Improved QRD-QLD Algorithm for Low Complexity MIMO Decoding

Nimrod Peer, Yonathan Murin, *Student Member, IEEE*, and Ron Dabora, *Member, IEEE*

***Abstract*—In this paper we present a decoding algorithm for multiple-input multiple-output (MIMO) communications. The algorithm is based on the QRD-QLD algorithm (see, e.g., Radosavljevic *et al.* 2012), proposes an improved path metric computation, and achieves a low probability of error which is close to the single tree search sphere decoder, at low implementation complexity. The proposed improved path metric computation increases the reliability of combining the results of the partial searches into sequences representing the maximum likelihood (ML) and the counter ML hypotheses in soft input MIMO decoding. The excellent performance of the improved algorithm are demonstrated via numerical simulations.**

***Index Terms*—MIMO decoding, QR-QL decomposition.**

I. Introduction

MULTIPLE-input multiple-output (MIMO) wireless systems have a central role in modern wireless communications due to their ability to enhance spectral efficiency and reliability. The practical implementation of MIMO communications requires an efficient decoding algorithm that has a manageable complexity and a small decoding delay. Since the straightforward implementation of the maximum likelihood (ML) decoder has a high computational complexity [1], the practical implementation of MIMO decoding is based on approximating the exact ML decoding scheme. The importance of efficient MIMO decoding has motivated an extensive research effort on the subject, and many algorithms have been developed. A major approach for practical MIMO decoding is the sphere decoding (SD) algorithm [1]. The SD algorithm applies the QR decomposition (QRD), see [2], to transform the MIMO decoding process into a tree search in which the search for the best candidate for the decoded sequence takes place in a bounded region around the received vector. The SD algorithm has inspired a large number of approximate ML decoding algorithms, which can be divided into two general classes. The first class is breadth-first algorithms. These algorithms are typically characterized by a constant complexity and delay. Important members of this class include the QRD-M, which is a tree search algorithm that maintains M surviving paths after processing each layer [3]; the smart order candidate adding (SOCA) algorithm [4]; and a combination of a partial search on the QRD tree with a partial search based on a tree obtained via the QL decomposition (QLD), referred to as the QRD-QLD algorithm [5]. The second class of tree search algorithms implements a depth-first search. These algorithms have variable complexity and delay, and include the list SD (LSD) algorithm [6]; the single tree search (STS) [7] algorithm; and the tuple-search (TS) algorithm [8]. In this paper we present an improved breadth-first algorithm that builds upon the QRD-QLD scheme. The fundamental idea in QRD-QLD is to replace the search on a single tree with two smaller searches where one search is implemented using the QRD and the second search is implemented using the QLD. Differently from [5], we combine the two searches by first generating a small set of candidate sequences and using this set for path metric *computation*. Thus, we refer to the algorithm as QR-QL with improved path metric computation (QRQLIPMC). The complexity of the new algorithm is analytically derived and shown to be fixed. The performance of our algorithm is compared with those of the major tree search algorithms, and the numerical simulations demonstrate a substantial improvement in the bit error rate (BER) versus complexity performance achieved by the new algorithms.

The rest of this paper is organized as follows: Section II describes the system model. Section III briefly revisits the fundamentals of the SD approach. Section IV provides a detailed description of the proposed QRQLIPMC algorithm and its associated computational complexity. Section V details the numerical simulation study and, lastly, Section VI presents concluding remarks.

II. Notations and System Model

Notations: In this work column vectors are denoted with boldface lowercase letters, e.g., $\mathbf{p}$, and matrices are denoted with boldface capital letters, e.g., $\mathbf{P}$; in particular, the $M \times M$ identity matrix is denoted with $\mathbf{I}_M$. We use $(\cdot)^H$ to denote conjugate transposition, and $\|\cdot\|$, $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ to denote the norm, floor and ceil operators, respectively. Slightly abusing common terminology, we call an $N \times M$ matrix $\mathbf{P}$, with $N \geq M$, unitary if $\mathbf{P}^H\mathbf{P} = \mathbf{I}_M$. Lastly, $E[\cdot]$ denotes the stochastic expectation, and $\mathbb{C}$ denotes the set of complex numbers.

