

Complexity reduced Soft-In Soft-Out Sphere Detection based on Search Tuples

Björn Mennenga and Andreas von Borany and Gerhard Fettweis

Vodafone Chair Mobile Communication Systems

Technische Universität Dresden

Dresden, Germany

Email: mennenga, andreas.borany, fettweis@ifn.et.tu-dresden.de, andreas.borany, fettweis@ifn.et.tu-dresden.de

Abstract—Depth-first tree search algorithms provide a promising approach to solve the detection problems in MIMO systems. Realizations like the List Sphere Detector (LSD) or the Single Tree Search (STS) enable near max-log detection at reduced but still high complexity. In this paper we show how the complexity of List Sphere Detection can be significantly reduced by MMSE preprocessing in combination with a novel unbiased and separated candidate handling. Therefor, we propose an extension of the LSD by search tuples. Without any performance loss, the resulting Tuple Search (TS) algorithm enables major reduction of sphere sizes and enables moreover a detection with flexible performance respectively complexity. Avoiding loss of useful status information, caused by unbiased MMSE preprocessing or small candidate storage, is provided by a novel matched candidate determination, leading also to reduced hardware complexity. The combination of these methods enable high-performance soft-out detection at very low complexity. More specifically, this enables a performance improvement up to 1 dB at half the complexity of common LSD or STS algorithms.

I. INTRODUCTION

Future mobile communication systems will make use of multiple-input multiple-output (MIMO) techniques to enhance spectral efficiency. While performant detection of the spatial multiplexed data streams, with respect to the channel conditions, is essential for good performance, its complexity is limiting the overall throughput. Many recent research try to solve this problem by introducing complexity reduced detection algorithms based on tree search techniques. Application of depth-first, breadth-first or metric-first search strategies [1] leads to a large variety of detection algorithms like the sphere detector [2], m-algorithm [3], LISS-algorithm [4] or modifications of them [5]–[7]. Reduction of the number of evaluated tree nodes is achieved by techniques like Schnorr-Euchner (SE) enumeration [8] or sorted QR decomposition (SQRD) [9]. MMSE preprocessing with bias reduction [10] enables further complexity reduction, but also significantly worsens the achievable performance in common detection algorithms.

However, especially for higher order systems, near maximum-likelihood (ML) detection still involves a too high complexity to be applicable in high-throughput systems. Otherwise, detectors with reasonable complexity, like linear detection, hard-output LSDs [11] or Successive Interference Cancellation (SIC) [12] detectors, do not provide a detection accuracy close to ML, limiting their field of application. Moreover, flexible implementations, adaptive to the environment, are still too inefficient to be applied.

In order to further reduce detection complexity, we introduce a novel Tuple Search algorithm. Based upon the LSD, we propose a separation of the candidate list and the search tuple, defining the search sphere. Hereon, we demonstrate how adaptation of search radii enables a reduction of evaluated nodes, involving also a flexible adaptation of detection complexity respectively performance. Auxiliary, this furthermore enables a novel matched candidate processing, minimizing the disadvantageous effects of the unbiased MMSE

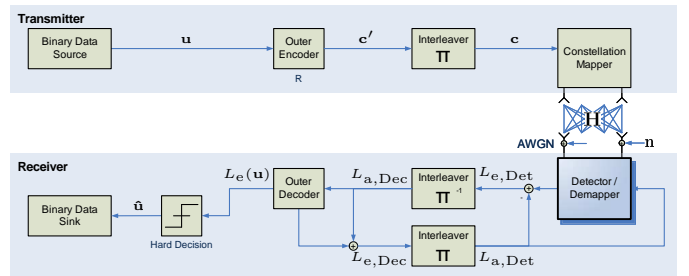


Fig. 1. System model with BICM transmitter and iterative receiver.

detection. Since usage of candidate lists involve a possible loss of useful status information, we additionally propose the replacement of candidate lists by bit-by-bit processing and storage.

The remainder of this paper is organized as follows: Subsequent to the introduction, section II discusses the system model and the parameters used for the simulations. Section III reviews and discusses tree search based soft-in soft-out detection, the LSD and the STS algorithms, as well as applied complexity reduction techniques. Based on this, we describe the proposed Tuple Search in section IV. Based on the Tuple Search we introduce the matched candidate determination including a lossless involvement of unbiased MMSE detection in section V. Finally we evaluate the resulting algorithms in section VI and conclude the paper in section VII.

