A Layer-Adaptive M algorithm for Multiple-Input Multiple-Output Channel Detection

David L. Milliner and John R. Barry

School of ECE, Georgia Institute of Technology
Atlanta, GA 30332-0250 USA, {dlm, barry}@ece.gatech.edu

***Abstract* — We present a new detection algorithm for multiple-input multiple-output channels called the CLAM algorithm. The CLAM algorithm is similar to the classical 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 to significantly outperform the M algorithm. For example, on a 4-input 4-output Rayleigh-fading channel with 64-QAM inputs, the CLAM algorithm outperforms the M algorithm by 2 dB at a BER of 10^{-3}, falling only 0.6 dB short of the joint maximum-likelihood detector, while simultaneously reducing the average search complexity by 6%.**

I. Introduction

Multiple-input multiple-output (MIMO) symbol detection is a well-studied problem with a myriad of potential solutions. The optimal joint maximum-likelihood (JML) detector can be implemented using the sphere detector [1], but its complexity is high. Of the many suboptimal MIMO detection algorithms that have been developed [2]-[7], the M detection algorithm is of particular interest [3][4]. Often referred to as the QR-decomposition M (QRD-M) algorithm, it achieves near-JML performance at a fraction of the complexity.

The M algorithm is a breadth-first strategy for searching the detection tree. It rejects all but M nodes at a given level of the tree before advancing to the next level [3]-[5]. Specifically, the b "best" children (having the lowest branch metrics) are extended from each of the M retained nodes, and of the bM contenders that result, only the M best are retained. The performance of the M algorithm approaches that of the JML detector when M is large, but falls off significantly as M is decreased [6].

In this paper we present a new algorithm called the *channel-based level-adaptive M (CLAM)* algorithm. Similar to the M algorithm, the CLAM algorithm rejects all but the best nodes at each stage of the detection tree. Unlike the M algorithm, however, the CLAM algorithm varies the b parameter (representing the number of children extended from each retained node) from one stage to the next. This simple enhancement of the M algorithm, together with a careful choice of the b parameters, results in both reduced average search complexity and significantly improved performance.

The adaptive complexity QRD-M algorithm (AC-QRD-M) of [8] is an alternative enhancement of the M algorithm that differs from the CLAM algorithm in two substantial ways. First, the AC-QRD-M algorithm varies the number of retained nodes (the M parameter) rather than the number of children extended from each retained node (the b parameter). Second, the AC-QRD-M algorithm is adaptive across orthogonal frequency-division multiplexing (OFDM) tones, but uses the same M and b for all stages of a given tree. The CLAM and AC-QRD-M algorithm are thus complementary. For example, in an OFDM system the AC-QRD-M algorithm could be used to select the M parameters on a tone-by-tone basis, while the CLAM algorithm could be used to select the b parameters within each tree.

The remainder of this paper is organized as follows. Section II presents the channel model and relates the JML detector to a tree search. Section III presents both the M algorithm and the proposed CLAM algorithm. Section IV provides a complexity analysis of the CLAM algorithm and Section V presents simulation results. Section VI concludes the paper.

II. Channel Model and Tree Search

We consider an N-input L-output channel with complex inputs $\boldsymbol{a} = [a_1, \ldots a_N]^T$ and outputs $\boldsymbol{r} = [r_1, \ldots r_L]^T$, where:

$$\boldsymbol{r} = \mathbf{H}\boldsymbol{a} + \boldsymbol{w}, \tag{1}$$

where $\mathbf{H}$ is an $L \times N$ channel matrix, where $\boldsymbol{w}$ is additive white Gaussian noise satisfying $E[\boldsymbol{w}\boldsymbol{w}^*] = N_0\mathbf{I}$, and where $\boldsymbol{w}^*$ denotes the conjugate transpose of $\boldsymbol{w}$. The entries of $\mathbf{H}$ are assumed i.i.d. complex Gaussian random variables with zero-mean and unit variance. We assume that the receiver knows the channel perfectly, and that $L \geq N$. The channel inputs are assumed to be chosen uniformly and independently

