Complexity Assessment of Sphere Decoding Methods for MIMO Detection

Johannes Fink*, Sandra Roger*, Alberto Gonzalez*, Vicenc Almenar*, Victor M. Garcia†
finjo@teleco.upv.es, sanrova@iteam.upv.es, agonzal@dcom.upv.es, valmenar@dcom.upv.es, vmgarcia@dsic.upv.es
*Instituto de Telecomunicaciones y Aplicaciones Multimedia,
Universidad Politécnica de Valencia, Spain
†Departamento de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia, Spain

Abstract—*Sphere Decoding (SD) algorithms have been shown to provide maximum likelihood (ML) detection over Gaussian multiple input-multiple output (MIMO) channels with lower complexity than the exhaustive search. These methods are based on a closest lattice point search over a limited search space (hypersphere). There exist several implementations of these algorithms pursuiting different search strategies and working either within a set of real numbers, thus called Real Sphere Decoders (RSD), or performing the search directly within a set of complex numbers, commonly known as Complex Sphere Decoders (CSD). In this paper, a performance comparison between the real and the complex version of the Schnorr-Euchner (SE) sphere decoder has been carried out in order to find out which algorithm is the most suitable depending on the application. Furthermore a recently appeared fixed-complexity version of the SE decoder (FSD) has been evaluated both in terms of complexity and performance and the results have been compared with the original version. In contrast to yet existing complexity analyses, not only the number of visited nodes has been investigated but also the total number of operations.*

I. Introduction

Maximum likelihood (ML) detection over Gaussian multiple input-multiple output (MIMO) channels can achieve the lowest Bit Error Rate (BER) for a given scenario, but at the expense of prohibitive complexity [1]. Thus, there is a continuous search for computationally efficient detectors, such as the Sphere Decoding (SD) algorithms, which are a set of tree search detectors with reduced complexity compared to the ML exhaustive search detector due to setting a radius constraint [2]. These algorithms perform a closest lattice point search for each component of the received vector, which is feasible due to the fact that the constellation set to which the transmitted symbols belong is known in advance. The existing SD algorithms can be implemented to operate within a finite set of real numbers, thus called Real Sphere Decoders (RSD) [3], or to perform the search directly within a finite set of complex numbers, commonly known as Complex Sphere Decoders (CSD) [4]. Since these detectors provide the ML solution to the detection problem, their evaluation focuses only on their complexity.

Recently, a fixed complexity sphere decoder (FSD) derived from the Schnorr-Euchner (SE) sphere decoder has been presented in [5]. This detector achieves quasi-ML performance with the advantage of predetermined runtime, thus it is applicable in time-constrained scenarios.

Various complexity investigations have been done on SD algorithms, some of them based on analytical evaluations [2], [6] and others obtained by means of simulations. In this paper, to evaluate the complexity of the SE sphere decoders, both methods have been combined; i.e. Monte Carlo simulations, which are used in order to obtain the number of visited and explored nodes in the tree, are combined with analytical examination of the operations carried out per node. Furthermore, the nodes are counted separately in each tree level, as the operations needed per node vary according to the level. The complexity analysis for the FSD turns out to be completely analytical, since the number of visited nodes is predetermined. The main focus in this paper lies in the above mentioned comparison between the CSD and the RSD using different modulation schemes and in the comparison between the CSD and the FSD.

II. Data detection in MIMO systems

Let us consider a block fading MIMO system with n_T transmit antennas, n_R receive antennas ($n_R \geq n_T$) and a signal to noise ratio denoted by ρ. The baseband equivalent model for this MIMO system is given by

$$\mathbf{x} = \mathbf{H}\mathbf{s} + \mathbf{v}, \tag{1}$$

where $\mathbf{s}$ represents the baseband signal vector that is transmitted during each symbol period, which is composed of elements chosen from the constellation $\mathbf{\Omega}$ consisting of $|\mathbf{\Omega}|$ points. Vector $\mathbf{x}$ in Eq. (1) denotes the received symbol vector, and $\mathbf{v}$ is a complex white Gaussian noise vector. The Rayleigh fading channel matrix $\mathbf{H}$ is assumed to be known at the receiver. This matrix is formed by $n_R \times n_T$ complex-valued elements, H_{ij}, which represent the complex fading gain from the j-th transmit antenna to the i-th receive antenna. Given the received signal $\mathbf{x}$, the ML detection problem consists in determining the transmitted vector $\mathbf{s}$ with the highest a posteriori probability, i.e. solving the following least squares problem:

$$\mathbf{s}_{\text{ML}} = \arg \min_{\mathbf{s} \in \mathbf{\Omega}^{n_T}} \|\mathbf{x} - \mathbf{H}\mathbf{s}\|^2. \tag{2}$$

