

COMPUTATIONAL COMPLEXITY BOUNDS FOR LIST MIMO DETECTION

David L. Milliner*, Ernesto Zimmermann†, John R. Barry* and Gerhard Fettweis†

*School of Electrical and Computer Engineering
 Georgia Institute of Technology, Atlanta, Georgia 30332-0250
 Email: {dlm,barry}@ece.gatech.edu

†Vodafone Chair Mobile Communications Systems
 Technische Universität Dresden, D-01062 Dresden, Germany
 Email: {zimmere,fettweis}@ifn.et.tu-dresden.de

ABSTRACT

This work establishes bounds on the computational complexity of list MIMO detectors. Specifically, upper and lower bounds are introduced, where computational complexity is measured in terms of the number of branch metric computations performed during detection. The lower complexity bound corresponds to a “genie tree search detector” while the upper bound is determined using the list sequential detector. For illustration, results are provided for several MIMO setups.

I INTRODUCTION

An ideal communications receiver would enable capacity achieving performance whilst requiring a negligible amount of computational complexity. Such a receiver is, of course, impossible to realize. Yet, there has been steady improvement in the performance achieved by multiple-input multiple-output (MIMO) detectors for a given computational complexity. Numerous approaches now exist for finding close-to-optimal solutions to the MIMO detection problem at acceptable complexity. Many of these schemes reformulate the detection task into a tree search problem, in which leaf nodes maximizing a certain metric have to be found. When more than one leaf node is maintained at the output of the tree search stage, the detector is called a list detector. One main attraction of such list MIMO detection schemes (e.g. [1–3]) is that, by choosing the list size appropriately, they allow a flexible adjustment of detection complexity subject to given performance requirements.

The computational complexity of list MIMO detection can be measured in many ways, e.g. using the number of floating point operations, silicon area, or amenability to parallelization. Yet, most of these figures of merit depend on the specific target architecture – e.g. fixed vs. floating point operation, ASIC vs. FPGA, etc. As a well accepted, relatively architecture-agnostic metric [4–6], this work uses the number of branch metric computations as the measure of computational complexity. This figure corresponds to an upper bound on the number of visited nodes, since visiting a node in the tree requires calculating the corresponding branch metric. We deem the number of branch metric computations the more useful metric as it takes the overhead due to discarded nodes into account (e.g. the ones dropped after sorting in M-Algorithm based tree search).

In recent years, it has become steadily more difficult to achieve further reductions in the complexity of list MIMO detectors. This evidently raises the question, whether some fundamental limits exist, under which the detection complex-

ity must not fall if a certain system performance has to be achieved. The aim of this work is to determine upper and lower bounds on the computational complexity of list MIMO detectors. These results are useful as criteria for system designers seeking to assess whether there is room for a further reduction in the complexity of a list MIMO detection algorithm, if a certain performance target has to be met.

The remainder of this paper is organized as follows: after discussing the employed system model in Section II, Section III details relevant fundamentals of MIMO detection. Section IV provides boundary conditions on the computational complexity of list MIMO detection. In Section V we provide numerical results for these bounds for various MIMO setups. Finally, conclusions are drawn in Section VI.

II SYSTEM MODEL

Consider an $N_T \times N_R$ MIMO system based on a BICM transmit strategy as depicted in Fig. 1: the vector \mathbf{u} of i.i.d. information bits is encoded and interleaved. The resulting coded bit stream is partitioned into blocks \mathbf{c} of $N_T \cdot L$ bits and mapped onto a vector symbol $\mathbf{x} \in \mathcal{X}$ whose components are taken from some complex constellation \mathcal{C} (e.g. Gray mapped 64-QAM). Here, L denotes the number of bits per complex symbol, resulting in $q = |\mathcal{C}| = 2^L$ different constellation points. We consider transmission over a flat fading channel.

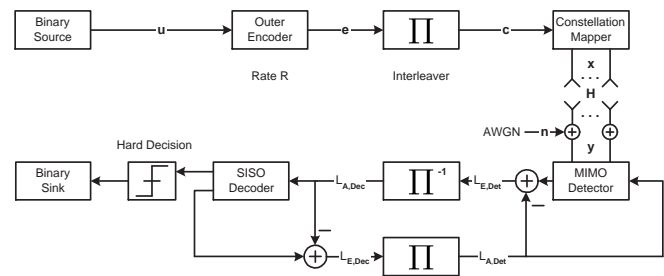


Figure 1: System model using a BICM transmit strategy.

