Channel-Adaptive Coding for Coarsely Quantized MIMO Systems

Thomas Lang, Amine Mezghani and Josef A. Nossek
Institute for Circuit Theory and Signal Processing, Technische Universität München
Theresienstraße 90, 80290 Munich, Germany
Email: {Lang, Mezghani, Nossek}@nws.ei.tum.de

***Abstract*—We present iterative algorithms which efficiently compute an optimized subset of equiprobable input symbols that achieves near-capacity on a large asymmetric discrete memoryless channel (DMC). The optimized subset defines a delay-free channel-adapted inner code which can be combined with a standard code. The introduction of a dense data structure renders it possible to apply channel-adaptive coding to 2-bit quantized MIMO systems, which exhibit a very large output set. The coding approach is shown to be robust to imperfect channel state information (CSI). Finally, we present a binary index switching algorithm that minimizes the BER by optimizing the mapping from code bits to input symbols.**

I. Introduction

In high-speed MIMO communication systems, e.g. Ultra-Wideband (UWB) communications, the use of low resolution analog-to-digital converters (ADCs) simplifies the circuit design and saves power as well as chip area [1]. Additionally, the loss in channel capacity due to 1-bit and 2-bit analog-to-digital conversion is surprisingly small [2]. However, coarse quantization implies that even for zero-noise some transmit symbols are not distinguishable at the receiver after quantization. For instance, in a 1-bit quantized 4×4 Gaussian MIMO system the uncoded BER saturates around 10^{-1} with maximum-likelihood (ML) decoding [3]. Thus, reliable communication with code rates close to the i.u.d. channel capacity $C_{\text{i.u.d.}}$ (which is the mutual information induced by an independently and uniformly distributed source) would necessitate the use of turbo-like codes (Turbo codes, LDPC, etc.) with very long code lengths. Moreover, distribution shapers should be applied, because coarsely quantized Gaussian MIMO channels are very asymmetric in that their capacity C is considerably higher compared to the i.u.d. capacity [4]. Thus, the use of turbo-like codes (which are very effective on any binary input channel and on non-binary symmetric channels [5]) to approach the capacity would involve a considerable average communication delay and a high decoding complexity [6].
In [4] a channel-adaptive coding approach is presented that takes the coarse quantization operation fully into account. The idea is to select a reduced subset of equiprobable input symbols which are well distinguishable at the receiver despite quantization. This channel-adaptive coding approach has several advantages. First of all, the overall system is simplified as no distribution shapers are needed to tightly approach the capacity. Secondly, the reduced symbol packing virtually transforms the very asymmetric DMC into an almost symmetric one. Thirdly, the optimized input symbol selection can be used as an inner delay-free code that can easily be combined with an outer standard code. Thereby, the channel-adapted code simplifies the task of the outer code by reducing the SER while increasing the mutual information. This holds true, if the rate of the inner code is sufficiently high for the given signal-to-noise ratio (SNR) [4]. Though, finding a subset of equiprobable input symbols which achieves near-capacity on large asymmetric DMCs is a non-polynomial (NP) hard problem. The algorithms in [4] solve this optimization problem well, but their computational complexity is quite high.

The main contribution of this paper are efficient algorithms which iteratively solve the symbol selection optimization problem. We demonstrate that the symmetry of the received signal constellation can be exploited to decrease the computational complexity. In MIMO systems with many transmit and receive antennas as well as in MIMO systems with 2-bit ADCs the number of channel transition probabilities becomes a bottleneck. Therefore, we introduce a dense data structure that considers only the significant channel transition probabilities. If the optimized subset of input symbols serves as an inner code that is combined with an outer code, we would like to minimize the bit-error ratio (BER) of the inner code so as to minimize the SER of the combined code. The binary switching algorithm [7] can be used to minimize the BER by optimizing the mapping from code bits to input symbols. We present a binary index switching algorithm which uses a compact formula of the BER to efficiently optimize the bit-to-symbol mapping. Finally, we demonstrate that the proposed channel-adaptive coding approach is robust to imperfect CSI.

The remainder of this paper is organized as follows. Section 2 introduces the system model of quantized MIMO systems. Section 3 describes the symbol selection optimization problem and presents algorithms that solve it efficiently. Section 4 demonstrates the robustness of the coding approach to imperfect CSI. Section 5 deals with the optimization of the bit-to-symbol mapping. Section 6 shows the performance of channel-adaptive coding when combined with convolutional coding. Conclusions are given in Section 7.

II. System Model

We consider the point-to-point quantized MIMO system in Fig. 1. The channel matrix $\boldsymbol{H} \in \mathbb{C}^{N \times M}$ connects the M transmit antennas with the N receive antennas and it

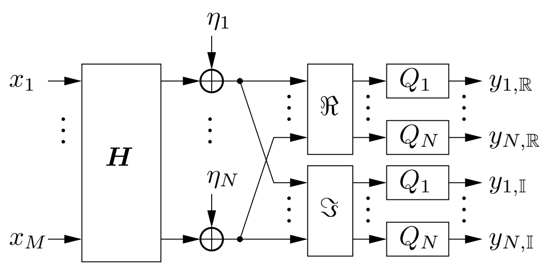

Figure 1. Quantized MIMO System.

is composed of i.i.d. zero-mean complex Gaussian random variables $[\boldsymbol{H}]_{i,j}$ of unit variance. Each source vector $\boldsymbol{x}$ comprises M source symbols x_i which are drawn from a discrete QPSK modulation alphabet. The average transmit energy of $\boldsymbol{x}$ is normalized to one. The unquantized output

$$\boldsymbol{r} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{\eta}, \quad \boldsymbol{r} \in \mathbb{C}^N \tag{1}$$

is perturbed by additive white Gaussian noise (AWGN) $\boldsymbol{\eta}$ with covariance matrix $\mathrm{E}[\boldsymbol{\eta}\boldsymbol{\eta}^{\mathrm{H}}] = \sigma_\eta^2 \mathbf{I}_N$. The signal-to-noise ratio (SNR) is defined by

$$\mathrm{SNR} = \sigma_\eta^{-2}.$$

After reception, the real-part $r_{i,\mathbb{R}} = \Re(r_i)$ and imaginary-part $r_{i,\mathbb{I}} = \Im(r_i)$ of the received signal component r_i are quantized separately by scalar b-bit resolution quantizers

$$y_{i,c} = Q_i(r_{i,c}), \quad c \in \{\mathbb{R}, \mathbb{I}\}, \quad 1 \le i \le N. \tag{2}$$

The overall quantization process is summarized by

$$\boldsymbol{y} = \mathcal{Q}(\boldsymbol{r}), \tag{3}$$

where $\mathcal{Q}(\bullet)$ is a short notation for the operation of the two identical banks of quantizers $Q_1, Q_2, \ldots, Q_N$. In this paper, we consider uniform symmetric quantizers. The quantized receive alphabet for the i-th antenna is given by

$$\mathcal{A}_i = \{\varDelta_i \cdot (k - 2^{b-1} - 2^{-1}) : k = 1, \ldots, 2^b\} \ni y_{i,c}, \tag{4}$$

where $\varDelta_i$ is the quantizer step-size of the i-th antenna. As in [3], we assume that the unquantized received signals $r_{i,c}$ are Gaussian distributed and calculate the step-size as follows

$$\varDelta_i = \varDelta \cdot \sqrt{\mathrm{E}[r_{i,c}^2]}. \tag{5}$$

Here, $\varDelta$ is the optimum output level spacing (with respect to the mean-squared error criterion) for a zero-mean Gaussian source with unit variance [8]. The lower and upper boundaries of the quantization intervals are given by

$$d^{\mathrm{low}}(y_{i,c}) = \begin{cases} y_{i,c} - \frac{\varDelta_i}{2} & \text{for } y_{i,c} \ge -\varDelta_i(2^{b-1} - 1) \\ -\infty & \text{otherwise,} \end{cases} \tag{6}$$

and

$$d^{\mathrm{up}}(y_{i,c}) = \begin{cases} y_{i,c} + \frac{\varDelta_i}{2} & \text{for } y_{i,c} \le \varDelta_i(2^{b-1} - 1) \\ +\infty & \text{otherwise.} \end{cases} \tag{7}$$