A straightforward method for solving Eq. (2) would be an exhaustive search over the total $|\mathbf{\Omega}|^{n_T}$ lattice points $\mathbf{s}$, which is commonly known as ML method. However, such an implementation is cumbersome in practice. For that reason, among others, the SD techniques appeared.

III. Sphere Decoding algorithms

Sphere decoding algorithms are a subset of decision feedback tree-search-decoders. They perform the detection of MIMO data symbols by iterating through a detection tree, in which the tree levels, also referred to as dimensions, correspond to the elements of the received symbol. Those detectors differ basically in the way how they search along the tree. At this point, the difference between visited nodes and explored nodes should be pointed out. Any node that is not discarded is considered a visited node (VN). A subset of the VN are the explored nodes (EN). These are all VN with branching child nodes. The goal always is to visit and explore as little nodes as possible to keep the computational cost low. Various strategies exist to achieve this goal, all of which can or even have to be combined in order to work properly:

- Setting a initial radius[1] and only searching for solutions inside the so determined hypersphere.
- Updating the radius constraint whenever a possible solution is found.
- Ordering the symbols to detect prior to decoding according to their post-detection noise amplification.
- Ordering the nodes branching from a parent node incremental with respect to their metric increment as done in the SE sphere decoder.

The following will be a brief summary of the real and complex SD algorithms using SE node enumeration as described in [4]. Afterwards the key ideas behind the FSD as proposed in [5] will be explained. Without loss of generality, it is assumed that $n_\mathrm{R} = n_\mathrm{T} = m$.

The key idea in SD is doing an exhaustive search over only those points that lie inside a m-dimensional hypersphere of radius r, instead of searching over all possible elements $\mathrm{s} \in \mathbf{\Omega}^m$, as it could be naively done to solve Eq. (2). This constraint can be expressed by

$$\|\mathbf{x} - \mathbf{Hs}\|^2 \leq r^2. \tag{3}$$

Therefore, besides choosing an appropriate initial sphere radius, the core problem that any SD algorithm has to solve is finding all points $\mathrm{s} \in \mathbf{\Omega}^m$ that satisfy (3) without having to examine the distance to *all* elements in $\mathbf{\Omega}^m$, as this would mean no difference to an exhaustive search, speaking in terms of complexity. This core problem in SD is solved by dividing the problem of determining which points lie within a m-dimensional hypersphere into m problems of determining which symbols lie in a 1-dimensional (real case) or in a 2-dimensional (complex case) hypersphere. This means that in the real case it has to be evaluated which numbers of the set $\mathbf{\Omega}$ lie in a given interval, whereas in the complex case all symbols that lie inside a given circle have to be found.

Introducing $\hat{\mathrm{s}} = \mathbf{H}^\dagger \mathbf{x} = \left(\mathbf{H}^*\mathbf{H}\right)^{-1}\mathbf{H}^*\mathbf{x}$ and canceling out constant terms, Eq. (3) may be written as $(\mathrm{s} - \hat{\mathrm{s}})^*\mathbf{H}^*\mathbf{H}(\mathrm{s} - \hat{\mathrm{s}}) \leq r^2$. To break the problem down into subproblems, as described above, it is necessary to factorize the channel matrix $\mathbf{H}$ using for example the Cholesky factorization, such that $\mathbf{U}^*\mathbf{U} = \mathbf{H}^*\mathbf{H}$. Exploiting the upper triangular property of the $m \times m$ matrix $\mathbf{U}$ leads to

$$\begin{aligned}&(\mathrm{s} - \hat{\mathrm{s}})^*\mathbf{U}^*\mathbf{U}(\mathrm{s} - \hat{\mathrm{s}}) \\ &\quad = \sum_{i=1}^{m} \mathrm{U}_{ii}{}^2 \left| s_i - \hat{s}_i + \sum_{j=i+1}^{m} \frac{\mathrm{U}_{ij}}{\mathrm{U}_{ii}}(s_j - \hat{s}_j) \right|^2 \leq r^2, \end{aligned} \tag{4}$$

