Automatic Joint Optimization of Iterative MIMO-OFDM Receiver Algorithms on a Meta Level

Andreas Ibing, Holger Boche
Chair of Theoretical Information Technology
Technische Universität München
Arcisstrasse 21, 80333 München, Germany

***Abstract*—We present a method for automatically optimized selection and composition of algorithm components for iterative MIMO-OFDM receivers. Complexity is measured as clock cycles of the target processor cores (suitable for software defined radio based implementation). A receiver description language is used to specify the design space, and to enable search in the design space using graph algorithms. Performance prediction of a composition of candidate algorithms uses a recently published method from stochastic convergence analysis of iterative processing. We illustrate the method in an example scenario, where the optimized receivers show both an extended operational SNR range compared to the standard receiver architecture, as well as significantly reduced complexity compared to iterative processing according to round-robin iteration scheduling using the same algorithm components.**

I. Introduction

The aim of finding the 'best' receiver with respect to a certain optimization criterion, where the receiver components are selected from the large body of published algorithms, faces several difficulties.

While the structure of a receiver is clear and can be derived from the interrelations of transmission variables, the optimization lies in the selection and concatenation of component algorithms. For each receiver component, a whole 'algorithm zoo' is available, containing algorithm candidates which are Pareto-optimal [1] regarding a performance/complexity tradeoff – especially for MIMO demapping and channel estimation. Attainment of receiver 'optimality' further requires evaluation of a component in the complete receiver: while receiver complexity is the sum of component complexities, there is no such simple relation for the performance. Complexity of a component may also be implementation-dependent, i.e. hardware-dependent.

Modern receivers follow a trend from sequential to iterative processing. This has become obvious for channel decoding (e.g. Turbo decoding instead of Viterbi algorithm), but also for other receiver components like channel estimation and demapping, the gains of maximum a posteriori (MAP) processing compared to the (non-iterative) maximum likelihood (ML) method have been demonstrated [2], [3]. For optimization of iterative receivers, usage of different component algorithms in each iteration is possible.

The possible receiver design space has become so large that it is unmanageable by the traditional approach of Monte-Carlo link-level simulation. Extrinsic information transfer charts (EXIT charts, [4]) have proven useful to quickly predict performance of iterative decoding of concatenated codes. EXIT charts have also been applied to predict performance of iterative receivers, especially including iterative demapping. On the other hand, EXIT charts were found to be not applicable in the case of fading MIMO channels [5], [6]. We therefore use the fast prediction method from [5], which is shortly recapitulated in subsection II-C.

In this paper we present a method to find the 'optimal' receiver as concatenation of component algorithms chosen from a given set of algorithm candidates, without the need for link-level simulation of the receiver candidates. As complexity measure we count the number and type of elementary operations of the algorithms candidates and map them to processor cycles on a target implementation hardware (assuming a common multicore processor architecture shared between the components). For different (processor based) hardware, a 'retargeting' of the complexity metric is possible. We further use a language to formally describe a receiver 'design space' as composition (schedule) of component algorithms. This allows enumeration of and search in the design space (using branch and bound graph search), where for each receiver candidate performance (in terms of bit error rate or mutual information) and complexity can be quickly assessed.

II. Preliminaries

A. Generic MIMO-OFDM Receiver

The generic receiver structure can be obtained from Bayesian inference, where conditional independencies between transmission variables are exploited by corresponding factorization of the joint probability density function

This work was accomplished while the authors were at Heinrich-Hertz-Chair for Mobile Communications, Technische Universität Berlin, Germany.

[7]. The generic structure includes standard non-iterative receivers as special case.

