The BEAST for maximum-likelihood detection in non-coherent MIMO wireless systems

Hug, Florian; Rusek, Fredrik

Published in:
[Host publication title missing]

DOI:
10.1109/ICC.2010.5501872

Published: 2010-01-01

The BEAST for Maximum-Likelihood Detection in Non-Coherent MIMO Wireless Systems

Florian Hug and Fredrik Rusek

Dept. of Electrical and Information Technology, Lund University,
P. O. Box 118, SE-22100 Lund, Sweden, Email: {florian, fredrik}@eit.lth.se

***Abstract*—Next generation wireless systems have to be able to efficiently deal with fast fading environments in order to achieve high spectral efficiency. Using multiple-input multiple-output (MIMO) systems and exploiting receive diversity, the spectral efficiency can be greatly increased. Commonly, the channel is estimated via training symbols, before data detection is carried out based on the obtained channel estimate. While this significantly simplifies the process of data detection, it leads in general to suboptimal results. A better approach is to carry out joint channel estimation and data detection; we turn our attention to joint maximum-likelihood (ML) detection which is the optimal strategy.**

In this paper, the BEAST — Bidirectional Efficient Algorithm for Searching code Trees — is proposed as an alternative algorithm for joint ML channel estimation and data detection and its complexity is compared with recently published algorithms in the literature.

I. Introduction

Achieving high-speed reliable data-transmission over dynamic wireless channels receives a lot of research attention from the information theory, signal processing and wireless communication communities. Such channels impose great challenges due to their time-variant behavior and fast fading properties. Commonly, a channel estimate $\widehat{H}$ for the time-varying channel H is estimated by training sequences and is subsequently used for data detection of the following data packet of length T, assuming the channel meanwhile to be constant. However, especially with fast fading channels, this approach becomes infeasible since the fraction of symbols spent on training may not be negligible, and one might resort to joint channel-estimation and data detection to improve the overall system performance [1]–[7].

Incorporating MIMO systems to increase spectral efficiency and exploiting receive diversity without channel state information at the receiver, additional challenges have to be solved [3], [8]–[10]. Generally, obtaining an channel estimate by training sequences before data detection indues a significant penalty in power and bandwidth consumption. A way to overcome this problem is however given by considering the problem of joint ML channel estimation and data detection for MIMO systems.

Suboptimal iterative receiving structures for joint channel estimation and data detection for multiple antenna systems have been studied in [1], [11], [12]. For single-input multiple-output (SIMO) systems its was shown that the problem of joint ML channel estimation and data detection is an integer least-squares problem under certain constant circumstances. Recently the problem of joint ML channel estimation and data detection was extended to MIMO systems [13], the corresponding optimization problem was formulated, and a branch-estimate-and-bound tree search algorithm for efficiently finding the ML solution was proposed.

Having a broader look at the field of coding theory one might notice that ML decoding of block and convolutional codes is often considered as finding a path of lowest weight through a given trellis or tree. The BEAST [14]–[16] — Bidirectional Efficient Algorithm for Search code Trees — is one of the most efficient algorithm used for finding the spectral components of block and convolutional codes as well as for ML decoding of block codes and is realized as a bidirectional tree search.

Following [13] we formulate an optimization problem suitable for joint ML channel estimation and data detection. By slightly modifying the BEAST, an alternative, often more efficient, algorithmic approach for obtaining the ML solution will be presented. Simulation results of the previously introduced branch-estimate-and-bound algorithm as well as an ordinary branch-and-bound algorithm compared with the BEAST illustrate the reduced complexity.

