that, for all $i \in B$, $\mathrm{x}_i = \max(\mathcal{D})$ or $\mathrm{x}_i = \min(\mathcal{D})$. This splits the index of components $\{1, 2..., k-1\}$ into two sets, B and F, where for all $i \in F$, $\min(\mathcal{D}) < \mathrm{x}_i < \max(\mathcal{D})$. B is called the set of active constraints or components, whereas F is called the set of free constraints or components.

4.1.1. Minimization algorithm with known active constraints

Suppose that the final solution $\hat{\mathbf{sr}}^{k-1}$ is not known but that the set of active constraints B associated to $\hat{\mathbf{sr}}^{k-1}$ is known, as well as the values of the active variables (this is equivalent to know which side of the box $\hat{\mathbf{sr}}^{k-1}$ lies on, but the exact position of $\hat{\mathbf{sr}}^{k-1}$ on that side is not yet known). From here we can obtain the solution $\hat{\mathbf{sr}}^{k-1}$ straight away, as follows:

Let $\mathbf{x_B}$ be the (known) values of the active restrictions.

1. Split the matrix $\mathbf{R}$ into two matrices $\mathbf{R}_B$ and $\mathbf{R}_F$; $\mathbf{R}_B$ is composed of the columns of $\mathbf{R}$ whose index is in B, and $\mathbf{R}_F$ is similarly defined; in Matlab notation: $\mathbf{R}_B = \mathbf{R}(:, B); \mathbf{R}_F = \mathbf{R}(:, F)$. Also split $\mathbf{x}$ accordingly into $\mathbf{x}_B$ and $\mathbf{x}_F$.
2. $\mathbf{z_{aux}} = \mathbf{z} - \mathbf{R}_B \cdot \mathbf{x}_B$.
3. Solve a reduced least squares problem, only for the variables in F: $\mathbf{x}_F = \mathbf{R}_F \backslash \mathbf{z_{aux}}$.
4. The solution $\hat{\mathbf{sr}}^{k-1}$ is built by joining $\mathbf{x}_B$ and $\mathbf{x}_F$ together, according to the indexes B and F. We express this operation as: $\hat{\mathbf{sr}}^{k-1} = \mathbf{x}_B \cup \mathbf{x}_F$.

The main cost of this algorithm is the solution of the reduced least squares problem in step 3. For general matrices in $\mathbb{R}^{m,m}$ the theoretical cost of each solution is $O(m^3)$. However, by using the previous triangular structure of $\mathbf{R}$ and applying a QR update using Givens rotations [24] to triangularize $\mathbf{R}_F$, the cost is reduced to a small fraction. This algorithm allows the creation of zeros in selected locations of the matrix. Since the columns that will form part of $\mathbf{R}_F$ are chosen from a triangular matrix, they already have many zeros that do not need to be processed.

For an efficient implementation, it is important to note that 1) only $\mathbf{R}_F$ must be triangularized, and 2) when forming $\mathbf{R}_F$, it is important to keep the relative order in which the columns were located in the original matrix. This is so because the closer a matrix is to being triangular, the less Givens rotations will be needed to actually make it triangular.

Since the actual cost of this QR update depends on the sets of active/free variables, which in turns depends on the received signal, it is quite difficult to give a priori estimations or bounds of the number of flops. Instead, we have chosen to count the actual flops just by slightly modifying the code, by including flop counters.

4.1.2. Complete Minimization algorithm

The algorithm in the previous section proves that an accurate determination of the set of active constraints associated to the solution brings a fast solution to the box minimization problem.

Algorithms seeking to determine the set of active constraints are called "active set algorithms". They interchange variables between the sets B and F until the active set for the optimal solution is determined. The following algorithm, proposed in [17], belongs to this family:

Step 1. Start from a feasible point $\mathbf{x^0}$; determine the sets B and F. Split $\mathbf{x^0}$ according to the sets, in $\mathbf{x^0}_B$ and $\mathbf{x^0}_F$.

Step 2. Solve a reduced least squares problem, only for the variables in F:

$$\begin{gathered} \mathbf{R}_B = \mathbf{R}(:, B); \mathbf{R}_F = \mathbf{R}(:, F) \\ \mathbf{z_{aux}} = \mathbf{z} - \mathbf{R}_B \cdot \mathbf{x^0}_B \\ \mathbf{x^t}_F = \mathbf{R}_F \backslash \mathbf{z_{aux}} \ . \end{gathered} \tag{23}$$

Step 3. If the new tentative solution $\mathbf{x^t} = \mathbf{x^0}_B \cup \mathbf{x^t}_F$ is not feasible (out of the constellation), then the line going from $\mathbf{x^0}$ to $\mathbf{x^t}$ is computed. Then, the new solution moves along the line as far as possible, in direction of $\mathbf{x^t}$ while still remaining feasible. The solution obtained is renamed again $\mathbf{x^0}$; this process moves at least a component from F to B, so that these sets must be updated. Then, the algorithm goes back to step 2.