In fact, the quantized MIMO system in Fig. 1 represents a DMC with input $\boldsymbol{x} \in \mathcal{X}$, output $\boldsymbol{y} \in \mathcal{Y}$ and channel transition probabilities $\Pr(\boldsymbol{y}|\boldsymbol{x})$. The set $\mathcal{X}$ contains all possible transmit vectors and the set $\mathcal{Y}$ comprises all possible quantized receive vectors. The quantization process is called coarse, if it is not possible to distinguish between all possible transmit vectors at the receiver after quantization even though the noise power is zero [2].
Since the real part $n_{i,\mathbb{R}}$ and imaginary part $n_{i,\mathbb{I}}$ of the noise n_i at the i-th receive antenna are uncorrelated with variance $\sigma_\eta^2/2$, the conditional probability $\Pr(\boldsymbol{y}|\boldsymbol{x})$ factorizes

$$\Pr(\boldsymbol{y}|\boldsymbol{x}) = \prod_{c\in\{\mathbb{R},\mathbb{I}\}} \prod_{i=1}^{N} \phi(\boldsymbol{x}, y_{i,c}), \tag{8}$$

where

$$\begin{aligned} \phi(\boldsymbol{x}, y_{i,c}) &= \Phi\left(\sqrt{2}/\sigma_\eta \cdot (d^{\mathrm{up}}(y_{i,c}) - [\boldsymbol{H}\boldsymbol{x}]_{i,c})\right) \\ &\quad - \Phi\left(\sqrt{2}/\sigma_\eta \cdot (d^{\mathrm{low}}(y_{i,c}) - [\boldsymbol{H}\boldsymbol{x}]_{i,c})\right) \end{aligned} \tag{9}$$

and $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt$ stands for the cumulative normal distribution function.

III. Optimizing the Input Symbol Selection

We assume that the receiver has access to the channel state information (CSI), which can be obtained even with coarse quantization [9]. The receiver optimizes the input symbol selection and feeds its choice back to the transmitter. This requires at most as many bits as the initial number of transmit symbols. If the bit-to-symbol mapping is optimized as well, additional bits (may) have to be fed back.

A. The Optimization Problem

The optimization problem is to find a subset $\mathcal{X}' \subseteq \mathcal{X}$ of equiprobable input symbols $\Pr(\boldsymbol{x}) = 1/|\mathcal{X}'|$ which achieves near-capacity for a given DMC realization. One option is to maximize the mutual information between $\boldsymbol{x}$ and $\boldsymbol{y}$, i.e.,

$$\max_{\mathcal{X}'\subseteq\mathcal{X}} I(X;Y) = \max_{\mathcal{X}'\subseteq\mathcal{X}} \left\{ \log_2(|\mathcal{X}'|) + \frac{1}{|\mathcal{X}'|} \sum\nolimits_{\boldsymbol{x}\in\mathcal{X}'} \alpha_{\boldsymbol{x}} \right\}, \tag{10}$$

where

$$\alpha_{\boldsymbol{x}} = \sum\nolimits_{\boldsymbol{y}\in\mathcal{Y}} \Pr(\boldsymbol{y}|\boldsymbol{x}) \log_2 \left(\frac{\Pr(\boldsymbol{y}|\boldsymbol{x})}{\sum_{\boldsymbol{x}'\in\mathcal{X}'} \Pr(\boldsymbol{y}|\boldsymbol{x}')} \right). \tag{11}$$

As pointed out in [4], it is also possible to use the SER as optimization criterion, that is

$$\min_{\mathcal{X}'\subseteq\mathcal{X}} \mathrm{SER} = 1 - \max_{\mathcal{X}'\subseteq\mathcal{X}} \left\{ \frac{1}{|\mathcal{X}'|} \sum\nolimits_{\boldsymbol{x}\in\mathcal{X}'} \beta_{\boldsymbol{x}} \right\}, \tag{12}$$

where

$$\beta_{\boldsymbol{x}} = \sum\nolimits_{\boldsymbol{y}|\boldsymbol{x} = \arg\max_{\boldsymbol{x}'\in\mathcal{X}'}\{\Pr(\boldsymbol{y}|\boldsymbol{x}')\}} \Pr(\boldsymbol{y}|\boldsymbol{x}). \tag{13}$$

We require the subset size $S = |\mathcal{X}'|$ to be a power of 2 such that source (or code) bits can be mapped directly to input symbols. Hence, for $M = N$ and QPSK modulation the possible code rates R_{i} are

$$R_{\mathrm{i}} \in \{(1/2M), (2/2M), \ldots, 1\}. \tag{14}$$

Algorithm 1 Iterative Capacity Maximization (C-max)

```
1: initialization $\mathcal{X}' = \mathcal{X}$, $S = 4^{R \cdot M}$, $L_{\boldsymbol{y}|\boldsymbol{x}} = \log_2(\Pr(\boldsymbol{y}|\boldsymbol{x}))$
2: while $|\mathcal{X}'| > S$ do
3:     for all $\boldsymbol{y} \in \mathcal{Y}$ do
4:         $L_{\boldsymbol{y}} = \log_2 \left( \sum_{\boldsymbol{x} \in \mathcal{X}'} \Pr(\boldsymbol{y}|\boldsymbol{x}) \right)$
5:     for all $\boldsymbol{x} \in \mathcal{X}'$ do
6:         $\alpha_{\boldsymbol{x}} = \sum_{\boldsymbol{y} \in \mathcal{Y}} \Pr(\boldsymbol{y}|\boldsymbol{x}) \cdot (L_{\boldsymbol{y}|\boldsymbol{x}} - L_{\boldsymbol{y}})$
7:     $\boldsymbol{z} = \arg \min_{\boldsymbol{x} \in \mathcal{X}'} \{\alpha_{\boldsymbol{x}}\}$ and $\mathcal{X}' \leftarrow \mathcal{X}' \setminus \boldsymbol{z}$
```

Algorithm 2 Iterative SER Minimization (SER-min)

```
1: initialization: $\mathcal{X}' = \mathcal{X}$, $S = 4^{R \cdot M}$, $\beta_{\boldsymbol{x}} = 0 \ \forall \boldsymbol{x} \in \mathcal{X}$
2: for all $\boldsymbol{y} \in \mathcal{Y}$ do
3:     $\boldsymbol{x} = \arg \max_{\boldsymbol{x}' \in \mathcal{X}'} \{\Pr(\boldsymbol{y}|\boldsymbol{x}')\}$
4:     $\beta_{\boldsymbol{x}} = \beta_{\boldsymbol{x}} + \Pr(\boldsymbol{y}|\boldsymbol{x})$ and $\mathcal{Y}_{\boldsymbol{x}} \leftarrow \mathcal{Y}_{\boldsymbol{x}} \cup \boldsymbol{y}$
5: while $|\mathcal{X}'| > S$ do
6:     $\boldsymbol{z} = \arg \min_{\boldsymbol{x} \in \mathcal{X}'} \{\beta_{\boldsymbol{x}}\}$ and $\mathcal{X}' \leftarrow \mathcal{X}' \setminus \boldsymbol{z}$
7:     for all $\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{z}}$ do
8:         $\boldsymbol{x} = \arg \max_{\boldsymbol{x}' \in \mathcal{X}'} \{\Pr(\boldsymbol{y}|\boldsymbol{x}')\}$
9:         $\beta_{\boldsymbol{x}} = \beta_{\boldsymbol{x}} + \Pr(\boldsymbol{y}|\boldsymbol{x})$ and $\mathcal{Y}_{\boldsymbol{x}} \leftarrow \mathcal{Y}_{\boldsymbol{x}} \cup \boldsymbol{y}$
```

For a given target code rate R the maximizations in (10) and (12) simplify to

$$\max_{\mathcal{X}' \subseteq \mathcal{X}} \left\{ \sum\nolimits_{\boldsymbol{x} \in \mathcal{X}'} \alpha_{\boldsymbol{x}} \right\} \tag{15}$$

and

$$\max_{\mathcal{X}' \subseteq \mathcal{X}} \left\{ \sum\nolimits_{\boldsymbol{x} \in \mathcal{X}'} \beta_{\boldsymbol{x}} \right\} . \tag{16}$$

In the following subsections, we propose algorithms that efficiently compute almost optimal solutions of (15) and (16).