This research was supported in part by Texas Instruments, and by National Science Foundation grants 0431031 and 0121565.

from a quadrature-amplitude modulation (QAM) alphabet $\mathcal{A} = \{\pm\alpha, \pm 3\alpha, \ldots \pm(\sqrt{q}-1)\alpha\} + \sqrt{-1}\,\{\pm\alpha, \pm 3\alpha, \ldots \pm(\sqrt{q}-1)\alpha\}$, where $q = |\mathcal{A}|$. The constant α is chosen so that the alphabet has average energy E, i.e. $\alpha = \sqrt{1.5E/(q-1)}$.

The JML detector chooses the decision $\boldsymbol{a} \in \mathcal{A}^N$ that minimizes the following cost function:

$$J(\boldsymbol{a}) = \|\boldsymbol{r} - \mathbf{H}\boldsymbol{a}\|^2 \tag{2}$$

$$= \|\boldsymbol{y} - \mathbf{R}\boldsymbol{a}\|^2 \tag{3}$$

$$= \sum_{i=1}^{N} |y_i - \sum_{j=1}^{i} R_{ij} a_j|^2 , \tag{4}$$

where $\mathbf{H} = \mathbf{QR}$ is the QR decomposition of the channel matrix $\mathbf{H}$, where $\mathbf{R}$ is an $N \times N$ lower triangular matrix, and where $\boldsymbol{y} = \mathbf{Q}^*\boldsymbol{r}$.

We can evaluate the JML cost with the aid of a detection tree. The root of the tree is connected to q child nodes, one for each possible value for a_1, and each of these is connected to q child nodes, one for each possible a_2, and so on. Thus each possible $\boldsymbol{a}$ is associated with a unique leaf node [9]. In terms of this tree, we may interpret (4) as the sum of N branch metrics, one for each branch in a path from the root to a leaf node, where the metric for a branch in the i-th stage with path history $\{a_1, a_2, \ldots a_i\}$ is defined as

$$|y_i - \sum_{j=1}^{i} R_{ij} a_j|^2. \tag{5}$$

In terms of this tree, the JML detection problem becomes the problem of finding the leaf node with the lowest cost. An efficient strategy for searching the tree is the sphere detector [1]. The *MMSE* sphere detector [10] approximates the JML detector with lower complexity by basing the branch metrics of (5) on a QR decomposition of the following *extended* channel matrix [7]:

$$\underline{\mathbf{H}} = \begin{bmatrix} \mathbf{H} \\ \sqrt{N_0}\mathbf{I}_{N\text{x}N} \end{bmatrix}. \tag{6}$$

In particular, the MMSE branch metrics are computed using (5) with $\underline{\mathbf{H}} = \mathbf{QR}$ and $\boldsymbol{y} = \mathbf{Q}^*[\,\boldsymbol{r}^T,\ \mathbf{0}_{1\times N}\,]^T$.

Permuting the columns of $\mathbf{H}$ can help speed the search, so in practice it is common to perform a QR decomposition on $\mathbf{HP}$ or $\underline{\mathbf{H}}\mathbf{P}$ instead, where $\mathbf{P}$ is a permutation matrix [7][11].

III. The M and CLAM Algorithms

A. The M Algorithm

The M algorithm is a low-complexity but suboptimal strategy for searching the tree. Specifically, the M algorithm is a breadth-first search that views all the branches it will ever consider for a given stage of the detection tree and then rejects all but the $M \le |\mathcal{A}|$ best nodes before continuing on to the next stage [5]. Because the M algorithm is a breadth-first search algorithm, all path histories have the same length. As a result it has fixed computational load and/or memory, making it a good candidate for practical applications [12].