MIMO transmission at time instance t over the channel matrix $\mathbf{H}^{(t)}$ is denoted as (per-subcarrier model in MIMO-OFDM transmission):

$$\mathbf{y}^{(t)} = \mathbf{H}^{(t)} \cdot \mathbf{x}^{(t)}(\mathbf{b}^{(t)}) + \mathbf{n}^{(t)}. \tag{1}$$

where $\mathbf{b}^{(t)}$ is a vector of transmit bits as part of the complete codeword $\mathbf{b}$, $\mathbf{x}^{(t)}$ is the corresponding vector of modulated symbols. The complete set of received symbol values of the message (all time instances) is denoted $\mathbf{y}$. The transmitter in our example uses Turbo coding, so that the code word $\mathbf{b}$ consists of the information bits $\mathbf{u}$, parity bits $\mathbf{c}_1$ of the first constituent encoder and parity bits $\mathbf{c}_2$ of the second constituent encoder. For a single bit of the bit vector $\mathbf{b}$ at position i we write b_i.

Maximum receiver performance would be reached if computing the maximum likelihood solution on codeword basis:

$$\hat{\mathbf{u}} = \arg\max_{\mathbf{u}} P(\mathbf{u}|\mathbf{y})$$

As direct implementation by exhaustive search is infeasible due to complexity, the practical approach is an iterative local approximation of the information bit APPs with subsequent binary quantization. In our example we optimize iterative demapping-decoding, and assume perfect channel estimation. So the joint probability density can be factorized [5]:

$$\begin{aligned} P(\mathbf{u},\mathbf{y}) &= \Big(\prod_t f_{Dem}(\mathbf{b}^{(t)}, \mathbf{y}^{(t)}, \mathbf{H}^{(t)})\Big) \cdot f_{Dec1}(\mathbf{u}, \mathbf{c_1}) \cdot \\ &\quad \cdot f_{Dec2}(\mathbf{u}, \mathbf{c_2}) \end{aligned} \tag{2}$$

which corresponds to a detector for each different time instance and the two constituent decoders, The factorization is illustrated in Fig. 1 ([5], using a factor graph notation similar to [8]).

To reduce complexity, a receiver uses log-probability rations (LLRs) for bit densities. The receiver computes an LLR for each transmit bit c:

$$L(c) = \ln \frac{P(c=+1)}{P(c=-1)} \tag{3}$$

The messages passed in Fig. 1 are therefore vectors of LLRs. L_a denotes a priori LLR, L_p a posteriori LLR. A factor node computes a posteriori LLRs, but outputs only the extrinsic LLRs $L_e = L_p - L_a$ [7], [9], [10]. Variable nodes compute sums of the incident LLR vectors, so that the a posteriori values of information bits are [5]

$$L_p(u_i) = L_e^{(det)}(u_i) + L_e^{(dec1)}(u_i) + L_e^{(dec2)}(u_i) \tag{4}$$

B. Component algorithms performing approximate APP computation

For complexity reasons, practically all component algorithms of interest compute (only) approximations of the a posteriori probabilities. For our optimization example, we consider three algorithm candidates which are shortly described in this subsection.

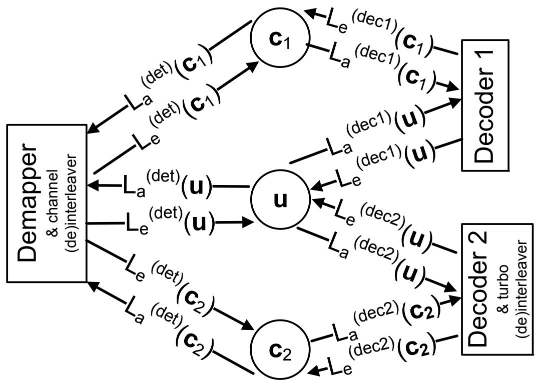

Fig. 1. Generic receiver structure for iterative demapping-decoding of Turbo-coded transmission [5].