II. SYSTEM MODEL

Throughout this paper, we consider a $N_T \times N_R$ MIMO system based on a bit-interleaved coded modulation (BICM) transmission strategy with N_T transmit and N_R receive antennas, as depicted in Figure 1. A vector \mathbf{u} of i.i.d. information bits is encoded by the outer channel code with rate R . The resulting stream of vectors \mathbf{c}' is bit-interleaved and portioned into blocks \mathbf{c} of $N_T \cdot L$ bits, where L denotes the number of bits per transmit symbol. For the transmission, the corresponding bits $\mathbf{c} \in \mathcal{C}$, covered in the set of permitted bit vectors, are mapped (e.g. gray mapping) onto complex constellation symbols $\mathbf{x}(\mathbf{c}) = [x_0, \dots, x_{N_T-1}]^T = \text{map}(\mathbf{c}) \in \mathcal{X}$, the set of valid transmit symbols with cardinality $\#\mathcal{X} = \#\mathcal{C} = 2^L$. We normalize the transmit energy such that $\mathcal{E}\{\mathbf{xx}^H\} = E_S/N_T\mathbf{I}$. On behalf of the transmission, we consider a flat fading channel and an additive noise vector $\mathbf{n} \in \mathbb{C}^{N_R \times 1}$ at the receiver with complex components of zero mean i.i.d. gaussian random variables of variance $N_0/2$ per real dimension ($\mathcal{E}\{\mathbf{nn}^H\} = N_0\mathbf{I}$). The considered passive channel is represented by $\mathbf{H} \in \mathbb{C}^{N_T \times N_R}$ with entries of a zero mean i.i.d. gaussian random process of variance 1 and is assumed to be perfectly known at the receiver. The received signal \mathbf{y} is therefore given by:

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

and the signal-to-noise-ratio ($SNR = E_s/N_0$) at the receiver applied to the energy of one information bit can be stated as: $E_b/N_0 = E_s N_R / N_0 N_T L R$.

In order to ensure comparability of the results, we use a simulation setup equivalent to the one used in e.g. [5], [10]. The simulations are carried out for a rate 1/2 PCCC with $(7_R, 5)$ convolutional codes, an information block size of 9216 bits (including tail bits), gray mapping, a 4×4 MIMO channel and spatial and temporal fading. The detection of the transmitted bits is carried out by a complex-valued soft-in soft-out (SISO) sphere detector in conjunction with a BCJR based decoder with 8 internal iterations and without detector \leftrightarrow decoder iterations.

III. MIMO DETECTION BASED ON TREE SEARCH

A. Fundamentals

The task of the focused detector is the determination of the bits c most likely sent and the calculation of reliability information for these bits. On behalf of the described system, this can be accomplished by calculating the corresponding log-likelihood ratios (L-values):

$$L(c_{m,l}|\mathbf{y}) = \ln \left(\frac{P(c_{m,l} = +1|\mathbf{y})}{P(c_{m,l} = -1|\mathbf{y})} \right) \approx -\frac{1}{N_0} \min_{c|c_{m,l}=+1} \{\lambda_0\} + \frac{1}{N_0} \min_{c|c_{m,l}=-1} \{\lambda_0\}, \quad (1)$$

where (1) results from the application of the max-log approximation. $c_{m,l} = \pm 1$ represents the l -th bit of the symbol sent by the m -th antenna and

$$\lambda_0(\mathbf{y}, \mathbf{c}, \mathbf{L}_a) = \|\mathbf{y} - \mathbf{H}\hat{\mathbf{x}}(\mathbf{c})\|^2 - \frac{N_0}{2} \sum_{i=0}^{N_T-1} \sum_{j=1}^L c_{i,j} L_a(c_{i,j})$$

represents the distance metric for a set of received symbols \mathbf{y} , a given \mathbf{c} and the a-priori knowledge \mathbf{L}_a . $\hat{\mathbf{x}}$ corresponds to a possible transmission symbol. As consequence, beside the most properly sent symbol $\arg \min_{\hat{\mathbf{x}}(\mathbf{c})|\mathbf{c} \in \mathcal{C}} \{\lambda_0\}$ - the detection

hypothesis - and its corresponding metric $\lambda_0(\mathbf{c}^{\text{ML}})$, the detector has to determine also the counter-hypotheses $\arg \min_{\hat{\mathbf{x}}(\mathbf{c})|\mathbf{c} \in \mathcal{C}, c_{m,l} \neq c_{m,l}^{\text{ML}}} \{\lambda_0\}$ with

their metrics for each bit.