Step 4. If the new tentative solution, $\mathbf{x^t} = \mathbf{x^0_B} \cup \mathbf{x^t_F}$ is feasible (in the constellation), the Lagrange multipliers (gradient of the active variables) are examined to check optimality. If the solution is optimal, $\hat{\mathbf{sr}}^{k-1} = \mathbf{x^t}$ and the algorithm finishes. If the solution is not optimal, the component that causes the non-optimality is sent from B to F, $\mathbf{x^0} = \mathbf{x^t}$, and the algorithm goes back to step 2.

In the following we discuss some of the details of this algorithm.

– The main cost per iteration (steps 2-4) is caused by the solution of a linear unconstrained least squares problem in step 2. The algorithm to solve this problem is the based on a QR update with Givens rotations, which is outlined in the previous section.

–For a given feasible point $\mathbf{x}$, the Lagrange multipliers are computed as the gradient of the problem $\mathbf{R}^T \cdot (\mathbf{z} - \mathbf{R} \cdot \mathbf{x})$. If the Lagrange multipliers corresponding to active variables are positive, the point $\mathbf{x}$ is optimal.

–In each iteration of steps (2-4) a component goes from B to F or from F to B. If the initial active set B is very different from the final one, then the number of iterations required for convergence will be quite large. Fortunately, this problem can be resolved because of the special features of the SD algorithm. Since in all levels, a minimization problem has probably been solved in the previous level, the solution of that minimization problem provides an excellent starting point for the minimizations needed in the lower levels.

To describe this procedure precisely, suppose that the SD algorithm is in the level k; then, it must be determined if the node corresponding to s_k should be expanded or not. Let us also suppose that the inequality (7) is verified, so inequality (16) must be checked to determine if the present node is expanded. Therefore, the box minimization algorithm must be executed.

Previously, in the $k+1$ level, either $\hat{\mathbf{s}}^k$ or $\hat{\mathbf{sr}}^k$ must have been computed (or both). If $\hat{\mathbf{sr}}^k$ has been computed, then the initial point for the box minimization $\mathbf{x^0}$ is selected as the first $k-1$ components of $\hat{\mathbf{sr}}^k$; if $\hat{\mathbf{sr}}^k$ has not been computed, then the initial point $\mathbf{x^0}$ is selected as the first $k-1$ components of $\hat{\mathbf{s}}_k$.

This initial point selection works well because the subproblems in level $k-1$ are just projections of the subproblem in the previous level k, so the variables that belong to B in the estimator $\hat{\mathbf{sr}}_k$ are very likely to belong to B also in the estimators $\hat{\mathbf{sr}}_{k-1}$ of the next level. We will try to provide some insight with the example shown in Figure 2.

Figure 2 (left) shows a typical situation, in which the remaining subproblem is three dimensional and the constellation is $\mathcal{D} = \{-3/2, -1/2, 1/2, 3/2\}$. The discontinuous lines mark the limits of the constellation. In this case, the $\hat{\mathbf{s}}^3$ estimator is out of the constellation, which means that the $\hat{\mathbf{sr}}^3$ estimator is located on a side of the constellation (usually, on the side closest to $\hat{\mathbf{s}}^3$). In this case, the components of the $\hat{\mathbf{sr}}^3$ would be $x = 3/2, y = 3/2$ and the value of z would be between $-3/2$ and $3/2$. Let this value be called z_0.

Let us suppose that the next component whose values will be examined is the y component. The y component has four possible values, all of which are associated to a two–dimensional problem. These problems are obtained projecting the three–dimensional problem over each possible value of y. If it were necessary to solve minimization problems, the initial minimization point $\mathbf{x^0}$ for all of them would be $(x = 3/2, z = z_0)$.

One of these subproblems, for $y = 1/2$, is shown in Figure 2 (right). The associated subproblems for the other possible values of y $\{3/2, -1/2, -3/2\}$ should look quite similar. Clearly, the desired solution for the minimization subproblem ($\hat{\mathbf{sr}}^2$) would be located very close to the initial minimization point and on the same side of the constellation. When this happens, the minimization algorithm converges in a single iteration as shown in 4.1.1.

In section 6 we will show empirically that in a very high percentage of the cases, the initial vector $\hat{\mathbf{sr}}_{1:k-1}$ or $\hat{\mathbf{s}}_{1:k-1}$ is already on the correct side, so the box minimization algorithm converges in a single iteration.