II. Problem Description

Hereinafter we will consider a wireless MIMO transmission scheme with M transmit antennas and N receive antennas. The MIMO channel matrix $H \in \mathcal{C}^{N\times M}$ is given by its individual complex channel gains $h_{ij} \in \mathcal{C}$ with $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, M$. Denote the T_p pilot symbols by the pilot matrix $P \in \Omega^{M\times T_\mathrm{p}}$ and the information symbols by the information matrix $X \in \Omega^{M\times T}$, both taken from a certain constellation Ω (*e.g.,* BPSK, QPSK, etc.). We assume a block fading channel model so that the MIMO channel matrix H is constant during the transmission of a data packet of length $T_\mathrm{p} + T$. Then channel output matrix Y can be written as

$$Y = H\,[P\,X] + N \tag{1}$$

where $N \in \mathcal{C}^{N\times(T_\mathrm{p}+T)}$ is a noise matrix with its elements being i.i.d. complex Gaussian random variables with zero mean and standard deviation σ ($\mathcal{CN}(0, \sigma^2)$). Note, that the location of the columns of the pilot matrix P is immaterial. Furthermore, we assume the entries of X being i.i.d.symbols and the channel matrix H to be considered as deterministically unknown with no *a priori* information given [1], [4], [17]. Clearly, wireless channels may have different distributions,

like Rayleigh, Rican, Nakagami or any other kind of fading statistics and that there are most likely correlations between the N different receive antennas. In fact, these distributions are needed for a joint ML channel estimation and data detection, but are generally unknown at the receiver side. However, for simplicity, we will use the unknown deterministic model at the receiver side, while the actual channel follows an i.i.d. Gaussian distribution with unit variance.

Under these assumptions, following [13], the problem of joint ML channel estimation and data detection can be reduced to solving

$$\min_{H,S\in\Omega^{M\times T}} \left\| Y - H\,[P\,S] \right\|^2 \tag{2}$$

where $\Omega^{M\times T}$ denotes the $M\times T$-dimensional signal lattice and S is any possible transmitted symbol matrix, since X corresponds to the actual transmitted symbol matrix. Denote the columns i to j, with $i<j$ of the possible transmitted symbol matrix S checked by the receiver and the corresponding channel output matrix Y by $S_{[i,j]}$ and $Y_{[i,j]}$, respectively. According to (1), a channel estimate $\widehat{H}_i$ can be calculated under the assumption of any partially transmitted symbol matrix $S_{[1,i]}$ by

$$\widehat{H}_i = Y_{[1,T_\mathrm{p}+i]}\left[P\,S_{[1,i]}\right]^\dagger \tag{3}$$

where $(\cdot)^\dagger$ denotes the Moore-Penrose pseudoinverse of a matrix. Replacing the unknown channel H in (2) by the complete channel estimate $\widehat{H}_T$ (3), the problem of joint ML channel estimation and data detection is given by solving

$$\min_{S\in\Omega^{M\times i}} \left\| Y - Y\,[P\,S]^\dagger\,[P\,S] \right\|^2. \tag{4}$$

It is easy to verify that (4) can be evaluated in a tree structure, the corresponding weight at depth i equals

$$\min_{S_{[1,i]}\in\Omega^{M\times i}} \left\| Y_{[1,T_\mathrm{p}+i]} - Y_{[1,T_\mathrm{p}+i]}\left[P\,S_{[1,i]}\right]^\dagger\left[P\,S_{[1,i]}\right] \right\|^2. \tag{5}$$

Furthermore (5) is monotonically increasing with i due to properties of the Frobenius norm,which implies that both the BEAST and the branch-estimate-and-bound tree search algorithm [13] can efficiently solve (2).

III. Algorithms for Searching a Tree

The final solution of (4) is given by a transmitted symbol matrix $S\in\mathcal{C}^{M\times T}$, which can be represented by tree of depth T stemming from the *root* (at depth 0), with a path of smallest metric weight through the tree corresponding to the solution of (4).

Denote a single node in a tree by ξ and let ξ^P be its parent node and ξ^C its children nodes. Every node is characterized by two parameters: depth $\ell(\xi)$ and metric weight $\omega(\xi)$. $|\Omega|^M$ branches are connecting every node ξ to its children nodes, being labeled by a $M\times 1$ column-vector. This column-vector corresponds to the assumed M symbols in the ith column of the possible transmitted symbol matrix S, where i is the current depth of the tree. Clearly, the M symbols are chosen out of the $|\Omega|^M$ possible combinations, where $|\Omega|$ denotes the number of symbols in Ω.