B. Tree Search Basics

Since brute force (maxlogAPP) detection of (1) is known to be of exponential growing complexity with the number of transmit antennas, several close to optimal detection strategies have been lately proposed to find relevant $\arg \min \{\lambda_0\}$. Some of the most promising are based on tree search techniques. As depicted in detail in [5], transforming the detection problem is permitted by QR-decomposition (QRD) of $\mathbf{H} = \mathbf{Q}\mathbf{R}$, where \mathbf{Q} is unitary and \mathbf{R} an upper triangular matrix. With the modified receive symbols $\mathbf{y}' = \mathbf{Q}^H \mathbf{y}$ and the potential sent symbols \hat{x}_i , $i = 0 \dots (N_T - 1)$, the euclidian distance in the detection

$$\|\mathbf{y}' - \mathbf{R}\hat{\mathbf{x}}(\mathbf{c})\|^2 \quad (2)$$

can be interpreted as tree search. The search tree and the relevant notations are drafted in Fig. 2 for a 2 QAM and $N_T = 4$ transmit antennas. The root node of the tree is defined as layer $i = N_T$. In each of the layers i , $i = (N_T - 1) \dots 0$, 2^L possible transmission symbols \hat{x}_i are existing for one parent node, represented by the nodes of the tree and connected to the parent via branches. Layer $i = N_T - 1$, corresponding to the lowest row of (2) and a path from $i = N_T - 1$

to $i = 0$ represents a complete set of sent symbols $\hat{\mathbf{x}}$, mapped to the leaves of the tree. Resulting from this, λ_0 can be recursively calculated by the layered branch metric

$$\lambda_i = \underbrace{\lambda_{i+1}}_{\substack{\text{metric from} \\ \text{already} \\ \text{estimated} \\ \text{symbols}}} + \underbrace{\left| \underbrace{y_i'' - r_{ii}\hat{x}_i}_{\substack{\text{interference} \\ \text{reduced} \\ \text{symbol}}} \right|^2}_{\substack{\text{interference} \\ \text{reduced} \\ \text{symbol}}} - \underbrace{\frac{N_0}{2} \sum_{j=1}^L c_{i,j} L_a(c_{i,j})}_{\substack{\text{a-priori} \\ \text{information}}}, \quad (3)$$

$$y_i'' = y_i' - \sum_{j=i+1}^{N_T-1} r_{ij}\hat{x}_j,$$

out of the squared distance between the nodes and an interference reduced symbol, the a-priori information and the metric of the corresponding parent node, whereas the root metric is defined to $\lambda_{N_T} = 0$.

Aim of the detection algorithm is the selection and analysis of nodes relevant for (1). Complexity reduction of the search is achieved by eliminating unfavorable paths from the search. Within this paper we focus on variations of sphere detection algorithm [5] [13], a so called depth-first search, which can be briefly summarized as follows: The sphere search is started in layer $i = N_T$ with an unlimited sphere. At each level i the algorithm analyzes the children nodes with their λ_i , see (3), and selects one of the nodes within the search sphere, which wasn't extended so far. The selected node is extended by analyzing its children nodes in the layer $i - 1$. Whenever a leaf is reached ($i = 0$), the search radius is adapted. Whenever all children nodes from a parent node within the sphere were extended, the search level is increased and the search continued. As soon the root is reached again, the search is completed. The operations of a sphere detection are drafted exemplarily in Fig. 2 with an adaptation of the radius in operations ④, ⑨ and a ML symbol found in operation ⑨. Paths excluded from the search are illustrated with thin branches, paths examined with thick ones.

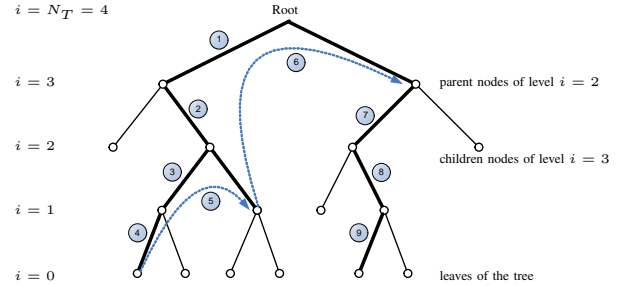


Fig. 2. Example sphere based tree search

Assuming an implementation with one extended node per cycle, as described in [11], the decisive factor for the throughput is the number of extended nodes. The resulting hardware complexity heavily depends on the actual implementation, the included complexity reduction techniques and optimizations. Throughout this paper, hence the number of extended nodes is chosen as complexity measurement.