1) MIMO Demapper: The optimal APP demapper for time instance t computes and outputs for each bit b_i of the codeword fragment $\mathbf{b}^{(t)}$:

$$\begin{aligned} P(b_i) &= \sum_{\mathbf{x}_{Qi}} P(b_i|\mathbf{x}_{Qi}, \mathbf{H}^t, \mathbf{y}^t) P(\mathbf{x}_{Qi}) \\ &= \sum_{\mathbf{x}_{Qi}} P(b_i|\mathbf{x}_{Qi}, \mathbf{H}^t, \mathbf{y}^t) \prod P(b_i) \end{aligned} \tag{5}$$

where probabilities are conditioned on all valid modulation vectors, and apriori information about the modulation vectors (from bit probabilities) is used.

We consider two approximative algorithms, namely the 'unbiased MMSE' and the 'Max-Log-APP' demapper.

a) UMMSE demapper: With channel noise variance σ^2, the MMSE receiver filter matrix for separation of the streams is given by:

$$G_{\text{bias}} = (H^H H + \sigma^2 I)^{-1} H^H$$

The unbiased version is $G_{\text{unb}} = S G_{\text{bias}}$, where S is the diagonal matrix which removes the bias introduced by the MMSE criterion [11]:

$$S = \text{diag}\Big(\frac{1}{(G_{\text{bias}}H)_{1,1}}, \ \ldots, \ \frac{1}{(G_{\text{bias}}H)_{N_T,N_T}}\Big)$$

The equalized symbol vector is:

$$\hat{\mathbf{x}} = G_{\text{unb}} \mathbf{y}$$

The LLR for transmit antenna i and bit position j is (Max-Log approximation per stream):

$$L(c_{i,j}) = -\frac{1}{2\sigma_{eq}^2}\Big(\min_{x_i \in \mathcal{X}_{j=1}^1} |\hat{x_i} - x_i|^2 - \min_{x_i \in \mathcal{X}_{j=-1}^1} |\hat{x_i} - x_i|^2\Big)$$

where σ_{eq}^2 is the noise variance on the stream after filtering. $x_i \in \mathcal{X}_{j=1}^1$ means the set of symbols where the bit whose LLR is to be computed has the value 1, $\mathcal{X}_{j=-1}^1$ is the complement set.

b) Max-Log-APP demapper: for joint Max-Log-APP demapping of all streams, the extrinsic LLRs are [12]:

$$\begin{aligned} L_e(c_i) \approx\ & \underset{\mathbf{x}\in\mathcal{X}_i^+}{max}\Big(-\frac{1}{2\sigma^2}||\mathbf{y}-\mathbf{H}\mathbf{x}(\mathbf{c})||^2 \\ & + \sum_{n\neq i}\min(c_n L_a(c_n);0)\Big) \\ & - \underset{\mathbf{x}\in\mathcal{X}_i^-}{max}\Big(-\frac{1}{2\sigma^2}||\mathbf{y}-\mathbf{H}\mathbf{x}(\mathbf{c})||^2 \\ & + \sum_{n\neq i}\min(c_n L_a(c_n);0)\Big) \end{aligned} \quad (6)$$

2) Constituent Decoder: An APP Decoder computes and outputs for each bit b_i of $\mathbf{u}$ and $\mathbf{c_1}$:

$$P(b_i) = \sum_{v\in V} P(b_i|v)P(v) = \sum_{v\in V} P(b_i|v)\prod P(b_i)$$

The summation is over the set $V_i^{b_i}$ of all valid codewords $\mathbf{v} = [\mathbf{u}\ \mathbf{c_1}]$ of the constituent code (and exploits a priori probabilities of codewords) which have the bit value b_i at position i. The computation can be reformulated as summation over states of the code trellis and efficiently implemented using the BCJR algorithm [13].

C. Convergence prediction for iterative MIMO-OFDM processing

We use the approach from [5], which models any algorithm candidate as a mapping from the probability density functions (PDF) of input variables to the PDFs of the output variables. Conditional LLR PDFs are modelled using a 2-parametric Gaussian model (mean and variance). The PDF mapping can be either a function or a table look-up from a pre-computed table. Performance prediction for a receiver consisting of a sequency of processing block updates reduces to a concatenation of table look-ups. Sufficiency of prediction accuracy has been verified [5]. Prediction of the bit error rate (BER) is obtained for the PDF of a posteriori LLRs as tail distribution:

$$BER = Q(\frac{\mu_p}{\sigma_p}), \quad (7)$$

where μ_p and σ_p are mean and standard deviation of the conditional LLR density [5].