The depth $\ell(\xi)$ is equal to the length (in branches) and the metric weight $\omega(\xi)$ is calculated according to (4) by

$$\begin{aligned}\omega(\xi) \;=\; & \Big\| Y_{[1,T_\mathrm{p}+\ell(\xi)]} - Y_{[1,T_\mathrm{p}+\ell(\xi)]} \\ & \times\left[P\,S_{[1,\ell(\xi)]}\right]^\dagger\left[P\,S_{[1,\ell(\xi)]}\right]\Big\|^2\end{aligned}$$

where $S_{[1,\ell(\xi)]}$ corresponds to the first $\ell(\xi)$ columns of the transmitted symbol matrix S, which are the $M\times 1$ column-vectors on the $\ell(\xi)$ branches of the path arriving from the root ξ_root at the node ξ.

Among all nodes ξ at depth $\ell(\xi)=T$, the one with smallest metric weight $\omega(\xi)$ corresponds to the solution of the joint ML channel estimation and data detection optimization problem (2). Note, that the memory introduced in the decoder is due to the channel estimation $\widehat{H}$, which is based on the previously $M\cdot i$ symbols $S_{[1,i]}$.

Besides an exhaustive tree search, several different algorithms will be presented and discussed in what follows. In [13] a so-called branch-estimate-and-bound algorithm was proposed for joint ML channel estimation and data detection. However, it is sufficient, and often less complex, to use an ordinary branch-and-bound algorithm to obtain the same result. Additionally we will propose the BEAST as an efficient alternative for joint ML channel estimation and data detection. Due to the fact that the BEAST was originally designed as an efficient tree searching algorithm for obtaining the spectrum of both block and convolutional codes and ML decoding of block codes, it needs to be slightly modified.

We will compare the complexity and efficiency of these algorithms with each other in terms of the average number of memory elements per decoded bit. In order to obtain a fair and realistic comparison, the maximum number of visited nodes is limited in all algorithms by m_nodes.

A. Branch-and-Bound Algorithm

In the branch-and-bound (BnB) algorithm, the currently best node ξ in terms of smallest metric weight $\omega(\xi)$ is extended during each iteration, until it has reached depth $\ell(\xi)=T$ and the corresponding path $\xi_\text{root}\to\xi$ is returned.

Initializing a list with the root node ξ_root of both, depth $\ell(\xi)$ and metric weight $\omega(\xi)$ zero, the algorithm removes the best node ξ_best in terms of smallest metric weight from the list during each iterations. The depth and metric weight of its $|\Omega|^M$ children nodes are calculated and appended to this list, before continuing with the next iteration. As soon as the depth of the best node $\ell(\xi_\text{best})$ is equal to T, the optimization problem (2) is solved and the algorithm terminates.

Additionally, if the number of visited nodes (*i.e.,* the number of nodes for which the depth and metric weight were previously calculated) exceeds the threshold m_nodes, the algorithm is terminated as well. In this case the path $\xi_\text{root}\to\xi_\text{largest}$, where ξ_largest has largest depth among all previous visited nodes, is used to obtain a channel estimate $\widehat{H}$ according to (3). Using

this channel estimate $\widehat{H}$, the remaining symbols for a path of length T are decoded independently symbols-wise, using (2). Note, that this might lead to non-ML results.

B. Branch-Estimate-and-Bound Algorithm

The branch-estimate-and-bound (BEnB) algorithm, as proposed by [13], starts by obtaining an initial threshold T_{tr} (*cf.* Section III-D). Afterwards a depth-first search is performed, as long as the metric weight $\omega(\xi)$ is smaller than or equal to the threshold T_{tr}. Thereby a path of length T stemming from the root ξ_{root} and weight smaller than or equal to the initial threshold T_{tr} is obtained quickly. Finally the corresponding path $\xi_{\text{root}} \to \xi$ of length T is stored and the threshold T_{tr} is updated by the metric weight $\omega(\xi)$, before other possible paths are considered.

Starting from the last visited node ξ with depth $\ell(\xi) = T$ and going back towards the root node ξ_{root}, all other possible, not previously checked, paths are considered as long as their metric weight is smaller than or equal to the current threshold T_{tr}. Having found a node ξ with depth $\ell(\xi) = T$ and metric weight smaller than T_{tr}, the threshold is updated accordingly and the path $\xi_{\text{root}} \to \xi$ is stored as the current best result.