where U_{ij} denotes the entry (i,j) of $\mathbf{U}$. The term $i = m$ in the sum of Eq. (4) only depends on s_m, the term $i = m-1$ on $\{s_m, s_{m-1}\}$ and so on. Therefore, we can now establish bounds to find candidate values for s_m. All the elements that lie in between the bounds are called *candidates*. Their metric increments μ_m are calculated and, as proposed in [7], they are sorted according to increasing metric (SE node enumeration). Then, depending on the chosen candidate, bounds for s_{m-1} will be computed. The algorithm continues in this fashion until one of two things happens: either there are no more candidates to examine in the current level, or level $k = 1$ is reached.

If there are no more candidates in the current level, the decoder backtracks, going up one level at a time until reaching a level that still contains unexamined nodes[2]. Then, a new candidate is chosen and new bounds for the level below are established and so on.

If the lowest level $k = 1$ is reached, each chosen candidate s_1 completes the vector s and thus determines a possible solution. The distance d' of this solution is calculated and if d' is smaller than the distance of any formerly calculated estimates, we have a new, better estimate for s_ML. When there are no more points left in level $k = m$, the search has finished and s represents the ML-estimate s_ML. This decoding process is illustrated in Fig. 1.

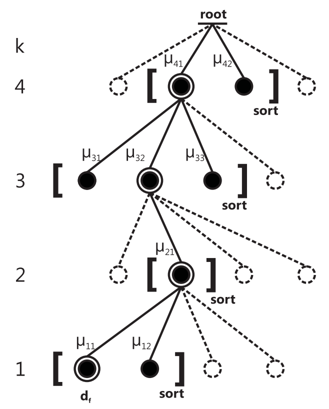

Fig. 1. Tree search in decoding process of a SE sphere decoder.

[1]The initial radius can be determined by means of statistics or by calculating the distance of the Babai estimate as described in [2].

[2]A *node* in level k can be interpreted as a tuple of $m - k + 1$ chosen candidates. All nodes in level $k = 1$ are called *leaves* and are elements of $\mathbf{\Omega}^m$.

Looking at Eq. (4) and defining ${r'_m}^2 = r^2$ and $\hat{s}_{m|m+1} = \hat{s}_m$, we find the expressions (5) to (7) that are used by the detector in order to calculate bounds for each s_k, with $k = (m-1), \dots 1$:

$$\begin{aligned} \hat{s}_{k|k+1} &= \hat{s}_k - \sum_{j=k+1}^{m} \frac{\mathrm{U}_{k,j}}{\mathrm{U}_{k,k}} (s_j - \hat{s}_j), & (5) \\ \mu_k &= {\mathrm{U}_{k,k}}^2 \left| s_k - \hat{s}_{k|k+1} \right|^2, & (6) \\ {r'_k}^2 &= {r'_{k+1}}^2 - \mu_{k+1}. & (7) \end{aligned}$$

denoting $\hat{s}_{k|k+1}$ as the element k of the skewed received vector $\mathbf{H}^\dagger \mathbf{x}$ altered by the effect of the so far chosen candidates for $s_l, \quad l > k$ (decision feedback). The μ_k variable represents the metric increment caused by the chosen candidate and the r'_k variable can be considered as the the new radius constraint due to the so far chosen candidates. Once level $k = 1$ is reached, the quadratic distance measure d'^2 of the estimate $\mathbf{Hs} = \mathbf{H} (s_1, s_2, \dots, s_m)^\mathrm{T}$ to the received vector $\mathbf{x}$ is calculated using Eqs. (6) and (7) as

$$d'^2 = {r'_m}^2 - {r'_1}^2 + \mu_1 = \sum_{k=1}^{m} \mu_k. \tag{8}$$

To decrease the size of the decoding tree and thus improve the speed of the detectors, whenever a path is completed, its final distance measure d'^2 is used as new radius constraint for level $k = m$ and the constraints of lower levels are adjusted accordingly. This process is referred to as *radius update*.

