A Parallel Smart Candidate Adding Algorithm for Soft-Output MIMO Detection

Ernesto Zimmermann and Gerhard Fettweis
Vodafone Chair Mobile Communications Systems
Technische Universität Dresden
D-01062 Dresden, Germany
Email: zimmere,fettweis@ifn.et.tu-dresden.de

David L. Milliner and John R. Barry
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332–0250
Email: dlm,barry@ece.gatech.edu

Abstract

Tree search schemes are an efficient means for solving the detection problem in multiple-input multiple-output systems. One option for implementing such a tree search is the smart candidate adding approach: using first an unconstrained search for the MAP estimate and then multiple constrained searches for counter-hypotheses. An inherent disadvantage of this strategy is that tree nodes might be visited multiple times, resulting in high detection complexity. This paper presents a parallel smart candidate adding algorithm, where the initial and the constrained searches are combined into a single detection step. Thus, complexity can be significantly reduced compared to previous proposals, at the cost of a minor performance loss. Furthermore, the algorithm has fixed detection complexity as opposed to most prior smart candidate adding proposals.

1 Introduction

Future wireless communications systems will make use of multiple antennas at the transmitter and the receiver to increase spectral efficiency. The main challenge for such MIMO systems lies in the nonorthogonality of the transmission channel, which renders difficult the calculation of reliability information (soft output) on the transmitted bits. In this context, tree search based detection techniques are known to enable a performance close to the optimal a posteriori probability (APP) detector while avoiding its prohibitively high complexity. Sphere [1], sequential [2] and M-algorithm based detection [3] are representative examples of such schemes. However, in order to ensure a highly accurate soft output, the "list" versions of the aforementioned algorithms need to generate a very large number of hypotheses on the transmit sequence – which obviously entails high detection complexity. Therefore, different authors [4]–[6] proposed to generate the soft output by using multiple instances of a Schnorr-Euchner sphere detector, each of which searches only for a single leaf node (list size 1). At first, a search for the MAP estimate is performed, followed by a set of searches for counter-hypotheses to this estimate. The term smart candidate adding (SCA) has been coined for this strategy in [4]. Recently [7], the authors extended and applied the SCA approach to other tree search detection algorithms, including the classical M-algorithm [8], yielding the SCA-M algorithm. An obvious disadvantage of the SCA strategy is that the different searches might visit similar parts of the detection tree – the total tree search complexity is thus somewhat higher than necessary to solve the task set to it.

Motivated by the above arguments, this contribution presents a Parallel SCA (PSCA) algorithm which avoids the aforementioned problem. Specifically, we propose to perform a single tree search which conveniently combines both the search for the MAP estimate and the constrained searches for appropriate counter-hypotheses to it. By removing the redundancy between the different tree searches, this approach allows for substantially reduced detection complexity.

The remainder of this paper is structured as follows: after discussing the employed system model in Section 2, Section 3 provides an introduction to tree search based MIMO detection. This is followed by a description of smart candidate adding and the proposed parallel algorithm in Section 4. Section 5 presents performance and complexity results. Finally, conclusions are drawn in Section 6.

2 System Model

Consider a $N_T \times N_R$ MIMO system based on a BICM transmit strategy as depicted in Figure 1: the vector $\mathbf{u}$ of i.i.d. information bits is encoded and interleaved. The resulting code 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 symbol, resulting in $Q = |\mathcal{C}| = 2^L$ different constellation points.

We consider transmission over a flat fading channel. In the equivalent discrete-time base-band model, the received signal $\mathbf{y}$ is given by:

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

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 $\mathcal{E}\{\mathbf{x}\mathbf{x}^{\mathrm{H}}\} = E_s/N_T\ \mathbf{I}$. 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: $\mathcal{E}\{\mathbf{n}\mathbf{n}^{\mathrm{H}}\} = N_0\ \mathbf{I}$. The signal-to-noise ratio (SNR) at each receive antenna is hence given by SNR $= E_s/N_0$.