The algorithm terminates if there are no more possible nodes to visit with metric weight smaller than the current threshold T_{tr}, and returns the last stored path of length T. For more detailed implementation aspects, we refer to [13].

Compared to the the initial proposal in [13], we modified the algorithm slightly by extending always in the direction of the children nodes with smallest metric weight. While this does not influence the obtained results and its performance in general, it is necessary to be able to terminate the algorithm and obtain a reasonable results by just returning the last stored path of length T, after having reached the threshold m_{nodes}. As for the branch-and-bound algorithm, the complexity limit leads to, in general, non-ML results.

C. The BEAST

The BEAST [14]–[16] was initially designed as an efficient algorithm for finding the spectral components of block and convolutional codes and later modified for ML decoding of block codes. However, as this algorithms performs an efficient bi-directional tree search it is not far fetched to adapt it to the concept to joint ML channel estimation and data detection.

While the previous algorithms perform only a search within a one-sided tree, the BEAST starts searching from both sides at the same time. In order to distinguish between the forward and backward tree, we denote the depth and the metric weight of a certain node ξ in the forward tree by $\ell_{\mathrm{F}}(\xi)$ and $\omega_{\mathrm{F}}(\xi)$, respectively. As the forward tree resembles the previous search algorithms, its parameters correspond to the definition as mentioned in Section III, that is, ℓ_{F} corresponds to the distance (in branches) from the root node ξ_{root} and

$$\begin{aligned}\omega_{\mathrm{F}}(\xi) \;=\; & \Big\| Y_{[1,T_{\mathrm{p}}+\ell_{\mathrm{F}}(\xi)]} - Y_{[1,T_{\mathrm{p}}+\ell_{\mathrm{F}}(\xi)]} \\ & \times \left[P\, S_{[1,\ell_{\mathrm{F}}(\xi)]}\right]^{\dagger} \left[P\, S_{[1,\ell_{\mathrm{F}}(\xi)]}\right] \Big\|^2.\end{aligned}$$

Similarity, for the backward tree, a search is started from the *toor*[1] node ξ_{toor} of both, depth $\ell_{\mathrm{B}}(\xi_{\text{toor}})$ and metric weight $\omega_{\mathrm{B}}(\xi_{\text{toor}})$ zero. While the definition of the depth ℓ_{B} corresponds to the distance (in branches) to the toor node ξ_{toor}, the metric weight is given by

$$\begin{aligned}\omega_{\mathrm{B}}(\xi) \;=\; & \Big\| Y_{[T_{\mathrm{p}}-\ell_{\mathrm{B}}(\xi),T_{\mathrm{p}}]} - Y_{[T_{\mathrm{p}}-\ell_{\mathrm{B}}(\xi),T_{\mathrm{p}}]} \\ & \times \left[P\, S_{[T_{\mathrm{p}}-\ell_{\mathrm{B}}(\xi),T_{\mathrm{p}}]}\right]^{\dagger} \left[P\, S_{[T_{\mathrm{p}}-\ell_{\mathrm{B}}(\xi),T_{\mathrm{p}}]}\right] \Big\|^2.\end{aligned}$$

Assume that the goal is to find a path through the tree of length T, that is, $\xi_{\text{root}} \to \xi_{\text{toor}}$, with metric weight smaller than or equal to T_{tr}. For such a path, there exists an intermediate node ξ, such that

$$\omega_{\mathrm{F}}(\xi) \geq \frac{T_{\mathrm{tr}}}{2}, \quad \omega_{\mathrm{B}}(\xi) \leq \frac{T_{\mathrm{tr}}}{2}, \quad \ell_{\mathrm{F}}(\xi) + \ell_{\mathrm{B}}(\xi) = T.$$