Since the manner of how the bounds are established differ in the two implementations of the SE algorithm considered, the explanations hereafter will be distinct.

A. Real SE Sphere Decoder (RSD)

Eqs. (5) and (7) provide the bounds in level k for s_k:

$$\left\lceil \frac{-r'_k}{\mathrm{U}_{k,k}} + \hat{s}_{k|k+1} \right\rceil \leq s_k \leq \left\lfloor \frac{r'_k}{\mathrm{U}_{k,k}} + \hat{s}_{k|k+1} \right\rfloor, \tag{9}$$

denoting $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ as functions that give the next higher or lower number in the real symbol-space $\mathbf{\Omega}_\mathbb{R}$, respectively.

The RSD is also capable to process complex constellations if the complex system is converted into its real representation of double dimensions $2n_R \times 2n_T$, as described in [8]. For that reason, $m = 2n_\mathrm{T}$ in the real case.

B. Complex SE Sphere Decoder (CSD)

In the complex case, the term to be evaluated, derived from Eq. (4) and using the definitions (5) and (7) in any level k is

$$\left| s_k - \hat{s}_{k|k+1} \right|^2 \leq \frac{{r'_k}^2}{{\mathrm{U}_{k,k}}^2}, \tag{10}$$

which is a circle in the complex plane, centered at $\hat{s}_{k|k+1}$ with a radius of $\frac{r'_k}{\mathrm{U}_{k,k}}$. The complex candidates are easily found when the symbols lay on a circle, as in PSK-modulations. Then, the angular bounds of the arc defined by the intersection of Eq. (10) and the constellation circle have to be found. This is illustrated for both the QPSK and the 16-QAM case in Fig. 2, which also shows the division of a QAM constellation into symbols on n_c concentric circles.

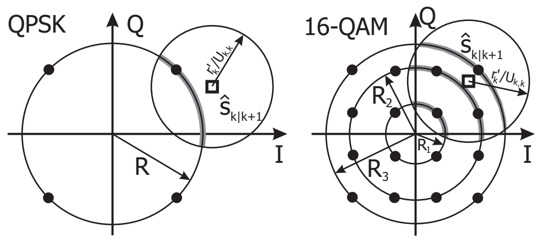

Fig. 2. Search disc for a QPSK and a 16-QAM constellation and concentric circle division of 16-QAM.

Let $s_k = a_k e^{i\Theta_k}$ and $\hat{s}_{k|k+1} = \hat{a}_{k|k+1} e^{i\hat{\Theta}_{k|k+1}}$, with Θ_k being taken from the set of angles of the constellation symbols and a_k being the according radius. Then Eq. (10) becomes for a given k:

$${a_k}^2 + \hat{a}_{k|k+1}^2 - 2 a_k \hat{a}_{k|k+1} \cos(\Theta_k - \hat{\Theta}_{k|k+1}) \leq \frac{{r'_k}^2}{{\mathrm{U}_{k,k}}^2}, \tag{11}$$

which yields

$$\begin{aligned} &\cos(\Theta_k - \hat{\Theta}_{k|k+1}) \\ &\qquad \geq \frac{1}{2 a_k \hat{a}_{k|k+1}} \left[{a_k}^2 + \hat{a}_{k|k+1}^2 - \frac{r'^2_k}{{\mathrm{U}_{k,k}}^2} \right] =: \eta. \quad (12) \end{aligned}$$

Three different cases for η have to be distinguished. If $\eta > 1$ the search disc is empty. If $\eta < -1$ the disc includes the entire constellation and if $-1 \leq \eta \leq 1$, the arc is defined by $|\Theta_k - \hat{\Theta}_{k|k+1}| \leq \cos^{-1} \eta$. Thus the range of allowable points is delimited by the following bounds[3]:

$$\left\lceil (\hat{\Theta}_{k|k+1} - \cos^{-1} \eta) \right\rceil \leq \Theta_k \leq \left\lfloor (\hat{\Theta}_{k|k+1} + \cos^{-1} \eta) \right\rfloor. \tag{13}$$

C. Fixed Complexity Sphere Decoder (FSD)

The Fixed Complexity Sphere Decoder (FSD) was proposed in [5] to overcome two main problems in SD:

1) The sequential nature of the SE process.
2) The variable detection complexity.