In the equivalent discrete-time base-band model, the received signal \mathbf{y} is given by:

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \quad (1)$$

where $\mathbf{H} \in \mathbb{C}^{N_R \times N_T}$ is the channel transfer matrix which is assumed to be perfectly known at the receiver. The entries of \mathbf{H} are realizations of zero mean i.i.d. complex

Gaussian random processes of variance 1 (passive subchannels). The average transmit energy is normalized such that $\mathcal{E}\{\mathbf{x}\mathbf{x}^H\} = E_s/N_T \mathbf{I}$. The vector $\mathbf{n} \in \mathbb{C}^{N_R \times 1}$ represents the receiver noise whose components are zero mean i.i.d. complex Gaussian random variables with variance $N_0/2$ per real dimension: $\mathcal{E}\{\mathbf{nn}^H\} = N_0 \mathbf{I}$. The signal-to-noise ratio (SNR) at each receive antenna is hence given by $\text{SNR} = E_s/N_0$.

III FUNDAMENTALS

III.A Soft-Output Detection

The task of the detector in Fig. 1 is to determine the a posteriori probability for each of the code bits $c_{m,l}$ in \mathbf{x} , where $m \in \{1, \dots, N_T\}$ is the symbol index, and $l \in \{1, \dots, L\}$ the bit index in the m -th symbol. Since we are dealing with binary numbers, this information is conveniently expressed in the form of log-likelihood ratios (LLRs):

$$\begin{aligned} L(c_{m,l}|\mathbf{y}) &:= \ln \frac{P[c_{m,l} = +1|\mathbf{y}]}{P[c_{m,l} = -1|\mathbf{y}]} \\ &\approx \max_{\hat{\mathbf{x}} \in \mathcal{X}_{m,l}^{+1}} \left\{ \frac{-\|\mathbf{y} - \mathbf{H}\hat{\mathbf{x}}\|^2}{N_0} + \sum_{j=1}^{N_T} \sum_{k=1}^L \ln P(c_{j,k} = \hat{c}_{j,k}) \right\} \\ &\quad - \max_{\hat{\mathbf{x}} \in \mathcal{X}_{m,l}^{-1}} \left\{ \frac{-\|\mathbf{y} - \mathbf{H}\hat{\mathbf{x}}\|^2}{N_0} + \sum_{j=1}^{N_T} \sum_{k=1}^L \ln P(c_{j,k} = \hat{c}_{j,k}) \right\}, \end{aligned} \quad (2)$$

where the second line follows from the application of the MaxLog-approximation. Here, $\mathcal{X}_{m,l}^{\pm 1}$ denotes the set of $2^{N_T \cdot L - 1}$ symbols $\hat{\mathbf{x}} \in \mathcal{X}$ for which $\hat{c}_{m,l} = \pm 1$, and $\hat{\mathbf{x}}$ denotes a certain hypothesis on the vector transmit sequence with $\hat{\mathbf{c}}$ as the corresponding vector of code bits. Using the MaxLog-approximation is widely accepted because of its relatively small performance loss and accompanying large complexity savings [1, 7]. Evaluating (3) by a brute-force approach (MaxLogAPP detection) is well known to require an effort growing exponentially in the number of transmitted bits per vector symbol. However, only a few hypotheses in $\mathcal{X}_{m,l}^{\pm 1}$ actually maximize each of the respective terms in (3). Several close-to-optimal detection strategies therefore construct a subset list $\mathcal{L} \subset \mathcal{X}$ of size $|\mathcal{L}|$ from which LLRs are determined.