M Algorithm

Inputs: $\boldsymbol{r}$, $\mathbf{H}$, M, b. Output: $\hat{\boldsymbol{a}}$

(A-1) QR decomposition of channel matrix $\mathbf{HP} = \mathbf{QR}$
(A-2) $\boldsymbol{y} = \mathbf{Q}^*\boldsymbol{r}$
(A-3) $S_0 =$ root node with branch metric of zero.
(A-4) $S_1 = \{M \text{ best children of } S_0\}$
(A-5) for $i = 2$ to N
(A-6) `contenders` = $\bigcup_{\text{node} \in S_{i-1}} \{b \text{ best children of } \texttt{node}\}$
(A-7) $S_i = M$ best `contenders`
(A-8) end
(A-9) $\mathbf{P}\hat{\boldsymbol{a}} =$ best of S_N

Fig. 1. The M algorithm [4].

A pseudocode description of the M algorithm is given in Fig. 1. Let S_i denote the retained (*survivor*) nodes at level i of the tree, with S_0 initialized to the root node. The M algorithm begins at the root node S_0 and calculates branch metrics for all possible values of $a_1 \in \mathcal{A}$. The M contender nodes with the lowest metrics are then retained, yielding S_1.

The algorithm continues by extending the b best children of each of the M nodes retained from the previous stage in the detection tree. Of the bM contenders that result, the M best are retained; the remaining $(b-1)M$ contenders are rejected [5]. Once the M best nodes have been selected the algorithm moves to the next detection stage. This process continues through the N-th detection stage.

The process of rejecting all but the M best contenders (A-7) may be performed using a sorting procedure on the branch metrics subject only to a partial ordering constraint, rather than the more restrictive constraint required of a total ordering. For this reason an appealing implementation of the sorting procedure is the heapsort algorithm [13].

B. The CLAM Algorithm

The M algorithm uses the same b parameter at each stage of the tree after the first. If we define b_i as the b parameter used at stage i, then the vector of parameters used by the M algorithm is

$$\boldsymbol{b} = [M, b, \ldots b]. \tag{7}$$

CLAM Algorithm

Inputs: $\boldsymbol{r}$, $\mathbf{H}$, M, b. Output: $\hat{\boldsymbol{a}}$

(B-1) QR decomposition of channel matrix $\mathbf{HP} = \mathbf{QR}$
(B-2) $\boldsymbol{y} = \mathbf{Q}^*\boldsymbol{r}$
(B-3) $\boldsymbol{b} = \texttt{generate\_b}(b, \mathbf{R})$
(B-4) $S_0 =$ root node with branch metric of zero.
(B-5) $S_1 = \{b_1$ best children of $S_0\}$
(B-6) for $i = 2$ to N
(B-7) $\quad \texttt{contenders} = \bigcup_{\texttt{node} \in S_{i-1}} \{b_i$ best children of $\texttt{node}\}$
(B-8) $\quad S_i = M$ best $\texttt{contenders}$
(B-9) end
(B-10) $\mathbf{P}\hat{\boldsymbol{a}} =$ best of S_N

Fig. 2. The CLAM algorithm.

In contrast, the CLAM algorithm varies the b parameter from one stage to the next, so that the vector of parameters takes the most general form:

$$\boldsymbol{b} = [b_1, b_2, \ldots b_N]. \tag{8}$$

Fig. 2 summarizes the CLAM algorithm. Note that the only differences between the M algorithm and the CLAM algorithm are line (B-3), where the b parameters are generated, line (B-5) where b_1 nodes are retained at the end of the first detection layer, and the subscript on b_i in line (B-7). Note further that the CLAM algorithm reduces to the M algorithm when $\boldsymbol{b}$ is given by (7). Furthermore, both the CLAM algorithm and the M algorithm reduce to the decision-feedback detector in the trivial case when $M = 1$ and $b = 1$.