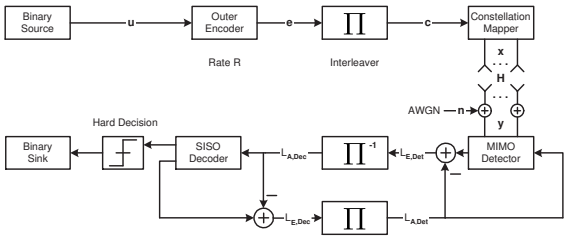

Figure 1. System model using a BICM transmit strategy.

3 Tree Search MIMO Detection

3.1 Fundamentals

The task of the detector is to calculate the a posteriori probability for each of the code bits $c_{m,l}$ in $\mathbf{x}$, where $m \in \{1, \ldots, N_T\}$ is the symbol index, and $l \in \{1, \ldots, L\}$ is 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):

$$L(c_{m,l}|\mathbf{y}) := \ln \frac{P\left[c_{m,l} = +1|\mathbf{y}\right]}{P\left[c_{m,l} = -1|\mathbf{y}\right]} \tag{2}$$
$$\approx \max_{\mathbf{x} \in \mathcal{X}_{m,l}^{+1}} \left\{ \frac{-\left\|\mathbf{y} - \mathbf{H}\mathbf{x}\right\|^2}{N_0} + \sum_{m=1}^{N_T} \sum_{l=1}^{L} \ln P[c_{m,l}] \right\}$$
$$- \max_{\mathbf{x} \in \mathcal{X}_{m,l}^{-1}} \left\{ \frac{-\left\|\mathbf{y} - \mathbf{H}\mathbf{x}\right\|^2}{N_0} + \sum_{m=1}^{N_T} \sum_{l=1}^{L} \ln P[c_{m,l}] \right\},$$

where the second line follows from the application of the so-called max-log approximation. Here, $\mathcal{X}_{m,l}^{\pm 1}$ denotes the set of $2^{N_T \cdot L - 1}$ symbols $\mathbf{x} \in \mathcal{X}$ for which $c_{m,l} = \pm 1$. Evaluating (2) 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 (2). Several close-to-optimal detection strategies therefore construct a subset list $\mathcal{L} \subset \mathcal{X}$ from which the LLRs are determined. The subset should on the one hand include only a fraction of the elements from $\mathcal{X}$ to minimize complexity. On the other hand, it should be large enough to allow one to approximate the true detector LLRs as closely as possible, to maximize performance. Let the size of the list $\mathcal{L}$ be denoted as $M = |\mathcal{L}|$. Tree search based MIMO detection techniques construct $\mathcal{L}$ using a back-substitution approach. After a QR-decomposition of the channel matrix, $\mathbf{H} = \mathbf{Q}\mathbf{R}$, the LLRs can be determined using the per-antenna metric increments Λ_m:

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

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

$$\Lambda_m = -\frac{1}{N_0} \left| \tilde{y}_m - \sum_{j=m}^{N_T} r_{m,j} x_j \right|^2 + \sum_{l=1}^{L} \ln \Pr[c_{m,l}] \tag{3}$$

with $\tilde{\mathbf{y}} = \mathbf{Q}^{\mathrm{H}}\mathbf{y}$. The detector starts in layer $m = N_T$ and works its way up until layer $m = 1$ is reached. For each branch in the tree, Q different choices are possible for the signal estimate x_m. 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* $\sum_{m=n}^{N_T} \Lambda_m$ to control which tree nodes are added to the working stack and in which order. Tree search schemes can generally be broken down into three representative classes of algorithms (see, e.g. [8]): *depth-first search*, *metric-first search* and *breadth-first search*.

Depth-first schemes (such as the *sphere decoder* [1]) consider only a single tree node at a time, whose path metric is checked against a certain threshold (the sphere radius), to prune paths from the search tree which are unlikely to produce leaf nodes with good metrics. Metric-first schemes (with the *sequential decoder* [2] as classical representative) keep multiple nodes in parallel in a stack structure and always extend the node with the currently best path metric. A detailed description and analysis of these two schemes is beyond the scope of this paper. An overview of the performance and complexity of all three tree search approaches in the context of MIMO detection is given in [7].

3.2 Breadth-First Tree Search

In this work, we will focus on breadth-first search algorithms. These algorithms extend the detection tree layer-by-layer. At each depth, the M nodes with the largest path metrics are retained and all other nodes are dropped. The classical example for this approach is the M-Algorithm [3]. The advantage of this technique, over depth-first or metric-first search techniques, is the fixed detection complexity. However, the achievable performance is limited by error propagation, particularly for low values of M.

