

# Efficient Soft Demodulation of MIMO QPSK via Semidefinite Relaxation

Mehran Nekuii, Mikalai Kisialiou, Timothy N. Davidson, and Zhi-Quan Luo

## Abstract

Two efficient list-based ‘soft’ demodulators are developed for iterative receivers in multiple-input multiple-output (MIMO) communication systems with QPSK signalling. The proposed demodulators are based on the semidefinite relaxation (SDR) technique, and hence their computational costs are bounded by a low-order polynomial of the number of bits transmitted per channel use. The first demodulator applies the SDR technique once per demodulation-decoding iteration, and generates list members via the randomization procedure that is inherent in the SDR technique. The second demodulator is based on an approximation of that randomization procedure by a set of independent Bernoulli trials, and this approximation reduces the number of semidefinite programs that need to be solved to just one per channel use. List-free implementations of the proposed demodulators are also developed, and these implementations reduce the memory requirements of the demodulators. Analysis and simulation results show that the proposed demodulators offer an attractive trade-off between performance and computational cost. In particular, in the scenarios that we consider, the second of the proposed demodulators provides error rates that are lower than those of the minimum mean square error soft interference canceller (MMSE-SIC) and close to those of the list sphere decoder, and does so at a significantly lower computational cost.

This work was supported in part by NSERC and the NSF (DMS-0610037). The work of the third author is also supported by the Canada Research Chairs program. M. Nekuii and T. N. Davidson are with the Department of Electrical and Computer Engineering, McMaster University, Hamilton, Ontario, Canada, (nekuiim,davidson@mcmaster.ca). M. Kisialiou was with and Z.-Q. Luo is with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN, (kisi0004,luozq@umn.edu). M. Kisialiou is now with Intel Corporation, Portland, OR. Preliminary versions of portions of this work appear in *Proc. IEEE Wkshp Signal Processing Advances in Wireless Commun.*, 2007, and will appear in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 2008.

## I. INTRODUCTION

The provision of multiple antennas at both the transmitter and receiver of a wireless communication system offers the potential for reliable communication at data rates that are substantially higher than those of single antenna systems [1]. One of the core challenges in the design of such multiple-input multiple-output (MIMO) systems is to obtain good performance at high data rates without incurring unreasonable computational cost. A popular transceiver architecture for moving toward that goal is that of MIMO bit-interleaved coded modulation (BICM) with iterative “soft” demodulation and decoding (IDD), e.g., [2]; see also Fig. 1. In this paper we will focus on narrowband multiple antenna systems, and when configured for such systems, the MIMO-BICM transmitter encodes the information bits using an outer binary code, interleaves the encoder output and then maps blocks of the interleaved codeword to points on a (space-time) matrix constellation for transmission over the channel. At the receiver, the demodulator and outer decoder iteratively exchange the extrinsic components of their estimates of the posterior log likelihood-ratio (LLR) of each encoded bit, until a hard decision is taken. Although this IDD scheme has many desirable features, the computational cost of the (exact) soft demodulator increases exponentially with the number of (encoded) bits transmitted per channel use, and hence there has been considerable interest in the development of approximate soft demodulation schemes with lower complexity; e.g., [2]–[10].

One approach to lower-complexity soft demodulation is to apply the so-called “max-log” approximation [11], under which the LLR of each bit is approximated by the difference between the optimal values of a pair of “hard” demodulation problems; e.g., [7]–[9]. However, each of these hard demodulation problems is also hard in the NP sense. Tree search methods (e.g., [12]), such as sphere decoding (e.g., [13], [14]), can be used to find optimal solutions to these problems (e.g., [7], [8]), but both the average and worst-case computational costs remain exponential in the problem size [15], and the “tail” of the distribution of the computational cost can be quite significant at low SNRs or for large problem sizes; e.g., [16]. As an alternative, semidefinite relaxation methods (e.g., [17], [18]) can be used to efficiently generate approximate solutions to the hard demodulation problems [9], and these methods have the advantage that the growth of the computational cost is bounded by a low-order polynomial in the problem size. However, the number of semidefinite programs that must be solved in each demodulation-decoding iteration grows linearly in the number of (encoded) bits transmitted per channel use.

Another approach to approximate soft demodulation is to apply the principles of list decoding, in which one seeks to efficiently identify a list of bit-vectors that dominate the LLRs; e.g., [2]–[6]. The LLRs can then be approximated by marginalizing over the list. Most of the existing techniques are based on the

use of tree search algorithms to identify members of the list (e.g., [2]–[6]), and hence can be rather computationally expensive, especially at SNRs close to the (ergodic capacity) threshold for the chosen rate. In some list demodulation schemes (e.g., [2]) the list for each channel use is generated once, in the first demodulation-decoding iteration, and is stored for use in the subsequent iterations. This may require substantial memory resources. In other schemes (e.g., [3]–[6]), the list for each channel use is regenerated in each demodulation-decoding iteration, but the memory requirements can still be significant.

In this paper we develop a semidefinite relaxation (SDR) approach to list-based soft demodulation, and propose two new demodulators, both of which regenerate the list in each demodulation-decoding iteration. The first demodulator applies the semidefinite relaxation technique once per demodulation-decoding iteration, and generates list members via the randomization procedure that is inherent in SDR techniques [17], [18]. The second demodulator is based on an approximation of this randomization procedure by a set of independent Bernoulli trials. This approximation allows us to reduce the number of semidefinite programs to be solved to just one per channel use. Furthermore, we develop a list-free implementation of both demodulators that reduces the memory resources required for implementation. Our simulation results will show that the resulting computational and memory efficiencies are obtained without incurring a significant degradation in performance.

The paper is organized as follows. In Section II we provide an overview of the MIMO-BICM-IDD system, and in Section III we review the SDR approach to hard demodulation of MIMO QPSK. In Sections IV and V we develop the proposed demodulators, which we will call the List-SDR and Single-SDR demodulators, respectively. In Section VI we describe the list-free implementation of the proposed methods, and in Section VII the computational cost of implementing the demodulators is analyzed. The results of simulation experiments that compare the performance and computational cost of the proposed demodulators against those of several existing demodulators will be presented in Section VIII.

## II. SYSTEM MODEL AND ITERATIVE RECEIVER

We consider a narrowband multiple antenna system with  $N_t$  transmit antennas and  $N_r$  receive antennas. If we let  $\mathbf{s}_n$  denote the signal vector transmitted at the  $n$ th channel use, the corresponding received signal vector can be written as

$$\mathbf{y}_n = \mathbf{H}_n \mathbf{s}_n + \mathbf{v}_n, \quad (1)$$

where  $\mathbf{H}_n$  is the  $N_r \times N_t$  matrix of channel coefficients and is assumed to be known at the receiver, and  $\mathbf{v}_n$  is a vector of additive white circular complex Gaussian noise samples with variance  $\sigma^2$  per real dimension.

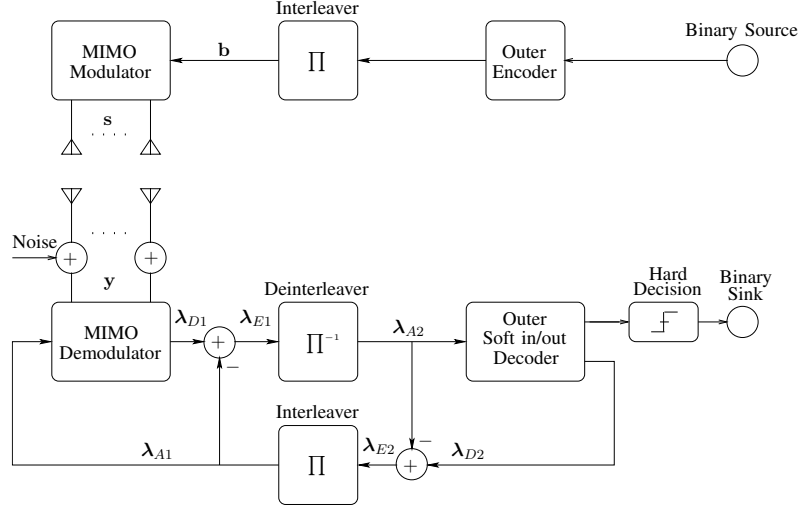


Fig. 1. MIMO BICM-IDD transceiver.

We will consider a MIMO-BICM-IDD transceiver for this system, e.g., [2]; see Fig. 1. For simplicity, we will focus on V-BLAST transmission [19], but by using the equivalent channel concept in [20], the proposed demodulators extend directly to signalling schemes based on general linear dispersion codes. We will let  $\mathbf{b}_n$  denote the sub-block of the interleaved outer codeword that is to be transmitted in the  $n$ th channel use, and we will let  $\mathcal{M}(\mathbf{b})$  denote the mapping used by the MIMO modulator; i.e.,  $\mathbf{s}_n = \mathcal{M}(\mathbf{b}_n)$ . We will consider systems in which this mapping is to QPSK symbols, and we will let  $N$  denote the number of channel uses required to transmit one codeword of the outer code.

At the receiver, the soft MIMO demodulator and the soft-input soft-output outer decoder iteratively exchange (extrinsic) information regarding the bit probabilities, using the log likelihood-ratio (LLR) format; see Fig. 1. This iterative process is based on the turbo principle [21], and since the channel in (1) is memoryless, the role of the soft demodulator is to compute (or approximate) the posterior LLR of each element of the block of the interleaved outer codeword that is transmitted in a given channel use, under the assumption that these blocks are independent from each other. That is, for the  $i$ th element of the bit-vector transmitted in the  $n$ th channel use,  $b_{n,i}$ , the soft demodulator computes (e.g., [2])

$$\lambda_{D1,n,i} = \log \frac{P\{b_{n,i} = +1 | \mathbf{y}_n\}}{P\{b_{n,i} = -1 | \mathbf{y}_n\}}, \quad (2)$$

where, for notational simplicity, we have not explicitly stated the conditioning on  $\mathbf{H}_n$  that is inherent in coherent demodulation. Once these LLRs have been computed for each of the  $N$  channel uses required to transmit a codeword of the outer code, they are passed through the de-interleaver to the outer decoder.