There are potentially many ways to generate an adaptive $\boldsymbol{b}$ yielding performance superior to that of the M algorithm. We propose the strategy shown in Fig. 3, which shows a function `generate_b` whose inputs are the scalar b and the effective channel $\mathbf{R}$. The first operation performed by the `generate_b` function is to create a vector whose elements are inversely proportional to the SNR for their respective detection layers (C-1), where the SNR of the i-th detection layer is $SNR_i = ER_{ii}^2 / N_0$. This ensures that b_i is large when SNR_i is small, and vice versa, thereby allocating more of the complexity budget to the detection stages most likely to be in error.

The remainder of the `generate_b` function is used to ensure that $\{b_i\}$ satisfy the following constraints:

$$b_i \geq 1 \quad \forall i \tag{9}$$

$$b_i \leq q \quad \forall i \tag{10}$$

`generate_b` Function

Inputs: b, $\mathbf{R}$ Output: $\boldsymbol{b}$

(C-1) $\boldsymbol{b} = [R_{11}^{-2}, \ \ldots, R_{NN}^{-2}]$
(C-2) $\boldsymbol{b} = (Nb/\sum_i b_i)\boldsymbol{b}$
(C-3) $i_{\min} = \text{find}(\boldsymbol{b} < 1)$
(C-4) $i_{\max} = \text{find}(\boldsymbol{b} > q)$
(C-5) $i_{\text{other}} = \{1, \ldots N\} \setminus \{i_{\min} \cup i_{\max}\}$;
(C-6) $\rho = \sum_i b_i$
(C-7) $b_{i_{\min}} = 1$
(C-8) $b_{i_{\max}} = q$
(C-9) $\rho = \rho - \sum_i b_i$
(C-10) $b_{i_{\text{other}}} = \lceil b_{i_{\text{other}}} - \rho/\text{length}(i_{\text{other}}) \rfloor$
(C-11) $d = \sum_i b_i - Nb$
(C-12) while $(d \neq 0)$
(C-13) $\quad i_{\min} = \text{argmin}_i(\{b_i\})$; $i_{\max} = \text{argmax}_i(\{b_i\})$
(C-14) $\quad$ if $(d > 0)$
(C-15) $\quad\quad b_{i_{\max}} = b_{i_{\max}} - 1$; $d = d - 1$
(C-16) $\quad$ end
(C-17) $\quad$ if $(d < 0)$
(C-18) $\quad\quad b_{i_{\min}} = b_{i_{\min}} + 1$; $d = d + 1$
(C-19) $\quad$ end
(C-20) end
(C-21) $i = N$
(C-22) while $(n_c(\boldsymbol{b}) > M + (N-1)bM)$,
(C-23) $\quad b_i = b_i - 1$
(C-24) $\quad$ while $b_i = 1$, $i = i - 1$, end
(C-25) end;

Fig. 3. A function for generating the parameters $\{b_i\}$, where b_i is the number of children extended from each of the M retained nodes at the i-th stage of the tree.

$$\boldsymbol{b} \in \mathbb{Z}^N \tag{11}$$

$$\frac{1}{N}\sum_{i=1}^{N} b_i \leq b. \tag{12}$$

Specifically, line (C-2) normalizes the metric from (C-1). Lines (C-3)-(C-10) are used to satisfy the constraints (9)-(11). Lines (C-11) through (C-20) are used to satisfy constraint (12). After (C-20), the inequality in (12) is an equality. The remaining lines (C-21)-(C-25) decrement the b_i parameters as necessary to ensure that the average complexity of the CLAM algorithm (with $M > 1$) is strictly less than that of the M algorithm; it is described in Section IV, with $n_c(\boldsymbol{b})$ defined in (13).

IV. Complexity Analysis

In this section we quantify and compare the complexities of the CLAM and M algorithms. We separate our analysis of the CLAM algorithm into a preprocessing stage to generate $\boldsymbol{b}$, and a core processing stage for the rest of the computations. Because the complexity of the QR decomposition is common to both algorithms, we will ignore it in our analysis. We assume both algorithms begin by operating on $\boldsymbol{y}$.