B. Maximizing the Uniform Capacity

The *iterative capacity maximization* (C-max) algorithm in Algorithm 1 iteratively solves (15). In each iteration the C-max algorithm evaluates the (negative) contribution $\alpha_{\boldsymbol{x}}/|\mathcal{X}'|$ of each individual $\boldsymbol{x} \in \mathcal{X}'$ to the mutual information. Subsequently, the input symbol with the smallest $\alpha_{\boldsymbol{x}}$ is removed from the set of remaining input symbols in line 7. After $(|\mathcal{X}| - S)$ iterations, the algorithm delivers the optimized symbol selection $\mathcal{X}'$.

C. Minimizing the Symbol Error Ratio

The *iterative SER minimization* (SER-min) algorithm in Algorithm 2 successively deletes bad symbols from the input set $\mathcal{X}' \subseteq \mathcal{X}$ until the target code rate R is reached. In each iteration of the while loop, the symbol with the lowest probability of correct detection $\beta_{\boldsymbol{x}}/|\mathcal{X}'|$ is removed from the set of remaining input symbols $\mathcal{X}'$. The list $\mathcal{Y}_{\boldsymbol{x}}$ keeps track of all quantization regions $\boldsymbol{y} \in \mathcal{Y}$ that are assigned to the symbol $\boldsymbol{x} \in \mathcal{X}'$ by the ML decoder. After the deletion of a symbol $\boldsymbol{z}$ only the quantization regions $\boldsymbol{y} \in \mathcal{Y}_{\boldsymbol{z}}$ of the deleted symbol have to be reallocated.

The C-max and SER-min algorithm have in common that the decision to remove a symbol is final. In contrast, a binary switching algorithm has the freedom to correct decisions [4]. In [7], a binary switching algorithm was proposed for optimizing the mapping from source symbols to code symbols. As the complexity of a binary switching algorithm crucially depends on the costs per switch, we choose (16) as optimization criterion.

Algorithm 3 Binary Symbol Switching (BSS)

```
1: initialization: $\mathcal{X}' \subset \mathcal{X}$,  $S = |\mathcal{X}'|$,  $u = 0$,  $\mu = 1000$
2: loop
3:     $\mathcal{X}'' \leftarrow \mathcal{X} \setminus \mathcal{X}'$
4:     $\beta_{\boldsymbol{x}} = 0 \ \forall \boldsymbol{x}$
5:     for all $\boldsymbol{y} \in \mathcal{Y}$ do
6:         $\boldsymbol{x} = \arg \max_{\boldsymbol{x}' \in \mathcal{X}'} \{\Pr(\boldsymbol{y}|\boldsymbol{x}')\}$
7:         $\beta_{\boldsymbol{x}} = \beta_{\boldsymbol{x}} + \Pr(\boldsymbol{y}|\boldsymbol{x})$
8:     Sort $\boldsymbol{x}_k \in \mathcal{X}'$ such that $\beta_{\boldsymbol{x}_1} \leq \beta_{\boldsymbol{x}_2} \leq \ldots \leq \beta_{\boldsymbol{x}_S}$
9:     for $k = 1, \ldots, S$ do
10:        $u = u + 1$
11:        if $u > \mu$ then
12:            return $\mathcal{X}'$ as solution
13:        $\mathcal{X}' \leftarrow \mathcal{X}' \setminus \boldsymbol{x}_k$
14:        for all $\boldsymbol{y} \in \mathcal{Y}$ do
15:            $\lambda_{\boldsymbol{y}} = \max_{\boldsymbol{x}' \in \mathcal{X}'} \{\Pr(\boldsymbol{y}|\boldsymbol{x}')\}$
16:        for all $\boldsymbol{x}'' \in \mathcal{X}''$ do
17:            $\gamma_{\boldsymbol{x}''} = \sum_{\boldsymbol{y} \in \mathcal{Y}} \max\{\lambda_{\boldsymbol{y}} \ , \ \Pr(\boldsymbol{y}|\boldsymbol{x}'')\}$
18:        $\gamma_{\boldsymbol{z}} = \max_{\boldsymbol{x}'' \in \mathcal{X}''} \{\gamma_{\boldsymbol{x}''}\}$
19:        if $\gamma_{\boldsymbol{z}} > \sum_{\boldsymbol{x} \in \mathcal{X}'} \beta_{\boldsymbol{x}} + 10^{-10}$ then
20:            $\mathcal{X}' \leftarrow \mathcal{X}' \cup \arg\{\boldsymbol{z}\}$
21:            goto line 3
22:        else
23:            $\mathcal{X}' \leftarrow \mathcal{X}' \cup \boldsymbol{x}_k$
24:        if $k \equiv S$ then
25:            return $\mathcal{X}'$ as solution
```

The *binary symbol switching* (BSS) algorithm in Algorithm 3 finds a locally optimal symbol selection through systematic symbol switching. Initially, the algorithm sorts the selected input symbols $\boldsymbol{x} \in \mathcal{X}'$ according to their probability of correct detection $\beta_{\boldsymbol{x}}/S$. Then, the algorithm tries to decrease the SER by replacing the worst symbol $\boldsymbol{x}_1$ with a currently unused symbol $\boldsymbol{x}'' \in \mathcal{X}''$. If such a symbol exists, the symbol $\boldsymbol{z} = \arg \max_{\boldsymbol{x}'' \in \mathcal{X}''} \{\gamma_{\boldsymbol{x}''}\}$ which leads to the lowest $\text{SER} = 1 - \gamma_{\boldsymbol{z}}/S$ is added to the reduced subset in line 20 and the algorithm is restarted with the new subset $\mathcal{X}'$. Otherwise, the second worst symbol $\boldsymbol{x}_2$ is tried next, and so on. If no symbol switch further improves the SER, $\mathcal{X}'$ is returned as solution. The variable u ensures that the algorithm terminates after at most $\mu = 1000$ executions of the lines $10 - 25$. For good initial subsets $\mathcal{X}'$, the algorithm is likely to halt at a local optimum after a moderate number of symbol switches.

This argument motivates the application of the BSS algorithm to the optimized symbol selection of the C-max algorithm. The combined algorithm is called *C-max BSS*.

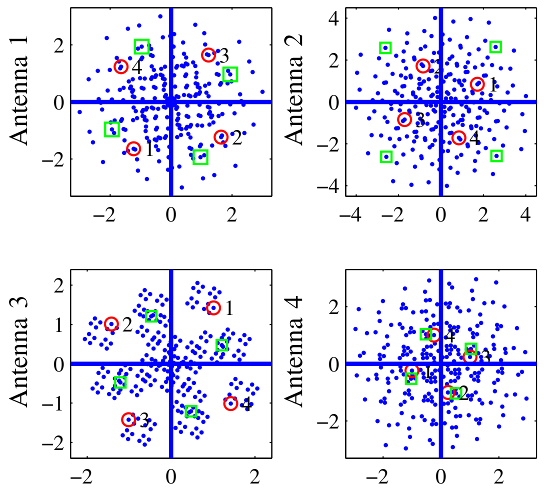

Figure 2. Noise-free received signal constellation. The bold lines through the origin divide the space into 4^N hyper-rectangular quantization regions. The circles (squares) mark a rotational symmetry group. The numbers $1, 2, 3$ and 4 tag the noise-free received signals of one rotational symmetry group.