III.B Tree-Search MIMO Detection

Tree search based MIMO detection techniques construct \mathcal{L} using a back-substitution approach. After an orthogonal-triangular decomposition of the channel matrix, e.g. $\mathbf{H} = \mathbf{Q}\mathbf{R}$, the LLRs can be determined using the per-antenna metric increments Λ_j :

$$L(c_{m,l}|\mathbf{y}) \approx \max_{\hat{\mathbf{x}} \in \mathcal{L} \cap \mathcal{X}_{m,l}^{+1}} \left\{ \sum_{j=1}^{N_T} \Lambda_j \right\} - \max_{\hat{\mathbf{x}} \in \mathcal{L} \cap \mathcal{X}_{m,l}^{-1}} \left\{ \dots \right\},$$

which are referred to as *branch metrics* and given by

$$\Lambda_j = -\frac{1}{N_0} \left| \tilde{y}_j - \sum_{i=j}^{N_T} r_{j,i} \hat{x}_i \right|^2 + \sum_{k=1}^L \ln P(c_{j,k} = \hat{c}_{j,k}), \quad (4)$$

with $\tilde{\mathbf{y}} = \mathbf{Q}^H \mathbf{y}$. The detector starts in layer $j = N_T$ and works its way up until layer $j = 1$ is reached. At each layer j , there are q possible choices for the signal component x_j . The detection process can hence be interpreted as a search for leaf nodes in a tree structure. Different types of tree search based detectors can be implemented by using the *path metrics* $\sum_{i=j}^{N_T} \Lambda_i$ (with j as the current layer index) to control which tree nodes are added to the working stack and in which order.

Since most list MIMO detection algorithms can be stated as a tree search problem, comparing the complexity of such schemes using a tree search based complexity metric is a logical undertaking. A common complexity metric for tree search schemes is the number of branch metric computations performed during detection [5, 6], here denoted as μ .

In [6] it was shown that for the specific – but from an implementation perspective attractive – case of fixed-complexity breadth-first MIMO detection algorithms, this figure, for the complex-valued channel model, can be expressed as:

$$\mu = \sum_{i=1}^{N_T} (\zeta_{i-1} b_i + S_i \kappa_i), \quad (5)$$

where b_i represents the number of children extended from each parent at the i th layer in the tree, and S_i determines whether or not to perform parallel smart candidate adding (PSCA) [8] at the i th layer in the detection tree. The total number of branch metric computations μ is a function of the number of nodes retained for a given layer in the detection tree ζ_i :

$$\zeta_i = \begin{cases} 1 & i = 0 \\ \min(\zeta_{i-1} b_i + S_i \kappa_i, M_i) & i > 0 \end{cases}, \quad (6)$$

where M_i represents the number of nodes retained at the i th layer in the detection tree, and κ_i denotes the number of branch metrics calculated as part of the smart candidate adding at layer i . κ_i is given by:

$$\kappa_i = \max\left(L - 2(\sqrt{b_i} - 1), 0\right), \quad (7)$$

when it is assumed that b_i is the square of an integer. The final list size is $|\mathcal{L}| = \zeta_{N_T}$.

III.C List Sequential Detection

The list-sequential detector (LISS) [3] is an extension of the stack algorithm to the soft output case, by considering list sizes larger than one, $|\mathcal{L}| > 1$. It keeps track of several paths simultaneously during the tree search, storing them in an ordered list, referred to as the *stack*. The paths in the stack are ordered according to the current path metrics, where paths with larger metrics are ordered to the top of the stack. The search tree is constructed by always extending the path which currently has the largest metric and has not yet reached full length. The

search is stopped once a predefined number of full length paths have reached the top of the stack.

It was shown in [4] that under certain conditions¹ the LISS using a largest branch metric first (“Schnorr-Euchner” [10]) enumeration strategy computes the fewest branch metrics required to ensure that the constructed list \mathcal{L} is optimal in the sense that it contains the $|\mathcal{L}|$ hypotheses for the transmit sequence whose associated metrics are larger than those of any other hypotheses not contained in \mathcal{L} . For the case where the LISS detector is without a priori knowledge, it thus finds the $|\mathcal{L}|$ lattice points closest to \mathbf{y} at a minimum detection effort amongst all tree search schemes.

IV COMPUTATIONAL COMPLEXITY

IV.A Computational Complexity Bounds

Lower Bound: A lower bound on the computational complexity of tree search detection schemes can be derived by considering a *genie tree search detector*: for a given list size $|\mathcal{L}|$, this detector generates exactly $|\mathcal{L}|$ leaf nodes which can then be used in the calculation of the LLRs based on the MaxLog-approximation. It furthermore prunes all nodes from the tree which are not needed to reach these leaf nodes, and thus constructs the *minimum spanning tree* for the considered list size $|\mathcal{L}|$. It thus provides a lower bound on computational complexity by performing the minimum number of branch metric computations required to generate a list of given length under the MaxLog-approximation. We therefore also refer to this lower bound as the minimum spanning tree bound.