5. Modified SD algorithm

In this section, we put together all the ingredients of the proposed new SD algorithm. We have taken the standard Schnorr-Euchner adaptive radius Sphere Decoding as basic algorithm, and have carried out several modifications to include our proposals. We have included an initial radius estimate based on the $\hat{\mathbf{s}}$ estimator (if $\hat{\mathbf{s}}$ is in the constellation) or in the $\hat{\mathbf{sr}}$ estimator (if $\hat{\mathbf{s}}$ is out of the constellation) .

Either $\hat{\mathbf{s}}$ or $\hat{\mathbf{sr}}$ must be propagated down in the tree to be used as starting points for the minimizations.

First, we show the modifications made to the preprocessing needed for the SD–SE:

Algorithm 1. *Preprocessing for SD–SE with minimization bound.*

```
1.  Computation of the QR decomposition of the channel matrix $\mathbf{H}$
    (which may already be computed).
2.  Computation of $\mathbf{z} = \mathbf{Q}^T \cdot \mathbf{y}$.
3.  Computation of $\hat{\mathbf{s}} = \mathbf{R} \backslash \mathbf{z}$.
4.  If $\hat{\mathbf{s}}$ is in the constellation,
5.      set initial radius as: $r_{ini} = \|\mathbf{R} \cdot \hat{\mathbf{s}}_q - \mathbf{z}\|$.
6.  End If
7.  If $\hat{\mathbf{s}}$ is out of the constellation
8.      compute $\hat{\mathbf{sr}}$.
9.      set initial radius as: $r_{ini} = \|\mathbf{R} \cdot \hat{\mathbf{sr}}_q - \mathbf{z}\|$.
10. End If
```

If the distance of $\hat{\mathbf{s}}$ to the constellation is larger than zero but small, the radius computed using $\hat{\mathbf{s}}_q$ may be smaller than the radius computed using $\hat{\mathbf{sr}}_{\mathbf{q}}$. In this case, the simplest possibility is to compute both radii and use the smallest.

Now we turn to the phase where the tree is traversed. The algorithm is a Schnorr-Euchner adaptive radius sphere decoding. We assume an algorithm using recursion over the tree levels k, where the algorithm starts on level m and ends on level 2 (level 1 would be unnecessary, since the remaining subproblems are unidimensional). We focus the description on the modifications in the logic that are needed to determine whether or not a node is expanded:

Algorithm 2. *SD–SE search with minimization bound. Procedure to expand a node at level k*

```
1.  For each $\mathrm{s}_k \in \mathcal{D}$ (ordered using Schnorr-Euchner)
2.      If condition (7) is verified
3.          compute $\hat{\mathbf{s}}^{k-1}$ solving (17)
4.          If $\hat{\mathbf{s}}^{k-1}$ in the constellation
5.              expand node corresponding to $\mathrm{s}_k$
6.          else
7.              compute $\hat{\mathbf{sr}}^{k-1}$ through box minimization
8.              compute $c$ as in (21)
9.              If condition (16) is verified
10.                 expand node corresponding to $\mathrm{s}_k$
11.             Else
12.                 discard $\mathrm{s}_k$
13.             End If
14.         End If
15.     Else
16.         discard $\mathrm{s}_k$
17.         Exit For loop
18.     End If
19. End For
```

In each node that is not pruned by the standard condition, either $\hat{\mathbf{sr}}^{\mathbf{k}}$ or $\hat{\mathbf{s}}^{\mathbf{k}}$ are computed. These vectors must be propagated down in the tree to be used as the initial values of the minimizations that are needed on the lower levels of the tree.

6. Numerical results

MIMO Detection algorithms are usually evaluated in terms of accuracy (computing the Bit-Error-Rate, BER vs. SNR) and in terms of efficiency (where the units can be the number of flops, the number of visited nodes, or the execution times, in seconds). In this case, since the proposed method is a ML method, its BER curve would look exactly like the BER curve of any ML method, which is not too interesting. Therefore, we will concentrate on the efficiency of the method. All the computing times were obtained with Matlab implementations, running on a Core 2 computer, with 4 Gb of RAM, using Matlab v 11.0. The flops were recorded by modifying the codes to include flop counters throughout the algorithm.

The experiments have been carried out for SNRs between 1 and 30 dBs. The noise variance σ^2 was chosen to obtain the desired SNR, taking into account that a SNR of k dBs corresponds to $\frac{P}{\sigma^2} = 10^{k/10}$. P represents the signal power at the receiver. Then, each signal to be detected $\mathbf{y}$ was generated as $\mathbf{y} = \mathbf{H} \cdot \mathbf{s} + \mathbf{n}$, where the entries of the vector $\mathbf{s}$ were randomly selected from the elements of the constellation used, each entry of the channel matrix $\mathbf{H}$ was independently drawn from a complex $N(0, 1)$ distribution, and each component of the noise vector $\mathbf{n}$ was independently drawn from a complex $N(0, \sigma^2)$ distribution.