One approach for improving the performance of the M-algorithm is the channel-based level-adaptive M-algorithm (CLAM) [9], which is similar to the M-algorithm for searching the detection tree, except that it varies from one stage to the next the number of children extended from each retained node. These numbers are optimized based on knowledge of the channel at the receiver. This simple enhancement not

only reduces complexity, it also enables the CLAM algorithm to significantly outperform the M algorithm in both hard-detection and soft-detection systems. Another breadth-first algorithm of interest is the fixed-complexity sphere decoder (FSD) [10]. Similar to the CLAM algorithm, the FSD determines the number of candidates to extend from each retained node from the previous stage in the detection tree. Unlike the CLAM algorithm, these numbers are not based on the detection layer SNRs.

An analysis of the CLAM and FSD algorithms reveals in both cases that the number of candidates extended from each retained node from the previous detection stage should be largest early in the detection tree. This is because more candidates need to be considered in the first detection levels due to interference from the other levels, while decision-feedback equalization (DFE) reduces the number of candidates that need to be considered in the last levels [10]. We will exploit this observation in the PSCA algorithm.

4 Parallel Smart Candidate Adding

4.1 Fundamentals

A major problem for all "conventional" tree search schemes are missing counter-hypotheses: whenever $\mathcal{L} \cap \mathcal{X}_{m,l}^{\pm 1} = \emptyset$, the magnitude of the LLR for the corresponding bit cannot be determined from the entries of $\mathcal{L}$. To avoid this situation (and thus ensure a reasonable quality of the soft output) the size of $\mathcal{L}$ must be chosen large enough. However, this often requires an undesirably high detection effort. Another solution is to simply clip the magnitude of the soft output to a certain predefined value [1]. But the system performance will then be very sensitive to the choice of the clipping level [3], particularly for small list sizes.

The alternative is to directly address the counter-hypothesis problem using smart candidate adding [4]–[6]. From (2) it is easily seen that the LLRs at the output of the maxLogAPP detector may also be written as the difference between the metric of the *MAP estimate* $\mathbf{x}^{\text{MAP}}$ (i.e., the hypothesis which maximizes the a posteriori probability) and the metric of an appropriate counter-hypothesis for each bit:

$$L(c_{m,l}|\mathbf{y}) \approx c_{m,l}^{\text{MAP}} \left(\sum_{1}^{N_T} \Lambda_m \left(\mathbf{x}^{\text{MAP}} \right) \quad \dots \right.$$
$$\left. - \max_{\mathbf{x} \in \mathcal{X}_{m,l}^{-\text{MAP}}} \left\{ \sum_{1}^{N_T} \Lambda_m \left(\mathbf{x} \right) \right\} \right). \quad (4)$$

with $\mathbf{c}^{\text{MAP}}$ as the bit pattern of the MAP estimate and $\mathcal{X}_{m,l}^{-\text{MAP}}$ the set of potential counter-hypotheses, for which $c_{m,l} = -c_{m,l}^{\text{MAP}}$. The maxLogAPP detection problem may hence be solved by first finding the MAP estimate and then performing $N_T \cdot L$ searches which cover only a subset of the transmitter signal set. It should be noted that bounding the complexity of the different tree searches may still lead to overestimated LLR magnitudes – which would necessitate the use of LLR clipping. Fortunately, the performance of reduced complexity Smart Candidate Adding has been found to be very robust to the choice of the clipping level [7].

4.2 SCA M-Algorithm

Applying SCA techniques to the classical M-algorithm yields the SCA-M algorithm [7]. This algorithm serves as motivation for our parallel breadth-first search algorithm and can be broken down into two stages:

- **Stage 1:** The *search for the MAP estimate* covers the entire signal set $\mathcal{X}$ and should be constructed such that errors in the hard output of the MIMO detector are avoided, i.e., the MAP estimate has to be found with high probability. This stage is implemented using an M-algorithm with a sufficiently large list size (denoted as M_1).
- **Stage 2:** Each *search for a counter-hypothesis* covers only a constrained signal set $\mathcal{X}_{m,l}^{-\text{MAP}}$, and is referred to as a *constrained search*. This is the computationally most expensive part, since the number of constrained searches scales with the number of transmitted bits per vector symbol. An M-Algorithm with M_2 is used for this task.