The number of operations required by the preprocessing function `generate_b` can be shown to be linear in N. The preprocessing stage of the CLAM algorithm therefore has linear complexity in terms of the number of operations.

A common metric for quantifying the complexity of a tree search algorithm is to count the number of nodes visited [4]. In our case, this is equivalent to counting the number n_c of contender nodes that are considered. The M algorithm considers precisely $n_c = M + (N-1)bM$ contenders, regardless of the channel. The CLAM algorithm, however, bases its $\{b_i\}$ parameters on the particular channel realization, resulting in a search complexity that varies with the channel. It is easy to show that the number of nodes n_c considered by the CLAM algorithm satisfies the following recursive relationship:

$$n_c = \underbrace{b_1}_{f_1} + \underbrace{b_1 b_2}_{f_2} + \underbrace{\min(f_2, M) b_3}_{f_3} + \ldots + \underbrace{\min(f_{N-1}, M) b_N}_{f_N}. \quad (13)$$

This reduces to $n_c = b_1 + b_1 b_2 + M(\sum_{i>2} b_i)$ when $b_1 b_2 \geq M$.

To ensure that the average complexity of the CLAM algorithm is strictly less than that of the M algorithm, we decrement $\{b_i\}$ one at a time in lines (C-21)-(C-25), starting with b_N and decrementing until b_N is equal to one. Then, if necessary, we proceed with the same operation on b_{N-1} down to b_1 until n_c is no greater than $M + (N-1)bM$, the total number of nodes searched by the M algorithm. *Consequently, the average complexity required by the CLAM algorithm (with $M > 1$) to search the detection tree is strictly less than that of the M algorithm.* The mean n_c for the CLAM algorithm (averaged over one million distinct 4×4 Rayleigh channels) is 48.22, for a 6% savings relative to the M algorithm. The probability mass function for n_c, when using the CLAM algorithm and a 4×4 Rayleigh channel with 16-QAM inputs and $M = 4$ and $b = 4$, is approximated in Fig. 4 by a histogram. Note that the CLAM algorithm will always consider at least $n_c = Nb$ contenders, in this case 16. Additionally, because $Nb \leq q$ in this example, the same histogram of Fig. 4 will result when the alphabet is 64-QAM. The dashed line at $n_c = 52$ indicates the fixed tree search complexity of the M algorithm.

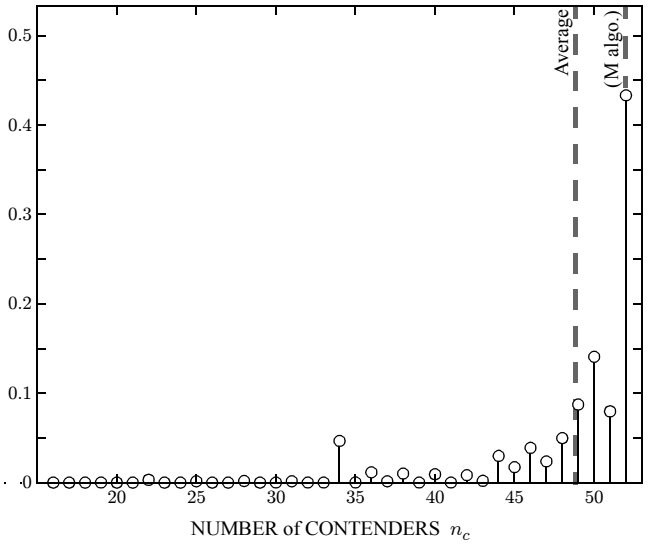

Fig. 4. Histogram for the number of contenders searched by the CLAM algorithm with $M = b = 4$, assuming a 4×4 Rayleigh channel with 16-QAM (or 64-QAM) inputs.