III. Optimization Criteria

For given transmission mode (modulation, MIMO scheme and code rate) and channel characteristics, receiver processing quality can be described by operational SNR (minimal SNR where reception works with predefined error rate), receiver complexity and processing delay. In the 3-dimensional (complexity, target SNR, processing delay) receiver algorithm utility space, the following optimization criteria could be chosen (among any other weighted combinations):

1) Minimum complexity for fixed operational SNR. Which receiver satisfies the operational requirements with minimum computational effort? The answer to this question could be used (apart from choosing the algorithms) to choose adequate hardware, e.g. the number of processor cores, clock speed or width of parallel processing (single instruction multiple data, SIMD).
2) Minimum operational SNR for fixed target hardware. Given a fixed hardware with certain computational power, what are the achievable operating conditions (and with which algorithms can this be reached)?
3) Minimum delay for fixed operational SNR. How far can the processing delay be reduced by parallelizing node updates using multiple processor cores?

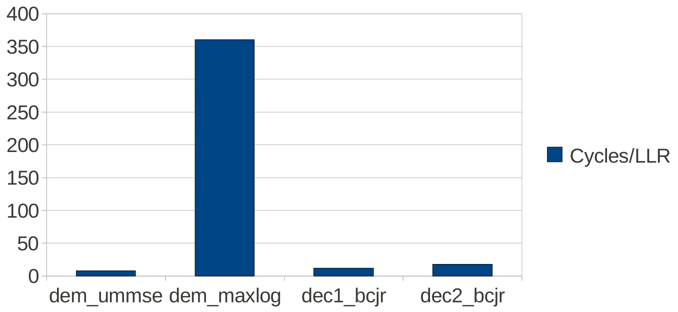

Fig. 2. Computational effort of the considered signal processing blocks on target processor core.

A. Complexity measure

a) Hardware-independent algorithm complexity measure: For the hardware independent assessment of complexities, the elementary operations of the considered signal processing blocks are counted. Demappers like UMMSE and Max-Log use multiply-accumulate (MAC) operations for the linear algebra part, and table look-ups (LU) for the soft demapping itself (modulation with separable I/Q components like e.g. in LTE [14] is assumed). The Max-Log demapper also uses select operations (SEL, conditional move). The turbo decoder uses Add-Compare-Select (ACS) operations for trellis traversal and Compare-Select (CS) operations for LLR computation.

b) Hardware-dependent measure, theoretical clock cycles on Cell SPU: To enable comparison and joint optimization, the different operations have to be expressed in a common metric. For this hardware dependent mapping, theoretically achievable clock cycles on the target hardware under the assumption of full utilization of processor resources are counted. By using this theoretical upper limit, the measure is hardware-dependent, but independent from the actual implementation code quality (different from using benchmark results as complexity measure). The Cell processor SPU has been chosen as target hardware due to its general-purpose signal processing architecture and high performance. It is used in an SDR testbed described in [15], [16], which give implementation benchmarks and also discuss how close an actual implementation with reasonable

Operation	SPU Cycles
real-valued Multiply-Accumulate (MAC)	0.25
table look-up (LU, demodulator)	1
Select (conditional move)	0.25
Add-Compare-Select (decoder, 16bit)	0.5
Compare-Select (decoder, 16bit)	1
Add (decoder, 16bit)	0.125
QPP read (turbo interleaver)	1
QPP write (turbo interleaver)	1

TABLE I
MAPPING COMPONENT ALGORITHM COMPLEXITY TO (HARDWARE-DEPENDENT) IMPLEMENTATION COMPLEXITY; THEORETICALLY ACHIEVABLE PROCESSOR CYCLE COUNT.