Since the focus of this paper is on the soft demodulator, we will drop the subscript  $n$  in (1) and (2) and will consider a generic channel use. Applying Bayes' Rule enables us to rewrite (2) as

$$\lambda_{D1,i} = \log \frac{\sum_{\mathcal{L}_{i,+1}} p(\mathbf{y}|\mathbf{b})p(\mathbf{b})}{\sum_{\mathcal{L}_{i,-1}} p(\mathbf{y}|\mathbf{b})p(\mathbf{b})}, \quad (3)$$

where  $\mathcal{L} = \{\mathbf{b} \in \{-1, +1\}^{2N_t}\}$  denotes the (complete) list of possible transmitted bit-vectors and  $\mathcal{L}_{i,\pm 1} = \{\mathbf{b} \in \mathcal{L} | b_i = \pm 1\}$ . Given the model in (1) and the fact that we are employing coherent demodulation, the likelihood function  $p(\mathbf{y}|\mathbf{b})$  in (3) satisfies

$$p(\mathbf{y}|\mathbf{b}) \propto \exp(-\|\mathbf{y} - \mathbf{H}\mathcal{M}(\mathbf{b})\|^2/(2\sigma^2)).$$

Furthermore, an estimate of the prior probability  $p(\mathbf{b})$  can be obtained from the output of the previous iteration of the decoder, by assuming that the elements of  $\mathbf{b}$  are independent; i.e.,  $p(\mathbf{b}) \approx \prod_i p(b_i)$ . If we define  $\lambda_{A1,i} = \log\left(\frac{p(b_i=+1)}{p(b_i=-1)}\right)$ , so that  $\boldsymbol{\lambda}_{A1}$  denotes the vector that represents these probabilities in LLR form, then  $\prod_i p(b_i) \propto \exp(\boldsymbol{\lambda}_{A1}^T \mathbf{b}/2)$ . Using these expressions, the LLR in (3) can be written as (e.g., [2])

$$\lambda_{D1,i} \simeq \log \frac{\sum_{\mathcal{L}_{i,+1}} \exp(-D(\mathbf{b})/(2\sigma^2))}{\sum_{\mathcal{L}_{i,-1}} \exp(-D(\mathbf{b})/(2\sigma^2))}, \quad (4)$$

where

$$D(\mathbf{b}) \triangleq \|\mathbf{y} - \mathbf{H}\mathcal{M}(\mathbf{b})\|_2^2 - \sigma^2 \boldsymbol{\lambda}_{A1}^T \mathbf{b}. \quad (5)$$

The number of elements in the lists  $\mathcal{L}_{i,\pm 1}$  in (4) grows exponentially as the number of (encoded) bits transmitted per channel use increases, and hence so does the computational cost of the demodulator. Many of the existing approaches to reducing this computational cost employ one of the following approximations:

$$\lambda_{D1,i} \approx \log \frac{\sum_{\hat{\mathcal{L}}_{i,+1}} \exp(-D(\mathbf{b})/(2\sigma^2))}{\sum_{\hat{\mathcal{L}}_{i,-1}} \exp(-D(\mathbf{b})/(2\sigma^2))} \quad (6)$$

$$\approx \frac{1}{2\sigma^2} \left( \min_{\mathbf{b} \in \hat{\mathcal{L}}_{i,-1}} D(\mathbf{b}) - \min_{\mathbf{b} \in \hat{\mathcal{L}}_{i,+1}} D(\mathbf{b}) \right), \quad (7)$$

where  $\hat{\mathcal{L}} \subseteq \mathcal{L}$ . Each of these equations reveals a class of approximate soft MIMO demodulators.

The first class is based on selecting  $\hat{\mathcal{L}} = \mathcal{L}$  and solving the two binary quadratic optimization problems in (7) for each encoded bit. Solutions to these ‘‘hard’’ demodulation problems can be obtained using tree-search algorithms, as they are in [7], [8], but, as mentioned in the Introduction, these algorithms can be rather computationally expensive. Alternatively, approximate solutions can be found in polynomial time using the semidefinite relaxation technique [9].

The second class of soft demodulators is based on efficiently selecting a list  $\hat{\mathcal{L}}$  of bit-vectors that generate small values for  $D(\mathbf{b})$  and then approximating the LLR either by marginalizing over  $\hat{\mathcal{L}}_{i,\pm 1}$  in (6), e.g., [4], or by performing an exhaustive search over  $\hat{\mathcal{L}}_{i,\pm 1}$  to solve the minimization problems in (7); e.g., [2]. The key challenge in this class of methods is the efficient selection of the members of  $\hat{\mathcal{L}}$ . Most approaches are based on tree-search ideas, and hence are potentially rather computationally expensive; e.g., [2]–[6]. In addition, as we will explain in Section VI, these methods may require significant memory resources in order to store the generated list members.

One important approximate soft demodulator that does not fall into one of these two classes is that based on the minimum mean square error soft interference canceller (MMSE-SIC) in [22]; see also [23]–[25]. For each transmitted symbol, this demodulator first forms the unbiased conditional MMSE estimate of the symbol, where the conditioning is on the probabilities of the other symbols transmitted in the channel use of interest. The demodulator then approximates the residual interference by a Gaussian random variable, and computes the LLR of each bit in the symbol as if the channel was a scalar additive white Gaussian noise channel. The conditional MMSE estimate has a canonical decomposition into two steps: subtraction of the mean of the interfering symbols, followed by (unbiased) linear MMSE estimation of the relevant symbol, and hence the moniker MMSE-SIC.

The two demodulators proposed in this paper fall into the second class of approximate soft demodulators, but they are based on semidefinite relaxation rather than a tree search, and hence their computational cost is bounded by a low-order polynomial of the problem size. Furthermore, the second of these demodulators requires the solution of only one semidefinite program per channel use, and both demodulators can be implemented without explicitly storing the list of candidate bit-vectors. Both the proposed demodulators exploit the properties of the randomization step that is inherent in the approximation of the solution of a binary quadratic problem by semidefinite relaxation, and before we introduce those demodulators we will provide a brief overview of the application of the semidefinite relaxation technique [17], [18] to hard demodulation [26], [27].

### III. HARD DEMODULATION USING SDR

Consider the real-valued equivalent representation for (1) with QPSK signalling,

$$\tilde{\mathbf{y}} = \tilde{\mathbf{H}}\mathbf{b} + \tilde{\mathbf{v}}, \quad (8)$$

where  $\tilde{\mathbf{y}}$  and  $\tilde{\mathbf{v}}$  are the concatenations of the real and imaginary parts of  $\mathbf{y}$  and  $\mathbf{v}$ , respectively, and we have considered an arbitrary channel use. Given prior information on the bit probabilities in the form of

$\lambda_{A1}$  in (5), the bit-vector  $\mathbf{b}$  that maximizes the *a posteriori* probability is the solution to the following binary optimization problem:

$$\min_{\mathbf{b} \in \{+1, -1\}^{2N_t}} D(\mathbf{b}) = \min_{\mathbf{b} \in \{+1, -1\}^{2N_t}} \|\tilde{\mathbf{y}} - \tilde{\mathbf{H}}\mathbf{b}\|_2^2 - \sigma^2 \boldsymbol{\lambda}_{A1}^T \mathbf{b}. \quad (9)$$

Using the definitions [9], [26]

$$\tilde{\mathbf{b}} \triangleq \begin{bmatrix} \check{\mathbf{b}} \\ c \end{bmatrix}, \quad \check{\mathbf{b}} \triangleq c\mathbf{b}, \quad \mathbf{Q} \triangleq \begin{bmatrix} \tilde{\mathbf{H}}^T \tilde{\mathbf{H}} & \mathbf{a} \\ \mathbf{a}^T & 0 \end{bmatrix}, \quad \mathbf{a} \triangleq -\tilde{\mathbf{H}}^T \tilde{\mathbf{y}} - 0.5\sigma^2 \boldsymbol{\lambda}_{A1}, \quad (10)$$

in which  $c \in \{+1, -1\}$ , the problem in (9) can be stated as the following (NP-hard) binary quadratic programming (BQP) problem:

$$\min_{\tilde{\mathbf{b}} \in \{+1, -1\}^{2N_t+1}} \tilde{\mathbf{b}}^T \mathbf{Q} \tilde{\mathbf{b}}. \quad (11)$$