As an example of the minimum spanning tree bound, consider a list of length 2, where the leaf nodes used for LLR calculation are assumed to be “siblings” of each other at the final layer in the detection tree. For this case, the minimum spanning tree corresponds to $\mu = (N_T - 1) + 2 = N_T + 1$ branch metric computations. Note, however, that the performance of such a scheme would evidently suffer relative to an algorithm generating the $|\mathcal{L}|$ nodes with largest metrics (as for example the LISS). The minimum spanning tree bound is easily extended to list lengths greater than two.

Upper Bound: To establish an upper bound on the computational complexity of list MIMO detection, we use the LISS as a *practical* algorithm for finding the M leaf nodes with largest metrics. The LISS computes the *minimum* number of branch metrics required to fulfil this task [4] provided that no genie knowledge is available – thus the contrast to the genie lower bound. If the number of branch metric computations is the sole measure of complexity (and storage and sorting overhead of the LISS are cast aside), it would evidently make no sense to employ any other tree search algorithm requiring more branch

metric computations to achieve the same performance as the LISS. It is thus reasonable to use the complexity of the LISS as an upper limit on what effort should be spent by a list detector in order to achieve a certain performance.

However, the complexity of the LISS is obviously a random variable. Evaluating the mean of this random variable is only a meaningful figure of merit if the detection delay is allowed to tend to infinity. This will evidently not be the case in practice, where hardware has to be designed to accommodate the worst case detection complexity and still meet delay requirements. Unfortunately, the worst case complexity of the LISS is equivalent to that of the brute force MaxLogAPP detector. However, this worst case occurs only in a marginal number of cases. A much more meaningful upper bound on complexity can be established by allowing the LISS, in a certain fraction of the cases, to stop the tree search before the optimum solution is found. Provided the number of errors additionally introduced by this limitation is small compared to the number of errors produced by the optimal detector, this can be expected to result in a negligible performance loss. This has been confirmed by the results from [4, 8]. With raw input error rates of 10% and above for state-of-the-art error correction codes (rate 1/2 and below), a fraction of 1% of prematurely stopped tree searches appears acceptable. We therefore elect the 99th percentile metric on the number of branch metric computations as the 1% *upper bound* on computational complexity.

IV.B Other Relevant Computational Complexities

We now present other computational complexities which do not serve as bounds, but are relevant in the context of list MIMO detection. These complexities are all based on the genie tree search detector.

Genie MaxLogAPP Complexity: The number of leaf nodes required to compute the soft-output for MaxLogAPP detection ranges from a minimum of 2, in the case where for all bit positions the same candidate vector is used to provide the counter-hypothesis to the JML decision, to a maximum of $N_T L + 1$, in the case where for each bit position a different leaf node provides the counter-hypothesis. In the former case, which is evidently rather unlikely, at least $\mu = 2N_T$ branch metrics must be computed. We will refer to this complexity as the “genie MaxLogAPP best case” computational complexity. The “genie MaxLogAPP worst case” computational complexity is found by considering the spanning tree for the case where each bit position requires a unique leaf node as the best counter-hypothesis to the JML estimate. The resulting computational complexity for the genie MaxLogAPP worst case is given by:

$$\mu = N_T + \frac{N_T(N_T + 1)L}{2}. \quad (8)$$

Note that the complexity figure given by (8) is exactly the same as that of the PSCA algorithm [8] with parameterization $\mathbf{M} = \infty$, $\mathbf{b} = [1..1]$ and $\mathbf{S} = [1..1]$ [6]. In fact, the PSCA generates a tree of the form just described to ensure that

¹Assuming infinite storage for the stack, and using no bias (“length term”) on the branch metrics. The proof from [4] thus generalizes a result from [9] which only treated the special case where $|\mathcal{L}| = 1$ and no a priori knowledge is available. Note that only tree search algorithms not making use of metric prediction techniques were considered in this analysis.

a counter-hypothesis is found for all detected bits. However, its performance is inferior to that of the MaxLogAPP detector. This is due to the fact the leaf nodes generated by the PSCA are not necessarily the ones with maximum metric under the given constraint on the value of a certain bit – like all other breadth-first tree search algorithms, it is prone to error propagation from early detected layers.