Reducing the complexity of the search for counter-hypotheses is thus crucial to achieving a good performance-complexity trade-off. In this respect, breadth-first search algorithms offer a major advantage: Due to their layer-by-layer operation, constraints on certain bits don't propagate root-wards in the tree structure. Therefore, it is possible to build up a "tree of search trees" for the second stage, such that no tree node is visited twice over the set of all searches for counter-hypotheses. In [7] it was shown that good performance is achievable when the constrained search list length is very small (i.e. $M_2 = 1$). The redundancy between the first and the second search stage can be reduced by exploiting the fact that the initial search produces M_1 leaf nodes, some of which already provide a counter-hypothesis for certain bits. The constrained searches have to be performed only for a subset of the transmitted bits. However, this approach leads to a slightly variable complexity for the SCA-M algorithm. Compared to conventional tree search schemes, the complexity in terms of the number of branch metric computations was reduced by a factor of 1.5-2 in an iterative detection-decoding setup [7]. Yet, there is still redundancy between the first and the second stage, which motivates the design of more efficient schemes.

4.3 Parallel SCA Algorithm

The principal difference between the parallel SCA algorithm and the SCA-M algorithm is that the constrained searches for counter-hypotheses are found *concurrently* with the MAP estimate as the breadth-first search

proceeds through the detection tree, rather than through supplemental searches. This is similar to the "parallel sphere detector" approach taken in [11]. However, since the search tree is constructed layer-by-layer, the counter-hypothesis for a bit of interest can only be found relative to the best *partial* MAP estimate at the current level in the detection tree – as opposed to the *full* MAP estimate for the traditional approach. The price paid for the complexity savings realized by the PSCA algorithm is thus a small loss in performance.

Two further important modifications to the SCA-M algorithm are necessary: firstly, to avoid the complexity of checking all nodes in the detection tree whether they already constitute a counter-hypotheses, the counter-hypotheses are created by enumerating appropriate "sibling nodes" of the current partial MAP estimate. As a positive side-effect, the generated nodes will very likely have good metrics as they are "close" to the node in the tree with the metric which is currently the best. Secondly, it must be avoided that any path providing a counter-hypothesis is pruned during the search process. This is done by extending *all* generated nodes until they have reached full length, by employing a Schnorr-Euchner/SIC-like strategy (i.e., at each depth only the child node with the best metric is retained; the other child nodes are not considered). The sort-and-select stage of the M-Algorithm can thus be omitted. This is similar to the approach followed by the FSD [10].

The growth rate of the search tree is controlled by the number of child nodes which are enumerated per parent node retained from the previous level. We denote this (level-dependent) figure by b_m, which is summarized in the parameter vector $\mathbf{b} \in \mathbb{Z}^{N_T \times 1}$, where $b_m \geq 1 \forall m$. Tuning $\mathbf{b}$ allows trading tree search complexity for detection performance. An appropriate choice of $\mathbf{b}$ can be found by using an approach similar to the `generate_b` function of the CLAM algorithm [9].

A pseudo code description of the PSCA algorithm is provided in Fig. 2. Let S_m denote the set of retained (survivor) nodes at level m of the tree, with $\mathcal{S}_{N_T+1}$ initialized to the root node. Furthermore, $\mathcal{S}_m^{\text{PMAP}}$ denotes the partial MAP estimate at level m, and $\mathcal{C}_l^{-\text{PMAP}}$ is the set of symbols which can potentially provide a counter-hypotheses for bit l of this estimate. The algorithm starts at the root node and finds its b_{N_T} best children, yielding $\mathcal{S}_{N_T}$. The best child ($\mathcal{S}_{N_T}^{\text{PMAP}}$) is defined to be the one possessing the branch metric with the minimum value. The additional child nodes (optionally) enumerated beyond $\mathcal{S}_{N_T}^{\text{PMAP}}$ will already provide counter-hypotheses for some of the bits. For the remaining bits in $\mathcal{S}_{N_T}^{\text{PMAP}}$, the node with the best metric emanating from the root and having the bit of interest flipped w.r.t. $\mathcal{S}_{N_T}^{\text{PMAP}}$ is found and added to $\mathcal{S}_{N_T}$. In this contribution, we constrain ourselves to the case $b_{N_T} \leq 2$ and Gray mapping. The branch enumerated beyond the partial MAP estimate thus provides a counter-hypothesis only for a single bit – which can be easily identified.