C. List Sphere Detector

Computing the L-values in (1) requires the determination of a detection hypothesis and all counter-hypotheses as depicted in III-A. Since an explicit search for all needed minimums leads to an insufficient high complexity [14], the key to an efficient detection is the adequate usage of information gathered during the search in combination with an adapted search strategy. One possibility is the

application of the so called List Sphere Detector (LSD) [5] [15]. Instead of searching all possible minima, the algorithm searches a set of most likely leaves. Whenever a node is examined during the search, the corresponding metric λ_0 is compared with the leaves already visited and the $K = \#\mathcal{K}$ best leaves are kept for the calculation of the L-values in a candidate list $\mathcal{K} := \{\lambda_0(\mathbf{c}_1), \lambda_0(\mathbf{c}_2), \dots, \lambda_0(\mathbf{c}_K)\}$:

$$\mathbf{c} \mapsto \arg \max_{\mathbf{c}_t | \mathbf{c}_t \in \mathcal{K}} \{\lambda_{\min, t}\}, \quad \lambda_0 \mapsto \max_{\mathbf{c}_t | \mathbf{c}_t \in \mathcal{K}} \{\lambda_{\min, t}\}. \quad (4)$$

Therefore this list is typically sorted and initialized with elements $\lambda_0(\mathbf{c}) = \infty$. During the search, the search radius is always set to the maximum of this list:

$$R = \max_{\mathbf{c}_t | \mathbf{c}_t \in \mathcal{K}} \{\lambda_{\min, t}\}.$$

The Radius is initialized with $R = \infty$ and scaled down during the search such that only K nodes remain in the sphere. Subsequent to the search, the calculation of the L-values in (1) is carried out over the hypothesis and the counter-hypotheses stored in \mathcal{K} , whereas the min-searches are reduced to:

$$\min_{\mathbf{c} | c_{m,l} = \pm 1} \{\lambda_0\} \mapsto \min_{\mathbf{c} \in \mathcal{K}, c_{m,l} = \pm 1} \{\lambda_0\}, \quad \mathbf{c} \in \mathcal{K} \subseteq \mathcal{C}.$$

Since the counter hypotheses for the L-value calculation are generated only from a subset of the leaves examined during the tree search, suitable counter hypotheses were possibly not found for every bit. Consequently, the L-values have to be clipped: $|L_e| \leq L_{\max}$ and definition a convenient clipping level L_{\max} is crucial for good performance [6]. Hence, for our simulations the clipping level is chosen such that the average mutual information at the detector output is maximized [16].

A regularized data flow of such a LSD is given in Figure 3.

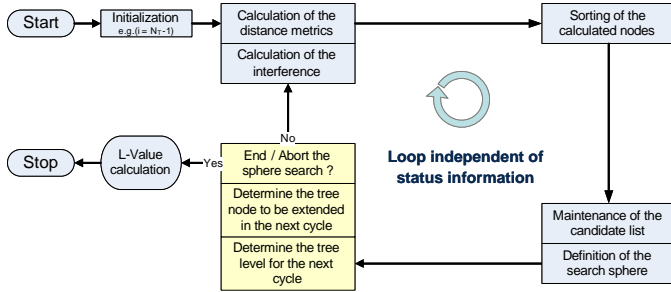


Fig. 3. Regularized data flow of a List Sphere Detector

D. Single Tree Search

Another solution for a tree search adapted to the L-value calculation is the Single Tree Search (STS) introduced by [14] and carried out in [7]. As described in detail in [7], the STS searches for all bit values, $c_{m,l} = \pm 1$ the leaves with the lowest metric $\min_{\mathbf{c} \in \mathcal{C}, c_{m,l} = \pm 1} \{\lambda_0\}$ within a maximum sphere. Involved by the bit-specific search, the search sphere varies depending on the examined nodes and has to be established for each node separately. The sphere radius of a node representing a partially symbol vector \mathbf{c}_i (with defined bits $c_{m,l}$, $m \geq i$) is dependant on the best metrics $\lambda_{\min, c_{m,l}}$ already found for the bits $c_{m,l}$ and the bit values contained in \mathbf{c}_i :

$$R(\mathbf{c}_i) = \max \left\{ \begin{array}{ll} \lambda_{\min, c_{m,l}} \quad \forall c_{m,l} \neq c_{m,l}^{\text{ML}} \in \mathbf{c}_i, & m \geq i, \\ \lambda_{\min, c_{m,l}} \quad \forall c_{m,l}, & m < i. \end{array} \right\}.$$

As for the LSD, the L-values might be clipped for complexity reduction. For a non iterative detection the radius of the sphere is limited by the maximum aimed L-value L_{\max} and the metric of the current hypothesis $\lambda_0(\mathbf{c}^{\text{ML}})$:

$$R_{\text{Clipped}}(\mathbf{c}_i) = \min \left\{ R(\mathbf{c}_i), \lambda_0(\mathbf{c}^{\text{ML}}) + N_0 L_{\max} \right\}.$$

Besides the additional complexity needed for the sphere determination, the STS incurs an integrated search for each bit and hence still a high complexity. The complexity of the L-value calculation is however reduced since all $L(c_{m,l} | \mathbf{y})$ in (1), with $|L(c_{m,l} | \mathbf{y})| \leq L_{\max}$, can be easily calculated out of the determined $\lambda_{\min, c_{m,l}}$.