System Model: We consider a MIMO system with N_T transmit antennas and $N_R \geq N_T$ receive antennas. Communication is based on a bit-interleaved coded modulation (BICM) scheme as depicted in Fig. 1: A data stream of independent and identically distributed (i.i.d.), equally likely bits, are encoded by a turbo encoder with rate R. The coded stream is then interleaved, and partitioned into blocks of size $N_T \cdot L$ bits. Each block is mapped into N_T channel symbols which are transmitted via N_T antennas. Each transmitted symbol is taken from a constellation whose cardinality is 2^L. The symbols are received at the output of a flat Rayleigh fading channel with additive white Gaussian noise at each receive antenna. The $N_R \times 1$ noise vector is obtained as a realization of a complex Normal random vector $\mathbf{n} \in \mathbb{C}^{N_R \times 1}$, with $E[\mathbf{n}] = \mathbf{0}$ and $E[\mathbf{n}\mathbf{n}^H] = N_0\mathbf{I}_{N_R}$. The entries of the channel matrix $\mathbf{H} \in \mathbb{C}^{N_R \times N_T}$ are taken from an i.i.d. complex Normal random process with zero mean and unit variance. $\mathbf{H}$ is assumed to be perfectly known at the receiver. Let $\mathbf{s} \in \mathbb{C}^{N_T \times 1}$ be the vector of transmitted symbols. The received signal $\mathbf{y} \in \mathbb{C}^{N_R \times 1}$ is given by:

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}. \tag{1}$$

Manuscript received July 4, 2014; accepted August 14, 2014. Date of publication August 22, 2014; date of current version October 8, 2014. This work was supported by the Israeli Ministry of Economy through the Israeli Smart Grid Consortium. The associate editor coordinating the review of this paper and approving it for publication was D.-A. Toumpakaris.

The authors are with the Department of Electrical and Computer Engineering, Ben-Gurion University, Beer-Sheva 84105, Israel (e-mail: nimrodpe@post.bgu.ac.il; moriny@ee.bgu.ac.il; ron@ee.bgu.ac.il).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/LCOMM.2014.2350976

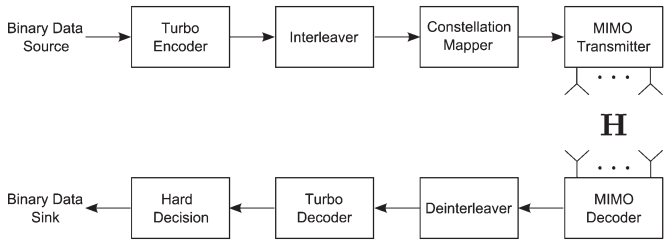

Fig. 1. The system model.

Let E_S be the total average energy of the transmitted constellation symbols vector, and E_b be the average energy per information bit at the receiver. The signal-to-noise ratio (SNR) per information bit at the receiver is $\text{SNR}_b = E_b/N_0 = (E_S N_R)/(N_0 N_T L R)$. Each received vector is first processed by a MIMO decoder, which outputs a soft (coded) bit stream. This bit stream is deinterleaved and then passed to a turbo decoder whose output is passed to the data sink, see Fig. 1. We measure the *computational complexity* of the MIMO decoder by the average number of visited tree nodes per vector decoding.