A novel channel matrix ordering is combined with restricting the search space to a very small subset. A search space restriction denoted by $\mathbf{n}_s = (1, 1, 1, |\mathbf{\Omega}|)$ means that all nodes in level $m = 4$ are explored, but only the best one (with the smallest metric) in all lower levels. Thus, a total of $\mathcal{P} = |\mathbf{\Omega}|$ paths are visited. Furthermore, the candidates are chosen without calculating bounds.

[3]Here, the functions $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ should not be interpreted in the integer sense but in the sense that they are referring to the next greater / smaller symbol-angle Θ in the constellation.

In contrast to the SE sphere decoder, the FSD does not guarantee to find the ML estimate, thus it is a suboptimal detector. But, as seen in Fig. 3, which shows a performance comparison between the bit-error-rates (BER) of optimal ML decoders and those achieved by the FSD for different modulations, its performance comes stunningly close to that of an optimal ML decoder.

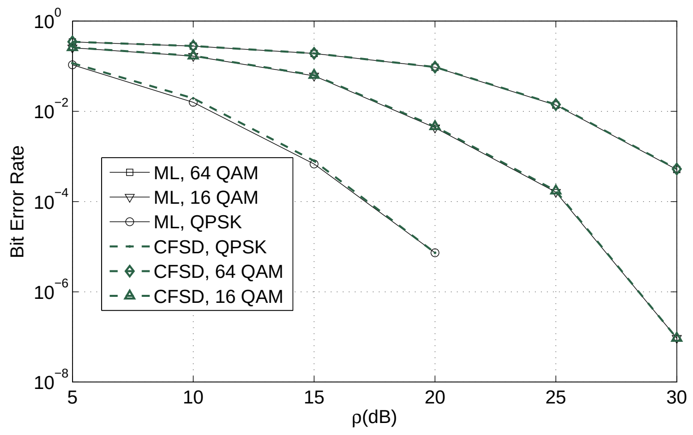

Fig. 3. BER achieved by the CSD, the RSD and the FSD in 4×4 MIMO system using QPSK, 16-QAM and 64-QAM.

IV. Complexity analysis

The complexity of tree search algorithms is determined by two criteria: the number of nodes that have to be examined and the operational cost per node. In SD, the number of visited nodes depends on the initial sphere radius and on the reduction of the the radius constraints due to a radius update. The operational costs per node depends on the algorithm used.

In the following, we will determine whether working directly in complex numbers (as the CSD) can effectively reduce the complexity of SD, or if actually working in the real case (as the originally proposed RSD [3]) is less expensive. Furthermore we will consider the FSD as a practically implementable decoder in our comparison to get a feeling for its behavior. Therefore, we will first analyze the decoding algorithms in order to get the number of operations executed per visited node. Then, the mean number of visited and explored nodes by both SE sphere detectors will be obtained using simulations, thus the yielding results represent an expected average complexity. In the FSD these node counts are fixed. By adding up all corresponding products of the node counts and the cost per node for each level k, the overall complexity values will be achieved. These results can be useful in deciding on an appropriate decoder under given circumstances.

A. Counting nodes

Looking at Eq. (5), it becomes obvious that it is not sufficient to just count the total number of visited and explored nodes by the detectors, but that it is rather necessary to count them separately for each level k since the number of operations that have to be calculated per node increases as k decreases, which is due to the triangular shape of $\mathbf{U}$. Furthermore, in order to evaluate the complexity correctly, in the case of the CSD, we have to count how often the condition $-1 < \eta < 1$ holds, since only then bounds have to be calculated as in (13). The node counts used in the complexity estimations consist of mean values over a representative number of detection runs.

B. Cost of nodes RSD, CSD

The time-complexity of an algorithm can be counted in terms of number of clock cycles that are needed to process the algorithm. Since the number of clock cycles that are needed to complete a certain operation differs on different hardware solutions (Digital Signal Processors, Microprocessors, Multi-Purpose-Processors, FPGAs, ...), we have to set rules how to calculate the cost, in order to fairly compare the different algorithms. Complex calculations are broken down into number of real sums and real products. So, one addition of two complex numbers can be seen as two real additions, and one product of two complex numbers can be regarded as the equivalent cost of 4 real products and 2 real sums. All other operations, both logical ones and arithmetic ones are assumed to be processed within one clock cycle, as this would be the goal of a fast specialized hardware implementation. Considering this, the cost per node for all three algorithms can now be determined.