E. Methods for Complexity Reduction

More efficient application of the described algorithms is permitted by inclusion of methods that allow a further reduction of the average number of nodes extended by the search algorithms.

Crucial for a reduction of the number of examined nodes is the size of the search sphere. Since the search sphere is reduced by means of determined leaves, it is essential to find proper leaves as soon as possible. A solution to enable this is the so call Schnorr-Euchner enumeration strategy [8]. The analyzed child nodes are extended in ascending order of their distance metric (3). As result the probability of finding nodes with a small λ_0 early is increased and the average number of extended nodes hence reduced.

A second method to reduce the complexity is provided by layer ordering. Choosing a wrong node in a high layer causes unnecessary analysis and extensions of child nodes. Wrong selections in lower layers hence have a less negative effect on the complexity. Detection of the most reliable symbols first minimizes the probability of wrong decision in a high layer. A common approach to apply the layer ordering is the inclusion of sorted QRD (SQRD) as proposed in [9]. Since both techniques provide a lossless complexity reduction, we will apply them in the subsequent without any further discussion.

Additionally to these methods, MMSE preprocessing [17] might be used to further reduce the complexity. This can be done by utilizing an extended channel matrix for the (S)QR-decomposition:

$$\bar{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \sigma \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{Q} & \mathbf{Q}_3 \\ \mathbf{Q}_2 & \mathbf{Q}_4 \end{bmatrix} \begin{bmatrix} \mathbf{R} \\ \mathbf{0} \end{bmatrix}, \quad \bar{\mathbf{y}} = \begin{bmatrix} \mathbf{y} \\ \mathbf{0} \end{bmatrix}.$$

Following this procedure, the MMSE preprocessing incurs a data dependant bias $\sigma^2 \|\hat{\mathbf{x}}\|^2$ on the metrics:

$$\|\bar{\mathbf{y}} - \bar{\mathbf{H}}\hat{\mathbf{x}}\|^2 = \|\mathbf{y} - \mathbf{H}\hat{\mathbf{x}}\|^2 + \sigma^2 \|\hat{\mathbf{x}}\|^2.$$

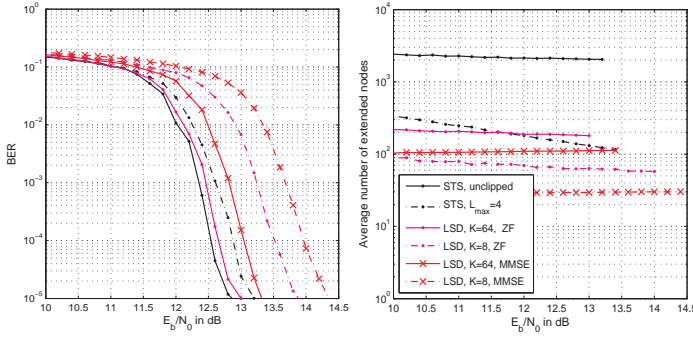
This bias remains on the metrics λ_0 after the detection and jams the L-value calculation. Hence, it should be removed for an accurate detection by subtraction to avoid performance degradations [10]:

$$\lambda_{0, \text{ub}}(\mathbf{y}, \mathbf{c}, \mathbf{L}_a) = \lambda_0(\mathbf{y}, \mathbf{c}, \mathbf{L}_a) - \sigma^2 \|\hat{\mathbf{x}}(\mathbf{c})\|^2. \quad (5)$$

F. Discussion and Reference Results

The complexity and the BER performance of the resulting algorithms are shown in Figure 4. As performance reference the maximum achievable maxlogAPP detection is selected, provided by the unclipped STS. A comparison to linear detection, SIC or hard-output tree searches is passed, since their achievable performance is incommensurable bad.

While the unclipped STS provides the best BER-performance, since all minima in (1) were determined, its complexity is unusable high. By introducing the described clipping the complexity is significantly reduced at a reasonable BER performance. But with approx 120 node extensions at a BER of 10^{-5} it is still to high for an efficient detector implementation.



(a) Influence on the BER performance (b) Influence on complexity
Fig. 4. Comparison of the complexity reduced algorithms for a 4×4 MIMO, 64QAM and SQRD.