III. The Sphere Decoding Approach

The first step in the application of efficient tree-based SD algorithms, is expressing $\mathbf{H}$ via the QRD as $\mathbf{H} = \mathbf{QR}$, where $\mathbf{Q}$ is an $N_R \times N_T$ unitary matrix and $\mathbf{R}$ is an $N_T \times N_T$ upper triangular matrix [2]. By left multiplying both sides of (1) with $\mathbf{Q}^H$ we obtain:

$$\mathbf{y}' \triangleq \mathbf{Q}^H \mathbf{y} = \mathbf{R}\mathbf{s} + \mathbf{Q}^H \mathbf{n}. \tag{2}$$

Note that the distribution of the noise vector $\mathbf{Q}^H\mathbf{n}$ is identical to the distribution of $\mathbf{n}$, see [7]. Also, note that $\mathbf{y}'(N_T)$ depends only on the transmitted symbol $\mathbf{s}(N_T)$, $\mathbf{y}'(N_T - 1)$ depends only on the symbols $\mathbf{s}(N_T)$ and $\mathbf{s}(N_T - 1)$, and so on. This structure allows solving the MIMO decoding problem through a tree search where the first layer of the tree corresponds to the equation for $\mathbf{y}'(N_T)$, the second layer corresponds to the equation for $\mathbf{y}'(N_T - 1)$, and so on. As such, decoding requires processing a tree with N_T layers. Let $\hat{\mathbf{s}}$ denote a vector of the estimated transmitted symbols. Using the max-log approximation (see e.g., [9]), the objective of the decoder is to find the vector $\hat{\mathbf{s}}$ with the minimal Euclidean distance metric λ, where

$$\lambda = ||\mathbf{y}' - \mathbf{R}\hat{\mathbf{s}}||^2. \tag{3}$$

During the processing of the tree nodes, decoding complexity is reduced by applying pruning to nodes which pass a certain accumulated metric threshold, see e.g., [3]. Once the processing of the tree is completed, the symbol vector with the lowest metric λ is taken as the *ML hypothesis*. We denote this vector as $\hat{\mathbf{s}}_{\text{ML}}$, and the value of this metric as λ_{ML}. Let $\hat{B}_i \in \{0, 1\}$ be the bit in the i'th location out of the $N_T \cdot L$ bits represented by the vector $\hat{\mathbf{s}}_{\text{ML}}$. Then, for each bit in the ML hypothesis, the *counter-hypothesis* is defined as the symbol vector with the lowest metric λ in which the i'th bit in its binary representation is flipped from its value in the ML hypothesis $\hat{\mathbf{s}}_{\text{ML}}$. We denote the metric of the counter-hypothesis for bit $\hat{B}_i$, computed via (3), as $\lambda_{\text{Counter}}(\hat{B}_i)$. Using the path metrics computed during the processing of the tree, the MIMO decoder generates, for each bit $\hat{B}_i$, $i = 1, 2, \ldots, N_T \cdot L$, a measure called log-likelihood ratio (LLR), which is computed via [7, Eq. (5)]:

$$\text{LLR}(\hat{B}_i) = \left(\lambda_{\text{ML}} - \lambda_{\text{Counter}}(\hat{B}_i)\right)(-1)^{\hat{B}_i}. \tag{4}$$

In recent years, several preprocessing algorithms based on the QRD were developed. One of the most common preprocessing algorithms among the SD-type schemes is the sorted QRD (SQRD) [2]. Another possible matrix decomposition for generating a search tree is the QLD, which decomposes the matrix $\mathbf{H}$ as $\mathbf{H} = \mathbf{QL}$ where $\mathbf{Q}$ is an $N_R \times N_T$ unitary matrix and $\mathbf{L}$ is an $N_T \times N_T$ *lower* triangular matrix. An algorithm that uses both the QRD and the QLD was recently proposed in [5]. In this algorithm, referred to in this work as the QRD-QLD algorithm, instead of a single search that goes through all the layers of the tree, two shorter searches are carried out. One search is implemented on the QRD tree and the other search is implemented on the QLD tree. Each search proceeds up to the middle of its tree (if there is an odd number of layers the QLD tree search will go through one extra layer). During the processing of each of the trees, after processing each layer, all nodes except for the M nodes with the lowest metrics, are pruned. Thus, at the end of each of the two tree searches, the algorithm obtains from each tree a set of M partial vectors, each containing an estimate of only half of the transmitted symbols. Then the QRD-QLD algorithm generates symbol vectors of length N_T by merging all possible combinations of partial vectors from both trees. This results in M^2 vectors of length N_T. The metric of each of these vectors is set to the sum of the metrics of its two partial vectors. Lastly, from the M^2 vectors, the QRD-QLD algorithm selects the vector with the lowest metric to be the ML hypothesis and from the other $M^2 - 1$ vectors the corresponding counter-hypotheses are selected, and then the LLR values are computed from the ML and counter-ML hypotheses.