Using the substitution  $\mathbf{X} = \tilde{\mathbf{b}}\tilde{\mathbf{b}}^T$ , the problem in (11) can be reformulated as

$$\min_{\mathbf{X}} \text{Trace}(\mathbf{X}\mathbf{Q}) \quad (12a)$$

$$\text{s.t. } \mathbf{X} \succeq \mathbf{0}, \quad \text{rank}(\mathbf{X}) = 1, \quad (12b)$$

$$[\mathbf{X}]_{ii} = 1, \quad i = 1, \dots, 2N_t + 1, \quad (12c)$$

in which the computational difficulties manifest themselves in the rank-1 constraint. The semidefinite relaxation approach to approximating the solution to (11) is to relax the rank-1 constraint and solve the following semidefinite program (SDP):

$$\min_{\mathbf{X}} \text{Trace}(\mathbf{X}\mathbf{Q}) \quad (13a)$$

$$\text{s.t. } \mathbf{X} \succeq \mathbf{0}, \quad (13b)$$

$$[\mathbf{X}]_{ii} = 1, \quad i = 1, \dots, 2N_t + 1. \quad (13c)$$

This problem is convex and can be efficiently solved using the interior point method in [28]; see Section VII. (See also [29] for some recent developments.) When  $\mathbf{X}_{\text{opt}}$ , the optimal solution to (13), is rank 1, its factorization generates an optimum solution to (11). In the more common event that the solution to (13) is not rank 1, a randomization procedure [17], [18] can be used to extract an approximation of the solution to (11) from  $\mathbf{X}_{\text{opt}}$ . That procedure involves the construction of a (Cholesky) factor  $\mathbf{V}$  of  $\mathbf{X}_{\text{opt}}$ , (i.e.,  $\mathbf{X}_{\text{opt}} = \mathbf{V}^T \mathbf{V}$ ), and the generation of a sequence of random vectors  $\mathbf{u}$  from the uniform distribution on the unit hypersphere. For each vector  $\mathbf{u}$  we compute  $\tilde{\mathbf{x}} = \text{sign}(\mathbf{V}^T \mathbf{u})$ , construct the vector  $\mathbf{x} = \tilde{x}_{2N_t+1} \times [\tilde{x}_1, \dots, \tilde{x}_{2N_t}]^T$ , and compute  $D(\mathbf{x})$  using (5). If this value of  $D(\mathbf{x})$  is smaller than the smallest encountered in the previous steps, then  $\mathbf{x}$  is retained as  $\mathbf{b}_{\text{sdr}}$ , the current approximation of the

solution to (11). A key feature of the SDR approach is that even for the worst-case channel, the expected value of  $D(\mathbf{x})$  over the randomizations is guaranteed to be within a (reasonably small) constant factor of the optimal value of (11), independent of the number of bits to be detected [18]. Furthermore, since each choice of the random vector  $\mathbf{u}$  is made independently, the probability that  $D(\mathbf{b}_{\text{sdr}})$  is higher than the expected value of  $D(\mathbf{x})$  decreases exponentially with the number of randomization iterations [18].

In [9], Steingrímsson *et al* developed a soft MIMO demodulator from the first class in Section II that was based on the semidefinite relaxation technique described above. For each channel use, that demodulator solves  $2N_t + 1$  SDPs per demodulation-decoding iteration, and hence we will call that scheme the “multi-SDR” method. In the next section, we will propose a list-based soft demodulation scheme that requires the solution to just one SDP in each demodulation-decoding iteration for each channel use, and in Section V we will propose a scheme that requires the solution of only one SDP per channel use.

#### IV. LIST-SDR METHOD FOR SOFT DEMODULATION

One of the properties of the SDR approach to hard demodulation is that, on average, the bit-vectors generated by the randomization procedure yield small values for the objective in (9). This suggests that, on average, those bit-vectors are good candidates for membership of the list in a list-based approach to soft MIMO demodulation, and this is the essence of the proposed approach. We will construct a preliminary list  $\hat{\mathcal{L}}'$  by simply storing each (unique) bit-vector generated by the randomization procedure. Since it is possible that there may be bit positions for which  $\hat{\mathcal{L}}'_{i,+1}$  or  $\hat{\mathcal{L}}'_{i,-1}$  is empty, once the randomizations have been completed we will construct an enriched list  $\hat{\mathcal{L}}$  consisting of  $\hat{\mathcal{L}}'$  plus all those bit-vectors with Hamming distance of 1 of the bit-vectors in  $\hat{\mathcal{L}}'$ . (This enrichment is based on ideas in [10] and can be implemented by “flipping” individual bits of each element of  $\hat{\mathcal{L}}'$ .) Once this enriched list has been constructed, we adopt the standard list-based approach to approximate the soft information using (6) or (7) over the constructed list  $\hat{\mathcal{L}}$ . Since the computational cost of this enumeration grows linearly with the cardinality of  $\hat{\mathcal{L}}$ , one may wish to bound the cardinality of  $\hat{\mathcal{L}}'$ , and to use this bound to enable early termination of the randomization procedure should  $\hat{\mathcal{L}}'$  be sufficiently rich. The resulting list generation algorithm is presented in Table I. Since the *a priori* information  $\lambda_{A1}$  is updated in each demodulation-decoding iteration the cost function  $D(\mathbf{b})$  in (5) changes, and hence so does  $\mathbf{X}_{\text{opt}}$ . Therefore, for each iteration we regenerate the list using the procedure in Table I.

TABLE I  
LIST GENERATION COMPONENT OF THE LIST-SDR ALGORITHM.

- 
- Data:  $\mathbf{X}_{\text{opt}}$ , the solution to (13), or an approximation thereof.
  - Parameters:  $M$ , the number of randomization iterations;  $K$ , the maximum size of the preliminary list.
  - Output:  $\hat{\mathcal{L}}$ , the enriched list.
1. Initialize  $\hat{\mathcal{L}}'$  and  $\hat{\mathcal{L}}$  empty,  $m = 0$ , and  $k = 0$ .
  2. Compute a (Cholesky) factor  $\mathbf{V}$  of  $\mathbf{X}_{\text{opt}}$  such that  $\mathbf{X}_{\text{opt}} = \mathbf{V}^T \mathbf{V}$ .
  3. Choose a random vector  $\mathbf{u}$  from the uniform distribution on the unit sphere.
  4. Compute  $\tilde{\mathbf{x}} = \text{sign}(\mathbf{V}^T \mathbf{u})$  and increment  $m$ .
  5. Construct  $\mathbf{x} = \tilde{x}_{2N_t+1} \times [\tilde{x}_1, \dots, \tilde{x}_{2N_t}]^T$ . If  $\mathbf{x}$  is not in  $\hat{\mathcal{L}}'$ , add it to  $\hat{\mathcal{L}}'$  and increment  $k$ .
  6. If  $k < K$  and  $m < M$ , return to 3.
  7. Construct  $\hat{\mathcal{L}}$  as the union of  $\hat{\mathcal{L}}'$  and all the single bit-flippings of the bit-vectors in  $\hat{\mathcal{L}}'$ .
- 

## V. SOFT DEMODULATION USING SINGLE SDR

An interesting property of the SDR approach to approximating the solution to a binary quadratic problem is that an analytic expression can be obtained for the mean value of each element of the candidate bit-vectors  $\mathbf{x}$  that are generated by the randomization procedure described in Section III. The mean value of the  $i$ th element can be computed by using the fact that if the inner products of the random vector  $\mathbf{u}$  with columns  $\mathbf{v}_i$  and  $\mathbf{v}_{2N_t+1}$  of the Cholesky factor  $\mathbf{V}$  have the same sign then  $x_i = +1$ , otherwise  $x_i = -1$ ; cf. [17], [18]. Since the random vector  $\mathbf{u}$  is uniformly distributed on the unit sphere, the mean value for  $x_i$  over the randomization iterations depends on the angle,  $\theta_{i,2N_t+1}$ , between  $\mathbf{v}_i$  and  $\mathbf{v}_{2N_t+1}$  and can be written as

$$\mu_i = \frac{\pi - 2\theta_{i,2N_t+1}}{\pi}. \quad (14)$$

Using the fact that  $\mathbf{v}_i^T \mathbf{v}_{2N_t+1} = \|\mathbf{v}_i\| \|\mathbf{v}_{2N_t+1}\| \cos(\theta_{i,2N_t+1})$ , and that the constraint  $[\mathbf{X}]_{ii} = 1$  in (13) ensures that all  $\|\mathbf{v}_i\| = 1$ , the mean value can be expressed directly in terms of the columns of  $\mathbf{V}$ ,

$$\mu_i = \frac{2}{\pi} \arcsin(\mathbf{v}_i^T \mathbf{v}_{2N_t+1}). \quad (15)$$

The first observation in the development of the proposed demodulator is that the expression in (15) suggests that for the purposes of soft demodulation, one could consider generating a sequence of bit vectors with properties similar to those generated by the formal randomization process by making the

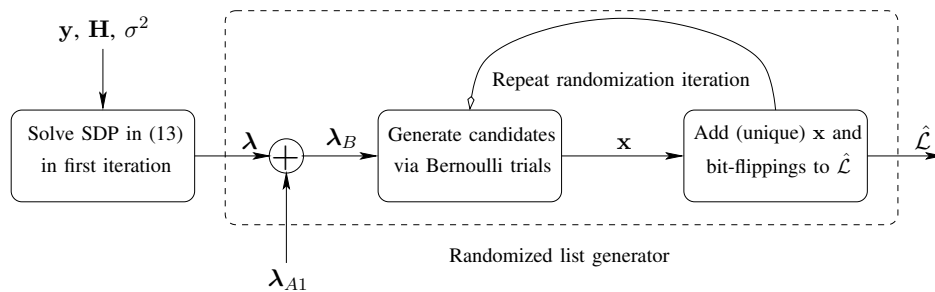


Fig. 2. List generation scheme using the Single-SDR algorithm.

approximation that the elements of  $\mathbf{x}$  are independent, and generating each element of  $\mathbf{x}$  via a scalar (antipodal) Bernoulli trial. Such an approach would avoid the cost of computing  $\mathbf{V}^T \mathbf{u}$  in each instance of the formal randomization procedure.

The second observation is that this Bernoulli trial approach provides an opportunity to separate the processing of the information provided by the channel output from the processing of the extrinsic information fed back from the previous iteration of the decoder. At each iteration, the decoder updates the extrinsic information that it provides to the demodulator (which we have denoted by  $\lambda_{A1}$ ). The expression for  $D(\mathbf{b})$  in (5) suggests that the demodulation procedure needs to be repeated at each iteration (as it is in [3], [4], [6] and in the List-SDR algorithm proposed in Section IV). However, as we will show below, the Bernoulli trial approach to randomization allows us to extract the SDP from the iterative demodulation and decoding loop so that we need only solve one SDP per channel use.