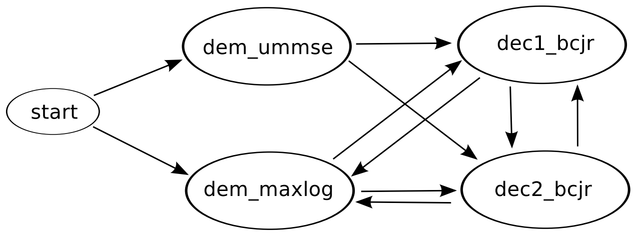

Fig. 3. Example search graph specifying receiver design space $\mathcal{R}_{\text{iterative-1}}$ (only useful state transitions are shown, double updates etc. omitted). Each state can be an end state.

programming effort (C-language using vector intrinsics) can reach the theoretical cycle count.

The numbers of elementary operations per cycle on the SPU using SIMD parallel implementation are given in table I. The numbers in the table are a consequence of SIMD processing with 128 bit width, where signal processing operations (like MAC) are performed on 32bit (single-precision) float numbers and decoding uses 16bit integer representation for LLRs. Load/store operations can be done in parallel to arithmetic operations (different processor pipelines) for all blocks except the turbo (de-)interleaver, where they have to be counted explicitly. Retargeting the complexity measure for a different architecture would use a different table.

Resulting block complexities (in cycles/LLR) are illustrated in Fig. 2 for UMMSE and Max-Log demapper (both for 4x4 QPSK), decoder 1 and decoder 2 using the BCJR algorithm. The decoder 2 block is a bit more complex then decoder 1 because Turbo interleaver and Turbo deinterleaver are counted as belonging to this block. For the Max-Log demapper's complexity, independent computation of all candidate vector metrics is assumed (no usage of intermediate partial metrics).

IV. Receiver Algorithm Description Language

To describe node update schedules where different algorithms for each update are possible, a description language is introduced in this section. It has a regular grammar and can thus be parsed by a finite state automaton (Chomsky hierarchy type 3 language [17]). A receiver then corresponds to a path through the finite state automaton.

A. *Factor graph notation: directed bipartite graph*

A factor graph F is given by the sets of its vertices (nodes) and directed edges $F = \{V, E\}$, with the property that the graph is bipartite: the set of nodes consists of two disjoint subsets, where every edge is between nodes belonging to different sets. For the factor graph describing the generic receiver architecture (Fig. 1 in our example), the first node subset are the factor nodes:

$$V_1 = \{\texttt{dem}, \texttt{dec1}, \texttt{dec2}\} \tag{8}$$

The second subset are the variable nodes:

$$V_2 = \{u, c_1, c_2\}. \tag{9}$$

The complete set of nodes is:

$$V = V_1 \cup V_2 \tag{10}$$

The general set of edges $E \subseteq V \times V$ with the bipartite graph property is

$$E = E_1 \cup E_2 \text{ ; with } E_1 \subseteq V_1 \times V_2 \text{ , } E_2 \subseteq V_2 \times V_1 \tag{11}$$

where the adjacency matrix can be described as:

$$M = \begin{pmatrix} 0 & M_1 \\ M_2 & 0 \end{pmatrix} \tag{12}$$

In the example case the edges are (compare Fig. 1):

$$\begin{aligned} E \;=\; & \{(u, dec1), (c_1, dec1), (u, dec2), (c_2, dec2), \\ & (u, dem), (c_1, dem), (c_2, dem), (dem, u), \\ & (dem, c_1), (dem, c_2), (dec1, u), (dec1, c_1), \\ & (dec2, u), (dec2, c_2)\} \end{aligned} \tag{13}$$

B. *Algorithm notation*

After naming the factor nodes in the last section, now the mapping of an algorithm to a node is described. The set of algorithm abbreviations is

$$A = \{\texttt{ummse}, \texttt{maxlog}, \texttt{bcjr}\} \tag{14}$$

To map an algorithm to a factor node, the abbreviations of node and algorithm are concatenated. Algorithm and factor node must have the same type (e.g. the BCJR algorithm is not applicable for channel estimation). Our example sets of candidate algorithm and factor node names is listed in Tab. II.