V. Performance Results

We now present the simulated error performance of the CLAM algorithm in an uncoded system. We assume a 4-input 4-output Rayleigh-fading channel with 16-QAM and 64-QAM inputs under Gray mapping. The SNR per bit is $E_b/N_0 = E/(\log_2|\mathcal{A}|N_0)$. The sorted QR decomposition (SQRD) of [7] is used for both the M algorithm and the CLAM algorithm. The SQRD has complexity comparable to that of the unordered QR decomposition, but its ordering results in improved detection performance [7].

In Fig. 5. we compare the bit-error rate performance in Rayleigh fading of the JML detector, the CLAM algorithm, and the M algorithm. The CLAM and M algorithms utilize the same MMSE forward filter $\mathbf{Q}^*$. For the CLAM and M algorithms we set $M = 2$ and $b = 2$. For the case of 16-QAM inputs, the CLAM algorithm outperforms the M algorithm by 2.5 dB at a BER of 10^{-3}, falling only 1.4 dB short of JML performance. The gap between the CLAM and M algorithms jumps to 3.8 dB for the case of 64-QAM inputs, with the CLAM algorithm falling 3.0 dB short of JML performance.

Fig. 6 shows the same curves as Fig. 5 for the same 4×4 Rayleigh channel, only this time the CLAM and M algorithm parameters are changed from $M = b = 2$ to $M = b = 4$. For the case of 16-QAM inputs, the CLAM algorithm outperforms the M algorithm by 0.3 dB at a BER of 10^{-3}, falling only 0.4 dB short of JML performance. The gap

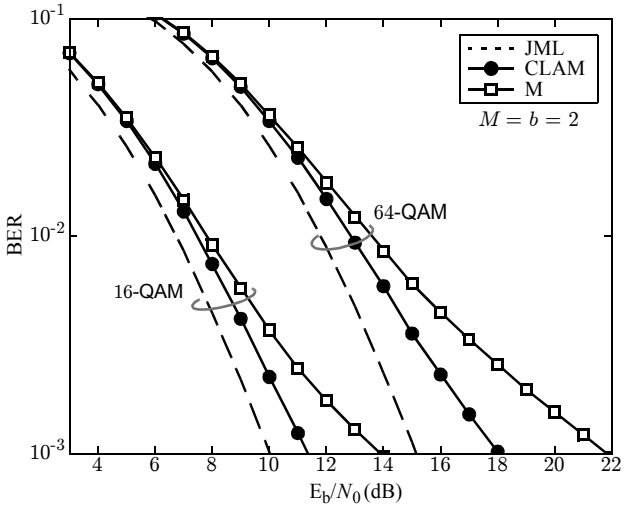

Fig. 5. BER performance with $M = b = 2$ on a 4×4 Rayleigh channel.

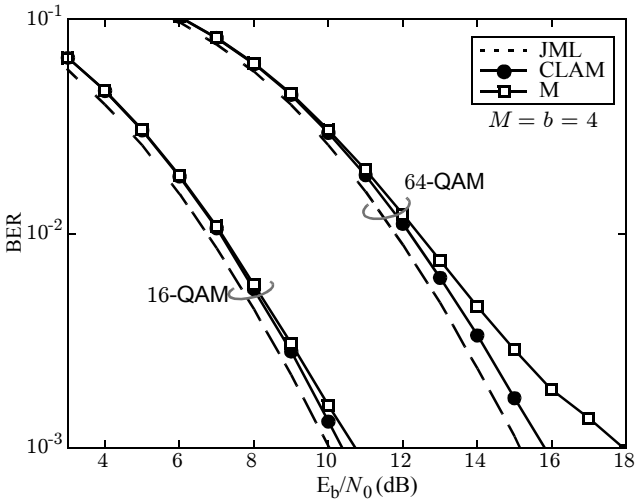

Fig. 6. BER performance with $M = b = 4$ on a 4×4 Rayleigh channel.

between the CLAM and M algorithms jumps to 2.0 dB for the case of 64-QAM inputs, with the CLAM algorithm falling only 0.6 dB short of JML performance. Furthermore, as described in the previous section, these gains come with a 6% *reduction* in average complexity.