The architecture of the proposed list generation technique is illustrated in Fig. 2. It consists of an SDR demodulator (which is invoked only in the first iteration), and a randomized list generator. The randomized list generator takes two inputs: (i) the vector  $\boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_{2N_t}]^T$  containing the mean values in (15) in LLR form, i.e.,

$$\lambda_i = \log\left(\frac{1 + \mu_i}{1 - \mu_i}\right); \quad (16)$$

and (ii) the vector  $\lambda_{A1}$  containing the extrinsic information (in LLR form) from the previous iteration of the decoder. The randomized demodulator then computes Bernoulli distributions that reflect these inputs (see (18) below), and generates a sequence of random binary vectors according to those distributions.

By construction, the extrinsic information provided by the decoder is independent of the soft information from the channel [21]. Therefore, if the randomized demodulator is to generate candidate bit-vectors via Bernoulli trials that reflect both the information from the channel and the extrinsic information

TABLE II  
LIST GENERATION COMPONENT OF THE SINGLE-SDR ALGORITHM

- 
- Data:  $\lambda$  in (16);  $\lambda_{A1}$ , the vector of extrinsic LLRs from the previous iteration of the decoder.
  - Parameters:  $M$ , the number of randomization iterations;  $K$ , the maximum size of the preliminary list.
  - Output:  $\hat{\mathcal{L}}$ , the enriched list.
1. Initialize  $\hat{\mathcal{L}}'$  and  $\hat{\mathcal{L}}$  empty,  $m = 0$ , and  $k = 0$ .
  2. Compute  $\lambda_B$  in (17) and subsequently  $\mu_B$ .
  3. Generate each element of  $\mathbf{x}$ ,  $x_i$ , independently according to the (antipodal) Bernoulli distribution with mean  $\mu_{B,i}$  and increment  $m$ .
  4. If  $\mathbf{x}$  is not in  $\hat{\mathcal{L}}'$ , add it to  $\hat{\mathcal{L}}'$  and increment  $k$ .
  5. If  $k < K$  and  $m < M$ , return to 3.
  6. Construct  $\hat{\mathcal{L}}$  as the union of  $\hat{\mathcal{L}}'$  and all the single bit-flippings of the bit-vectors in  $\hat{\mathcal{L}}'$ .
- 

from the decoder, the LLR representation of the mean of that Bernoulli distribution should be

$$\lambda_B = \lambda + \lambda_{A1}. \quad (17)$$

The  $i$ -th entry of the corresponding mean vector  $\mu_B$  is

$$\mu_{B,i} = 1 - 2/(1 + \exp(\lambda_{B,i})). \quad (18)$$

Having computed  $\mu_B$ , the demodulator randomly generates the bit-vectors that will form the preliminary list,  $\hat{\mathcal{L}}'$ . The  $i$ th bit of each of these vectors is generated by running an independent (antipodal) Bernoulli trial with mean  $\mu_{B,i}$ . An enriched list  $\hat{\mathcal{L}}$  is then constructed by adding to  $\hat{\mathcal{L}}'$  all the single bit-flippings of the bit vectors in  $\hat{\mathcal{L}}'$ . A formal statement of list generation using this algorithm is presented in Table II. After construction of the list  $\hat{\mathcal{L}}$  the soft information from demodulator can be approximated using (6) or (7).

## VI. LIST-FREE IMPLEMENTATION

One of the bottlenecks in the implementation of list-based soft demodulators is the requirement of a significant amount of memory. This is an especially important issue in list demodulation schemes in which the list is generated in the first demodulation-decoding iteration and stored for use in the subsequent iterations. In those schemes (e.g., [2]), the system must provide enough memory to save the list associated

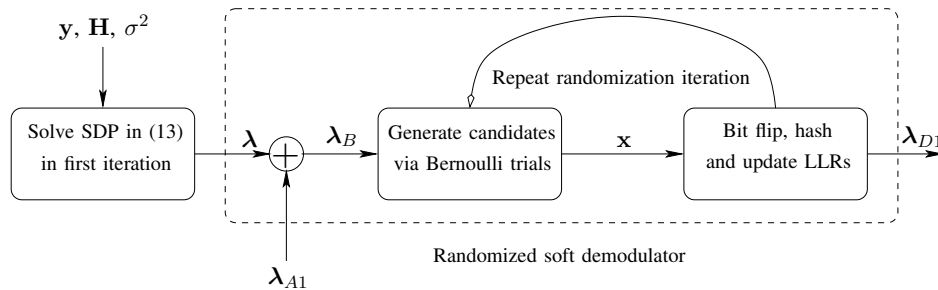


Fig. 3. The proposed list-free soft demodulator.

with each of the  $N$  channel uses required to send a complete codeword. (Since these lists are based on information from the channel only, they need to be quite long.) In schemes that regenerate the list at each iteration, the receiver has only one list to store, but the resulting memory requirement can still be quite significant.

Another issue that arises in list-based demodulation schemes in which the list members are generated via a randomization procedure, such as those proposed in Sections IV and V, is that candidate list members may be duplicated. In order to avoid redundant computation of the metric  $D(\cdot)$ , and in order to avoid storage of repeated list members, this duplication ought to be avoided; see Step 5 in Table I and Step 4 in Table II. However, it is important that the computational cost of any scheme that is used to avoid this duplication is small.

In this section we will show that for list demodulators that regenerate the list at each iteration and use the max-log approximation on the list to approximate the LLR (cf. (7)), only two real vectors of size  $N_t$  need to be stored. We will also show that repeated generation of candidate list members can be efficiently detected via a hashing strategy. A block diagram of the proposed list-free implementation is provided in Fig. 3, and a formal statement of the algorithm is provided in Table III. We point out that unlike the algorithms in the previous sections in which the output was the enriched list  $\hat{\mathcal{L}}$ , from which the soft information can be extracted using (6) or (7), in the list-free implementation the soft information is generated directly.

One of the features of many list demodulators, including the List-SDR and Single-SDR demodulators presented in the previous sections, is that the candidate list members are generated one at a time. In such schemes we can update the optimal values of the  $2N_t$  optimization problems in (7) as each list member is generated, rather than waiting for the whole list to be constructed. We can then discard that list member and simply store the  $2N_t$  real values that are the current optimal values for the problems in (7). Doing

so leads to a list-free implementation. In our demodulators, the list  $\hat{\mathcal{L}}$  is generated in two phases, the randomization phase and the enrichment phase (which is based on bit-flipping). The enrichment phase can be incorporated into the list-free implementation by simply evaluating the neighbouring vectors at Hamming distance 1 of each randomized candidate just after it is generated.

This list-free implementation highlights the potential for redundant computation of  $D(\cdot)$ . One efficient way in which that can be avoided is via a hashing scheme [30]. The hashing scheme computes an integer ‘signature’ for each generated bit vector via a deterministic injective function, and then stores this signature in an ordered array. Before processing the next candidate bit-vector,  $\mathbf{x}$ , the demodulator computes its signature and checks if that value is in the signature array. The candidate is discarded if its signature is present, and if that value is not found, its signature is added to the array, and the candidate bit-vector is processed. In the simulations in Section VIII we chose the signature to be the decimal equivalent of the binary number  $(\mathbf{x}^T + \mathbf{1})/2$ , where  $\mathbf{1}$  is a conformally-sized vector of ones. As we will quantify in the next two sections, the computational costs of computing the signatures and searching the (ordered) signature array are small with respect to the overall computational cost of the demodulator. Implementing the hashing scheme does require additional memory. Although the maximum required memory,  $\min\{M, K\}$ , is, quite naturally, the same as that required to store the list, in practice the average length of the signature array will be significantly shorter, and the memory requirement will often be outweighed by the computational cost reduction that is obtained by (efficiently) avoiding redundant computation.

An additional advantage of the list-free implementation described above is that the parameters of the algorithm can be adapted dynamically. This offers the potential for the demodulator to dynamically adjust its operating point on its performance-complexity trade-off in response to changes in the characteristics of the channel or in the requirements of the application.

TABLE III  
LIST-FREE IMPLEMENTATION OF SINGLE-SDR ALGORITHM FOR RANDOMIZED SOFT DEMODULATOR

- 
- Data:  $\lambda$  in (16);  $\lambda_{A1}$ , the vector of extrinsic LLRs from the previous iteration of the decoder.
  - Parameters:  $M$ , the number of randomization iterations;  $K$ , the maximum size of the signature list.
  - Output:  $\lambda_{D1}$ , the vector of log likelihood-ratios
1. Compute  $\lambda_B$  in (17) and subsequently  $\mu_B$ .
  2. Initialize  $\mathbf{f}_{+1} = \{+\infty\}^{2N_t}$ ,  $\mathbf{f}_{-1} = \{+\infty\}^{2N_t}$ ,  $m = 0$ ,  $k = 0$
  3. Generate each  $x_i$  independently according to the (antipodal) Bernoulli distribution with mean  $\mu_{B,i}$ .
  4. Compute the signature of  $\mathbf{x}$ . If that value is not in the signature array, insert the value into the array and increment  $k$ , compute  $D(\mathbf{x})$ , and for each  $i = 1, 2, \dots, 2N_t$ , if  $x_i = +1$  then set  $[\mathbf{f}_{+1}]_i = \min\{[\mathbf{f}_{+1}]_i, D(\mathbf{x})\}$ , else set  $[\mathbf{f}_{-1}]_i = \min\{[\mathbf{f}_{-1}]_i, D(\mathbf{x})\}$ .
  5. For each  $i = 1, 2, \dots, 2N_t$ , set  $\tilde{\mathbf{x}}^{(i)} = \mathbf{x}$  and then  $\tilde{x}_i^{(i)} = -x_i$ . Repeat Step 4 for  $\tilde{\mathbf{x}}^{(i)}$ .
  6. Increment  $m$ . If  $m < M$  and  $k < K$  return to 3. Otherwise, return  $\lambda_{D1} = (\mathbf{f}_{+1} - \mathbf{f}_{-1}) / (2\sigma^2)$ .
- 

## VII. COMPUTATIONAL COST

As mentioned in the Introduction, an advantage of SDR methods over tree search methods, such as sphere decoding, is their polynomial (worst-case) computational cost. An advantage of the particular (list-free) SDR-based soft demodulators that we have proposed herein is that their computational costs are lower than those of some existing soft demodulators. In this section we quantify that claim by evaluating the computational costs of these SDR-based algorithms. For convenience, we have summarized the outcomes of this analysis in Table IV. We will begin our analysis by stating the computational cost of each of the components of the algorithms, in terms of floating point operations.