Input: $\mathbf{y}, \mathbf{H}, \mathcal{C}, L, \mathbf{b}$
Output: $\mathcal{L}$

1 $\mathbf{HP} = \mathbf{QR}$, $\tilde{\mathbf{y}} = \mathbf{Q}^{\text{H}}\mathbf{y}$
2 $\mathcal{S}_{N_T+1}$ = root node with metric of 0
3 $\mathcal{S}_{N_T} = \{b_{N_T}$ best children of $\mathcal{S}_{N_T+1}\}$, $\mathcal{S}_{N_T}^{\text{PMAP}}$ = best of $\mathcal{S}_{N_T}$
4 **for** $l = 1 : L$ **do**
5 | `node` = best child of $\mathcal{S}_{N_T+1} \in \mathcal{C}_l^{-\text{PMAP}}$
6 | $\mathcal{S}_{N_T} = \mathcal{S}_{N_T} \cup$ `node`
7 **end**
8 **for** $m = N_T - 1 : -1 : 1$ **do**
9 | $\mathcal{S}_m = \cup_{\texttt{node} \in \mathcal{S}_{m+1}} \{b_m$ best children
10 | of `node`$\}$
11 | $\mathcal{S}_m^{\text{PMAP}}$ = best of $\mathcal{S}_m$
12 | **for** $l = 1 : L$ **do**
13 | | `node` = best child $\in \mathcal{C}_l^{-\text{PMAP}}$ of parent of $\mathcal{S}_m^{\text{PMAP}}$
14 | | $\mathcal{S}_m = \mathcal{S}_m \cup$ `node`
15 | **end**
16 **end**
17 $\mathcal{L}$ = $\mathbf{P}\mathcal{S}_1$

Figure 2. Parallel SCA Algorithm. Depending on b_m, some counter-hypotheses are already available and do not have to be added explicitly (lines 4-7,12-15) – see text.

The algorithm continues to the next level by extending exactly b_{N_T-1} children (the ones with the best metrics) from each node in $\mathcal{S}_{N_T}$, yielding $\mathcal{S}_{N_T-1}$. The best node in this set becomes $\mathcal{S}_{N_T-1}^{\text{PMAP}}$. The PSCA algorithm continues by finding the remaining $1 + L - b_{N_T}$ counter-hypotheses to this partial MAP estimate (and its $b_{N_T-1}-1$ siblings), and appends them to $\mathcal{S}_{N_T-1}$ before advancing to the next detection level. This process continues until the layer $m = 1$ is reached, at which point the full candidate list $\mathcal{L}$ is available, without the need for a second stage search for counter-hypotheses.

A common metric for quantifying the complexity of a tree search algorithm is the number of visited nodes (or branch metric computations) n_c [7], [9]. If we consider an equivalent real-valued system model with $N_{2T} = 2N_T$ and $N_B = L/2$ and set $b_m = 1 \ \forall m$, the PSCA algorithm considers precisely

$$n_c = \sum_{m=1}^{N_{2T}} 1 + mN_B = N_{2T} + N_B \frac{N_{2T}(N_{2T}+1)}{2} \quad (5)$$

nodes and the final list size is $|\mathcal{L}| = 1 + N_T \cdot L$. Due to its recursive nature, the expression for the number of visited nodes is slightly more involved for a general $\mathbf{b}$, but it is easily seen that n_c is deterministic as well.

As will be shown in the next section, the number of nodes visited by the PSCA-M is generally lower than that of the M-Algorithm or the SCA-M achieving the same performance. Furthermore, the PSCA-M does not require any list sorting and is easier to parallelize – a major advantage for practical implementation.