The Zero Forcing (ZF) LSD obtains for list sizes of 64 a loss of about 0.2 dB compared to the BER performance of the STS at complexity one and a half times the clipped STS. By introducing the unbiased MMSE the performance is decreased, so the algorithm is outperformed by the clipped STS at comparable complexity. Regarding smaller list sizes the effect of MMSE preprocessing is even worse. On the one hand it decreases the complexity clearly, on the other hand it degrades the BER performance with decreasing list sizes. This is caused by the fact that unbiased the metrics with (5) changes the minima in (1):

$$\min_{\mathbf{c} \in \mathcal{K}, c_{m,l} = \pm 1} \{ \lambda_0 \} - \sigma^2 \|\hat{\mathbf{x}}(\mathbf{c})\|^2 \geq \min_{\mathbf{c} \in \mathcal{C}, c_{m,l} = \pm 1} \{ \lambda_0 - \sigma^2 \|\hat{\mathbf{x}}(\mathbf{c})\|^2 \},$$

which also incurs a possible switch of the hypothesis $\hat{\mathbf{x}}(\mathbf{c}^{\text{ML}})$ and the counter-hypotheses. As consequence, the introduction of MMSE preprocessing to a LSD with a small list size or to a STS (one element stored per bit) for further complexity reduction is not applicable. Easing the caused loss in the BER performance is only enabled by large list sizes, since the candidate list \mathcal{K} for large K contains several candidates and hence proper metrics might be determined

$$\min_{\mathbf{c} \in \mathcal{K}, c_{m,l} = \pm 1} \{ \lambda_0 - \sigma^2 \|\hat{\mathbf{x}}(\mathbf{c})\|^2 \} \gtrsim \min_{\mathbf{c} \in \mathcal{C}, c_{m,l} = \pm 1} \{ \lambda_0 - \sigma^2 \|\hat{\mathbf{x}}(\mathbf{c})\|^2 \}.$$

In Summary, the STS is accomplishing a complete search for all leaves within a given maximum clipping value. While this ensures the desired detection performance, it also causes a high complexity. The LSD searches a subset of most likely candidates. While the candidate list limits the detection complexity, it also limits the amount of candidates. Consequently for small list sizes only little candidates are retained. Most leaves determined during the detection are hence not usable for the L-value calculation. MMSE extension of the detection enhances this problem as discussed above. Resulting from this, currently known concepts for a SISO STS as well as for a SISO LSD are either not performant enough or too hardware demanding to be applicable for efficient MIMO detection.

IV. TUPEL SEARCH

Good BER performance requires an accurate determination of the L-values. Hence, accurate candidates are essential. As described in section III, the LSD is only able to regulate the quality of the determined candidates over the size of the candidate list. Leaves determined during the search containing a λ_0 greater than the search radius are not usable for the L-value calculation, since they were dropped during the search. Possible counter-hypotheses with metrics close to minimum $\min_{\hat{\mathbf{x}}(\mathbf{c}) \in \mathcal{C}, c_{m,l} \neq c_{m,l}^{\text{ML}}} \{ \lambda_0 \}$ will also provide valuable contribution to the L-value calculation if the minima weren't

found. Same for counter-hypotheses, which were found but dropped because of a too high metric. As consequence the LSD drops useful information. In order to ease this effect and to enable a more precise adjustment, we propose a modified sphere search.

For the Tuple Search we propose the usage of a search tuple for the sphere determination and the retention of leaf candidates in a candidate list, for the L-value determination. As the LSD described in section III-C, our proposed search identifies a set of most promising candidates (with respect to λ_0). Unlike the conventional LSD, the search sphere is determined over a T -tuple $\mathcal{T} := \{ \lambda_0(\mathbf{c}_1), \lambda_0(\mathbf{c}_2), \dots, \lambda_0(\mathbf{c}_T) \}$ of the identified candidates with lowest λ_0 , independent of the candidate List \mathcal{K} . The sphere radius is defined to the maximum tuple metric:

$$R = \max_{\mathbf{c}_t \in \mathcal{T}} \{ \lambda_{\min,t} \}.$$

As for the conventional LSD, the list is initialized with $\lambda_{0,t} = \infty$. The sphere search finds all leaves within the current sphere. Each leaf within the current sphere, identified by the search, represents a new element of the tuple, which replaces the worst element of the tuple:

$$\mathbf{c} \mapsto \arg \max_{\mathbf{c}_t \in \mathcal{T}} \{ \lambda_{\min,t} \}, \quad \lambda_0(\mathbf{c}) \mapsto \max_{\mathbf{c}_t \in \mathcal{T}} \{ \lambda_{\min,t} \}.$$

Resulting from this, the tuple always contains the T leaves with lowest λ_0 identified so far and the search finds definitely the T most promising candidates of the tree. The candidate list contains, as the LSD, the most promising candidates for the L-value calculation. This might include elements of \mathcal{T} as well as candidates with $\lambda_0(\mathbf{c}) > \max_{\mathbf{c}_t \in \mathcal{T}} \{ \lambda_{\min,t} \}$.