If we let  $\epsilon$  denote the accuracy to which the SDP is solved, the worst-case computational cost of solving the SDP in (13) using the interior point method in [28] is  $O((2N_t + 1)^{3.5} \log \epsilon^{-1})$ . The computational cost of generating each bit-vector in the conventional randomization procedure used in the Multi-SDR [9] and List-SDR (cf. Section IV) methods is  $O((2N_t + 1)^2)$ . The computational cost of the simplified randomization step in the Single-SDR method (cf. Section V) is  $O(2N_t)$ . Finally, since  $\mathbf{b}$  is binary, computing the metric  $D(\mathbf{b})$  requires  $2N_t(2N_t + 1)$  (signed) real additions.

The Multi-SDR method of [9] solves one SDP of size  $2N_t + 1$  and  $2N_t$  SDPs of size  $2N_t$  per demodulation-decoding iteration, and after solving each SDP it performs  $M$  randomization iterations and

computes  $D(\mathbf{b})$  for all the generated bit-vectors. If we perform  $T$  demodulation-decoding iterations, then the computational cost per channel use of the multi-SDR scheme is:

$$\begin{aligned} & T \times [O((2N_t + 1)^{3.5} \log \epsilon^{-1}) + M \times O((2N_t + 1)^2) + M \times O((2N_t)^2)] \\ & \quad + 2N_t T \times [O((2N_t)^{3.5} \log \epsilon^{-1}) + M \times O((2N_t)^2) + M \times O((2N_t)^2)] \\ & \quad \sim O(TN_t^{4.5} \log \epsilon^{-1}) + O(TMN_t^3) + O(TMN_t^3), \quad (19) \end{aligned}$$

where the first term corresponds to the cost of solving the SDPs, the second term corresponds to the cost of the randomization steps that follow each of these SDPs, and the last term denotes the cost of the metric computations.

The List-SDR approach proposed in Section IV requires the solution of only one SDP of size  $2N_t + 1$  per demodulation-decoding iteration. If we perform  $M$  randomization steps (i.e., if  $K = M$  in Table I), then the cardinality of the enriched list  $\hat{\mathcal{L}}$  is at most  $(2N_t + 1)M$ , and hence the enumeration approach to optimizing the terms on the right hand side of (7) requires at most  $(2N_t + 1)M$  evaluations of  $D(\mathbf{b})$  in each demodulation-decoding iteration. Therefore, the worst-case complexity of the proposed approach is

$$\begin{aligned} & T \times [O((2N_t + 1)^{3.5} \log \epsilon^{-1}) + M \times O((2N_t + 1)^2) + M(2N_t + 1) \times O((2N_t)^2)] \\ & \quad \sim O(TN_t^{3.5} \log \epsilon^{-1}) + O(TMN_t^2) + O(TMN_t^3), \quad (20) \end{aligned}$$

where these terms correspond to the cost of solving the SDP, the cost of constructing the list  $\hat{\mathcal{L}}$  (via randomization), and the cost of the enumerations in (7) over  $\hat{\mathcal{L}}_{i,\pm 1}$ , respectively. This expression shows that the computational cost of the SDP component of the proposed list-based SDR scheme is one order lower than that of the multi-SDR scheme.

The computational cost per channel use of the Single-SDR approach can be obtained in a similar way to those above, and is

$$O(N_t^{3.5} \log \epsilon^{-1}) + O(TMN_t) + O(TMN_t^3), \quad (21)$$

where the first term represents the complexity of solving the SDP, the second term represent the cost of the Bernoulli-based randomizations, and the third term represents the cost of computing  $D(\mathbf{b})$  for each bit-vector generated in the randomization step. (As in (20), we have assumed that  $K = M$  in Tables II and III.) These expressions reveal the computational advantage of only having to solve one SDP per channel use, and the advantage of the Bernoulli-based randomizations. The list-free implementation of this algorithm requires a small amount of additional computation; at most  $O(TMN_t)$  integer additions

TABLE IV  
DOMINANT COMPUTATIONAL COST PER CHANNEL USE OF VARIOUS MIMO SOFT DEMODULATORS FOR A SYSTEM WITH  $N_t$   
TRANSMIT ANTENNAS AND  $T$  DEMODULATION-DECODING ITERATIONS.

Demodulator	Dominant Computational Cost
Multi-SDR	$O(TN_t^{4.5})$
List-SDR	$O(TN_T^{3.5})$
Single-SDR	$O(N_t^{3.5})$
MMSE-SIC	$O(TN_t^4)$

per channel use to compute the signatures, and at most  $O(M \log M)$  operations to search for existing signatures in the (sorted) signature array.

As we will illustrate in Section VIII, in most practical implementations, the cost of solving the SDPs will be the dominant component of the computational cost of the SDR-based demodulators. We have summarized those costs in Table IV, and to help place those costs in context, we have included the computational cost per channel use of the MMSE-SIC demodulator in [22], which is  $TO((2N_t)^4) \sim O(TN_t^4)$ .

## VIII. SIMULATIONS

In this section we will compare the performance and computational cost of the proposed demodulators with those of the Multi-SDR demodulator [9], the list sphere decoder in [2], and the MMSE-SIC demodulator in [22]. We consider MIMO BICM systems that employ V-BLAST transmission of QPSK symbols over an i.i.d. Rayleigh fading channel. The transceiver parameters, including those of the outer codes and the iterative demodulation and decoding algorithm, are chosen from those used in [2], [22]. For the proposed demodulators we will evaluate the performance of the list-free implementation described in Section VI, using the hashing strategy to avoid redundant computation. For all demodulation schemes based on list-decoding ideas, an insufficiently rich list can result in under or over estimation of the soft information. In our simulations of the proposed demodulators and the list sphere decoder in [2], we will take a common approach to mitigating this effect [5], and will clip the estimated log likelihood-ratios to the interval  $[-5, +5]$ .

We will consider two types of outer code: a turbo code with a reasonably long block length, and a convolutional code with a rather short block length. For the (scalar) binary-input additive white Gaussian noise channel, the turbo code provides good performance at SNRs close to the capacity threshold for the

given rate. The convolutional code is a weaker code, but it can be implemented with significantly lower latency. Following [2], the turbo code was chosen to be a rate  $1/2$  punctured parallel concatenated turbo code with the  $(5, 7)$  recursive systematic convolutional code as the component codes and an (input) block length of 8192. The (different) interleavers in the turbo code and in the BICM transmitter were selected from randomly generated candidates in each Monte-Carlo iteration. We used the conventional BCJR algorithm [31] to decode the constituent convolutional codes of the turbo code, and 8 turbo decoding iterations were performed before we passed extrinsic information back to the demodulator. Following [22], the convolutional code was chosen to be the rate  $1/2$   $(23, 35)$  recursive systematic convolutional code with block length 256, and BCJR decoding was used. We have performed simulation experiments for  $8 \times 8$  and  $4 \times 4$  MIMO systems, and we will report some results from those experiments in the following sections, respectively.

#### A. $8 \times 8$ system

In our first set of simulation experiments, we will consider a MIMO system with  $N_t = 8$  antennas at the transmitter and  $N_r = 8$  antennas at the receiver. In Figs. 4 and 5 we consider systems with the turbo outer code and the convolutional outer code, respectively, and we compare the BER performance of i) the Single-SDR demodulator, ii) the List-SDR demodulator of Section IV, iii) the Multi-SDR demodulator in [9], iv) the list sphere decoder in [2] (with a list size of  $L = 512$ ), and v) the MMSE-SIC demodulator in [22]. In this comparison we specify the Single-SDR demodulator with the SDP accuracy of  $\epsilon = 10^{-2}$  and  $M = 50$  randomizations. Later in this section, we will examine the impact of these parameters on the complexity and performance of the Single-SDR demodulator and will justify this choice. For fair comparison, the value of  $\epsilon$  in the List-SDR and Multi-SDR demodulators was also chosen to be  $10^{-2}$ . For reference, we have indicated the SNR at which the mutual information for QPSK signals is equal to 8 bits per channel use (the data rate of the chosen scheme). That SNR is about 1.6 dB. From Fig. 4 it is apparent that the BER of the Single-SDR demodulator is better than that of the MMSE-SIC demodulator and is close to that of the other demodulators. The performance advantage of the Single-SDR demodulator over the MMSE-SIC demodulator is somewhat larger in the early iterations in the case of the convolutional outer code (see Fig. 5), but the relative weakness of this short code means that after four demodulation-decoding iterations the performance of all the considered demodulators is quite similar.

In order to show that the Single-SDR demodulator achieves the performance described above at low computational cost, we explicitly counted the number of floating point operations (FLOPs) required by

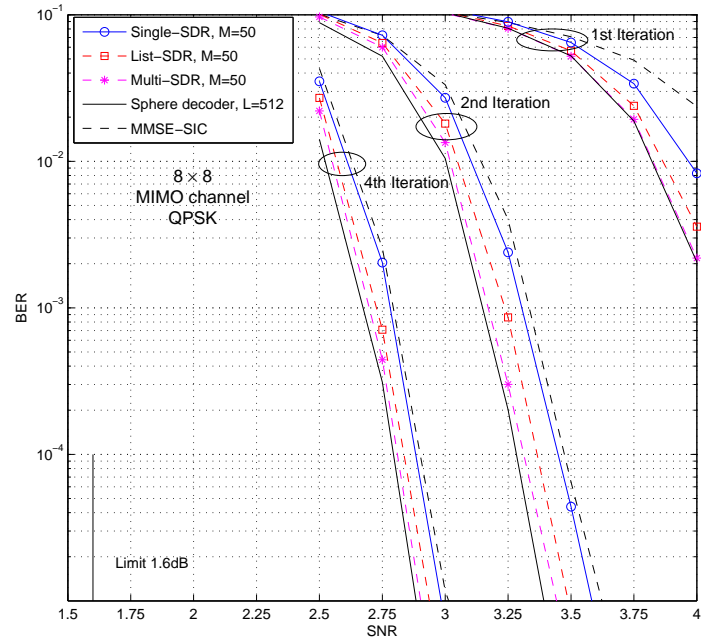


Fig. 4. Comparison of the BER performance of various demodulators for the  $8 \times 8$  MIMO system with the turbo outer code.