Hence, a search for all paths $\xi_{\text{root}} \to \xi_{\text{toor}}$ of metric weight smaller than or equal to T_{tr} can be split into a forward search for all paths $\xi_{\text{root}} \to \xi$ of weight greater than or equal to $T_{\mathrm{tr}}/2$, and a backward search for all paths $\xi \to \xi_{\text{toor}}$ of metric weight smaller than or equal to $T_{\mathrm{tr}}/2$. Based on these criteria, the BEAST finds all paths of metric weight smaller than or equal to T_{tr} as follows:

(1) *Forward search:* Starting at the root node ξ_{root}, grow a forward tree and obtain the set of nodes

$$\mathcal{F} = \left\{ \xi \;\middle|\; \omega_{\mathrm{F}}(\xi) \geq \frac{T_{\mathrm{tr}}}{2}, \omega_{\mathrm{F}}(\xi^{\mathrm{P}}) < \frac{T_{\mathrm{tr}}}{2}, \ell_{\mathrm{F}}(\xi) \leq T \right\}.$$

(2) *Backward search:* Starting at the toor node ξ_{toor}, grow a backward tree and obtain the set of nodes

$$\mathcal{B} = \left\{ \xi \;\middle|\; \omega_{\mathrm{F}}(\xi) \leq \frac{T_{\mathrm{tr}}}{2}, \ell_{\mathrm{B}}(\xi) \leq T \right\}.$$

(3) *Matching:* Find all pair of nodes $(\xi, \xi') \in \mathcal{F} \times \mathcal{B}$ such that $\ell_{\mathrm{F}}(\xi) + \ell_{\mathrm{B}}(\xi') = T$. Each such match uniquely describes a path $\xi_{\text{root}} \to \xi_{\text{toor}}$ with its overall metric ω being determined by (4). Discard all paths that do not fulfill $\omega < T_{\mathrm{tr}}$ (otherwise we might obtain a non-ML result). Among the remaining paths, choose the one of lowest metric weight and stop the algorithm. If no path is left, increase the threshold T_{tr} and go to step (1).

Note, that the forward and backward search can be carried out independently, allowing parallel implementations. During each iteration, the previously obtained sets can be reused, instead of restarting the search from the root and/or the toor node. Moreover, the efficiency of the BEAST can be improved by extending only the smaller one of the two sets (trees) during each iteration.

As in the previous algorithms, this algorithm is terminated if the threshold m_{nodes} is reached. In this case the longest path from the forward tree $\xi_{\text{root}} \to \xi_{\mathrm{F}}$ is combined with a suitable path from the backward tree $\xi_{\mathrm{B}} \to \xi_{\text{toor}}$. If some symbols in the middle are not yet determined, a channel estimate $\widehat{H}$ is obtained using the two parts and the remaining symbols are

[1]Corresponding to the root node in the forward tree, the toor node denotes the starting node in the backward tree (toor = root backwards)

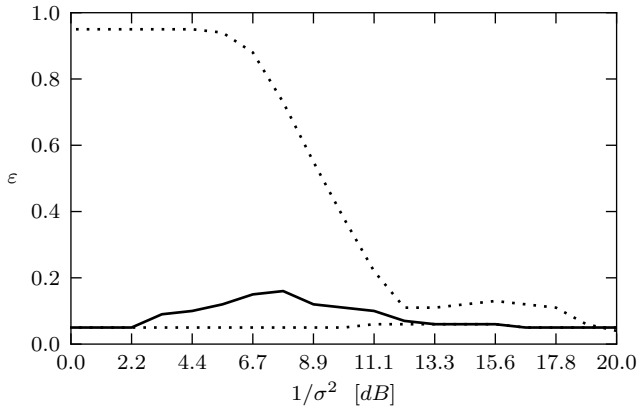

Fig. 1. Optimal Thresholds T_{tr} (*i.e.,* ε) for BEnB using different noise standard deviations σ