Note that such a tuple usually will be practicable small and can be easily created as sorted list. Hence, it can be implemented as portion of the candidate list $\mathcal{T} \subseteq \mathcal{K}$ causing no additional hardware complexity as depicted in Figure 5.

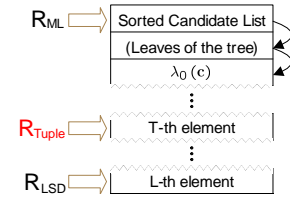


Fig. 5. Sample candidate list including radius determination

V. MATCHED CANDIDATE DETERMINATION

One main problem still remaining is caused by the MMSE preprocessing: Since the magnitude of unbiased metrics differs from biased metrics, the known strategies cannot provide the best possible candidate list. Hence, these methods are still suboptimal for MMSE detection. The following subsections will propose two approaches to ease this effect.

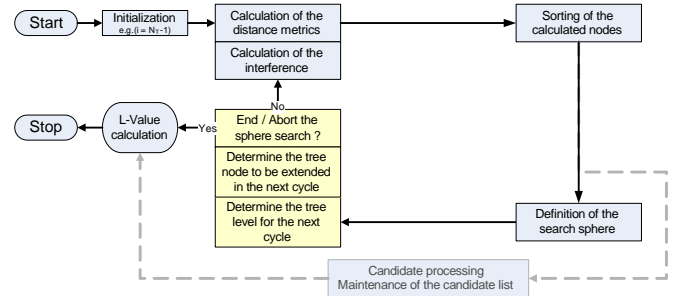


Fig. 6. Data flow of a the proposed Sphere Detector

A. Separation of Sphere and Candidate Determination

In order to further increase the possible detection quality and to optimize the adaptivity of the candidate quality, the sphere determination has to be separated from the candidates for the L-value calculation. For this e.g. a second list could be introduced: One optimized for the sphere determination and one optimized for the L-value calculation. For the sphere determination we propose usage of the described TS. Since only the metrics are relevant to determine the sphere, a storage of the corresponding symbols is unnecessary. Unlike the existing sphere detectors, we propose for the candidate list a storage of candidates respectively status information adapted to the current detection technique. For the proposed MMSE extended sphere detection this would cover the bias reduction. With respect to (5), updating the candidate list (4) has to be adapted as follows:

$$\mathbf{c} \mapsto \arg \max_{\mathbf{c}_t | \mathbf{c}_t \in \mathcal{K}} \{f_{\min}(\mathbf{c}_t)\}, \quad f_{\min}(\mathbf{c}) \mapsto \max_{\mathbf{c}_t | \mathbf{c}_t \in \mathcal{K}} \{f_{\min}(\mathbf{c}_t)\},$$

with $f(\mathbf{c}) \triangleq f(\lambda_0(\mathbf{c}), \sigma^2, \mathbf{c}) = \lambda_0 - \sigma^2 \|\mathbf{x}\|^2$ for the described bias-reduced MMSE detection. Resulting from this, the candidate list contains the most promising nodes examined during the search, with respect to $f(\mathbf{c})$. Note that beside the metric magnitude also the resulting hypothesis $\arg \min_{\mathbf{c}_t | \mathbf{c}_t \in \mathcal{K}} \{f_{\min}(\mathbf{c}_t)\}$ and also the counter hypotheses $\arg \min_{\mathbf{c}_t | \mathbf{c}_t \in \mathcal{K}, c_{m,l} \neq c_{m,l}^{\text{ML}}} \{f_{\min}(\mathbf{c}_t)\}$ may change due to the adapted search. Processing the adapted elements and maintaining the candidate list is independent of the sphere determination and hence can be performed in parallel as shown in Figure 6.

B. Bit-specific Candidate Determination

Although introducing a separate list enables the usage of candidates outside the search defining tuple, it doesn't solve the problem of dropping candidates. This is caused by the limited size of the underlying list. In order to solve this problem, we propose a bit-specific storage of status information for the L-value calculation. In case of the proposed bias reduced MMSE detection, the lowest $f(\mathbf{c})$ have to be stored for the hypothesis and for the corresponding counter hypotheses bits, as illustrated in Figure 7.