IV. QR-QL With Improved Path-Metric Computation

A. The Algorithm

Our proposed algorithm uses the QR and QL decompositions with a new method for combining the partial vectors. The new combining leads to a substantially improved performance at the cost of a small increase in the complexity compared to the QRD-QLD algorithm [5]. Thus, we refer to this algorithm as QRD-QLD with improved path metric computation (QRQLIPMC). The QRQLIPMC algorithm consists of three steps: preprocessing, double tree search, and path metric *computation* with final path selection. Next, we describe each step in detail.

Step A—Preprocessing: In the preprocessing step two sets of matrices are generated: $\mathbf{Q}_L$ and $\mathbf{L}$ constituting the QLD of $\mathbf{H}$, and $\mathbf{Q}_R$ and $\mathbf{R}$ constituting the QRD of $\mathbf{H}$. Note that in the QRQLIPMC algorithm it is not possible to use SQRD and SQLD since those algorithms will produce $\mathbf{R}$ and $\mathbf{L}$ matrices whose diagonal elements correspond to the same symbols. Instead of using SQRD, the work [5] proposed to permute the columns of $\mathbf{H}$ at the beginning of the preprocessing step, before applying the QRD and the QLD, such that the column of $\mathbf{H}$ with the highest norm correspond to the first layer of the QLD tree and the last layer of the QRD tree; the column of $\mathbf{H}$ with the second-highest norm correspond to the first layer of the QRD

tree and the last layer of the QLD tree, and so on. Finally, note that it is possible to obtain the QLD using the QRD algorithm via a permutation of the columns of the matrix $\mathbf{H}$, see [5].

Step B—Double Tree Search: After creating the QRD and the QLD of $\mathbf{H}$, the QRQLIPMC algorithm implements a breadth-first search (e.g., as in [3]) on each tree up to the middle of the tree, in the case that N_T is even. In case N_T is odd, the QLD tree will process one extra layer (e.g., for $N_T = 5$ the search on the QRD tree will process two layers and the search on the QLD tree will process three layers). After processing each layer, all but the M nodes with the lowest metrics are pruned. After processing both trees, the QRQLIPMC algorithm will have M partial vectors from each tree, each vector containing an estimate of only part of the transmitted symbols. The two M partial vectors from each search tree are next merged into M^2 vectors, each containing an estimate of the entire N_T transmitted symbols. The metric for each merged vector is set to be the sum of the metrics of the two partial component vectors composing the merged length-N_T vector. Up to this point the algorithm operates as the QRD-QLD algorithm.

Step C—Path Metric Computation: The QRQLIPMC algorithm prunes the M^2 symbol vectors keeping only the K candidate length-N_T vectors with the lowest metrics. Next, the decoder recalculates the metric for each of the K remaining candidate vectors *based on the QLD tree* (note that the metric computation for the first $\lceil N_T/2 \rceil$ symbols of each vector was already done during the QLD tree search in Step B). Once the path computation on the vectors is completed, each vector has a metric computed using *only* the QLD tree. We emphasize that this computation is not a search: From each node there is only one predetermined decedent node. Next, from the set of K vectors, the algorithm chooses the vector with the lowest metric to be the ML hypothesis. From the remaining $K-1$ candidate vectors the algorithm selects the counter hypotheses such that the counter hypothesis for $\hat{B}_i$ is the vector whose metric is the lowest among all non-ML vectors with the i'th bit flipped compared to $\hat{B}_i$. Then, the LLR values are computed via (4). The algorithm is summarized in Algorithm 1.