The set of valid algorithm mappings to factor nodes forms the alphabet Σ (set of symbols) of the receiver description language:

$$\Sigma = \Big\{ (v\_a) \mid v \in V_1,\ a \in A,\ \text{factor node } v \text{ and algorithm } a \text{ have same type} \Big\} \tag{15}$$

Examples

- `dem_maxlog`: MIMO demapping using the max-log algorithm.
- `dec2_bcjr`: constituent decoder 2 implementing the BCJR algorithm.

	Type	Abbreviation
Algorithm name		
unbiased MMSE	`demapping`	`ummse`
max-log	`demapping`	`maxlog`
BCJR	`decoding`	`bcjr`
Factor node name		
Demapper	`demapping`	`dem`
Decoder 1	`decoding`	`dec1`
Decoder 2	`decoding`	`dec2`

TABLE II
NAMING COMPONENT ALGORITHMS AND FACTOR NODES TO CONSTRUCT ALPHABET FOR RECEIVER DESCRIPTION LANGUAGE.

C. *Schedule notation: regular expression*

A schedule is a valid word from the regular receiver description language $\mathcal{L}$. The language can be defined by a starting set of valid words and 'construction rules' [17].
Starting set:

- $\mathcal{L}(\emptyset)$: 'empty' receiver is in $\mathcal{L}$.
- $\underset{s\in\Sigma}{\forall} \mathcal{L}(s) = s$: only one factor node update

Construction rules:

- $\underset{s,t\in\Sigma}{\forall} \mathcal{L}(s|t) = \mathcal{L}(s) \cup \mathcal{L}(t)$: alternative
- $\mathcal{L}(st) = \{\alpha\beta \mid \alpha \in \mathcal{L}(s) \wedge \beta \in \mathcal{L}(t)\}$: sequence
- $\mathcal{L}(a^*) = \bigcup_{i\geq 0} \mathcal{L}^i(a)$: repetition
- $\mathcal{L}(a^+) = \bigcup_{i\geq 1} \mathcal{L}^i(a)$: nonzero repetition (at least once)

A receiver design space (search space) is a subset $\mathcal{R} \subseteq \mathcal{L}$ and can be given as regular expression.
Example:

- $\mathcal{R}_{\text{ex-1}}$ = (`dem_ummse`)(`dec1_bcjr dec2_bcjr`)$^+$ describes a 'normal' linear receiver with turbo decoder (at least one turbo decoder iteration).

V. DESIGN SPACE ENUMERATION, BRANCH & BOUND GRAPH SEARCH

A design space as set of concatenations of symbols from Σ has graph structure (compare Fig. 3). Graph search algorithms (breadth first or depth first search) produce a search tree [17]. Efficient graph search in a potentially infinite design space uses branch and bound: the cost metric of the first found converging receiver is used to limit the search tree by pruning more expensive subtrees.

A. *Operational SNR versus Receiver Complexity*

This subsection illustrates the automatic receiver optimization by giving optimization results of iterative demapping-decoding for 4x4 QPSK for the target hardware (Cell SPU). Considered are concatenations of the four processing blocks illustrated in Fig. 2. For each SNR the objective is to find the cheapest (least complex) schedule which achieves convergence (defined here as $MI > 0.99$). The following three receiver spaces are considered:

- $\mathcal{R}_{\text{normal}}$ = (`dem_ummse`)(`dec1_bcjr dec2_bcjr`)*
'Normal' receiver with Turbo decoder. The number of Turbo decoder iterations is variable.
- $\mathcal{R}_{\text{round-robin}}$ = (`dem_maxlog dec1_bcjr dec2_bcjr`)*
Iterative MIMO demapping-decoding using the Max-Log demapper and a round-robin schedule.
- $\mathcal{R}_{\text{iterative-1}}$ = (`dem_ummse` | `dem_maxlog` | `dec1_bcjr` | `dec2_bcjr`)*
All possible concatenations of the four blocks are used as input to the optimization.