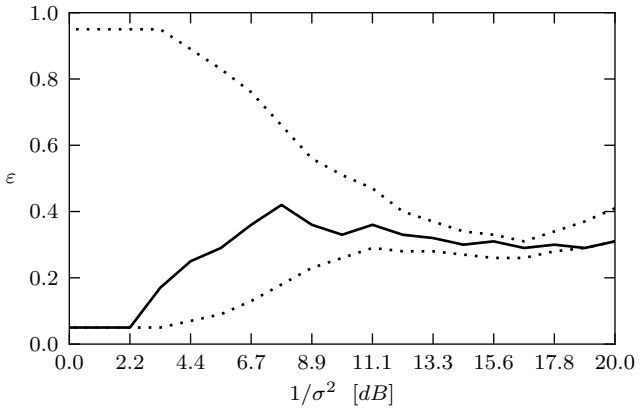

Fig. 2. Optimal Thresholds T_{tr} (*i.e.,* ε) for the BEAST using different noise standard deviations σ

decoded independently symbols-wise, using (2). Note, that this might lead to non-ML results as well.

D. Determining the Threshold

Following [13], the threshold T_{tr} can be determined probabilistically. As $\|N\|^2$ is chi-square distributed with $|\Omega|(T_{\mathrm{p}}+T)$ degrees of freedom, the threshold T_{tr} is naturally specified as

$$T_{\mathrm{tr}}^2 = \min\left\{x \;\middle|\; P\left(\|N\|^2 > x\right) \leq 1-\varepsilon\right\}$$

with $\varepsilon > 0$.

IV. Simulation Results

In the following simulations, we assume a 2×2 MIMO system with packet length T and QPSK signaling (*i.e.,* 2 bits/symbol and $4T$ bits in total). Additionally, the maximum number of nodes to be visited is limited to $m_{\text{nodes}} = 4T \times 2^7$ (corresponds to the ML decoding complexity of a trellis with 2^7 states).

In a first step a near-optimum threshold T_{tr} will be determined, which will be used later on to compare the complexity of the three different algorithm approaches.

A. Optimizing the Threshold

The efficiency of the BEnB algorithm and the BEAST depends largely on the chosen threshold T_{tr}, that is, the parameter ε.

For all combinations of the noise standard deviation $\sigma = 10^{-1}, \ldots, 10^{0}$ in 19 equal logarithmic steps and the parameter $\varepsilon = 0.05, 0.10, \ldots, 0.95$, a joint ML channel estimation and data detection using the BEnB algorithm and the BEAST is performed.

The parameter ε leading to the smallest number of visited nodes in each possible combination is illustrated as a function of σ in Figure 1 and Figure 2 for the BEnB algorithm and the BEAST, respectively. To compensate for slight deviations, the area $\pm 1\%$ of the absolute minimum is marked additionally by the two dashed lines.

Clearly, the range of optimal ε grows with increasing σ as both algorithms reach the complexity threshold m_{nodes} already during their first iterations. Additionally we note that the actual minimum does not depend on the chosen σ, which leads to the choice of $\varepsilon = 0.05$ and $\varepsilon = 0.3$ for the BEnB algorithm and the BEAST, respectively, leading to a reasonable near-optimum threshold T_{tr}.

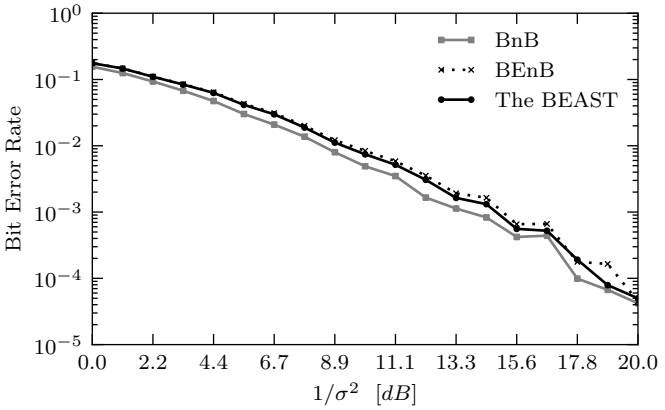

Fig. 3. Obtained bit error rate for a given noise standard deviation σ for all three algorithm approaches using packet length of $T = 50$

B. Complexity Comparison

Using the previously determined values for ε, the complexity of the three algorithms (BnB, BEnB, and BEAST) are compared. As the Moore-Penrose pseudoinverse has to be evaluated in all three algorithms for the same maximum matrix length T_p+T, we will consider the number of evaluations, that is the number of visited nodes as our complexity measurement.