Genie LISS Complexity: The minimum spanning tree lower bound can be expected to be rather loose, since the generated leaf nodes are not necessarily optimal in the MaxLogAPP sense. In order to obtain a tighter bound for algorithms which generate the $|\mathcal{L}|$ leaf nodes with largest metrics – and thus achieve the same performance as the LISS – we apply the idea of a genie detector to the list generated by the LISS. This “genie LISS” detector calculates only those metrics which are necessary to generate the spanning tree to the subset of leaf nodes (in the list generated by the LISS) which are actually used in the LLR calculation. Again, the mean and the 99th percentile metric will be evaluated, although these figures of merit are now computed based on the spanning tree.

We conclude this section by observing that the lower bound and the Genie MaxLogAPP bounds can be found analytically, while the 1% upper bound and the genie LISS bounds must be found via simulation.

V RESULTS

For simulation we use a setup equivalent to the one in [1]: transmission occurs over a spatially and temporally i.i.d. fading 4×4 MIMO channel using 4-QAM and 64-QAM modulation alphabets. The information block size (including tail bits) is 9216 bits, using a rate 1/2 PCCC based on $(7_R, 5)$ convolutional codes for channel coding and 8 internal iterations of logMAP decoding. The LLR magnitudes for bits without counter-hypothesis were set to the values found in Table 1 (“LLR clipping” [2]) for the 4×4 MIMO setup with 4-QAM and 64-QAM transmission. Justification and the original derivation for these clipping levels can be found in [4].

As a reference on performance, so that the complexity bounds of the previous section may be provided in context, we use the error rate performance of a list MIMO detector which finds the $M = |\mathcal{L}|$ leaf nodes with largest metrics² among all hypotheses in \mathcal{X} , and employs (3) to calculate the LLRs using the MaxLog-approximation, i.e. a MaxLogAPP list detector with list size M . The resultant error rate performance is evidently specific to a given antenna setup, modulation alphabet, list size and employed coding scheme.

²An open problem concerns the optimal list to be found by a list detector. Specifically, the authors conjecture that for a given list length $|\mathcal{L}| \ll |\mathcal{X}|$, choosing the list which maximizes the mutual information between channel input and detector output will yield better performance than the one which is optimal in terms of largest metrics, due to the problem of missing counter-hypotheses for some bits. Unfortunately, so far no results exist on how this optimal list might be characterized and accordingly generated.

| M | 2 | 4 | 8 | 16 | 64 |
|--------|-----|-----|---|----|----|
| 4-QAM | 3 | 3.5 | 4 | 5 | – |
| 64-QAM | 2.5 | 3 | 4 | 5 | 6 |

Table 1: Fixed LLR clipping levels L_{clip} determined in [4] for a spatially and temporally i.i.d. fading 4×4 MIMO channel.

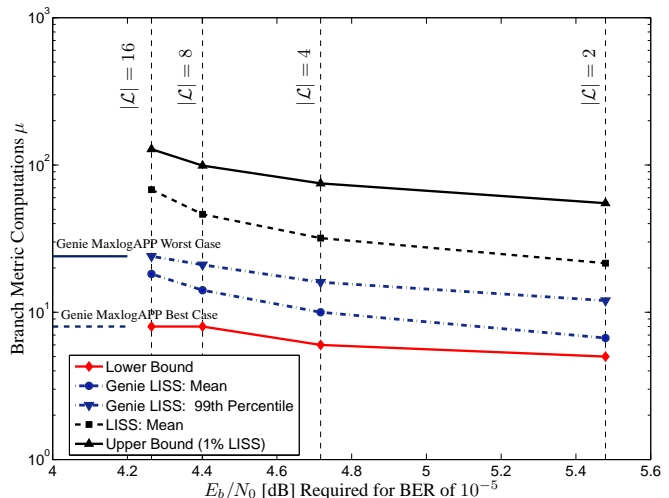


Figure 2: Complexity Bounds for 4×4 MIMO in Rayleigh fading using 4-QAM transmission and ZF-SQRD equalization.