Algorithm 1: QR-QL With Improved Path Metric Computation

Input : M, K, $\mathbf{H}$, $\mathbf{y}$
Output: LLR values

Apply the QRD to $\mathbf{H}$ and obtain (2) from (1)
for *layers 1 to* $\lfloor N_T/2 \rfloor$ **do**
  Test all the child nodes for every surviving node from the previous layer
  Prune all but the M child nodes with the lowest metrics
end for
Keep the list of the M vectors from this partial search as "QRD partial vectors"
Apply the QLD to $\mathbf{H}$ to obtain $\mathbf{Q}_L^H\mathbf{y} = \mathbf{L}\mathbf{s} + \mathbf{Q}_L^H\mathbf{n}$ from (1)
for *layers 1 to* $\lceil N_T/2 \rceil$ **do**
  Test all the child nodes for every surviving node from the previous layer
  Prune all but the M child nodes with the lowest metrics
end for
Keep the list of the M vectors from the partial search as "QLD partial vectors"
Merge both partial vectors and set the metric of a merged vector as the sum of the metrics of the partial vectors. Continue the QLD tree metric computation only with the K best candidates
Generate the ML and counter-ML hypotheses from the candidates and compute the LLR values

The main benefit of the QRQLIPMC algorithm, compared to the QRD-QLD algorithm, is the increased reliability of the computed path metrics. This follows from the fact that the partial QLD search ignores the information about the first $\lceil N_T/2 \rceil$ symbols, which is embedded in the last $\lfloor N_T/2 \rfloor$ QLD layers. A similar observation holds for the partial QRD search. On the other hand, the QRQLIPMC algorithm exploits this information via the computation which is based on the QLD tree in Step C. Note that this increased reliability comes at the cost of a minor increase in the computational complexity, compared to the QRD-QLD algorithm, as discussed next.

B. Analysis of the Computational Complexity

For each search tree, the number of visited nodes at the first layer is 2^L. Combined over both trees, there are $N_T - 2$ remaining layers, and at each of these layers, the number of visited nodes is $M \cdot 2^L$. Lastly, in the path computation step $\lfloor N_T/2 \rfloor$ layers are processed with K visited nodes at each layer. Thus, the overall computational complexity (the number of visited nodes) of the algorithm is ($M < 2^L$):

$$2 \cdot 2^L + M \cdot 2^L \cdot (N_T - 2) + K \cdot \left\lfloor \frac{N_T}{2} \right\rfloor. \tag{5}$$

Note that the term $K \cdot \lfloor N_T/2 \rfloor$ in (5) represents the incremental computational complexity of the QRQLIPMC algorithm compared to the complexity of the QRD-QLD algorithm. We emphasize that for practical values of computational complexity this increase is negligible. Further note that (5) does not include the complexity of the preprocessing in Step A, since this preprocessing needs to be done once per channel realization, and therefore its complexity can be neglected compared to that of steps B and C. Furthermore, while we measure the algorithm's complexity via the average number of visited nodes per MIMO decoding, as defined in Section II, it is possible to simultaneously process both trees, thereby decreasing by half the run time of the algorithm at the cost of adding hardware (see [5]). Finally, it should be noted that the QRQLIPMC algorithm has a drawback: The maximum number, M, of surviving paths that can be maintained by the algorithm is $2^{L(\lceil N_T/2 \rceil - 1)}$, which is considerably below the maximum number of paths that can be maintained by an algorithm that decodes using a single search tree, which is $2^{L(N_T-1)}$. Thus, the range of BER versus complexity offered by the QRQLIPMC algorithm is bounded from reaching best performance, as this requires maintaining a large number of paths. This disadvantage can be alleviated by taking the parameter K to be much larger than M (up to $2^{L(\lceil N_T/2 \rceil - 1)}$), but this eliminates the advantages of the QRQLIPMC algorithm over algorithms which use a single search.