D. *Exploiting the Received Signal Symmetry*

The symbol selection optimization problems (10) and (12) can be simplified by exploiting the rotational symmetry of the received signal constellation. Due to the QPSK modulation rotating a transmit vector $\boldsymbol{x}$ by $90^\circ, 180^\circ$ or 270° yields another transmit vector $\boldsymbol{x}'$. The noiseless receive vectors $\bar{\boldsymbol{r}} = \boldsymbol{H}\boldsymbol{x}$ and $\bar{\boldsymbol{r}}' = \boldsymbol{H}\boldsymbol{x}'$ exhibit the same rotational relation, that is

$$\boldsymbol{x}' = \alpha \cdot \boldsymbol{x} \;\leftrightarrow\; \bar{\boldsymbol{r}}' = \alpha \cdot \bar{\boldsymbol{r}} \quad \text{for} \quad \alpha \in \{+1, +\mathrm{j}, -1, -\mathrm{j}\}\ . \tag{17}$$

Fig. 2 shows the received signal constellation of the random 4×4 MIMO channel defined by (16) and (17) in [4]. In this figure, the circles (squares) mark a group of four received signals which are interconnected by (17). A rotational symmetry group is characterized completely by one of its four members. Each receive vector of a group exhibits the same distances to all other receive vectors and to the quantization boundaries, i.e., the distance structure of group members matches. Fig. 2 illustrates that receive vectors within one group are very well distinguishable. Hence, the task of selecting a well distinguishable subset of input symbols can be simplified by searching for well distinguishable rotational symmetry groups of four symbols instead. This argument is strong when N and $S = |\mathcal{X}'|$ are large enough.

Due to the symmetry, each $L_{\boldsymbol{y}}$, $L_{\boldsymbol{y}|\boldsymbol{x}}$ and $\beta_{\boldsymbol{x}}$ occurs fourfold in the C-max algorithm in Algorithm 1. The *iterative symmetric capacity maximization* (SC-max) algorithm in Algorithm 4 exploits this fact and executes line 6 only for $\boldsymbol{x} \in \mathcal{X}_s$, where $\mathcal{X}_s \subset \mathcal{X}$ contains one member of each remaining symmetry group. Also, the SC-max algorithm deletes all four symbols of the currently worst group at once. The function $\mathcal{G}(\boldsymbol{z})$ in line 8 delivers the four group members of the argument $\boldsymbol{z}$. As the computational effort is dominated by the calculations in line 6, the overall complexity of the SC-max algorithm is roughly $4 \cdot 4$ times lower compared to the C-max algorithm.

Algorithm 4 Iterative Symmetric Capacity Max. (SC-max)

```
1: initialization $S = 4^{R \cdot M}$, $\mathcal{X}' = \mathcal{X}$,
   $L_{\boldsymbol{y}|\boldsymbol{x}} = \log_2(\Pr(\boldsymbol{y}|\boldsymbol{x}))\ \forall \boldsymbol{x} \in \mathcal{X}_s \subset \mathcal{X}$
   $\mathcal{X}_s$ contains one member of each symmetry group
2: while $|\mathcal{X}_s| > S/4$ do
3:     for all $\boldsymbol{y} \in \mathcal{Y}$ do
4:         $L_{\boldsymbol{y}} = \log_2\left(\sum_{\boldsymbol{x} \in \mathcal{X}'} \Pr(\boldsymbol{y}|\boldsymbol{x})\right)$
5:     for all $\boldsymbol{x} \in \mathcal{X}_s$ do
6:         $\alpha_{\boldsymbol{x}} = \sum_{\boldsymbol{y} \in \mathcal{Y}} \Pr(\boldsymbol{y}|\boldsymbol{x}) \cdot (L_{\boldsymbol{y}|\boldsymbol{x}} - L_{\boldsymbol{y}})$
7:     $\boldsymbol{z} = \arg\min_{\boldsymbol{x} \in \mathcal{X}_s} \{\alpha_{\boldsymbol{x}}\}$ and $\mathcal{X}_s \leftarrow \mathcal{X}_s \setminus \boldsymbol{z}$
8:     $\mathcal{X}' \leftarrow \mathcal{X}' \setminus \mathcal{G}(\boldsymbol{z})$
```

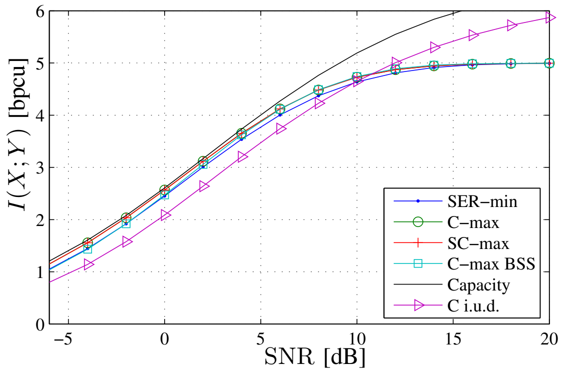

Figure 3. Mutual information increase through optimized selection of $S = 32$ input symbols for 1000 single-bit quantized 4×4 MIMO channels.

E. *Simulation Results for Single-Bit Quantization*

Fig. 3 and Fig. 4 illustrate the performance of the proposed algorithms for a 1-bit quantized QPSK MIMO system with $M = N = 4$ antennas and code rate $R = 5/8$. The subsets are optimized anew for each SNR value. In contrast to [4] the results are averaged over 1000 i.i.d. channel realizations. The selected subsets of the algorithms yield information rates which are well above the i.u.d. capacity and which tightly lower bound the capacity for $C < \log_2(S) - 1$ bits per channel use (bpcu). Remarkably, the SC-max algorithm, which exploits the received signal symmetry, performs practically as well as the C-max algorithm. At higher SNR values the subset selection can be further improved by applying the BSS algorithm to the pre-optimized subset selection of the C-max algorithm. The results for the C-max and C-max BSS algorithm show, that decreasing the SER can result in a lower mutual information at low SNR values.
Finally, the C-max algorithm outperforms the SER-min algorithm over the whole SNR-range for both performance criteria. The following intuitive argument helps to explain this result. At the start of the SER-min algorithm many input symbols exhibit an individual SER of 1, i.e., $\beta_{\boldsymbol{x}} = 0$. Consequently the SER-min algorithm deletes all these symbols at once. In contrast, $\alpha_{\boldsymbol{x}}$ in the C-max algorithm is a measure for the amount of interference that all symbols $\boldsymbol{x}' \in \mathcal{X}' \setminus \boldsymbol{x}$ cause to $\boldsymbol{x}$ and vice versa. Hence, the C-max algorithm is able to distinguish the quality of input symbols with $\beta_{\boldsymbol{x}} = 0$.

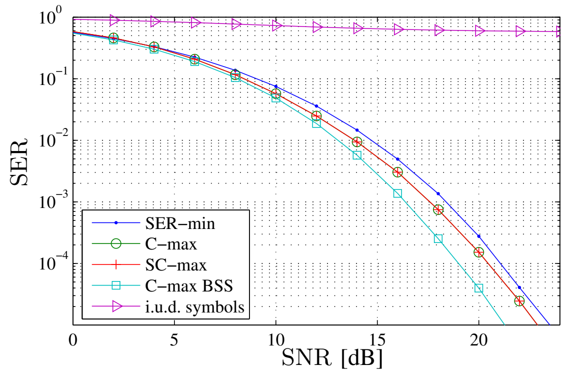

Figure 4. SER decrease through optimized selection of $S = 32$ input symbols for 1000 single-bit quantized 4×4 MIMO channels.