Whenever a leaf is reached, the current \mathbf{c} , $f(\mathbf{c})$ are compared with the stored values $\mathbf{c}^{\text{ML}}, f(\mathbf{c}^{\text{ML}})$

$$\begin{aligned} \mathbf{c}^{\text{ML}} &\leftarrow \arg \min \{f(\mathbf{c}), f(\mathbf{c}^{\text{ML}})\}, \\ f(\mathbf{c}^{\text{ML}}) &\leftarrow \min \{f(\mathbf{c}), f(\mathbf{c}^{\text{ML}})\}, \end{aligned}$$

and subsequent to this with the corresponding $f(c_i)$

$$f(c_i) \leftarrow \min \{f(\mathbf{c}), f(c_i)\}, \quad \forall c_i \in \mathbf{c} | c_i \neq c_i^{\text{ML}},$$

whereas the lowest values are stored. As for the separated list, the storage is independent to the determination of the sphere radius and hence can be processed optimized for the L-value calculation in parallel to the sphere algorithm. From there, the candidates as well as their status information don't have to be identical to the one stored for the sphere determination. For each possible bit the best values found will be stored. Hence, dropping useful status information is avoided. Due to the replacement of the list by bit-specific storage, costly maintenance of the list is replaced by simple comparisons. Additionally the TS only requires the storage of λ_0 in \mathcal{T} . On behalf of the L-value calculation the complexity is also reduced by the bit-specific storage.

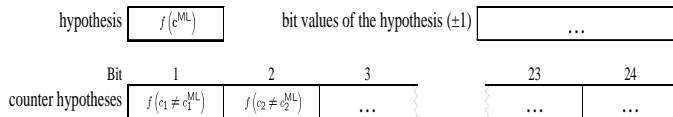


Fig. 7. Bit-specific storage for a 64 QAM and $N_T = 4$

VI. RESULTS

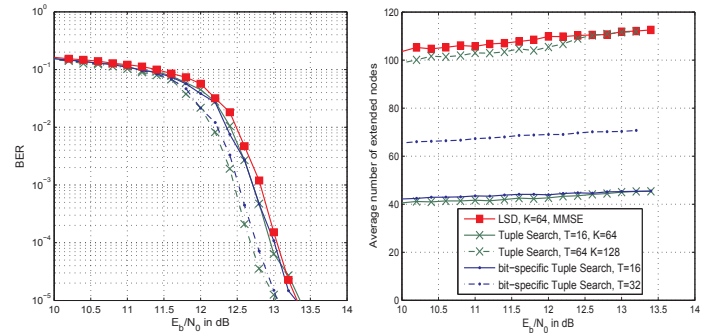
The results of applying the Tuple Search (TS) algorithm and the matched candidate determination are shown in Figure 8. An overview of the complexity and performance resulting from the proposed techniques is given in Figure 9 for different search tuple sizes respectively clipping values and a BER of 10^{-5} . The corresponding clipping values, used for the simulations, are given in Table I. For our simulations in Figure 8, we choose the LSD algorithm with a candidate list size $K = 64$ as performance reference, see Figure 4. With unbiased MMSE processing, the algorithm provides a performance close to maxlogAPP (max. 0.5 dB loss) at a complexity of approx. 120 node extensions. The candidate list size ensures comparability of the algorithms, since MMSE processing would increase performance degradations for smaller list sizes (see subsection III-F).

The inclusion of search tuples in the LSD enables a reduction of the lead node set, crucial for the sphere determination. Smaller search tuples result in an earlier pruned sphere, comparable to a LSD with a smaller candidate list. Due to a candidate list size $K \gg T$, the TS retains the most relevant candidates, with $\lambda_0 > R$, for the L-value calculation. Exemplarily, it is feasible to reduce the tuple size of a LSD with $K = 64$ to a TS with $T = 16$, without significant loss of BER performance but a complexity reduction to almost a third of the conventional LSD.

Besides this obvious reduction in complexity, the TS enables an adjustment of complexity and BER performance for a given hardware implementation to the systems requirements, by adaptation of the tuple size. Within the limits of the underlying LSD, this allows to adapt the performance and complexity reaching from close to hard output (searching only the ML-point) to a full LSD: $1 \leq T \leq K$. Additional limitation of the amount of sequential calculations furthermore enables to reduce the complexity to 4 sequential operations, leading to a SIC detection.

Separated candidate processing enables a MMSE detection without performance degradation. Also for smaller search tuples ($K \leq 64$), the algorithm detects reliable candidates with respect to the unbiased metrics and ensures hereby a high candidate quality. Consequently, the matched candidate processing enables a performance improvement for MMSE detection with given search complexity respectively search tuple size. While the performance improvement compared to a LSD with $K = 64$ (Figure 4) is already about 0.3 dB it increases to over 1 dB for smaller list sizes, see Figure 9.

Bit-specific candidate storage further improves the performance of the detection, compared to TS with medium K shown in Figure 8 and 9. Whenever the candidate list is too small to retain the determined candidates useful data is dropped. Consequently for accurate detection or small K the detection complexity can be further



(a) Influence on the BER performance

(b) Influence on complexity

Fig. 8. Comparison of the proposed search strategies with the conventional for a 4×4 MIMO, 64QAM, SQRD and MMSE preprocessing.