Finally, the visited nodes were counted taking into account all the nodes whose partial distance had been computed, not just those that had been expanded.

6.1. Comparison 1

Our first experiment was designed using 4×4 complex matrices and a 64-QAM constellation (which were transformed into a real model, of dimension 8×8, with constellation of cardinality 8). Experiments were carried out varying the SNR from 1 to 30, generating 50 channel matrices for each SNR and detecting 50 signals with each matrix. For each SNR, the average number of visited nodes, the average computing times, and the average number of flops were recorded. In this initial experiment, we compare three implementations of the SD–SE: the proposed SD–SE version with minimization bound, described in section 5; the version proposed in [14] and used in [22], using a bound based on the minimum singular value; and a standard adaptive radius SD–SE decoder.

Figure 3 clearly shows the large improvement obtained in terms of flops, using this new bound. The actual figures in flops, computing times, and visited nodes are shown in Table 1 for some of the SNRs in the experiment. Although the minimization algorithm may seem costly and complicated, figures comparing flops and computing times make clear that the reduction in number of nodes is large enough to compensate the cost of the minimizations.

It is important to note that the average complexity remains virtually constant across all the SNRs tested.

The same experiment was carried out, but recording different data, in order to check the efficiency of the box minimization algorithm proposed. The results are summarized in Figure 4.

The black line shows the average number of minimizations per detected signal. For low SNR, this figure oscillates between 25 and 35, which may seem too costly. However the overall reduction in visited nodes is so important that the global number of flops ends up being much smaller.

The blue line shows the maximum number of inner iterations of the minimization algorithm (steps 2–4 of the minimization algorithm in section 4.1.2), which coincides with the maximum number of linear systems solved for each minimization. In this case (where the number of dimensions of the minimization problem solved is 8), the maximum number of iterations is 4.

Finally, the red line shows the average number of inner iterations per minimization. The average number hardly reaches 1.5 for $SNR = 1$, decreasing towards 1 when the SNR increases. This means that, in a vast majority of the cases, the minimization algorithm converges to the solution in a single iteration. This happens only when the initial set of active variables is the right one, converging to the solution in a single iteration. This is clearly a consequence of using the first components of the estimator $\hat{\mathbf{sr}}^k$ as starting point for the minimizations in the level $k-1$.

6.2. Comparison 2

The effect of the minimization bound becomes more important when the size of the problem increases. Experiments like the one described in the subsection 6.1 were carried out, increasing the size of the problem to 6×6, (12×12 in the real-valued version). The difference between the SD–SE with minimization bound and the standard SD–SE was, in this case, too large, and the running times for Standard SD–SE were too long to carry out meaningful simulations. Therefore, we show only the results comparing the SD–SE versions with minimization bound versus the version with singular value bound. To obtain reasonable running times, we limited the number of experiments for each SNR, to 20 matrices and 20 signals decoded with each matrix. Only the computation of the case with $SNR = 1$ with the singular value bound lasted more than 2 hours; simulations with Standard SD–SE would have been far longer. The results are shown in table 2.

6.3. Comparison 3, increasing the dimension of the problem

Since the proposed algorithm can handle large problems with ease, it is natural to consider how it behaves when the size of the problem is increased. To this end, we carried out experiments with fixed SNR, increasing the size of the problem (increasing the number of antennas). The constellation was fixed to 64–QAM. The sizes of the matrices has varied from 5×5 to 16×16; these were the sizes of the complex-valued problems, which were cast into real-valued problems with double dimension (from 10×10 to 32×32). Two experiments were conducted, one with SNR=5 and other with SNR=20, generating 20 channel matrices for each size and detecting 20 signals with each matrix. The results are shown in Figure 5.

The results show an unusual property; when the size of the problem grows, the proposed algorithm performs better in presence of noise. This shows how effective the minimization bound is in chopping the tree, in situations with small SNR.

Another interesting point is that there are still some well–known techniques that can be applied to enhance the algorithm, such as reordering of the columns of the channel matrix (no reordering has been done in these experiments).

6.4. Discussion of the results

The application of the proposed technique delivers an important cost reduction. The keys to these results are the reduction in number of nodes (already reported in [14]) and the reduced number of average flops per node obtained with our implementation. However, Figure 5 shows that, when the number of antennas grows, the complexity of the search still grows exponentially (although the complexity grows much more without applying the technique proposed in this paper). So, while the proposed procedure allows ML detection for a relatively large number of antennas, even well–tuned implementations may take too long to decode a signal if the number of antennas is too large.