F. Exploiting the Sparse Transition Probability Matrix

So far we assumed that all $|\mathcal{X}| \cdot |\mathcal{Y}|$ entries $\Pr(\boldsymbol{y}|\boldsymbol{x})$ of the transition probability matrix $\boldsymbol{P}_{\boldsymbol{y}|\boldsymbol{x}}$ are known. In practice, $|\mathcal{X}|$ is bounded, as the receiver has to feed the indices of the selected input symbols back to the transmitter. In contrast, $|\mathcal{Y}|$ has no influence on the feedback delay. In a 4×4 MIMO system with 2-bit ADCs the memory requirements for storing $|\mathcal{X}| \cdot |\mathcal{Y}| = 4^4 \cdot 4^{2 \cdot 4} = 16777216$ transition probabilities as well as the computational complexity for the optimization is very high. However, most $\Pr(\boldsymbol{y}|\boldsymbol{x})$ are close to zero at medium to high SNR values and $|\mathcal{Y}| \to \min\{|\mathcal{Y}|, |\mathcal{X}|\}$ for SNR $\to \infty$. Thus, we introduce a dense data structure that captures only the significant channel transition probabilities. To this end, the calculation in (8) is split into two parts using the sets

$$P_c = \left\{ \prod_{i=1}^{N} \phi(\boldsymbol{x}, y_{i,c}) : y_{i,c} \in \mathcal{A}_i, \boldsymbol{x} \in \mathcal{X} \right\}, \; c \in \{\mathbb{R}, \mathbb{I}\}, \quad (18)$$

which satisfy

$$\Pr(\boldsymbol{y}|\boldsymbol{x}) \in \{p_1 \cdot p_2 : p_1 \in P_{\mathbb{R}}, p_2 \in P_{\mathbb{I}}\}. \quad (19)$$

Those elements of the sets $P_{\mathbb{R}}$ and $P_{\mathbb{I}}$ that are larger than some threshold T are sorted according to

$$a_i \in P_{\mathbb{R}}, \quad a_1 \geq a_2 \geq \ldots \geq a_A, \quad a_{A+1} < T, \quad (20)$$

$$b_i \in P_{\mathbb{I}}, \quad b_1 \geq b_2 \geq \ldots \geq b_B, \quad b_{B+1} < T. \quad (21)$$

Clearly, the computational burden for computing and sorting the (relatively small) sets $P_{\mathbb{R}}$ and $P_{\mathbb{I}}$ is negligible compared to the evaluation of all $\Pr(\boldsymbol{y}|\boldsymbol{x})$ via (19).
Our goal is to approximate the sparse $|\mathcal{Y}| \times |\mathcal{X}|$ matrix $\boldsymbol{P}_{\boldsymbol{y}|\boldsymbol{x}}$ with a dense $L \times |\mathcal{X}|$ matrix $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$. The approximation will be precise, if we consider the L largest $\Pr(\boldsymbol{y}|\boldsymbol{x})$ per $\boldsymbol{x} \in \mathcal{X}$ and choose L large enough. Unfortunately, it is not known a priori which $\Pr(\boldsymbol{y}|\boldsymbol{x})$ are large and also it is too complex to calculate and sort all $\Pr(\boldsymbol{y}|\boldsymbol{x})$ for a given $\boldsymbol{x} \in \mathcal{X}$ in order to find out the L largest ones. In Algorithm 5 we pursue a different approach that exploits the sorted vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.
Though, before we explain the details of this algorithm, it is important to explain how we access the data in $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$. Obviously, it is not possible to address an element $\Pr(\boldsymbol{y}|\boldsymbol{x})$ of $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$ by using its position as its (implicit) address within the matrix. In contrast, the vector $\boldsymbol{p}_{\boldsymbol{y}}$ that is composed of all possible sums

$$P_{\boldsymbol{y}} = \sum\nolimits_{\boldsymbol{x} \in \mathcal{X}} \Pr(\boldsymbol{y}|\boldsymbol{x}) \quad (22)$$

is not compressed, and hence the position of $P_{\boldsymbol{y}}$ within $\boldsymbol{p}_{\boldsymbol{y}}$ can be used as the address of $P_{\boldsymbol{y}}$. The index matrix $\boldsymbol{I}_{\boldsymbol{y}|\boldsymbol{x}}$ provides the connection from an entry of $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$ to the respective entry in $\boldsymbol{p}_{\boldsymbol{y}}$. More precisely, the (i,j)-th element of $\boldsymbol{I}_{\boldsymbol{y}|\boldsymbol{x}}$ contains the address of $P_{\boldsymbol{y}}$ within $\boldsymbol{p}_{\boldsymbol{y}}$ for the (i,j)-th element of $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$. The function index$(\boldsymbol{x})$ in line 4 of Algorithm 5 delivers the column index of $\boldsymbol{x}$ within $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$. The function index$(a_k, b_l)$ in line 10 yields the index of the element $P_{\boldsymbol{y}} = \boldsymbol{p}_{\boldsymbol{y}}(\bar{y})$ of which $\Pr(\boldsymbol{y}|\boldsymbol{x}) = a_k \cdot b_l$ is a summand of.
Clearly, the calculation of $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$ would be straightforward, if we were to know the value of the L-th largest $\Pr(\boldsymbol{y}|\boldsymbol{x})$ for each $\boldsymbol{x} \in \mathcal{X}$. Having no a priori information the key idea in Algorithm 5 is to use a threshold array $\boldsymbol{t}$ of length V satisfying

$$t_1 > t_2 > \ldots > t_V. \quad (23)$$

The Algorithm 5 works as follows. In the first iteration of the v-loop in line 5 the algorithm evaluates and stores all $\Pr(\boldsymbol{y}|\boldsymbol{x}) \geq t_1$. Thereby, the algorithm starts by saving all products which satisfy $a_1 \cdot b_l \geq t_1, \; l \in \{1, \ldots, B\}$ to $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$ and to $\boldsymbol{p}_{\boldsymbol{y}}$. If $a_1 \cdot b_l < t_1$ occurs, the algorithm remembers the current position $d_1 = l$ so that the processing of $a_1 \cdot b_l$ can be resumed later on, when $v = 2$. Thereafter, the algorithm computes all products $a_2 \cdot b_l \geq t_1, \; l \in \{1, \ldots, B\}$, and so forth. Finally, the algorithm has evaluated all products $\Pr(\boldsymbol{y}|\boldsymbol{x}) \geq t_1$ and continues with $v = 2$, etc.
If $a_k \cdot b_B \geq t_v$ is fulfilled for some k and v, the algorithm has already saved all products containing a_k, and consequently the k-loop will begin with $k = g + 1$ in the next iteration of the v-loop. At the end of the algorithm, $\boldsymbol{p}_{\boldsymbol{y}}$ and $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$ are normalized. As all probabilities are scaled with the same κ to avoid biasing, $\left(\sum_i \boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}(i, \bar{x})\right) \in [0, \kappa^{-1}]$ with $\kappa^{-1} \geq 1$.

Please note, that the v-loop in line 5 increases the complexity of Algorithm 5 only slightly, because the v-loop accounts at most for $A \cdot V$ executions of the lines $8 - 21$. In contrast, the maximum number of executions of the lines $8 - 21$ equals $A \cdot (B + V)$ and, in general, $B > V$.
The accuracy of the approximation $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$ can be adjusted by modifying L, T and $\boldsymbol{t}$. It is important to mention, that the algorithm finds most of the L largest $\Pr(\boldsymbol{y}|\boldsymbol{x})$ for each $\boldsymbol{x} \in \mathcal{X}$. Indeed, it considers all $\Pr(\boldsymbol{y}|\boldsymbol{x})$ above some threshold t_v plus some (sub-optimally and randomly) chosen values $\Pr(\boldsymbol{y}|\boldsymbol{x}) \geq t_{v+1}$. Also, if T is chosen such that $L > A \cdot B$, some entries of $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$ will remain 0. Clearly, the restriction that L is the same for all $\boldsymbol{x}$ (to obtain a simple dense data structure) is suboptimal, because the number of elements which fulfill $\Pr(\boldsymbol{y}|\boldsymbol{x}) > t_v$ varies considerably for different $\boldsymbol{x}$. Moreover, the performance of the algorithm could be further improved, if it were allowed to dynamically adjust the threshold array $\boldsymbol{t}$ during the algorithm's execution.