In Table 1 the Eqs. (5) to (8) have been tagged with a cost for both the real and the complex case. In addition, for the real case, Eq. (9) has been assigned cost values and for the complex case, this has been done for Eq. (12), which is necessary to get the η variable. For Eq. (13), which is the equation defining the bounds in level k, it is more difficult to assign a cost value because of the implementation of the floor and ceiling functions that work within a set of angles. These are subroutines with a determined cost, but they are only evaluated if $-1 < \eta < 1$ and thus we also measured how often this is the case to get a correct mean value of cost. As this percentage depends on the noise level and the level k of the decoder, we generated another vector denoted $\mathbf{n_b}$ additional to the visited and explored node counts, that represents the times that the subroutine calculating the bounds is actually evaluated. The total cost for the functional blocks can be seen in Table 2.

Additionally to those equations, the algorithms have some additional cost, for example comparing candidate values with the bounds and sorting the found candidates according to their metrics. A summary of all necessary steps and how often they are carried out in terms of number of visited and explored nodes can be seen in Table 3.

C. Complexity FSD

The fixed complexity of the FSD can be calculated entirely analytically: It depends basically on the node distribution $\mathbf{n}_s$ and on the constellation size $\mathcal{P}$. For $\mathbf{n}_s = (1, 1, 1, \mathcal{P})$ we can calculate the complexity as follows.

Table 1. Cost of used equations / functions expressed in real operations.

Equation / function	Operational Cost.			
	RSD		CSD	
	Sums	Products	Sums	Products
(5)	$2(m-k)$	$2(m-k)$	$6(m-k)$	$6(m-k)$
(6)	1	2	3	3
(7)	1	-	1	-
(8)	2	-	2	-
(9)	2	3	-	-
(12)	-	-	2	5

Table 2. Cost functional blocks of the algorithm.

functional block	Operational Cost.	
	RSD	CSD
c_{rounding}	1	$3+\frac{\mathcal{P}}{n_c}$;
c_{bounds}	$2(\log_2(\mathcal{P})-1)+1+3+2$	$22+2c_{\text{rounding}}$

Table 3. Subroutines considered for complexity estimation and according complexity in terms of evaluated equations plus additional cost.

Operation	Evaluated Eqs. / additional cost		How often
	RSD	CSD	
Calculating initial bounds for level m.	(9) $+c_{\text{bounds}}$	n_c times (12) $+5+n_b c_{\text{bounds}}$	Once per detection run.
Calculating metric increment.	(6)		Once per visited node (VN), independent of level k.
Sorting candidates according to their metric.	sort$(\cdot)$		Once per explored node (EN), all its child nodes have to be sorted[a].
Calculating $\hat{s}_{k\|k+1}$.	(5)		Once per EN.
Retrieving modulo and angle of $\hat{s}_{k\|k+1}$.	-	2	Once per EN.
Calculating new bounds.	(7), (9) $+c_{\text{bounds}}$	(7), n_c times (12) $+5+n_b c_{\text{bounds}}$	Once per EN.
Calculating final distance measure.	(8)		Once for every completed path.
Comparing selected node with bounds.	one comparison $\geq$		Once per VN.
Incrementing state	1 comparison + 1 sum = 2 op.		Once per VN.

[a]As a reference, we use the complexity of the well-known quicksort algorithm [9], which performs on average $2\ln(2)n\ln(n) \approx 1.39n\ln(n)$ comparisons. Since we only count the nodes visited per level k, we calculate the sorting complexity as if they were all in one list and get thus an upper limit for the average complexity.

Table 4. Number of carried out operations of the CFSD for different modulation schemes using a sequential implementation and a fully parallelized implementation.

Modulation	Number of carried out operations.	
	sequential V.	fully parallelized V.
QPSK	560	140
16-QAM	6272	392
64-QAM	89664	1401

In each level except in $k=1$, the detector explores $\mathcal{P}$ nodes. The number of visited nodes is $\mathcal{P}^2$ for every level except for $k=m$, where it is $\mathcal{P}$. This accumulates to a total of $(m-1)\mathcal{P}$ explored nodes and $(m-1)\mathcal{P}^2+\mathcal{P}$ visited nodes.