V. Numerical Simulations and Results

A. Simulation Setup

In the simulations we used a rate 1/2 parallel concatenated convolutional code (PCCC) with feedback polynomial $G_R(D) = 1 + D + D^2$ and feedforward polynomial $G(D) = 1 + D^2$, see [6]. Each transmitted block consisted of 9216 information and tail bits. A 64-QAM constellation and a 4×4 MIMO Rayleigh fading channel were used. For the purpose of complexity comparison we assumed that the preprocessing at Step A is negligible compared to other two steps. The PCCC is decoded by a turbo decoder based on a complex-valued soft-input soft-output BCJR detector [10] with 8 internal

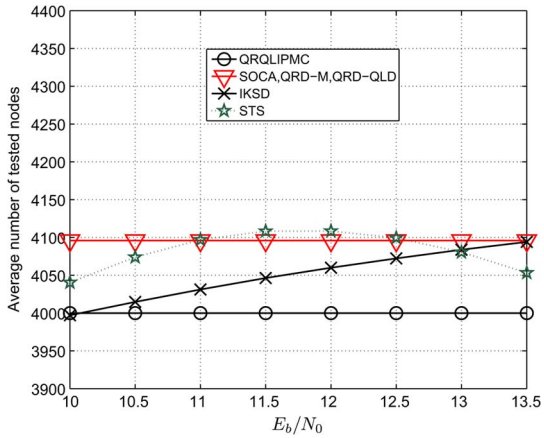

Fig. 2. Complexity vs. E_b/N_0.

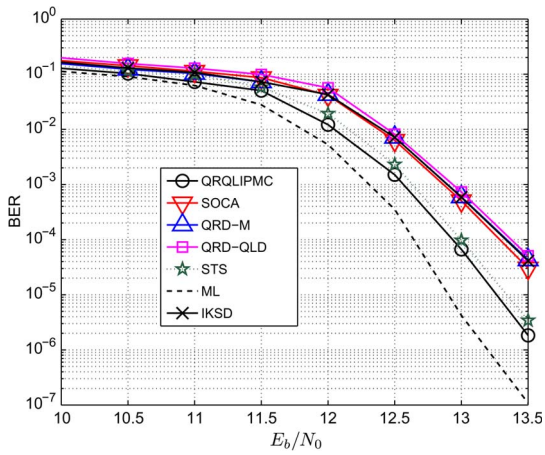

Fig. 3. BER vs. E_b/N_0.

iterations and with no iterations between the MIMO decoder and the turbo decoder. The results were compared with the optimal ML decoder and with five major SD algorithms: single tree search (STS) [7], QRD-M [3], improved K-best sphere decoder (IKSD) [11], smart order candidate adding (SOCA) [4] and QRD-QLD [5]. For each of the tested algorithms, the parameters controlling the tradeoff between complexity and BER were set such that the computational complexity was approximately 4000 visited nodes. It should be noted that in the STS algorithm the $Lmax$ parameter value is different from the values presented at [7], since [7] normalized the noise variance to be one. The parameters of the different algorithms were set as follows: For the SOCA algorithm, the b-parameter (determining how many children nodes will survive from a single father node) was taken to be the same as suggested in [4], while m was taken to be 21, resulting in a constant complexity of 4096 visited nodes. For the QRD-QLD algorithm the value of the parameter M was chosen to be 31, resulting in a constant complexity of 4096 visited nodes. For QRD-M the parameter M was taken to be 21, resulting in a constant complexity of 4096 visited nodes. For the STS algorithm the value of $Lmax$ was set to 0.257, setting the algorithm's complexity to be approximately 4000 (as the complexity of STS varies from search to search, and also depends on SNR_b it cannot be set to a constant value). Finally, for the IKSD algorithm we set K to 24 with a fixed threshold value of 0.2, setting the algorithm's complexity to be approximately 4000 (similarly to STS, IKSD does not have fixed complexity). In our proposed QRQLIPMC algorithm, M was set to 29 and K to 80, resulting in a constant complexity of 4000 visited nodes.