The choice of the coding scheme is relevant to the overall system performance in MIMO detection. If near-capacity performance is desired then the channel code has to be designed to fit the EXIT characteristic of the detector [11], and multiple iterations between the detector and decoder are required. In this work we use the aforementioned system setup for ease of comparison with previous works, e.g. [1, 5, 6].

Fig. 2 depicts the complexity bounds outlined in the previous section for a 4×4 MIMO channel with 4-QAM transmission using the simulation setup just described, where equalization is performed using the ZF sorted QR decomposition (SQRD) [12] ordering and performance is measured as the SNR required to obtain a BER of 10^{-5} . The lowest complexity curve, denoted with diamond markers, is the minimum spanning tree bound. The highest complexity curve, denoted with a solid curve and upward facing triangular markers, is the 1% upper bound based on the LISS. The average LISS complexity is also shown, using a dark dashed curve with square markers. Other references for the computational complexity shown are the best and worst case genie MaxLog detector, as well as the genie LISS complexity (mean and 1% upper bound). The best case genie MaxLog computational complexity for the given configuration is $\mu = 8$ and the worst case is $\mu = 24$.

As becomes clear from the figure, the variance in complexity of the LISS is relatively small – there is roughly a factor of 2 between the average and the 99th percentile of the complexity. This is consistent with the results from [4]. Note, however, that for a list size of 16, the 1% upper bound is already around $\mu \approx 150$ branch metric computations – about 2/3 of the brute

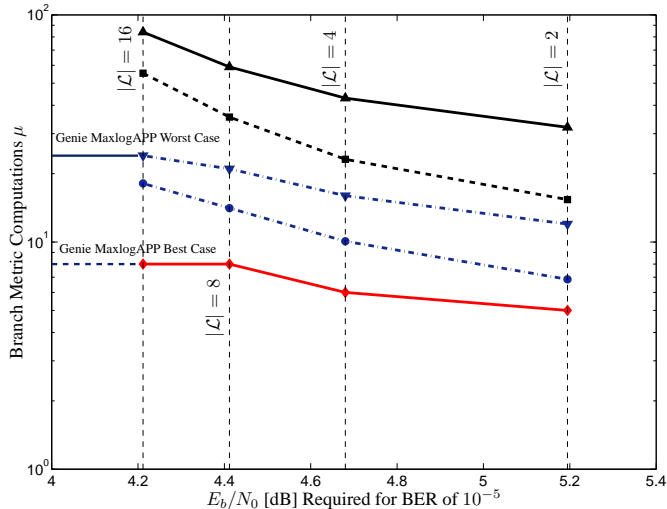


Figure 3: Complexity Bounds for 4×4 MIMO in Rayleigh fading using 4-QAM transmission and MMSE-SQRD equalization.

force MaxLogAPP detector. Considering the relative complexity of the practical LISS algorithm and the genie LISS, there is a factor of around 3-4 between the two cases if the average complexity is evaluated, and around 5, if the 1% worst case complexity is considered to be the most relevant design criterion. If a variable complexity of detection is acceptable, the room for improvement in terms of the available computational complexity is thus limited, at least for the considered (relatively small) problem size.

Using MMSE preprocessing is a well known concept to enhance the performance-complexity trade-off. Fig. 3 depicts results for the same system configuration as Fig. 2, except that now MMSE-SQRD preprocessing is employed. The lower bound does not change, but the upper bound changes significantly. Additionally, the genie MaxLogAPP computational complexities do not change, but the complexity of the genie LISS, like the realizable LISS, undergoes a reduction in the computational complexity due to the improved ordering. The impact of the MMSE-SQRD ordering is even more substantial for tree search schemes with higher variability in detection complexity than the LISS, such as the sphere decoder, and/or for larger problem sizes [4], which we consider now.