5 Results

To ensure comparability of results, we use a setup equivalent to the one in [1]: A rate 1/2 PCCC based on $(7_R, 5)$ convolutional codes is employed for transmission over a 4×4 MIMO channel which is spatially and temporally i.i.d. fading. The information block size (including tail bits) is 9216 bits. The PCCC decoder uses 8 internal iterations (logMAP decoding). Detection is performed based on the equivalent real-valued system model. The LLRs were clipped at a magnitude of ± 6 for all investigated techniques.

Figure 3 shows results for the case for 4-QAM transmission. As for all tree search schemes with fixed (or tightly bounded) detection complexity [12], both the SCA-M and the PSCA-M benefit significantly from the use of MMSE preprocessing. This enables the SCA-M algorithm with $M_1 = 4$ and $M_2 = 1$ (square markers) to achieve a performance 0.2dB from MaxLogAPP detection. The average number of visited nodes for this algorithm is $n_c = 84$, the upper bound is at $n_c^{\max} = 97$. The least complex version of the PSCA-M algorithm ($b_m = 1\ \forall m$, circle markers) suffers a performance loss of 0.2dB, but visits only $n_c = 44$ nodes – a complexity reduction by a factor of roughly 2. The loss in performance can be decreased to below 0.1dB by choosing $b_m = 2$ for the first 3 detected layers (diamond markers), at the cost of $n_c = 69$ visited tree nodes. Note that all schemes operate within 0.5dB of MaxLogAPP detection which visits $n_c^{\text{APP}} = 510$ nodes.

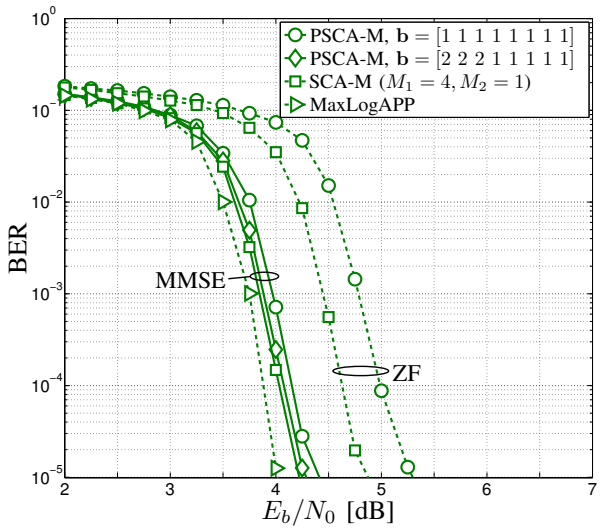

Figure 3. Performance of SCA-M and PSCA-M tree search detection (4×4 MIMO, 4-QAM). Dashed lines: ZF-SQRD, solid lines: MMSE-SQRD preprocessing (unbiased, cf. [12]).

The case of 64-QAM transmission depicted in Figure 4 is more challenging: The MaxLogAPP detector would perform $n_c^{\text{APP}} \approx 19 \cdot 10^6$ branch metric computations – which is clearly infeasible. Performance reasonably close to MaxLogAPP detection can be achieved with an M-Algorithm with $M = 64$. The number of visited nodes is $n_c = 3144$. This figure can be reduced to slightly above $1,000$ nodes[1] by employing the pragmatic Schnorr-Euchner-like enumeration strategy proposed in [7]. However, this approach has no longer fixed tree search complexity. Although the variation is minimal, this might be undesirable from an implementation perspective.

For the PSCA-M algorithm, the use of MMSE preprocessing provides a performance gain of almost 1dB, comparable to the 4-QAM case. The configuration with $b_m = 1\ \forall m$ then achieves performance within 0.2dB of the SCA-M algorithm with $M_1 = 4, M_2 = 1$ and within roughly 0.7dB for the SCA-M algorithm with $M_1 = 16, M_2 = 1$. Enumerating a greater number of tree nodes in the first detected layers again improves performance (by 0.5dB for $b_m = 2$ in the first 4 layers).