However, instead of using the absolute number of visited nodes $m_{\text{visited nodes}}$, we will relate them to the number of transmitted bits, leading to the average number of memory elements per decoded bit, according to

$$\log_2\left(\frac{m_{\text{visited nodes}}}{T\,|\Omega|}\right).$$

To include different channels, we vary the noise standard deviation $\sigma = 10^{-1}, \ldots, 10^{0}$ in 19 equal logarithmic steps. Using the resulting channel output matrix Y, a joint ML channel estimation and data detection is performed by each of the three algorithms.

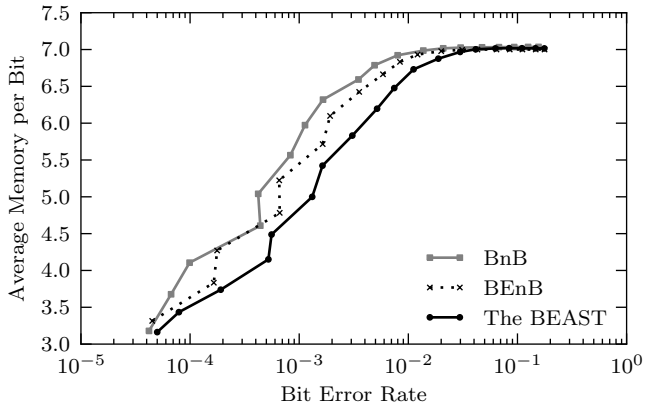

Fig. 4. Complexity in memory per bit over achieved bit error rate for packet length $T = 50$

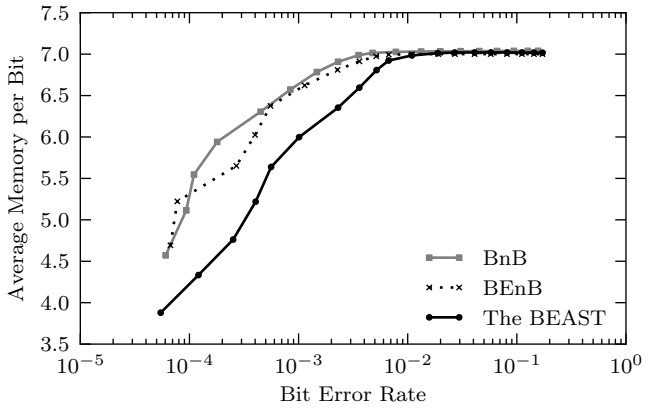

Fig. 5. Complexity in memory per bit over achieved bit error rate for packet length $T = 100$

In Figure 3 the obtained bit error rate for all three algorithms is illustrated over noise standard deviation σ for a packet length of $T = 50$.

The complexity is illustrated in Figure 4 and Figure 5 over the achieved bit error rate for a packet length of $T = 50$ and $T = 100$, respectively. Clearly, by using the BEAST, the average number of memory elements per decoded bit is reduced and decreases furthermore with an increasing packet length T compared to the BnB and BEnB algorithm.

Note that for larger bit error rates (*i.e.,* $\geq 10^{-1}$), all algorithms have to visit a major fraction of the tree and reach the complexity limit m_{nodes} already during their first iterations, leading to the same "flat" complexity performance as given in the illustrations. This behavior, however, can be overcome by increasing m_{nodes}.

V. Conclusions

The BEAST was introduced as an alternative algorithm for joint ML channel estimation and data detection. Its performance and complexity were considered and compared with those of the previously suggested Branch-Estimate-and-Bound and Branch-and-Bound algorithm. While the Branch-and-Bound algorithm obtains the best bit error rate for a fixed noise density, the BEAST achieves the lowest average complexity, in terms of the average number of memory elements per decoded bit for a given bit error rate. With increasing packet lengths, the complexity-gain of the BEAST increases even further compared to the other proposed algorithms.

Acknowledgements

This research was supported in part by the Swedish Research Council under Grant 621-2007-6281 and in part by the Swedish Foundation for Strategic Research through its Center for High Speed Wireless Communication at Lund University and in part by VINNOVA through the WILATI+ project.