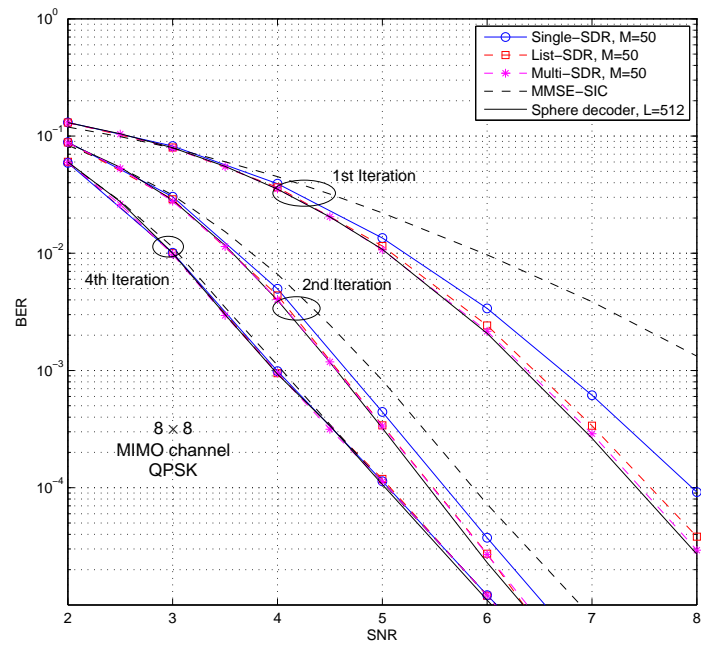


Fig. 5. Comparison of the BER performance of various demodulators for the  $8 \times 8$  MIMO system with the convolutional outer code.

each demodulator to perform each component of its algorithm at each demodulation iteration in each channel use. For the SDR-based demodulators we include the FLOPs required to solve the SDPs, to perform the randomization steps, and to compute the metrics and the hash functions. For the list sphere decoder we have included the FLOPs required to construct the list (which is only performed once per channel use), and those required to compute the metrics in each demodulation iteration. (For the list sphere decoder, we have excluded the computational cost of the trial and error scheme used to compute an appropriate radius for the sphere (cf. [2]), and hence the depicted results may be a little optimistic.) For the MMSE-SIC demodulator, for each bit we count the FLOPs required to compute and subtract the mean of the interfering symbols, and those required to compute and implement the unbiased linear MMSE estimator of the resulting zero-mean signal. Our explicit counting of the number of required operations revealed that, as stated in Section VII, the dominant component of the computational cost of the SDR-based demodulators is the cost of solving the SDP or SDPs. This observation is quantified in Fig. 6, where we have plotted the average computational cost per channel use of each algorithm against the SNR, for the case of the turbo outer code, and we have included results for several values of  $M$ , the number of randomizations.

Fig. 6 also quantifies the computational advantages of the Single-SDR demodulator over the List-SDR and Multi-SDR demodulators, and its computational advantages over the MMSE-SIC demodulator and the list sphere decoder. In particular, in the ‘waterfall’ region of the BER curves in Fig. 4, the average computational cost per channel use of the Single-SDR demodulator is about half that of the MMSE-SIC demodulator and about a third of that of the list sphere decoder. Furthermore, unlike the list sphere decoder, the distribution of the computational cost of the SDR demodulation methods is concentrated around the mean. To illustrate that fact, we have plotted in Fig. 7 the empirical probability density of the computational cost per channel use of several demodulators at an SNR of 2.75 dB. In this scenario, the whole empirical distribution of the computational cost of the Single-SDR demodulator lies below the computational cost of the MMSE-SIC demodulator. Fig. 7 also illustrates the rather ‘fat’ tail of the distribution of the computational cost of the list sphere decoder. This fat tail can make it rather awkward to provision an appropriate amount of computational resources for a list sphere decoder. The concentrated complexity distributions of the SDR-based methods make that provisioning much more straightforward.

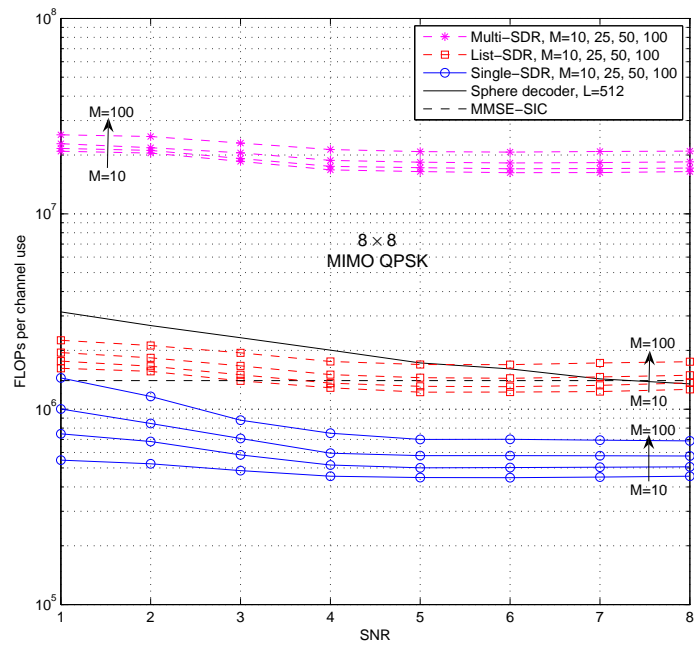


Fig. 6. Comparison of the average computational cost per channel use of the proposed demodulators and that of the Multi-SDR, list sphere decoding, and MMSE-SIC demodulators for the  $8 \times 8$  system with the turbo outer code. For the SDR based methods results for several values for  $M$ , the number of randomizations, are provided.

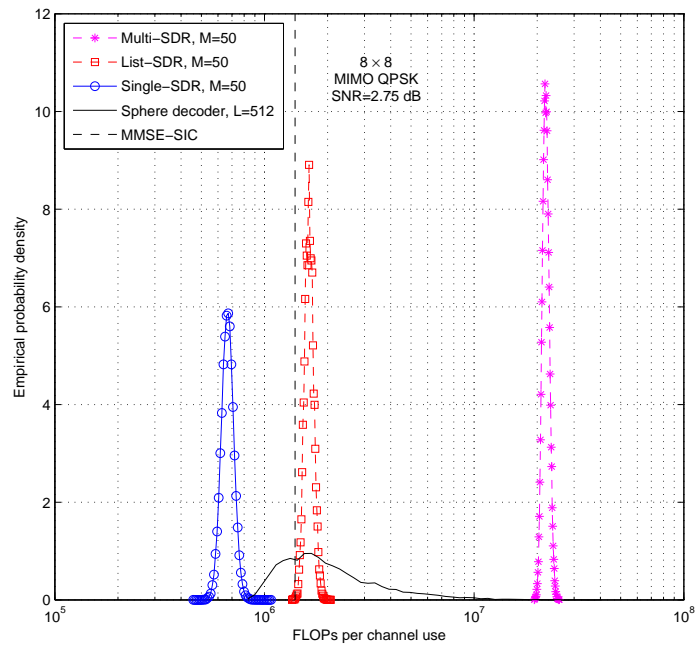


Fig. 7. Empirical probability density of the number of FLOPs per channel use in the  $8 \times 8$  system with the turbo outer code at an SNR of 2.75 dB.

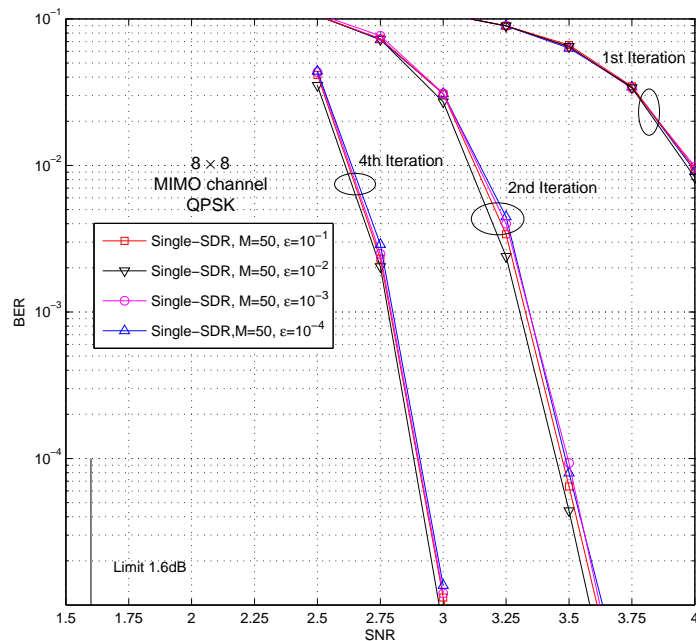


Fig. 8. BER performance of the  $8 \times 8$  MIMO system with the turbo outer code and the Single-SDR demodulator with  $M = 50$  randomizations and different accuracies to which the SDP is solved.