On the other hand, the number of simulations used for this study is small, causing irregular performance that can be observed in some points of Figure 3. However, as mentioned above, the time cost of these simulations is very high. Furthermore, the tests were designed to assess the difference in performance between the SD method with and without the new technique. From this point of view, we believe that the simulations prove clearly the effect of the technique proposed in this paper.

7. Conclusion

The results show that the algorithm proposed in this work generates a huge improvement over other known versions of SD–SE algorithms, when the constellations used are large enough. It presents a nearly constant complexity average over all the SNRs considered, meaning that it is virtually insensitive to noise. Furthermore, the technique proposed makes it possible to consider problems of larger sizes. In addition, there is room for even further improvement, since standard optimizations (such as reorderings) have not been used. It must be noted that the proposed algorithm may be especially appropriate for application in soft SD detectors.

Standard SD is usually not considered to be viable for practical applications, since signals may appear that are very costly to detect (even in low noise situations). The algorithm proposed prevents the appearance of these cases. Therefore, we believe that this is an interesting step towards a real use of Maximum Likelihood detection in practical applications.

8. Acknowledgements

This work has been partially funded by Universitat Politècnica de València through *Programa de Apoyo a la Investigación y Desarrollo de la UPV (PAID-06-11) and (PAID-05-12)*, by Generalitat Valenciana through projects *PROMETEO/2009/013* and *Ayudas para la realización de proyectos de I+D para grupos de investigación emergentes GV/2012/039*, and by Ministerio Español de Economía y Competitividad through project *TEC2012-38142-C04*.

References

[1] D. Micciancio, S. Goldwasser, Complexity of Lattice Problems, Springer, (2002).

[2] A. Hassibi, S. Boyd, Integer parameter estimation in linear models with applications to GPS, IEEE Transactions on Signal Processing, 46 (1998) 2938–2952.

[3] B. Hassibi, An efficient square-root algorithm for BLAST, in: ICASSP00: Proceedings of the Acoustics, Speech, and Signal Processing, 2000. IEEE International Conference, 2 (2000), II737–II740.

[4] R. Kannan, Improved algorithms for integer programming and related lattice problems, ACM Symp. Theory of Computing (1983).

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

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

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

[8] B. Hassibi, H. Vikalo, On sphere decoding algorithm. I. expected complexity, IEEE Transactions on Signal Processing, 53 (2005) 2806–2818.

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

[10] C.Z.W. Hassell, J.S. Thompson, Orthotope sphere decoding and parallelotope decoding- reduced complexity optimum detection algorithms for MIMO channels, Signal Processing, 86 (2006) 1518–1537.

[11] W. Zhao, G. Giannakis, Sphere Decoding Algorithms With Improved Radius search, IEEE Transactions on Communications, 53 (7) (2005) 1104–1110.

[12] G.J. Foschini, G.D. Golden, F. Reinaldo, A. Valenzuela, P.W. Wolniansky, Simplified Processing for High Spectral Efficiency Wireless Communication Employing Multi-Element Arrays, IEEE Journal On Selected Areas in Communications, 17 (11) (1999), 1841–1852.

[13] K. Su, I. J. Wassell, A New Ordering for Efficient Sphere. Decoding, International Conference on Communications (2005).

[14] 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.

[15] R.C. de Lamare, R. Sampaio-Neto, Minimum Mean Squared Error Iterative Successive Parallel Arbitrated Decision Feedback Detectors for DS-CDMA Systems, IEEE Trans. on Commun., 56 (5) (2008) 778 – 789.

[16] P. Li, R.C. de Lamare, R. Fa, Multiple Feedback Successive Interference Cancellation Detection for Multiuser MIMO Systems, IEEE Transactions on Wireless Communications, 10 (8)(2011), 2434–2439.

[17] A. Björck, Numerical Methods for Least Squares Problems, SIAM, Philadelphia, 1996.

[18] A. Yener, R.D. Yates, S. Ulukus, CDMA Multiuser Detection: A Nonlinear Programming Approach, IEEE Transactions on Communication 50 (6) (2002) 1016–1024.

[19] S. Park, H. Zhang, J. Kim, E.S. Kang, W. Hager, A Fast Suboptimal Algorithm for Detection of 16-QAM Signaling in MIMO Channels, in: Help Military Communications Conference, 2007, MILCOM 2007, IEEE International Conference.

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

[21] Y. Zakharov, J. Luo, C. Kasparis, Joint box-constraint and deregularization in multiuser detection, in: Proceedings of the European Signal Processing Conference, EUSIPCO'2006, Florence, Italy.