References

[1] P. Stoica and G. Ganesan, "Space-time block codes: Trained, blind and semi-blind detection," *Digital Signal Processing*, vol. 13, pp. 93–105, 2003.

[2] H. Vikalo, B. Hassibi, and P. Stoica, "Efficient joint maximum-likelihood channel estimation and signal detection," *IEEE Trans. Wireless Commun.*, vol. 5, no. 7, pp. 1838–1845, Jun. 2006.

[3] P. Stoica, H. Vikalo, and B. Hassibi, "Joint maximum-likelihood channel estimation and signal detection for SIMO channels," in *Proc. 2003 Int. Conf. on Acoust., Speech, Signal Process.*, vol. 4, 2003, pp. 13–16.

[4] W.-K. Ma, B.-N. Vo, T. Davidson, and P. Ching, "Blind ML detection of orthogonal space-time block codes: High-performance, efficient implementations," *IEEE Trans. Signal Process.*, vol. 54, no. 2, pp. 738–751, Feb. 2006.

[5] A. L. Swindlehurst and G. Leus, "Blind and semi-blind equalization for generalized space-time block codes," *IEEE Trans. Signal Process.*, vol. 50, no. 10, pp. 2589–2498, 2002.

[6] S. Shahbazpanahi, A. Gershman, and J. Manton, "Closed-form blind MIMO channel estimation for orthogonal space-time block codes," *IEEE Trans. Signal Process.*, vol. 53, no. 12, pp. 4506–4517, Jun. 2005.

[7] N. Jindal, A. Lozano, and T. L. Marzetta, "What is the value of joint processing of pilots and data in block-fading channels?" in *Proc. Int. Sym. Inf. Theory (ISIT'09)*, Seoul, South-Korea, Jun. 2009.

[8] B. Hochwald and T. Marzetta, "Unitary space-time modulation for multiple-antenna communication in Rayleigh flat fading," *IEEE Trans. Inf. Theory*, vol. 46, pp. 543–464, Mar. 2000.

[9] T. Marzetta, "Blast training: Estimating channel characteristics for high-capacity space-time wirless," in *Proc. 37th Annual Allerton Conf. Commun., Control, and Computing*, Sep. 1999.

[10] L. Zheng and D. Tse, "Communicating on the Grassmann manifold: A geometric approach to the non-coherent multiple antenna channel," *IEEE Trans. Inf. Theory*, vol. 48(2), pp. 359–383, Feb. 2002.

[11] C. Cozzo and B. Hughes, "Joint channel estimation and data detection in space-time communications," *IEEE Trans. Commun.*, vol. 51, no. 8, pp. 1266–1270, Aug. 2003.

[12] E. Larsson, P. Stoica, and J.Li, "Orthogonal space-time block codes: Maximum-likelihood detection for unknown channels and unstructured interferences," *IEEE Trans. Signal Process.*, vol. 51, no. 2, pp. 362–372, 2003.

[13] W. Xu, M. Stojnic, and B. Hassibi, "On exact maximum-likelihood detection for non-coherent MIMO wireless systems: a branch-estimate-bound optimization framework," *Proceedings of the IEEE International Symposium on Information Transaction*, pp. 2017–2021, Jun. 2008.

[14] I. Bocharova, M. Handlery, R. Johannesson, and B. Kudryashov, "A BEAST for prowling in trees," in *Proc. 39th Annual Allerton Conf. Commun., Control, and Computing*, Monticello, Illinois, USA, Oct. 2001.

[15] ——, "A BEAST for prowling in trees," *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1295–1302, Jun. 2004.

[16] I. Bocharova, R. Johannesson, B. Kudryashov, and M. Lončar, "BEAST decoding for block codes," *European Trans. on Telecommunications*, vol. 15, no. 4, pp. 297–305, Jul. 2004.

[17] E. Larsson, P. Stoica, and J.Li, "On maximum-likelihood detection and decoding for space-time coding systems," *IEEE Trans. Signal Process.*, vol. 50, no. 4, pp. 937–944, 2002.