In order to justify the choices of an SDP accuracy of  $\epsilon = 10^{-2}$  and  $M = 50$  randomizations for the Single-SDR demodulator in Figs. 4 and 5, we now evaluate the impact of these parameters on the BER performance of the receiver. In Fig. 8 we plot the average BER of the Single-SDR demodulator with  $M = 50$  (and  $K = M$ ) and different values for  $\epsilon$ . Fig. 8 suggests that in order to extract the benefits of the Single-SDR demodulator, it is sufficient to solve the SDP to two digits of accuracy. This is an important observation because solving the SDP is the dominant computational task in the Single-SDR demodulator, and the cost of solving this SDR grows as  $O(N_t^{3.5} \log \epsilon^{-1})$ . Fig. 8 also suggests that solving the SDP more accurately does not necessarily lead to improved performance. This is a consequence of the fact that accurate list demodulation requires a rather rich set of list members. In the case that the SDP is solved accurately, the resulting randomization procedure may generate a rather narrowly-focused list, and hence the slightly degraded performance in the case of  $\epsilon = 10^{-4}$  in Fig. 8. In order to promote the generation of rich lists, and in order to reduce the computational cost, it is natural to consider demodulators in which the SDP is only coarsely solved.

In Fig. 9 we examine the BER performance of the Single-SDR demodulator for an SDP accuracy of

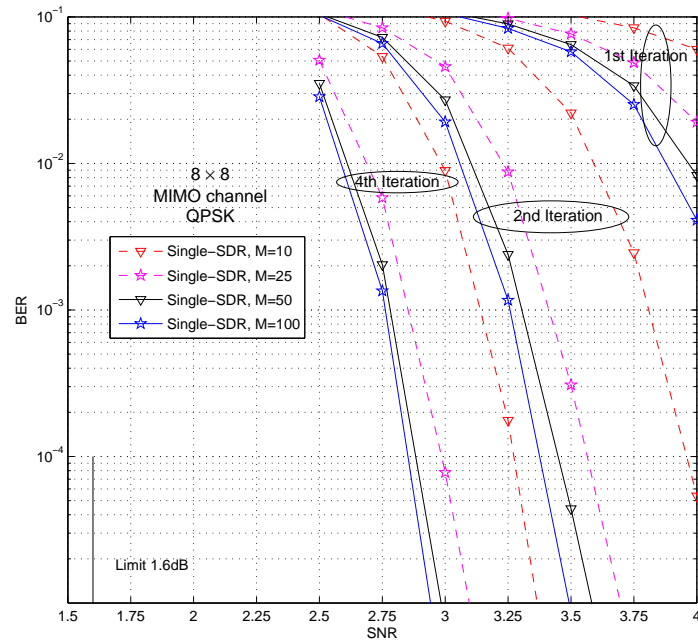


Fig. 9. BER performance of the  $8 \times 8$  MIMO system with the turbo outer code and the Single-SDR demodulator with an SDP solution accuracy of  $\epsilon = 10^{-2}$ , and different numbers of randomizations,  $M$ .

$\epsilon = 10^{-2}$  and various numbers of randomizations. Even though each randomization in the Single-SDR demodulator is rather cheap to implement (see Fig. 6), Fig. 9 suggests that the choice of  $M = 50$  is sufficient to demonstrate the potential performance of the Single-SDR demodulator.

### B. $4 \times 4$ system

In the second set of simulation experiments, we repeated the above analysis for a MIMO system with  $N_t = N_r = 4$ . In this case, the full demodulation list has only 256 elements, and hence full-list demodulation is chosen as the performance benchmark. (A list sphere decoder with a list size of 128 is also considered.)

Following experiments analogous to those discussed in the previous section, we chose to solve the SDPs to an accuracy of  $\epsilon = 10^{-2}$  and to employ  $M = 25$  randomizations. The average BERs of the various demodulators in the system with the turbo outer code are plotted in Fig. 10. (The SNR threshold of this system is 1.6 dB.) As in the  $8 \times 8$  case, the performance of the proposed demodulators is close to that of the benchmark demodulator. However, in this  $4 \times 4$  case, the performance of the proposed demodulators is also substantially better than that of the MMSE-SIC demodulator. The relative degradation in the performance of the MMSE-SIC is due to the fact that there are fewer interfering symbols in the  $4 \times 4$  case, and hence the inherent approximation that the residual interference is Gaussian is less accurate in this case.

The average BERs of these demodulators in the case of the convolutional outer code are provided in Fig. 11. In the early iterations, the Single-SDR demodulator has a significant performance advantage over the MMSE-SIC demodulator, but as was the case for the  $8 \times 8$  system, after four iterations the performance of all demodulators is quite similar.

In Fig. 12 we plot the average computational cost per channel use of each demodulator in the system with the turbo outer code, and in Fig. 13 we plot the empirical density of the computational costs at an SNR of 2.75 dB. These figures quantify the computational advantages of the Single-SDR demodulator. In particular, its average computational cost is just over two-thirds of that of the MMSE-SIC demodulator, and just over half of that of the full list demodulator. In this  $4 \times 4$  scenario, the computational advantages of the single-SDR demodulator over the MMSE-SIC demodulator are smaller than those in the  $8 \times 8$  case, but its performance advantage is significantly larger.

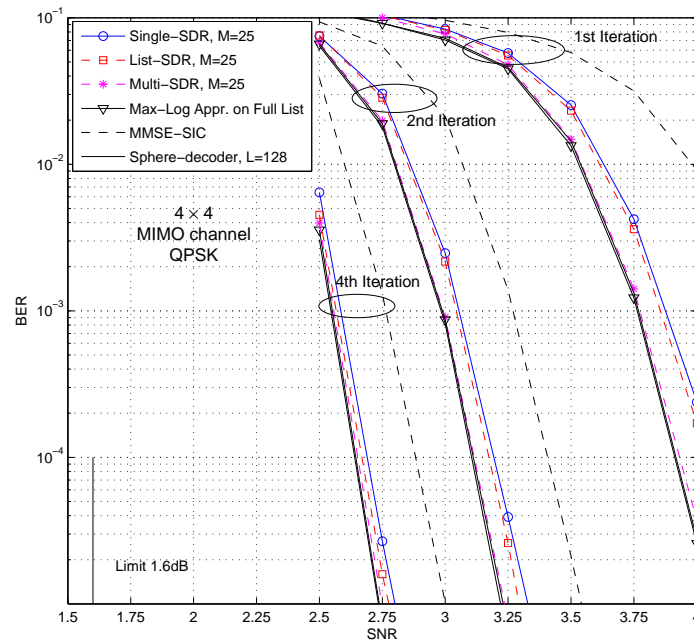


Fig. 10. Comparison of the BER performance of various demodulators for the  $4 \times 4$  MIMO system with the turbo outer code.

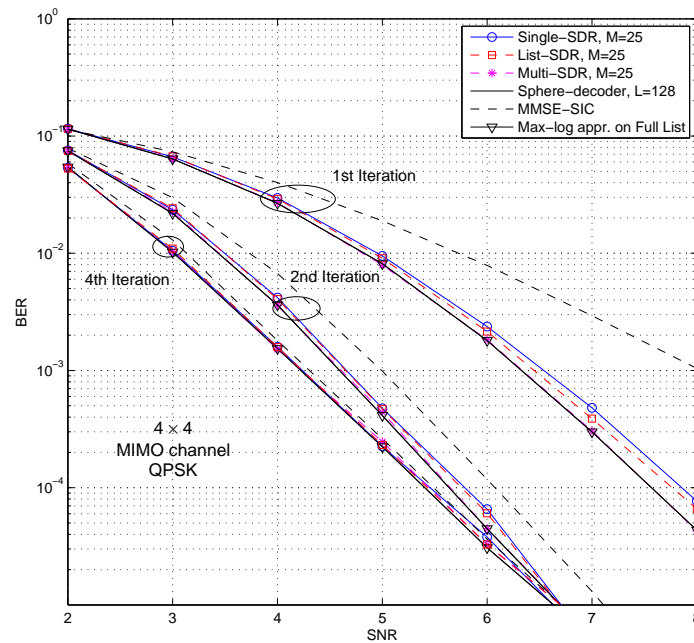


Fig. 11. Comparison of the BER performance of various demodulators for the  $4 \times 4$  MIMO system with the convolutional outer code.

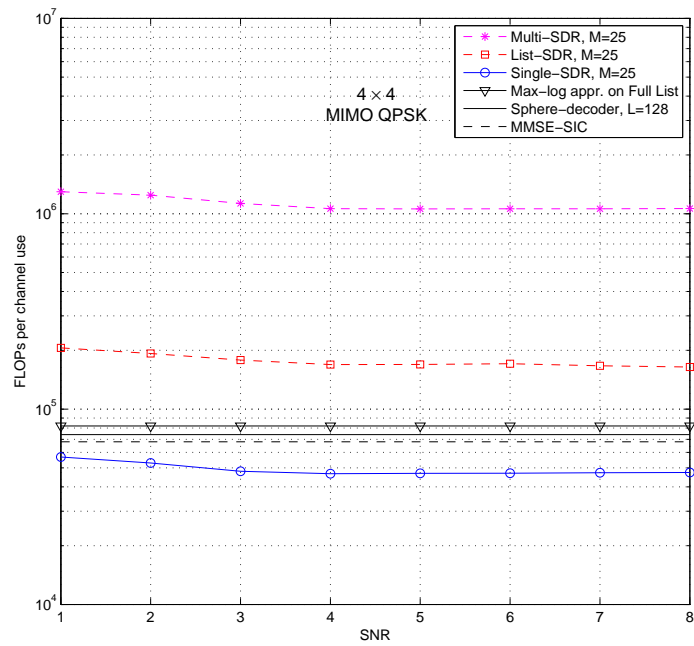


Fig. 12. Comparison of average computational cost per channel use of the proposed demodulators and that of the Multi-SDR, full-list, sphere decoding and MMSE-SIC demodulators for the  $4 \times 4$  system with the turbo outer code.