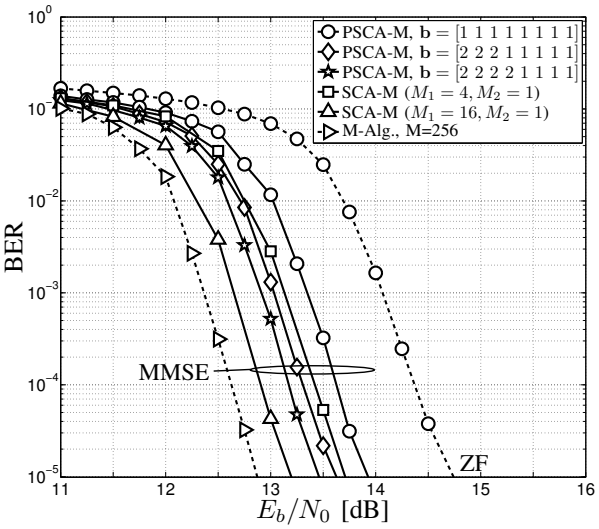

Figure 4. Performance of SCA-M and PSCA-M (4×4 MIMO, 64-QAM). Dashed lines: ZF-, solid lines: MMSE-SQRD (unbiased).

The performance and complexity of the investigated schemes, in terms of the number of visited tree nodes vs. the SNR required to achieve a BER of 10^{-5} is summarized in Figure 5 for the 64-QAM case. Here, N_2 denotes the number of levels from the root in which $b_m = 2$ for the PSCA-M algorithm. For the SCA-M and the M-Algorithm, both the figures for the standard approach (triangle markers) and for the Schnorr-Euchner approach (square and + markers, respectively) are given. In order to ensure a fair comparison, we consider only schemes with fixed complexity in the following, i.e., the standard M-Algorithm and the upper bound on the SCA-M complexity.

The SCA-M based on the standard M-algorithm and using $M_1 = 4$ and $M_2 = 1$ visits $n_c^{\max} = 347$ nodes in the worst case. This is to be compared with the PSCA-M algorithm with $N_2 = 0$ (i.e., $b_m = 1\ \forall m$) which visits only $n_c = 116$ nodes – a complexity reduction by a factor of 3 – and suffers a performance loss of roughly 0.2dB.

[1]The average number of child nodes per parent node is $\bar{b} = 2.7$.

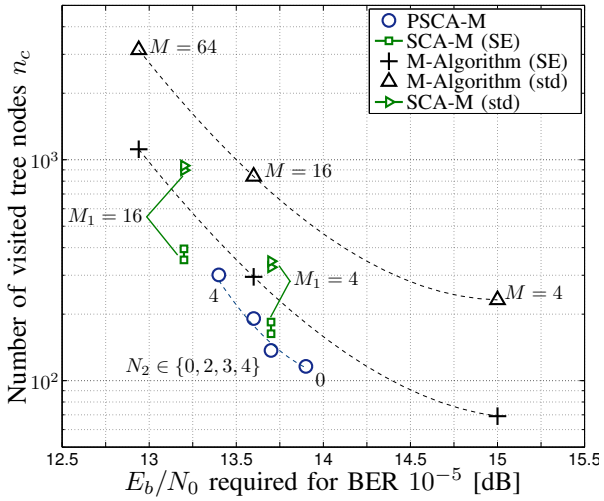

Figure 5. Performance-complexity trade-off for different tree search techniques (4×4 MIMO, 64-QAM, unbiased MMSE-SQRD).

This loss can be removed by setting $N_2 = 2$ which increases n_c to 137. Choosing $N_2 = 4$ increases performance to within 0.2dB of the SCA-M algorithm with $M_1 = 16$. The number of visited tree nodes is $n_c = 301$ and $n_c = 940$, respectively – again a complexity reduction by a factor of 3. Note that the PSCA approach with $N_2 = 3$ gains almost 1.5dB over the standard M-Algorithm with $M = 4$, at roughly a 20% lower number of visited visited tree nodes.