[22] J. Maurer, J. Jalden, D. Seethaler, G. Matz, Achieving a continuous diversity-complexity tradeoff in wireless MIMO systems via pre-equalized Sphere-Decoding, IEEE Journal of Selected Topics in Signal Processing, 3 (6) (2009) 986–999.

[23] The Mathworks Inc., MATLAB R14 Natick MA (2004).

[24] G.H. Golub and C.F. Van Loan. Matrix Computations. The Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996.

Captions:

Caption for figure 1: Position of the $\hat{\mathbf{s}}$ and $\hat{\mathbf{sr}}$ estimators.

Caption for figure 2: Three dimensional problem, and its projection to one of its four two dimensional subproblems.

Caption for figure 3: Average number of flops per decoded signal.

Caption for figure 4: Average number of minimizations per detected signal; Maximum number of inner iterations per minimization; Average number of inner iterations per minimization.

Caption for figure 5: (a)Average computing times; (b)Average flops.

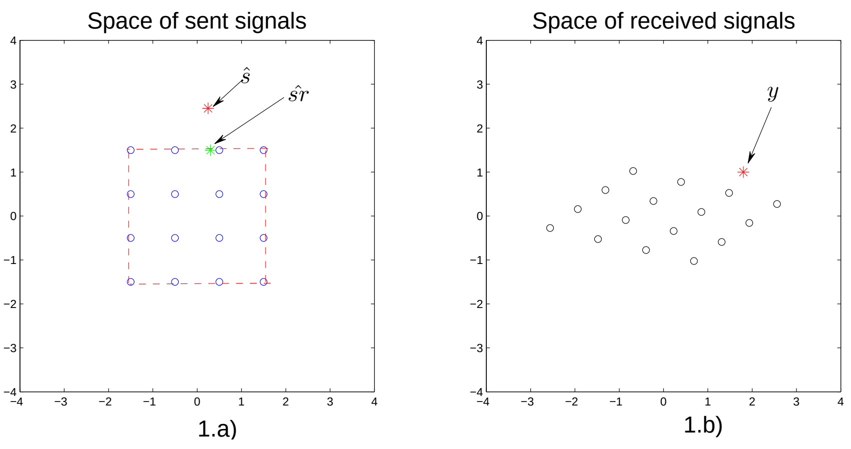
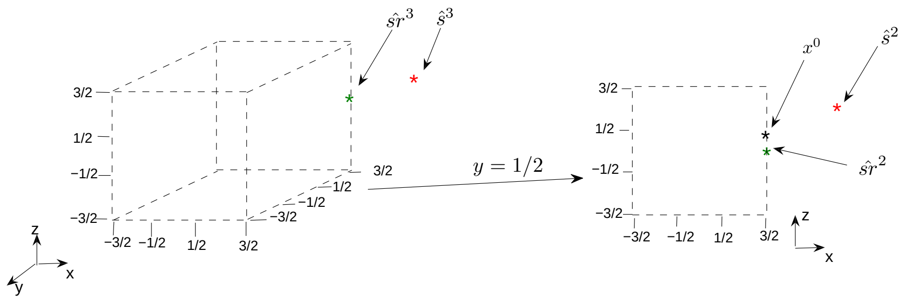

Space of sent signals
$\hat{s}$
$\hat{sr}$
1.a)
Space of received signals
y
1.b)

Figure 1:



$\hat{sr}^3$
$\hat{s}^3$
3/2
1/2
−1/2
−3/2
−3/2
−1/2
1/2
3/2
3/2
1/2
−1/2
−3/2
z
x
y
$y = 1/2$
x^0
$\hat{s}^2$
$\hat{sr}^2$
3/2
1/2
−1/2
−3/2
−3/2
−1/2
1/2
3/2
z
x

Figure 2:

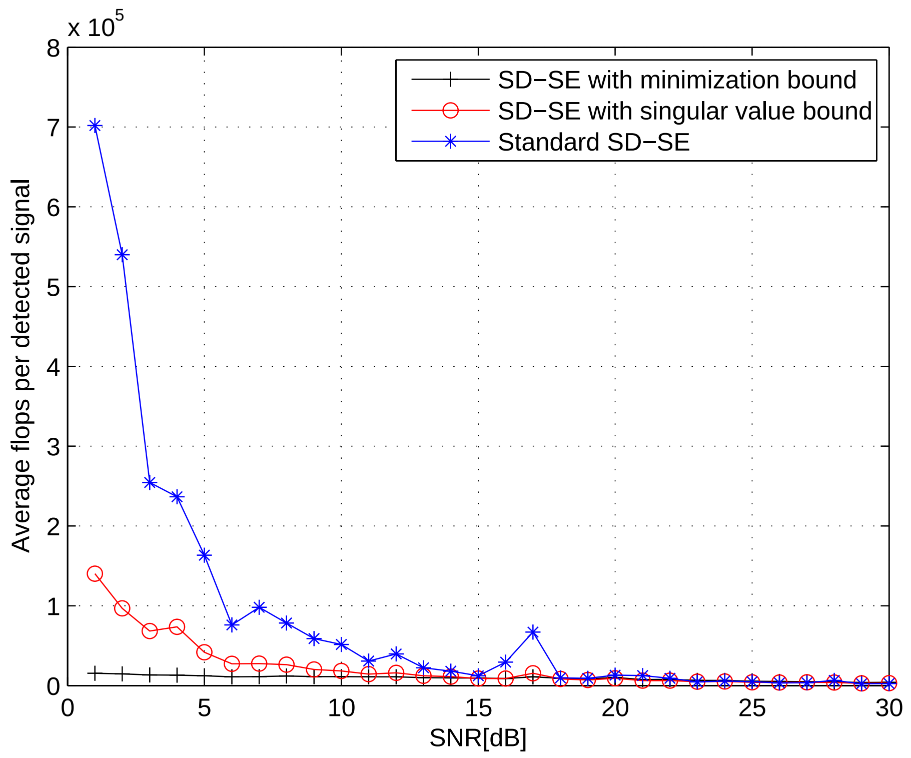

x 10^5
Average flops per detected signal
SNR[dB]
SD−SE with minimization bound
SD−SE with singular value bound
Standard SD−SE
0
1
2
3
4
5
6
7
8
0
5
10
15
20
25
30

Figure 3:

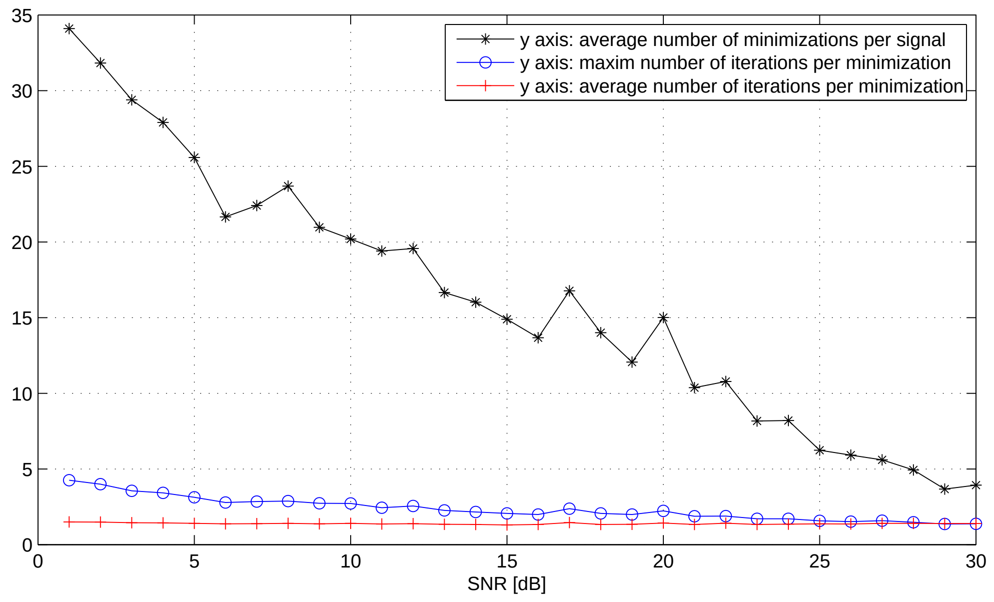
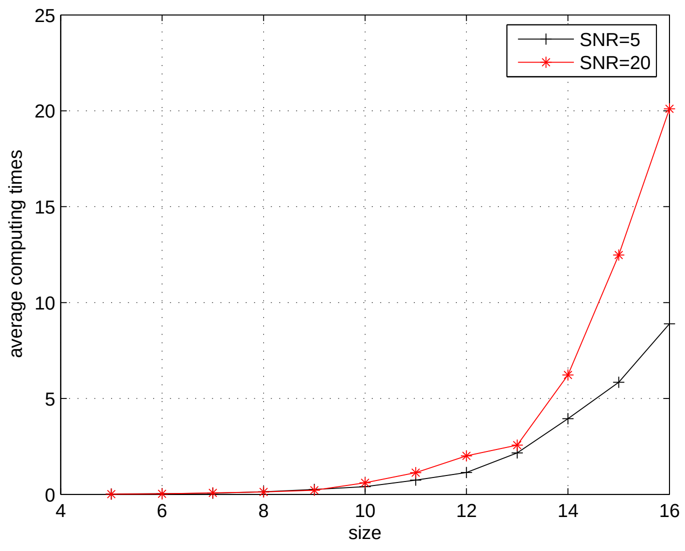
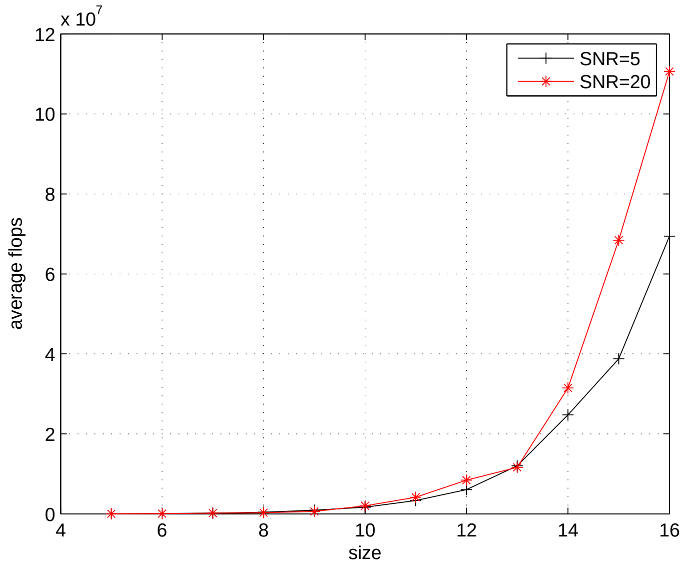