Algorithm 1, 2 and 3 can be properly adapted to work with $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$, $\boldsymbol{I}_{\boldsymbol{y}|\boldsymbol{x}}$ and $\boldsymbol{p}_{\boldsymbol{y}}$ instead of $\boldsymbol{P}_{\boldsymbol{y}|\boldsymbol{x}}$. Interestingly, the dense

Algorithm 5 Computation of the Dense Matrix $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$

```
1:  initialization: $\boldsymbol{p}_{\boldsymbol{y}} = \mathbf{0}_{|\mathcal{Y}|\times 1}$, $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}} = \mathbf{0}_{L\times|\mathcal{X}|}$, $\boldsymbol{I}_{\boldsymbol{y}|\boldsymbol{x}} = \mathbf{0}_{L\times|\mathcal{X}|}$, where $\mathbf{0}_{n\times m}$ is a $n \times m$ matrix of zeros.
2:  for all $\boldsymbol{x} \in \mathcal{X}$ do
3:    $i = 0,\ \ g = 1,\ \ d_k = 1$ for $k = 1, \ldots, A$
4:    $\bar{x} = \text{index}(\boldsymbol{x})$
5:    for $v = 1$ to $V$ do
6:      for $k = g$ to $A$ do
7:        for $l = d_k$ to $B$ do
8:          $c = a_k \cdot b_l$
9:          if $c \geq t_v$ then
10:           $\bar{y} = \text{index}(a_k, b_l)$
11:           $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}(i, \bar{x}) = c$
12:           $\boldsymbol{I}_{\boldsymbol{y}|\boldsymbol{x}}(i, \bar{x}) = \bar{y}$
13:           $\boldsymbol{p}_{\boldsymbol{y}}(\bar{y}) = \boldsymbol{p}_{\boldsymbol{y}}(\bar{y}) + c$
14:           $i = i + 1$
15:           if $i = L$ then
16:             continue with the next $\boldsymbol{x}$ in line 2
17:           if $l = B$ then
18:             $g = g + 1$
19:         else
20:           $d_k = l$
21:           continue with $k = k + 1$ in line 5
22: $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}} = \boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}/\kappa,\ \boldsymbol{p}_{\boldsymbol{y}} = \boldsymbol{p}_{\boldsymbol{y}}/\kappa$ with $\kappa = |\mathcal{X}|^{-1}\sum_{\bar{y}} \boldsymbol{p}_{\boldsymbol{y}}(\bar{y})$
```

data structure reveals another means to save complexity in Algorithm 1, 2 and 3. For instance, let us focus on Algorithm 1. At the end of an iteration of the C-max algorithm the symbol $\boldsymbol{z} \in \mathcal{X}'$ with $\alpha_{\boldsymbol{z}} \leq \alpha_{\boldsymbol{x}}, \forall \boldsymbol{x} \in \mathcal{X}'$ is removed from $\mathcal{X}'$. The crucial point is, that not all $\alpha_{\boldsymbol{x}}$ have to be recomputed after this deletion. On the one hand, only those entries $P_{\boldsymbol{y}}$ of $\boldsymbol{p}_{\boldsymbol{y}}$ have to be updated via $P_{\boldsymbol{y}} = P_{\boldsymbol{y}} - \Pr(\boldsymbol{y}|\boldsymbol{z})$ which contain some $\Pr(\boldsymbol{y}|\boldsymbol{z}) \in \boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$. On the other hand, $\alpha_{\boldsymbol{x}}$ has to be updated only for those $\boldsymbol{x} \in \mathcal{X}'$ which fulfill $\mathcal{Y}_{\boldsymbol{z}} \cap \mathcal{Y}_{\boldsymbol{x}} \neq \emptyset$. Here, $\mathcal{Y}_{\boldsymbol{z}}$ denotes the subset $\{\boldsymbol{y} : \Pr(\boldsymbol{y}|\boldsymbol{z}) \in \boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}\} \in \mathcal{Y}$. From a geometric point of view, only those input symbols $\boldsymbol{x} \in \mathcal{X}'$ are affected by the deletion of $\boldsymbol{z}$ which are so close to $\boldsymbol{z}$ that the truncated transition probability mass functions of $\boldsymbol{x}$ and $\boldsymbol{z}$ overlap.
In the following, the *efficient C-max* (EC-max) algorithm shall denote an implementation of the C-max algorithm that both operates on the dense data structure and efficiently updates $P_{\boldsymbol{y}}$ and $\alpha_{\boldsymbol{x}}$ as explained above. The dense data structure combined with the EC-max algorithm renders it possible to handle very large $|\mathcal{Y}|$ as well as quite large $|\mathcal{X}|$. One reason is that the average ratio $\mathrm{E}_{\boldsymbol{x}}[|\mathcal{Y}_{\boldsymbol{x}}|/|\mathcal{Y}|]$ decreases with increasing number of receive antennas N.

G. Simulation Results for Two-Bit Quantization

In this subsection, the performance of the C-max algorithm (which is based on the uncompressed transition probability matrix $\boldsymbol{P}_{\boldsymbol{y}|\boldsymbol{x}}$) is compared to the performance of the EC-max algorithm (which uses the dense transition probability matrix $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$). Similar to subsection III-E, the subsets are optimized

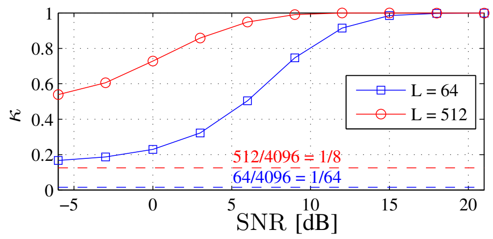

Figure 5. Dependence of the probability normalization factor κ on the SNR and L for 1000 two-bit quantized 4×4 MIMO channels.

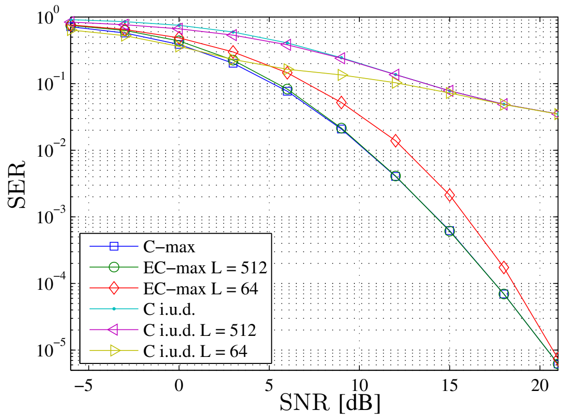

Figure 6. SER decrease through optimized selection of $S = 16$ input symbols for 1000 two-bit quantized 4×4 MIMO channels.

anew for each SNR value. As we want to average the results over 1000 channels and as the complexity of the C-max algorithm is very high for 2-bit quantization and $M = N = 4$ antennas, we simulate 1000 two-bit quantized 3×3 MIMO channels with code rates $R \in \{2/3, 5/6\}$. Fig. 5 indicates the accuracy of the approximation of the sparse $|\mathcal{Y}| \times |\mathcal{X}|$ matrix $\boldsymbol{P}_{\boldsymbol{y}|\boldsymbol{x}}$ by means of the $L \times |\mathcal{X}|$ matrix $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$, because $\kappa = |\mathcal{X}|^{-1}\sum_{\bar{y}} \boldsymbol{p}_{\boldsymbol{y}}(\bar{y})$ equals the proportion of the considered transition probabilities. The dashed lines in Fig. 5 illustrate that $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$ is compressed by the factor $\frac{|\mathcal{Y}|}{L} \in \{8, 64\}$ compared to $\boldsymbol{P}_{\boldsymbol{y}|\boldsymbol{x}}$. Fig. 6, Fig. 7 and Fig. 8 demonstrate that the EC-max algorithm performs nearly as well as the C-max algorithm for high SNR values at $L = 64$ and for medium to high SNR values at $L = 512$. Apparently, the C-max algorithm yields information rates that are well above the i.u.d. capacity and close to the capacity for $C < \log_2(S) - 1$ bits per channel use. Moreover, the curves C i.u.d. $L = 64$ and C i.u.d. $L = 512$ show that the i.u.d. capacity can be tightly approximated at medium to high SNR values by computing the i.u.d. capacity based on $\boldsymbol{B}_{\boldsymbol{y}|\boldsymbol{x}}$. Hence, the dense data structure makes it possible to accurately estimate the i.u.d. capacity of very large DMCs at moderate to high SNR values.