The receiver space $\mathcal{R}_{\text{iterative-1}}$ is illustrated in Fig. 3 in form of a state transition graph (finite state machine) containing the processing blocks as states.

The optimization consists of predicting performance for each receiver visited in the search tree. For the search space $\mathcal{R}_{\text{iterative-1}}$, the branch-and-bound costs can be initialized with the results from $\mathcal{R}_{\text{round-robin}}$: if there is a converging receiver for this SNR, then there is a converging receiver with round-robin schedule ([18] – although it is probably not the optimal receiver according to the chosen criterion).

Optimization results are shown in Fig. 4:

- standard receivers according to $\mathcal{R}_{\text{normal}}$ are computationally inexpensive, but do not work at very low SNR. In this scenario (4x4 QPSK, Rayleigh fading, rate 1/3), the necessary number of Turbo decoder iterations increases from 1 iteration at 5dB to 10 iterations at 2.1 dB.
- $\mathcal{R}_{\text{round-robin}}$ extends the operational SNR by 1 dB to the low SNR regime, although at largely increasing computational cost.
- optimization among $\mathcal{R}_{\text{iterative-1}}$ reduces the necessary cost compared to $\mathcal{R}_{\text{round-robin}}$ by a factor of 2 with the same operational SNRs. At the same complexity the operational SNR is reduced by around 0.5 dB. The standard receiver architecture $\mathcal{R}_{\text{normal}}$ is included as special case and is result of the optimization for the high SNR regime.

The Pareto-optimal receivers lie on a hyperbola in the operational SNR / complexity diagram. Receivers inside the hyperbola are suboptimal, dominated by the Pareto-optimal ones. Which receiver from the hyperbola is the optimal one depends on the chosen criterion. To the lower SNR region (left in the diagram), the hyperbola is limited by channel capacity (if arbitrary processing is allowed) and transmitter characteristics. The hyperbola may be improved (to the lower left of the diagram) by improving one of the contained APP computation components (towards better accuracy of APP approximation or towards complexity reduction). The optimization process itself does not have to change to include new algorithm developments.

VI. DISCUSSION

The number of factor node updates of the found 'optimal' receivers in the example is 3 for 5dB, 47 for $\mathcal{R}_{\text{round-robin}}$ at 1.1 dB and 56 for $\mathcal{R}_{\text{iterative-1}}$ at 1.1 dB. $\mathcal{R}_{\text{iterative-1}}$ contains

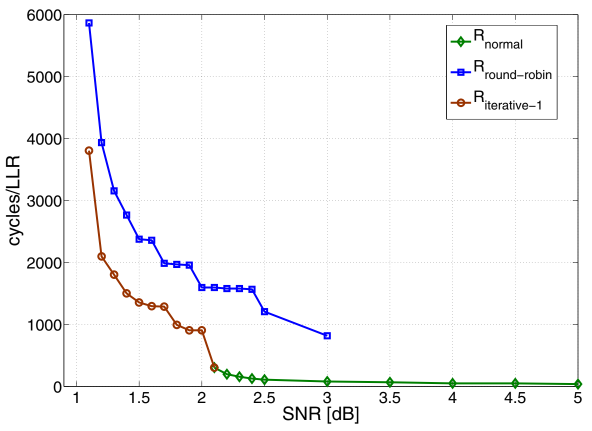

Fig. 4. Pareto-optimal receivers lie on a performance/complexity hyperbola: results for 4x4 QPSK.