VI. Conclusions

This paper presented a new algorithm for MIMO detection called the CLAM algorithm. The CLAM algorithm enhances the M algorithm by varying on a stage-by-stage basis the number of children extended from retained nodes in the detection tree, according to the receiver's knowledge of the channel matrix. The CLAM algorithm is on average less complex than the M algorithm while achieving significantly improved performance. For example, on a 4-input 4-output Rayleigh-fading channel with 64-QAM inputs, the CLAM algorithm outperforms the M algorithm by 2 dB at a BER of 10^{-3}, falling only 0.6 dB short of the joint maximum-likelihood detector, while simultaneously reducing the average search complexity by 6%.

References

[1] M. O. Damen, A. Chkeif, and J.-C. Belfiore, "Lattice Code Decoder for Space-Time Codes," *IEEE Communications Letters*, vol. 4, no. 5, pp.161.163, May 2000.

[2] L. G. Barbero and J. S. Thompson, "Performance Analysis of a Fixed-Complexity Sphere Decoder in High-Dimensional MIMO Systems," in *IEEE Int.Conf. on Acoustics, Speech, and Signal Processing (ICASSP '06)*, vol. 4, Toulouse, France, May 2006, p. 557-560.

[3] J. Yue, K. J. Kim, J. D. Gibson, and R. A. Iltis, "Channel Estimation and Data Detection for MIMO-OFDM Systems," in *Proc. GLOBECOM 2003*, pp. 581–585, 2003.

[4] W. H. Chin, "QRD Based Tree Search Data Detection for MIMO Communication Systems," in *Proc. IEEE VTC 2005 Spring*, vol. 3, pp. 1624-1627, Stockholm, Sweden, May 2005.

[5] J. B. Anderson and S. Mohan, "Sequential coding algorithms: A survey and cost analysis," *IEEE Trans. Commun*., vol. COM-32, no. 2, pp. 169–176, Feb. 1984.

[6] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE Journal on Selected Areas in Communications*, vol. 24, pp. 491–503, March 2006.

[7] D. Wübben, R. Böhnke, V. Kühn, and K.-D. Kammeyer, "MMSE Extension of V-BLAST Based on Sorted QR Decomposition," *IEEE Proc. Vehicular Technology Conference (VTC)*, Orlando, Florida, USA, October 2003.

[8] K. J. Kim, J. Yue, R.A. Iltis, and J.D. Gibson, "A QRD-M/Kalman filter-based detection and channel estimation algorithm for MIMO-OFDM systems," *IEEE Trans. Wireless Commun*., vol.4, no.2, pp.710–720, March 2005.

[9] J. R. Barry, E. A. Lee, D. G. Messerschmitt, *Digital Communication, Third Edition*, Boston: Kluwer Academic Publishers (2004).

[10] M. O. Damen, H. E. Gamal, and G. Caire, "On Maximum-Likelihood Detection and the Search for the Closest Lattice Point," *IEEE Trans. on Info. Th*., vol. 49, no. 10, Oct. 2003.

[11] D. Wübben and K.-D. Kammeyer, "Low Complexity Successive Interference Cancellation for Per-Antenna-Coded MIMO-OFDM Schemes by Applying Parallel-SQRD," *IEEE Proc. Vehicular Technology Conference (VTC)*, Melbourne, Australia, May 2006.

[12] S. G. Wilson and S. Husain, "Adaptive tree encoding of speech at 8000 bits/s with a frequency-weighted fidelity criterion," *IEEE Trans. Commun,*. vol. COM-27, pp. 165-170, Jan. 1979.

[13] D. E. Knuth, The Art of Computer Programming, Vol. III: Sorting and Searching. Reading, MA: Addison-Wesley, 1973.