IV. Robustness to an Imperfect CSI

In this section, we demonstrate that the channel-adaptive coding approach is robust to an imperfect *channel state information* (CSI). The imperfect CSI is defined by

$$\boldsymbol{H} = g \cdot \boldsymbol{H}_{\text{unknown}} + \sqrt{1 - g^2} \cdot \boldsymbol{H}_{\text{known}},$$

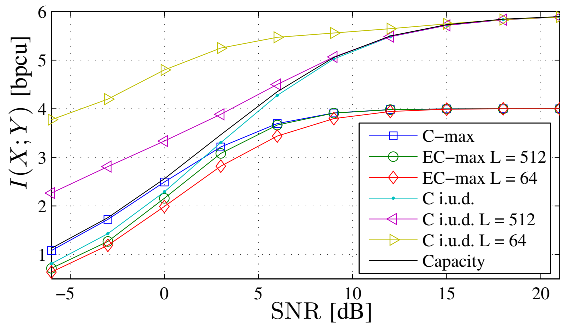

Figure 7. Mutual information increase through optimized selection of $S = 16$ input symbols for 1000 two-bit quantized 4×4 MIMO channels.

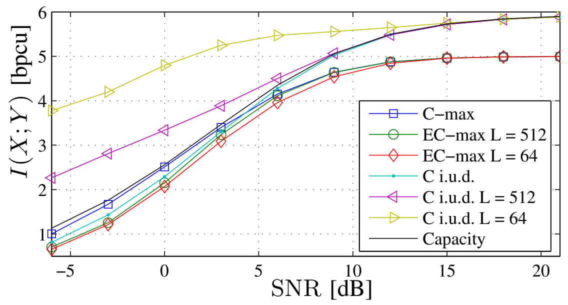

Figure 8. Mutual information increase through optimized selection of $S = 32$ input symbols for 1000 two-bit quantized 4×4 MIMO channels.

where both matrices $\boldsymbol{H}_{\text{unknown}}$ and $\boldsymbol{H}_{\text{known}}$ are composed of i.i.d. zero-mean complex Gaussian random variables of unit variance. Thereby, we assume that the receiver has only access to the matrix $\boldsymbol{H}_{\text{known}}$ as well as to the variance of the unknown channel taps g^2. In order to make the optimized input symbol selection robust against the imperfect CSI, the variance g^2 is factored in by using $\tilde{\sigma}_\eta^2 = \sigma_\eta^2 + g^2$ instead of σ_η^2 as the variance for the calculation of the transition probability matrix. Fig. 9 demonstrates, that the symbol optimization as well as the ML detection are robust, if they are based on the robust DMC estimate $\tilde{\boldsymbol{P}}_{\boldsymbol{y}|\boldsymbol{x}} = \boldsymbol{P}_{\boldsymbol{y}|\boldsymbol{x}}|_{\tilde{\sigma}^2=\sigma^2+g^2}$. In contrast to the previous simulations, the input symbol selection is optimized only once at SNR $=$ 15dB and evaluated over the whole SNR-range. Interestingly, at an SNR of 20dB the input subsets optimized at 20dB in Fig. 4 do not perform better than the input subsets optimized at 15dB with $g = 0$ in Fig. 9. This result supports the claim in [4], that the optimal input symbol selection does not strongly depend on the SNR.

V. Optimizing the Bit-to-Symbol Mapping

In this section, we show how to efficiently minimize the BER by optimizing the 1-to-1 mapping from the code (or source) vectors $\boldsymbol{c} = [c_1, c_2, \dots, c_{\log_2(S)}] \in \mathcal{C} = \{0,1\}^{\log_2(S)}$ to the selected input vectors $\boldsymbol{x} \in \mathcal{X}' \subset \mathcal{X}$ with $S = |\mathcal{X}'|$. The symmetric weight function

$$\mathcal{W}(i,j) = d_H(\boldsymbol{c}_i, \boldsymbol{c}_j) \cdot (\log_2(S))^{-1} \tag{24}$$

delivers the proportion of bits in which the i-th and j-th code vectors $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ differ. Here, $d_H(\boldsymbol{c}_i, \boldsymbol{c}_j)$ is the Hamming

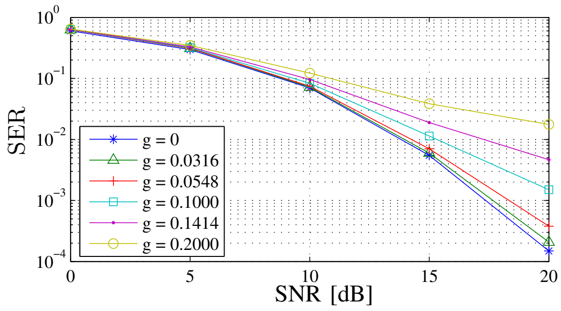

Figure 9. Robustness of the channel-adaptive coding approach for 1000 single-bit quantized 4×4 MIMO channels and different values of g.

distance between $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$. The BER can be expressed as

$$\text{BER} = \sum\nolimits_{\boldsymbol{y}\in\mathcal{Y}} \sum\nolimits_{j\in\{1,\dots,S\}\setminus\hat{i}_{\boldsymbol{y}}} \Pr(\boldsymbol{y}, \boldsymbol{c}_j) \cdot \mathcal{W}(j, \hat{i}_{\boldsymbol{y}}), \tag{25}$$

where

$$\hat{i}_{\boldsymbol{y}} = \arg \max_{i\in\{1,\dots,S\}} \Pr(\boldsymbol{y}, \boldsymbol{c}_i). \tag{26}$$

An optimization of the BER based on equation (25) is quite complex, as $|\mathcal{C}| \cdot |\mathcal{Y}| = S \cdot |\mathcal{Y}|$ products have to be calculated and summarized and as $|\mathcal{Y}| = 4^{b\cdot N}$ can be very large.
We define the following partition of the set of output vectors

$$\mathcal{Y} = \bigcup\nolimits_i \mathcal{Y}_i = \bigcup\nolimits_i \{\boldsymbol{y} \in \mathcal{Y} | \boldsymbol{c}_i = \arg\max_{\boldsymbol{c}\in\mathcal{C}} \Pr(\boldsymbol{y}, \boldsymbol{c})\}, \tag{27}$$

with $\mathcal{Y}_i \cap \mathcal{Y}_j = \emptyset$ for $i \neq j$. The strictly upper triangular matrix $\boldsymbol{U}$ satisfies

$$[\boldsymbol{U}]_{i,j} = \sum\nolimits_{\boldsymbol{y}\in\mathcal{Y}_i} \Pr(\boldsymbol{y}, \boldsymbol{c}_j) + \sum\nolimits_{\boldsymbol{y}\in\mathcal{Y}_j} \Pr(\boldsymbol{y}, \boldsymbol{c}_i) \tag{28}$$

for $i < j$ and $[\boldsymbol{U}]_{i,j} = 0$ for $i \geq j$. Roughly speaking, the value of $[\boldsymbol{U}]_{i,j}$ corresponds to the 'individual SER' that $\boldsymbol{x}_j$ exhibits due to $\boldsymbol{x}_i$ plus the 'individual SER' that $\boldsymbol{x}_i$ exhibits due to $\boldsymbol{x}_j$. The mapping from code bits to input symbols is defined by the permutation vector $\boldsymbol{\pi} = [\pi_1, \pi_2, \dots, \pi_S]$, which is a permutation of the indices $1, 2, \dots, S$ of the selected input symbols $\boldsymbol{x} \in \mathcal{X}'$. Therewith, the BER can be written as

$$\text{BER} = \sum\nolimits_{i=1}^{S} \sum\nolimits_{j=i+1}^{S} [\boldsymbol{U}]_{i,j} \cdot \mathcal{W}(\pi_i, \pi_j). \tag{29}$$

Now, the optimization problem can be formulated as

$$\min_{\boldsymbol{\pi}\in\mathcal{P}} \sum\nolimits_{i=1}^{S} \sum\nolimits_{j=i+1}^{S} [\boldsymbol{U}]_{i,j} \cdot \mathcal{W}(\pi_i, \pi_j), \tag{30}$$

where the set $\mathcal{P}$ contains all possible permutations of the indices $1, 2, \dots, S$.