For every explored node, Eq. (5) has to be evaluated, and for every visited node, the metric has to be calculated as in Eq. (6). Furthermore, in all levels except $k=m$, the minimum among a number of $\mathcal{P}$ metrics has to be found, which means $\mathcal{P}-1$ comparisons. Moreover, since a total of $\mathcal{P}$ paths are completed, this is also the number of times we have to calculate a final distance as in Eq. (8). These explanations lead to the following cost calculations with $n_{\text{EN},k}$ being the number of explored nodes per level and n_{EN} the total number of explored nodes. n_{VN} denotes the total number of visited nodes respectively. It should be noted that the FSD can be implemented based on the real version of the SE SD or on the complex version, but we only consider the complex version for our study.

$$C_{\text{total}} = \sum_{k=2}^{m} \left(12(m-k)n_{\text{EN},k}\right) + (m+2)\mathcal{P} - 1 + 3n_{\text{EN}} + 6n_{\text{VN}} + (\mathcal{P}-1)\mathcal{P}(m-1) \quad (14)$$

V. Results

The results for the complexity of the CFSD, considering a 4×4 MIMO system and different modulation schemes can be seen in Table 4. Since one of the main advantages of the FSD is its capability to be parallelized using FPGASs, it has also been calculated the number of clock cycles necessary to complete the algorithm if it is fully parallelized.

The results for the analysis of the complexity of the SE sphere decoders in a 4×4 MIMO system for different modulation schemes can be seen in Fig. 4 to 6. The start radius was set to the end of the scale and all simulations have been done using 1000 channel realizations and 32000 transmitted symbols. Fig. 4 shows the total number of operations needed in average by the CSD and the RSD using a QPSK constellation. It can be seen that the RSD works slightly more efficiently for this modulation scheme. In high SNRs the CFSD can only compete if its parallelized version is used, which requires more hardware resources. However, we should note, that the CSFD only needs basic operations (multiplication, addition and comparison), whereas the CSD needs to calculate the angle and the absolute value of a complex number and to evaluate the square-root of a real number.

In Fig. 5, we can see the same comparison for a higher order orthogonal constellation, in this case 16-QAM. Again the CFSD outperforms the other detectors in its parallel version for the obvious reasons. For the CSD and RSD we can see that their plots cross at around 15 dB. This is due to setting the initial radius to infinity, since in this case at least one complete path has to be visited. Note that visiting one full path is less costly in the RSD than in the CSD, due to the number of symbols to be searched is $\log 2|\Omega_{\mathbb{C}}|$, which overcompensates the double dimensions of the RSD. This means, searching one full path is less costly in the RSD.

In Fig. 6, the results are displayed for a 64-QAM modulation. The behavior is the same as in Fig. 4 and 5, but now the drop in complexity when going from 5 dB to 20 dB is almost 90 %, and for high SNRs, the SE sphere decoders come in average quite close to the fully parallelized version of the CFSD, which needs a lot more resources.

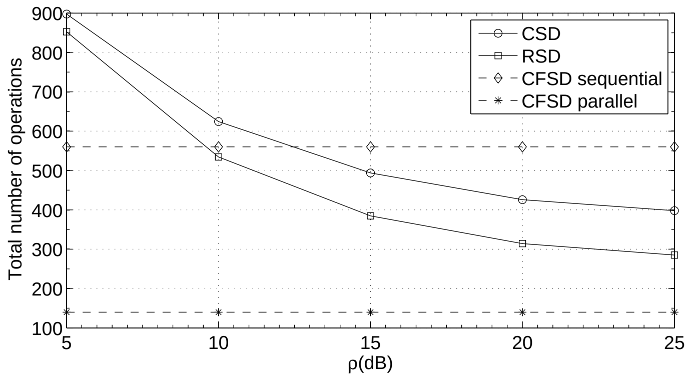

Fig. 4. Total number of operations for the CSD, RSD and the CFSD implemented sequentially and parallely detecting in a 4×4 MIMO environment using QPSK.