Fig. 4 depicts the complexity bounds outlined in the previous section for the case of 64-QAM transmission using MMSE-SQRD equalization. At this larger problem size, the difference between the genie based LISS and the upper bound (i.e. real LISS) become more evident – at a list size of 64, there is a factor of around 10 between the two cases, for the 1% bound on computational complexity. Also, the best case complexity of the genie MaxLogAPP detector remains unchanged at $\mu = 8$ while the worst case grows, due to the constellation growth, to $\mu = 64$. As can also be seen, the average number of branch metrics computed for the LISS spanning tree is roughly a factor 2 below the MaxLogAPP upper bound – some complexity can evidently be saved by not generating counter-hypotheses for

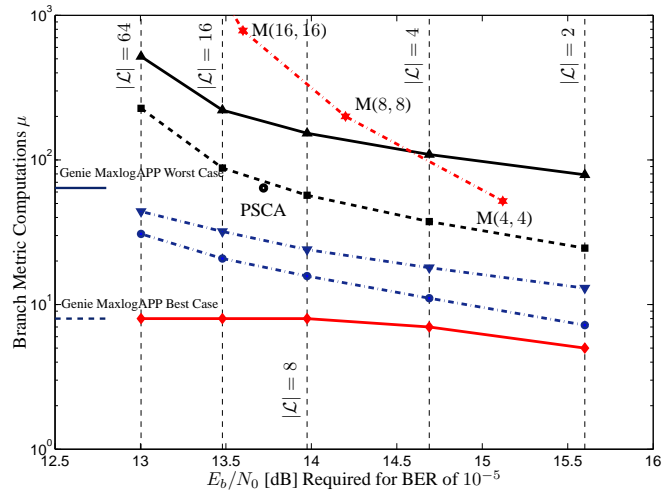


Figure 4: Complexity Bounds for 4×4 MIMO in Rayleigh fading using 64-QAM transmission and MMSE-SQRD equalization.

all detected bits. This constitutes an interesting path for further investigations into more efficient detection schemes.

In order to assess the complexity of tree search algorithms which are attractive for implementation, Fig. 4 also shows two fixed complexity algorithms: the parallel smart candidate adding (PSCA) algorithm with complexity equal to the worst case genie MaxLogAPP detector (i.e. PSCA parameterization $\mathbf{M} = \infty$, $\mathbf{b} = [1..1]$ and $\mathbf{S} = [1..1]$ [6]) and the M Algorithm [2, 6] with parameterization $M = b = 4, 8, 16$. This lowest complexity parameterization of the PSCA algorithm is within 1dB of MaxLogAPP performance, at a fixed detection complexity of $\mu = 64$ branch metric computations. The M algorithm with $M = b = 4$ is slightly less complex than the given PSCA algorithm, but its performance is more than 2dB worse than that of MaxLogAPP. The M algorithm with $M = b = 16$ performs 0.1 dB better than the given PSCA parameterization, but its complexity is over 12 times that of the PSCA. Note that the 99th percentile of the genie LISS, which can be considered the relevant complexity measure for the best possible fixed complexity scheme, is at roughly 30 branch metric computations for the performance corresponding to the PSCA. For the considered system setup, the employed PSCA configuration is thus, in terms of complexity, only a factor of two away from this very optimistic best case complexity metric. It should also be emphasized that the complexity of the PSCA is below the average complexity of the LISS, whilst requiring almost no sorting overhead and having substantially lower storage requirements. This indicates that the optimization criterion of the PSCA – ensuring the availability of counter-hypotheses – might be more suitable to achieve good performance, than an optimization solely based on the metrics of the leaf nodes. Note also the contrast to the M algorithm, which has consistently higher complexity than the average complexity of the LISS, and even beyond the 1% worst case complexity of the LISS, for list sizes of 8 and above. It is thus mainly attractive due to practical implementation considerations such as memory re-

quirements and control overhead. Additional information on the placement of state-of-the-art fixed complexity breadth-first list MIMO detection algorithms in the context of comparing performance and complexity can be found in [6].

Finally, Fig. 5 depicts a histogram for the number of nodes in the spanning tree of the genie LISS detector, for various $|\mathcal{L}|$. The data set from which this histogram was constructed were used to produce the LISS bounds (mean and 99th percentile) in Fig. 4. The lower bound for $|\mathcal{L}| = 2$ and the genie MaxLogAPP worst case are shown as reference. Interestingly, the distribution of nodes appears to converge to a Gaussian distribution for large enough list sizes. Exploiting this behavior for the design of even more efficient tree search schemes constitutes an interesting path for further research.