35
30
25
20
15
10
5
0
0
5
10
15
20
25
30
SNR [dB]
y axis: average number of minimizations per signal
y axis: maxim number of iterations per minimization
y axis: average number of iterations per minimization

Figure 4:



25
20
15
10
5
0
4
6
8
10
12
14
16
average computing times
size
SNR=5
SNR=20

(a)



x 10^7
12
10
8
6
4
2
0
4
6
8
10
12
14
16
average flops
size
SNR=5
SNR=20

(b)

Figure 5:

Table 1: Average flops, times and visited nodes for detection of a signal in a 4x4 complex MIMO system with constellation 64–QAM.

SNR	Minimiz. Bound			Min. s. v. Bound			Standard		
	flops	nodes	time	flops	nodes	time	flops	nodes	time
1	1.5e4	1.2e2	1.5e-2	1.4e5	4.5e3	2.5e-1	7.0e5	4.6e4	6.6e-1
3	1.3e4	1.3e2	1.3e-2	6.8e4	2.3e3	1.2e-2	2.5e5	1.7e4	2.5e-1
6	1.0e4	1.5e2	1.1e-2	2.7e4	9.6e2	4.7e-2	7.6e4	5.2e3	7.9e-2
10	1.0e4	1.8e2	1.1e-2	1.8e4	6.8e2	3.1e-2	5.1e4	3.5e3	5.4e-2
14	9.6e3	2.1e2	1.0e-2	1.1e4	4.2e2	1.8e-2	2.2e4	1.2e3	1.9e-2
18	9.1e3	2.2e2	1.0e-2	8.2e3	3.1e2	1.3e-2	6.7e4	6.7e2	1.0e-2
22	7.8e3	1.8e2	8.2e-3	6.0e3	2.1e2	9.2e-3	9.0e3	6.0e2	9.4e-3
26	5.2e3	1.2e2	5.3e-3	3.9e3	1.2e2	5.2e-3	3.2e3	1.8e2	3.2e-3
30	4.0e3	8.6e1	3.8e-3	3.0e3	8.6e1	3.4e-3	2.5e3	1.1e2	2.0e-3

Table 2: Average flops, times(seconds) and visited nodes for detection of a signal in a 6×6 complex MIMO system with constellation 64–QAM.

SNR	Minimiz. Bound			Min. s. v. Bound		
	flops	nodes	time	flops	nodes	time
1	8.7e4	2.9e2	5.3e-2	1.5e7	2.4e5	1.9e1
3	8.4e4	3.2e2	5.1e-2	6.9e6	1.1e5	8.9e0
6	8.4e4	4.7e2	5.4e-2	5.9e6	1.2e5	6.8e0
10	7.0e4	5.8e2	4.6e-2	5.0e5	9.3e3	6.5e-1
14	6.7e4	7.0e2	4.6e-2	3.9e5	8.7e3	4.9e-1
18	7.0e4	7.8e2	4.6e-2	9.3e4	2.0e3	1.2e-1
22	4.5e4	5.7e2	3.1e-2	3.2e4	7.1e2	3.9e-2
26	3.0e4	3.5e2	2.0e-2	3.2e4	6.6e2	4.2e-2
30	1.7e4	1.9e2	1.1e-2	1.3e4	2.3e2	1.4e-2