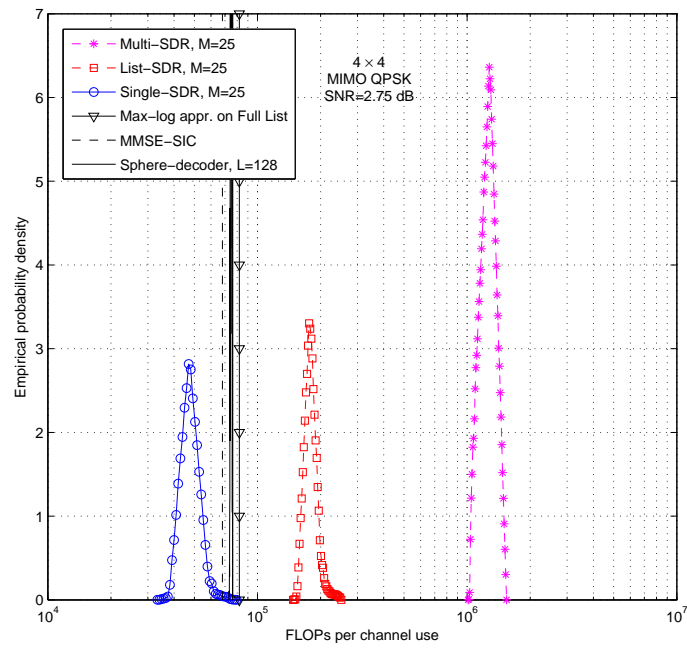


Fig. 13. Empirical probability density of the number of FLOPs per channel use in the  $4 \times 4$  system with the turbo outer code at an SNR of 2.75 dB.

## IX. CONCLUSION

In this paper, we have proposed two computationally-efficient soft MIMO demodulators based on an adaptation of the semidefinite relaxation (SDR) method for hard demodulation to list-based soft demodulation. We have also presented a list-free implementation of the proposed methods that can be implemented with a substantially smaller memory ‘footprint’ than conventional list demodulation algorithms. In contrast to list demodulators based on the principles of sphere decoding, the (worst-case) computational cost of the proposed demodulators is bounded by a (low-order) polynomial of the number of bits to be demodulated, and in contrast to the SDR-based demodulator in [9], one of the proposed demodulators requires the solution of one semidefinite program (SDP) per demodulation-decoding iteration for each channel use and the other requires the solution of only one SDP per channel use. Our simulation results suggest that these computational advantages are obtained without incurring a significant degradation in performance. In particular, the proposed Single-SDR demodulator provides better performance than the MMSE-SIC demodulator and performance that is close to that of the list sphere decoder, and it does so at a substantially lower computational cost. In this paper we have focussed on soft demodulators for MIMO-BICM systems that employ QPSK signalling. Extensions to systems that employ higher order QAM constellations are currently being developed, using the corresponding SDR approaches to hard demodulation in [32], [33].

## REFERENCES

- [1] I. E. Telatar, "Capacity of multiple antenna Gaussian channels," *Eur. Trans. Telecom.*, vol. 10, pp. 585–595, Nov. 1999.
- [2] B. M. Hochwald and S. ten Brink, "Achieving near capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.
- [3] S. Baro, J. Hagenauer, and M. Witzke, "Iterative detection of MIMO transmission using a list-sequential (LISS) detector," in *Proc. IEEE Int. Conf. Commun.*, vol. 4, Anchorage, May 2003, pp. 2653–2657.
- [4] H. Vikalo, B. Hassibi, and T. Kailath, "Iterative decoding for MIMO channels via modified sphere decoding," *IEEE Trans. Wireless Commun.*, vol. 3, no. 6, pp. 2299–2311, Nov. 2004.
- [5] Y. L. C. de Jong and T. J. Willink, "Iterative tree search detection for MIMO wireless systems," *IEEE Trans. Commun.*, vol. 53, no. 6, pp. 930–935, June 2005.
- [6] J. Hagenauer and C. Kuhn, "The list-sequential (LISS) algorithm and its application," *IEEE Trans. Commun.*, vol. 55, no. 5, pp. 918–928, May 2007.
- [7] J. Jaldén and B. Ottersten, "Parallel implementation of a soft output sphere decoder," in *Proc. Asilomar Conf. Sig. Sys. Comp.*, Monterey, CA, Oct. 2005.
- [8] R. Wang and G. B. Giannakis, "Approaching MIMO channel capacity with soft detection based on hard sphere decoding," *IEEE Trans. Commun.*, vol. 54, no. 4, pp. 587–590, Apr. 2006.
- [9] B. Steingrimsson, Z.-Q. Luo, and K. M. Wong, "Soft quasi-maximum-likelihood detection for multiple-antenna channels," *IEEE Trans. Signal Processing*, vol. 51, no. 11, pp. 2710–2719, Nov. 2003.
- [10] D. J. Love, S. Hosur, A. Batra, and R. W. Heath, Jr., "Space-time Chase decoding," *IEEE Trans. Wireless Commun.*, vol. 4, no. 5, pp. 2035–2039, Sept. 2005.
- [11] P. Robertson, E. Vilebrun, and P. Hoeher, "A comparison of optimal and suboptimal map decoding algorithms operating in the log domain," in *Proc. IEEE Int. Conf. Commun.*, vol. 2, Seattle, WA, June 1995, pp. 1009–1013.
- [12] A. D. Murugan, H. El Gamal, M. O. Damen, and G. Caire, "A unified framework for tree search decoding: Rediscovering the sequential decoder," *IEEE Trans. Inform. Theory*, vol. 52, no. 3, pp. 933–953, Mar. 2006.
- [13] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inform. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.
- [14] M. O. Damen, H. El Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inform. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.
- [15] J. Jaldén and B. Ottersten, "On the complexity of sphere decoding in digital communications," *IEEE Trans. Signal Processing*, vol. 53, no. 4, pp. 1474–1484, Apr. 2005.
- [16] W. K. Ma, B. N. Vo, T. N. Davidson, and P. C. Ching, "Blind ML detection of orthogonal space-time block codes: Efficient high-performance implementations," *IEEE Trans. Signal Processing*, vol. 54, no. 2, pp. 738–751, Feb. 2006.
- [17] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problem using semidefinite programming," *J. ACM*, vol. 42, pp. 1115–1145, 1995.
- [18] Y. E. Nesterov, "Quality of semidefinite relaxation for nonconvex quadratic optimization," Universite Catholique de Louvain, Brussels, Belgium, Tech. Rep. CORE, Mar. 1997.
- [19] G. J. Foschini, G. Golden, R. Valenzuela, and P. Wolniansky, "Simplified processing for high spectral efficiency wireless communication employing multi-element arrays," *IEEE J. Select. Areas Commun.*, vol. 17, no. 11, pp. 1841–1852, Nov. 1999.

- [20] B. Hassibi and B. Hochwald, "High-rate codes that are linear in space and time," *IEEE Trans. Inform. Theory*, vol. 48, no. 7, pp. 1804–1824, July 2002.
- [21] J. Hagenauer, "The turbo principle: Tutorial introduction and state of the art," in *Proc. Int. Symp. Turbo Codes and Related Topics*, Brest, France, Sept. 1997, pp. 1–11.
- [22] X. Wang and H. V. Poor, "Iterative (turbo) soft interference cancellation and decoding for coded CDMA," *IEEE Trans. Commun.*, vol. 47, no. 7, pp. 1046–1061, July 1999.
- [23] H. El-Gamal and E. Geraniotis, "Iterative multiuser detection for coded CDMA signals in AWGN and fading channels," *IEEE J. Select. Areas Commun.*, vol. 18, no. 1, pp. 30–41, Jan. 2000.
- [24] J. Boutros and G. Caire, "Iterative multiuser joint decoding: Unified framework and asymptotic analysis," *IEEE Trans. Inform. Theory*, vol. 48, no. 7, pp. 1772–1793, July 2002.
- [25] G. Caire, R. R. Müller, and T. Tanaka, "Iterative multiuser joint decoding: Optimal power allocation and low-complexity implementation," *IEEE Trans. Inform. Theory*, vol. 50, no. 9, pp. 1950–1973, Sept. 2004.
- [26] W. K. Ma, T. N. Davidson, K. M. Wong, Z.-Q. Luo, and P. C. Ching, "Quasi-maximum-likelihood multiuser detection using semi-definite relaxation with application to synchronous CDMA," *IEEE Trans. Signal Processing*, vol. 50, no. 4, pp. 912–922, Apr. 2002.
- [27] P. H. Tan and L. K. Rasmussen, "The application of semidefinite programming for detection in CDMA," *IEEE J. Select. Areas Commun.*, vol. 19, no. 8, pp. 1442–1449, Aug. 2001.
- [28] C. Helmberg, F. Rendl, R. Vanderbei, and H. Wolkowicz, "An interior point method for semidefinite programming," *SIAM J. Optim.*, vol. 6, no. 2, pp. 342–361, 1996.
- [29] M. Kisioliou and Z.-Q. Luo, "Efficient implementation of a quasi-maximum-likelihood detector based on semi-definite relaxation," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, vol. IV, Honolulu, Apr. 2007, pp. 1329–1332.
- [30] D. Knuth, *The Art of Computer Programming: Volume 3*. Reading, MA: Addison Wesley, 1973.
- [31] L. Bahl, J. Coke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, no. 3, pp. 284–287, Mar. 1974.
- [32] A. Wiesel, Y. C. Eldar, and S. Shamai, "Semidefinite relaxation for detection of 16-QAM signaling in MIMO channels," *IEEE Signal Processing Lett.*, vol. 12, no. 9, pp. 653–656, Sept. 2005.
- [33] N. D. Sidiropoulos and Z.-Q. Luo, "A semidefinite relaxation approach to MIMO detection for high-order QAM constellations," *IEEE Signal Processing Lett.*, vol. 13, no. 9, pp. 525–528, Sept. 2006.