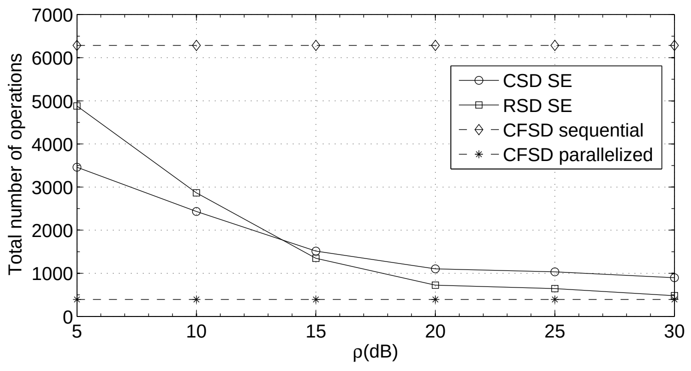

Fig. 5. Total number of operations for the CSD, RSD and the CFSD implemented sequentially and parallely detecting in a 4×4 MIMO environment using 16-QAM.

VI. Conclusions

A method for evaluating the complexity of SD algorithms not only in terms of number of visited nodes, but also in

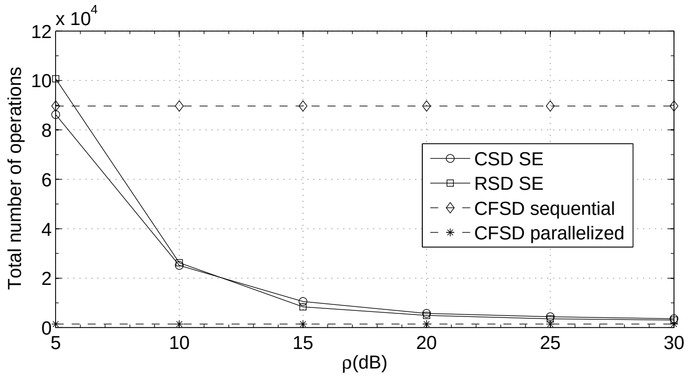

Fig. 6. Total number of operations for the CSD, RSD and the CFSD implemented sequentially and parallely detecting in a 4×4 MIMO environment using 64-QAM.

terms of total number of operations has been presented in this paper. A SD that works directly with complex constellation symbols (CSD) has been compared with a SD that works either with real constellations or with the real representation of complex constellations (RSD), and both of them have been compared with a fixed complexity tree-search-decoder (CFSD). The proposed method is applicable to any given constellation and channel. If one of them changes, the method only requires a new count of visited and explored nodes by means of simulations in order to get an adequate complexity estimation. The results show, that the CSFD can only compete with the SE sphere decoders if it is implemented in a parallel manner. The CSD and RSD behave quite similarly, however, the RSD performs better at high SNR levels and the CSD outperforms the RSD at low SNR regimes.

References

[1] A. J. Paulraj, D. A. Gore, R. U. Nabar, and H. Bölcskei, "An overview of MIMO communications - a key to gigabit wireless," *Proceedings of the IEEE*, vol. 92, no. 2, pp. 198–218, Feb. 2004.

[2] B. Hassibi and H. Vikalo, "On Sphere Decoding algorithm. Part I, the expected complexity," *IEEE Transactions on Signal Processing*, vol. 54, no. 5, pp. 2806–2818, August 2005.

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

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

[5] L. G. Barbero and J. S. Thompson, "Fixing the Complexity of the Sphere Decoder for MIMO Detection," *IEEE Transactions on Wireless Communications*, vol. 7, no. 6, pp. 2131–2142, 2008.

[6] J. Jalden and B. E. Ottersten, "On the complexity of sphere decoding in digital communications," *IEEE Transactions on Signal Processing*, vol. 53, no. 4, pp. 1474–1484, 2005.

[7] C. P. Schnorr and M. Euchner, "Lattice basis reduction: improved practical algorithms and solving subset sum problems," *Mathematical Programming*, vol. 66, pp. 181–199, 1994.

[8] T. Kailath, H. Vikalo, and B. Hassibi, *MIMO receive algorithms in Space-Time Wireless Systems: From Array Processing to MIMO Communications*. Philadelphia: Cambridge University Press, 2006.

[9] C. A. R. Hoare, "Quicksort." *Computer Journal*, vol. 5, no. 1, pp. 10–15, 1962.