B. Simulation Results

Fig. 2 depicts the complexity of each tested algorithm in the simulations. It can be verified that all the algorithms have approximately the same complexity (up to the differences explained Section V-A). Note that QRQLIPMC has lower complexity compared to the other algorithms. This further shows the strength of the new algorithm at lower complexity. Fig. 3 depicts the performance in terms of BER vs. E_b/N_0 for the tested algorithms. As can be seen from the figure, the results are much in favor of the new algorithm: When complexity is the same, the BER of QRQLIPMC is lower than the BER of all other algorithms. At BER of 10^{-3}, the SNR gain of the new algorithm is 0.3 dB over SOCA and over QRD-QLD, while at BER of 10^{-4} the SNR gain is 0.4 dB. These gains follow as the path metrics created by the new algorithm are much more reliable compared to those created by the QRD-QLD algorithm, see Section IV-A. Furthermore, observe that the performance of the new algorithm is very close to the STS, and a gain of 0.1 dB can be observed at BER 10^{-3}, while in contrast to the STS, the complexity of the new algorithm is fixed.

VI. Conclusion

In this paper we proposed an improved fixed complexity SD algorithm called QRD-QLD with improved path metric computation. The proposed algorithm is shown to achieve an excellent complexity vs. performance tradeoff for practical values of computational complexity.

References

[1] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Math. Comput.*, vol. 44, no. 170, pp. 463–471, Apr. 1985.
[2] D. Wubben, R. Bohnke, V. Kuhn, and K. Kammeyer, "MMSE extension of V-BLAST based on sorted QR decomposition," in *Proc. IEEE VTC*, Oct. 2003, vol. 1, pp. 508–512.
[3] Y. Dai, S. Sun, and Z. Lei, "A comparative study of QRD-M detection and sphere decoding for MIMO-OFDM systems," in *Proc. IEEE Int. Symp. PIMRC*, Berlin, Germany, Sep. 2005, pp. 186–190.
[4] L. Milliner, E. Zimmermann, J. Barry, and G. Fettweis, "A fixed-complexity smart candidate adding algorithm for soft-output MIMO detection," *IEEE J. Sel. Topics Signal Process.*, vol. 3, no. 6, pp. 1016–1025, Dec. 2009.
[5] P. Radosavljevic, K. J. Kim, H. Shen, and J. R. Cavallaro, "Parallel searching-based sphere detector for MIMO downlink OFDM systems," *IEEE Trans. Signal Process.*, vol. 60, no. 6, pp. 3240–3252, Jun. 2012.
[6] B. 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.
[7] C. Studer, A. Burg, and H. Bolcskei, "Soft-output sphere decoding: Algorithms and VLSI implementation," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 2, pp. 290–300, Feb. 2008.
[8] B. Mennenga, A. von Borany, and G. Fettweis, "Complexity reduced soft-in soft-out sphere detection based on search tuples," in *Proc. IEEE Int. Conf. Commun.*, Dresden, Germany, Jun. 2009, pp. 1–6.
[9] M. S. Yee, "Max-log-MAP sphere decoder," in *Proc. IEEE ICASSP*, Philadelphia, PA, USA, Mar. 2005, vol. 3, pp. 1013–1016.
[10] J. Vogy and A. Finger, "Improving the max-log-MAP turbo decoder," *Electron. Lett.*, vol. 36, no. 23, pp. 1937–1939, Nov. 2000.
[11] S. Han, T. Cui, and C. Tellambura, "Improved K-best sphere detection for uncoded and coded MIMO systems," *IEEE Wireless Commun. Lett.*, vol. 1, no. 5, pp. 472–475, Oct. 2012.