$4^{56} > 10^{33}$ receivers of schedule length exactly 56. These numbers clearly show the necessity of an optimization approach based on fast performance prediction, as it would have been unfeasible to obtain results for the presented optimization problem by Monte-Carlo simulation of the receivers. Receiver design spaces further grow exponentially if more algorithm alternatives are considered and if the optimization approach is extended to also include iterative channel estimation algorithms. Apart from applying the presented approach for comparing and optimizing receivers including different component algorithms not covered by our example, we see three main application extensions. Optimization for multi-user MIMO (multiple transmitters and/or multiple codeworts) is also possible, with the following differences: SNRs are different per user/transmitter, leading to different input receive value densities; some processing blocks jointly process signals of the different transmitters (like MIMO demapping), some blocks separately process the signals (like decoding); BER is determined per user/transmitter. The optimization can also be extended to include (possibly iterative) channel estimation. The assumption of Gaussian densities (mean, variance) can not only be applied to model conditional LLR densities, but also to model baseband symbol densities (complex Gaussian densities). A third possible extension is optimization of processing delay. When delay is taken into account in the optimization criterion, parallel processing makes sense (concurrent factor node updates). The presented receiver description language so far only considers serial schedules (sequential updates). For delay optimization, the language therefore needs extension towards parallel schedules.

References

[1] "Pareto efficiency," Wikipedia. [Online]. Available: http://en.wikipedia.org/w/index.php?title=Pareto_efficiency&oldid=376222647

[2] F. Sanzi, S. Jelting, and J. Speidel, "A comparative study of iterative channel estimators for mobile OFDM systems," *IEEE Trans. Wireless Communication*, vol. 2, no. 5, pp. 849–859, Sept. 2003.

[3] A. Ibing, D. Kühling, and H. Boche, "On the relation of MIMO APP detection and SIMO maximum ratio combining," in *IEEE Globecom 2009*.

[4] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Transactions on Communications*, vol. 49, no. 10, pp. 1727–1737, October 2001.

[5] A. Ibing and H. Boche, "On predicting convergence of iterative MIMO detection-decoding with concatenated codes," *IEEE Trans. Veh. Techn.*, vol. 59, no. 8, pp. 4134–4139, 2010.

[6] M. Fu, "Stochastic analysis of turbo decoding," *IEEE Transaction on Information Theory*, vol. 51, no. 1, pp. 81–100, Jan. 2005.

[7] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, 2nd ed. Morgan Kaufmann, 1988.

[8] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[9] F. Kschischang, B. Frey, and H. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Information Theory*, vol. 47, no. 2, Feb. 2001.

[10] R. McEliece, D. MacKay, and J. Cheng, "Turbo decoding as an instance of Pearl's "belief propagation" algorithm," *IEEE JSAC*, vol. 16, no. 2, Feb. 1998.

[11] E. Zimmermann, "Complexity aspects in near-capacity MIMO detection-decoding," Ph.D. dissertation, TU Dresden, 2007.

[12] B. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, March 2003.

[13] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions of Information Theory*, vol. 20, pp. 284–287, March 1974.

[14] *3GPP TSG RAN: TS36.300v8.6.0 E-UTRA and E-UTRAN; Overall Description; Stage 2*, 3GPP, Sept. 2008. [Online]. Available: http://www.3gpp.org/ftp/Specs/html-info/36300.htm

[15] A. Ibing, D. Kühling, M. Kuszak, V. Jungnickel, and C. Helmolt, "Flexible demonstrator platform for cooperative joint transmission and detection in next generation wireless MIMO-OFDM networks," in *Tridentcom 2008*.

[16] A. Ibing, D. Kühling, D. Wieruch, and H. Boche, "Software defined hybrid MMSE/QRD-M turbo receiver for LTE Advanced uplink on a Cell processor," in *IEEE ICC*, 2009.

[17] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[18] F. Brännström, L. Rasmussen, and A. Grant, "Convergence analysis and optimal scheduling for multiple concatenated codes," *IEEE Transactions on Information Theory*, pp. 3354–3364, Sept. 2005.