Algorithm	Setup	4-QAM	64-QAM
M-Algorithm	$M = 4$	744	1912
	$M = 16$	–	6600
SCA-M	$M_1 = 4, \ M_2 = 1$	1244	3228
	$M_1 = 16, \ M_2 = 1$	–	7640
PSCA	$N_2 = 0$	700	1668
	$N_2 = 2$	791	1937
	$N_2 = 3$	1032	2658

Table I
TREE SEARCH COMPLEXITY (NUMBER OF OPERATIONS)

However, the number of nodes visited during the tree search tells only part of the story. This is illustrated by the complexity figures[2] given in Table I: comparing the PSCA-Algorithm with $N_2 = 2$ and the SCA-M with $M_1 = 4, M_2 = 1$ shows that the complexity saving *reduces* from a factor of 2.5 (if n_c is considered) to 1.7 (in terms of the operation count). This is due to the fact that the search tree of the PSCA-M is constantly growing in width, in contrast to the M-Algorithm: Nodes visited in the last detected layers cost more effort for calculating the interference reduced received signal $\tilde{y}_m - \sum r_{m,j} x_j$. Hence, the complexity savings achieved by the reduced sorting effort are overcompensated by the added effort in metric calculation. This fact should be taken into account in the design of efficient tree search schemes and is the subject of ongoing research.

[2]The figures include calculating $\tilde{\mathbf{y}}$, the tree search stage and the LLR calculation. Multiplication, addition, comparison each count as one operation. The overhead for sorting is included.

6 Conclusions

In this contribution, we presented the parallel smart candidate adding algorithm as an efficient way to achieve near-capacity performance in MIMO systems. By performing a single directed search, rather than multiple directed searches, we substantially reduced the complexity compared to previous, serial, versions of the smart candidate adding approach, at the cost of a minor performance loss. It was shown that for a PCCC coded 4×4 MIMO system with 4-QAM modulation, the PSCA approach requires only half the number of branch metric computations of its serial counterpart, at only a 0.2dB loss in performance. For the case of 64-QAM modulation, the number of visited nodes could be reduced by a factor of 3, at the same low performance loss. The PSCA-M algorithm offers several advantages which render it particularly attractive for real-time implementation: it has a fixed complexity, requires no sorting and also offers a very high potential for parallelization.

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] J. Hagenauer and C. Kuhn, "The List-Sequential (LISS) Algorithm and Its Application," *IEEE Transactions on Communications*, vol. 55, pp. 918–928, May 2007.

[3] S. Haykin, M. Sellathurai, Y. de Jong, and T. Willink, "Turbo-MIMO for wireless communications," *IEEE Communications Magazine*, vol. 42, pp. 48– 53, Oct. 2004.

[4] P. Marsch, E. Zimmermann, and G. Fettweis, "Smart Candidate Adding: A new Low-Complexity Approach towards Near-Capacity MIMO Detection," in *13th European Signal Processing Conference (EUSIPCO'05)*, Antalya, Turkey, Sept. 2005.

[5] M. S. Yee, "Max-log-MAP sphere decoder," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'05)*, vol. 3, Mar. 2005, pp. 1013–1016.

[6] R. Wang and G. Giannakis, "Approaching MIMO channel capacity with Soft Detection Based on Hard Sphere Decoding," *IEEE Transactions on Communications*, vol. 54, pp. 587–590, Apr. 2006.

[7] E. Zimmermann and G. Fettweis, "Generalized Smart Candidate Adding for Tree Search Based MIMO Detection," in *ITG/IEEE Workshop on Smart Antennas (WSA'07)*, Vienna, Austria, Feb. 2007.

[8] J. Anderson and S. Mohan, "Sequential coding algorithms: A survey and cost analysis," *IEEE Transactions on Communications*, vol. 32, pp. 169–176, Feb. 1984.

[9] D. Milliner and J. R. Barry, "An Adaptive M-Algorithm for Detection of Multiple-Input Multiple-Output Channels," in *IEEE Signal Processing Advances in Wireless Communications, (SPAWC'07)*, Helsinki, Finland, 17.-20. June 2007.

[10] L. G. Barbero and J. S. Thompson, "A Fixed-Complexity MIMO Detector Based on the Complex Sphere Decoder," in *IEEE Workshop on Signal Processing Advances for Wireless Communications (SPAWC 06)*, Cannes, France, July 2006.

[11] J. Jalden and B. Ottersten, "Parallel Implementation of a Soft Output Sphere Decoder," in *Conference Record of the 39th Asilomar Conference on Signals, Systems and Computers*, Oct. 2005, pp. 581–585.

[12] E. Zimmermann and G. Fettweis, "Unbiased MMSE Tree Search MIMO Detection," in *International Symposium on Wireless Personal Multimedia Communications (WPMC'06)*, San Diego, USA, Sept. 2006.