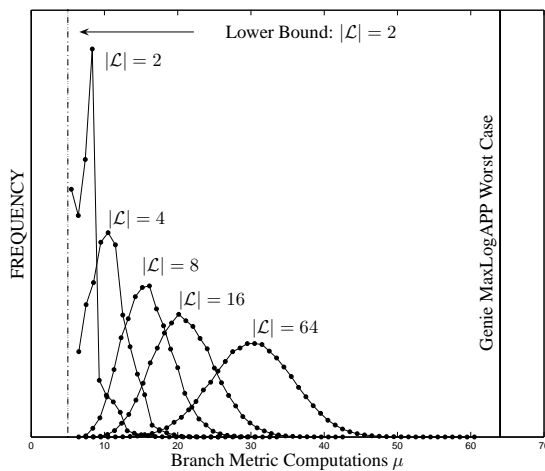


Figure 5: Histogram for the number of nodes μ in the MaxLog spanning tree for LISS with length $|\mathcal{L}|$.

VI CONCLUSIONS

In this contribution, we formulated upper and lower bounds on the computational complexity for list MIMO detection, based on the number of branch metrics computations. The list sequential decoder and a genie detector, having knowledge of the minimum spanning tree for a given list length, were used to establish the respective bounds. Evaluating representative examples of fixed complexity tree search schemes showed that for some relevant scenarios, the PSCA algorithm operates already within a factor of two of what can be considered the minimum complexity possible for a fixed complexity tree search scheme, required to achieve a performance within 1dB of the brute force MaxLogAPP detector.

The relative merit of different detection schemes will, of course, largely depend on the chosen channel code, the code rate, and the amount of diversity present in the channel. The results given in this work are thus representative for transmission over high diversity channels using powerful coding schemes. An extension of this analysis to other system configurations is the subject of ongoing work.

REFERENCES

- [1] B. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Transactions on Communications*, vol. 51, pp. 389–399, Mar. 2003.
- [2] Y. de Jong and T. Willink, "Iterative tree search detection for MIMO wireless systems," *IEEE Transactions on Communications*, vol. 53, pp. 930–935, Jun. 2005.
- [3] J. Hagenauer and C. Kuhn, "The List-Sequential (LISS) algorithm and its application," *IEEE Transactions on Communications*, vol. 55, no. 5, May 2007.
- [4] E. Zimmermann, "Complexity Aspects in Near-Capacity MIMO Detection-Decoding," Ph.D. dissertation, Technische Universität Dresden, Aug. 2007.
- [5] L. G. Barbero and J. S. Thompson, "Extending a Fixed-Complexity Sphere Decoder to Obtain Likelihood Information for Turbo-MIMO Systems," *IEEE Transactions on Vehicular Technology*, 2008, to appear.
- [6] D. L. Milliner, E. Zimmermann, J. R. Barry, and G. Fettweis, "A Framework for Fixed Complexity Single-Stage Breadth-First Detection," in *10th International Symposium on Spread Spectrum Techniques and Applications (ISSSTA'08)*, Bologna, Italy, 25.-28. Aug. 2008, invited paper.
- [7] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *IEEE International Conference on Communications (ICC 1995)*, Jun. 1995.
- [8] E. Zimmermann, D. L. Milliner, G. Fettweis, and J. R. Barry, "A parallel smart candidate adding algorithm for soft-output MIMO detection," in *Proc. 7th International ITG Conference on Source and Channel Coding (SCC 08)*, Ulm, Germany, Jan. 2008.
- [9] W. Xu, Y. Wang, Z. Zhou, and J. Wang, "A computationally efficient exact ML sphere decoder," in *IEEE Global Telecommunications Conference (GLOBECOM'04)*, vol. 4, Dec. 2004, pp. 2594 – 2598.
- [10] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," in *Mathematical Programming*, vol. 66, no. 1-3, Aug. 1994, pp. 181–199.
- [11] S. ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Transactions on Communications*, vol. 52, pp. 670–678, Apr. 2004.
- [12] D. Wübben, R. Böhnke, V. Kühn, and K. D. Kammeyer, "MMSE extension of V-BLAST based on sorted QR decomposition," in *Proc. IEEE Semiannual Vehicular Technology Conference (VTC2003-Fall)*, Orlando, USA, Oct. 2003.