The *binary index switching* (BIS) algorithm in Algorithm 6 efficiently finds a local optimum of (30). The BIS algorithm avoids to evaluate the quality of a symbol switch with (29). In line $6 - 10$, the BIS algorithm determines the differential BER variation d_b which would result from an exchange of the permutation indices π_a and π_b. The z-loop is performed only four times, as the BIS algorithm was very close to (or already reached) a local optimum after four iterations, in the simulations.

Algorithm 6 Binary Index Switching (BIS)

```
1:  initialization: $\boldsymbol{\pi} = [\pi_1, \pi_2, \dots, \pi_S] = [1, 2, \dots, S]$
    $\mathrm{BER} = \sum_{i=1}^{S} \sum_{j=i+1}^{S} [\boldsymbol{U}]_{i,j} \cdot \mathcal{W}(\pi_i, \pi_j)$
2:  for $z = 1, \dots, 4$ do
3:    $d_i = 0$ for $i = 1, \dots, S$
4:    for $a = 1, \dots, S$ do
5:      for $b = a, \dots, S$ do
6:        for $i = 1, \dots, S$ do
7:          $c_i = \mathcal{W}(\pi_a, \pi_i) - \mathcal{W}(\pi_b, \pi_i)$
8:          $d_b = \sum_{i=1}^{a-1} c_i \cdot ([\boldsymbol{U}]_{i,a} - [\boldsymbol{U}]_{i,b})$
9:              $+ \sum_{i=a+1}^{b-1} c_i \cdot ([\boldsymbol{U}]_{a,i} - [\boldsymbol{U}]_{i,b})$
10:             $+ \sum_{i=b+1}^{S} c_i \cdot ([\boldsymbol{U}]_{a,i} - [\boldsymbol{U}]_{b,i})$
11:       $\hat{d} = \max_{b \in \{a,\dots,S\}} \{d_b\}$
12:       if $\hat{d} > 10^{-14}$ then
13:         $b = \arg(\hat{d})$, $t = \pi_b$
14:         $\pi_b = \pi_a$, $\pi_a = t$
15:         $\mathrm{BER} = \mathrm{BER} - \hat{d}$
```

VI. Channel-Adaptive Coding combined with Convolutional Coding

In this section, we investigate the combination of channel-adaptive coding with convolutional coding[1]. A symbol selection that is optimized using the C-max BSS algorithm serves as a delay-free code of rate R_i. This inner code is combined with a convolutional code of maximum free distance. The convolutional code with constraint length 9 and rate R_o is decoded using a Viterbi decoder with a traceback length of 50 bits. We apply the same decoupled detection/decoding approach as in [2]. The rate of the combined code equals $R = R_\mathrm{i} \cdot R_\mathrm{o}$. The simulation results in Fig. 10 refer to a code rate of $R = 1/2$ and the simulation results in Fig. 11 refer to code rates of $R \in \{1/4, 1/3, 5/8\}$. The symbol selections are optimized at SNR $=$ 10dB in Fig. 11 and at SNR $=$ 15dB in Fig. 10 in order to highlight the robustness of the coding approach to SNR fluctuations. Clearly, the channel-adaptive coding approach is superior to pure convolutional coding. (A similar result was obtained for LDPC codes of block length 250 in [4]). For $R = 1/2$, it is even best to solely use an optimized selection of $S = 16$ input symbols. Moreover, the optimization of the mapping from convolutional code bits to input symbols by means of the BIS algorithm leads to an SNR gain of 0.5dB and 1.3dB for $R = 1/3$ and $R = 1/2$ at $\mathrm{BER} = 10^{-2}$, respectively.

VII. Conclusion

This paper presents various new algorithms with fixed and moderate complexity that compute a uniform input distribution that almost achieves the capacity of coarsely quantized MIMO channels. A new dense data structure reduces the complexity and memory requirements of the algorithms. Moreover, at medium to high SNR levels, this dense data structure renders it possible to accurately estimate the capacity of 2-bit quantized MIMO channels, which exhibit a very large output set. A binary index switching algorithm is derived that efficiently optimizes the mapping from code bits to input symbols. This algorithm is based on a compact formulation of the optimization cost function. Finally, the robustness of the channel-adaptive coding approach to imperfect CSI is demonstrated.

[1]We choose convolutional codes, as they exhibit a low decoding complexity and a small decoding delay.

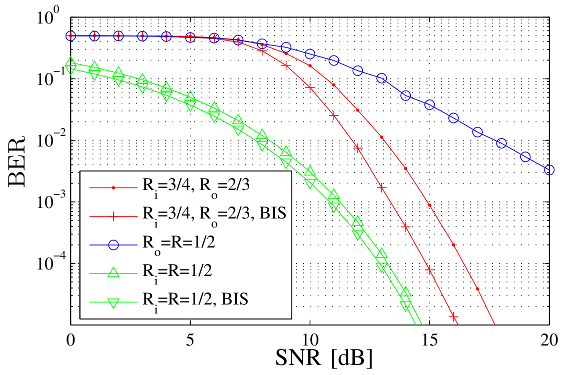

Figure 10. Channel-adaptive and convolutional coding for the single-bit quantized 4×4 MIMO channel specified by (16) and (17) in [4].

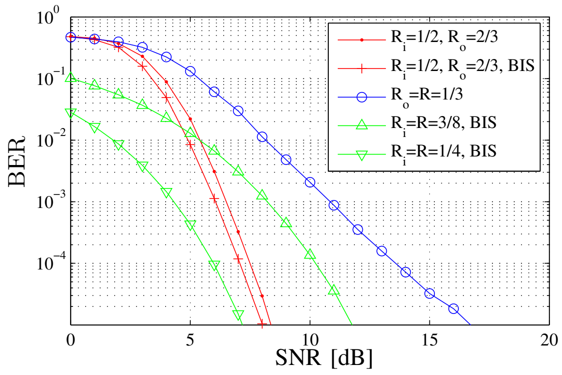

Figure 11. Channel-adaptive and convolutional coding for the single-bit quantized 4×4 MIMO channel specified by (16) and (17) in [4].

References

[1] R. Schreier and G. C. Temes, "Understanding Delta-Sigma Data Converters," *IEEE Computer Society Press*, 2004.

[2] J. A. Nossek and M. T. Ivrlac, "Capacity and Coding for Quantized MIMO Systems," *International Wireless Communication and Mobile Computing Conference (IWCMC)*, pp. 1387–1392, July 2006, (invited).

[3] A. Mezghani, M. S. Khoufi, and J. A. Nossek, "Maximum Likelihood Detection for Quantized MIMO Systems," in *Smart Antennas, 2008. WSA 2008. International ITG Workshop on*, Feb. 26–27, 2008, pp. 278–284.

[4] A. Mezghani, M. T. Ivrlač, and J. A. Nossek, "Achieving near-Capacity on Large Discrete Memoryless Channels with Uniform Distributed Selected Input," *International Symposium on Information Theory and its Applications (ISITA)*, Dec. 2008.

[5] R. J. McEliece, "Are Turbo-like Codes Effective on Nonstandard Channels?," *IEEE Inform. Theory Society Newsletter*, vol. 51, no. 4, pp. 1–8, Dec. 2001.

[6] A. Bennatan and D. Burshtein, "Design and Analysis of Nonbinary LDPC Codes for Arbitrary Discrete-memoryless Channels," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 549–583, Feb. 2006.

[7] K. Zeger and A. Gersho, "Pseudo-Gray Coding," *IEEE Transactions on Communications*, vol. 38, pp. 1100–2147, Dec. 1990.

[8] J. Max, "Quantizing for Minimum Distortion," *IRE Transactions on Information Theory*, vol. 6, no. 1, pp. 7–12, Mar. 1960.

[9] J. A. Nossek and M. T. Ivrlac, "On MIMO Channel Estimation with Single-Bit Signal-Quantization," in *Smart Antennas, 2007. WSA 2007. International ITG Workshop on*, Feb. 26–27, 2007.