A Stack algorithm with limited tree-search

Rym Ouertani, *Student Member IEEE*, Ghaya Rekaya Ben-Othman, *Member IEEE*
TELECOM ParisTech, 46 rue Barrault, 75013 Paris - France
{ouertani, rekaya}@telecom-paristech.fr

***Abstract*—Due to the capacity achievable, Multiple-Input Multiple-Output (MIMO) systems witnessed more interest in recent years. Several data detection algorithms are available for MIMO systems, especially the tree-search ones which offer lower complexity comparing to the ML receiver. In this paper, we are interested in one of tree-search algorithms, which is the Stack decoder. Nevertheless, the stack decoder is only proposed to decode MIMO systems employing finite constellations, as the QAM (Quadrature Amplitude Modulation). In this work, we introduce a new Stack decoder capable of decoding lattice and we call it the M-Stack decoder. The principle idea behind it is to generate M nodes at each tree-level and so to reduce the infinite alphabet of the lattice to only M components at each step. Then, we propose to adapt it to decode QAM constellations. This one is compared to the known QRD-Stack algorithm. While both visit the same number of nodes, the proposed M-Stack algorithm offers lower computational complexity and it is much faster than the QRD-Stack one.**

***Keywords:* MIMO, lattice decoding, tree-search algorithm, Stack decoder, complexity, QRD-Stack.**

I. Introduction

In this work, we are interested in the decoding of multi-antenna systems using spatial multiplexing [1] and linear space time block codes (STBC). In [2], a lattice representation of multi-antenna schemes has been presented, which makes it possible to decode such systems using lattice decoders. In the literature, different decoders are proposed such as the sphere decoder [3] and the Schnorr-Euchner algorithm [4]. In [5], another kind of decoders, based on sequential algorithms especially the Stack and the Fano ones were also used to decode MIMO systems. However, they are proposed to decode finite constellations like QAM and BPSK modulation but they are unsuitable for decoding codewords belonging to an infinite alphabet like lattice.

In this work, we propose a new Stack decoder that overcomes this problem. We then introduce the $M-$Stack algorithm. The purpose from it is to reduce the lattice to a finite alphabet. This is done by considering only one finite region in the lattice.

We focus here on the Stack decoder rather than the Fano one, as it was shown in [6] that the Stack decoder is six times faster and less computationally complex than the Fano. These observations are ones of our main interests in the Stack decoder.

Furthermore, we show that we can use the proposed algorithm to decode finite constellations. We compare then the $M-$Stack to the well known QRD-Stack algorithm [7]. We show that the two algorithms have several similarities however the $M-$Stack is less complex than the first one.

This paper is organized as follows. In section II, we introduce the system model. In section III, we recall the idea behind the conventional Stack and the QRD-Stack decoders. Then in section IV, we present the proposed $M-$Stack algorithm for lattice. This one will be further extended to the case of finite constellations. In section V, we give the simulation results, and finally we draw the conclusions in section VI.

II. System Model and notations

A. MIMO scheme with spatial multiplexing (SM)

Let us consider a MIMO system with n_t transmit and n_r receive antennas using spatial multiplexing scheme. The channel is assumed to be quasi-static, and the received signal is given by

$$\boldsymbol{y}^{\boldsymbol{c}}_{\boldsymbol{n_r}} = \boldsymbol{H}^{\boldsymbol{c}}_{\boldsymbol{n_r \times n_t}} \cdot \boldsymbol{x}^{\boldsymbol{c}}_{\boldsymbol{n_t}} + \boldsymbol{w}^{\boldsymbol{c}}_{\boldsymbol{n_r}} \quad (1)$$

where $\boldsymbol{H^c}$ is the channel transfer matrix with complex entries h_{ij} representing the fading coefficients between the i^{th} receive and the j^{th} transmit antennas and are modeled by independent Gaussian random variables of zero-mean and variance 0.5 per real and imaginary component. $\boldsymbol{x^c}$ is the complex information vector.

We call lattice case if the information symbols $\boldsymbol{x^c_i}$,$i = 1, \ldots, n_t$, are carved in $\mathbb{Z}[\iota]$, $\iota^2 = -1$. Otherwise, they are carved in finite constellations, such as the QAM constellation. For example for a 4-QAM, the real and imaginary parts of each component $\boldsymbol{x^c_i}$ belong to the interval $[-1, +1]$. However, for large constellation sizes such as 256-QAM, we can assume that we are in the lattice case as the real and imaginary parts of $\boldsymbol{x^c_i}$belong to a large interval $[-15, +15]$. $\boldsymbol{w^c}$ represents the i.i.d complex additive white Gaussian noise vector with zero-mean and variance σ^2.

A representation of the multi-antenna scheme by a lattice packing was proposed in [8]. This one is obtained by separating the real and imaginary parts as

$$\boldsymbol{y} = \begin{bmatrix} \Re(\boldsymbol{y^c}) \\ \Im(\boldsymbol{y^c}) \end{bmatrix} = \begin{bmatrix} \Re(\boldsymbol{H^c}) & -\Im(\boldsymbol{H^c}) \\ \Im(\boldsymbol{H^c}) & \Re(\boldsymbol{H^c}) \end{bmatrix} \cdot \begin{bmatrix} \Re(\boldsymbol{x^c}) \\ \Im(\boldsymbol{x^c}) \end{bmatrix} + \begin{bmatrix} \Re(\boldsymbol{w^c}) \\ \Im(\boldsymbol{w^c}) \end{bmatrix} \quad (2)$$

$$\boldsymbol{y} = \boldsymbol{H} \cdot \boldsymbol{x} + \boldsymbol{w} \quad (3)$$

$\boldsymbol{H}$ is therefore the equivalent lattice generator matrix of dimension $2n_r \times 2n_t$.

B. MIMO scheme with STBC

We now consider a coded system using a linear space time block code [9], such as the TAST codes [10] and the perfect codes [11][12]. In most current wireless standards (Wi-Fi, WiMAX and LTE), MIMO is combined with channel coding to further improve the system diversity (robustness). The received signal matrix is then given by

$$\boldsymbol{Y^c_{n_r \times T}} = \boldsymbol{H^c_{n_r \times n_t}} \cdot \boldsymbol{C^c_{n_t \times T}} + \boldsymbol{W^c_{n_r \times T}} \quad (4)$$

where $\boldsymbol{C^c_{n_t \times T}}$ represents the codeword matrix. We consider a MIMO symmetric system, *i.e*, $n_t = n_r$, and a square code, which means that the temporal code length T is equal to n_t. The lattice representation is obtained here by vectorization and separation of the real and imaginary parts of the received signal $\boldsymbol{Y^c_{n_r \times T}}$ [11]. The equation (4) becomes therefore

$$\boldsymbol{y^c_{n_r \cdot T}} = \begin{pmatrix} \boldsymbol{H^c_{n_r \times n_t}} & & 0 \\ & \ddots & \\ 0 & & \boldsymbol{H^c_{n_r \times n_t}} \end{pmatrix} \cdot \begin{pmatrix} \phi_{11} & \cdots & \phi_{1,n_tT} \\ \vdots & \ddots & \vdots \\ \phi_{n_tT,1} & \cdots & \phi_{n_tT,n_tT} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ \vdots \\ x_{n_tT} \end{pmatrix} + \begin{pmatrix} w_1 \\ \vdots \\ w_{n_rT} \end{pmatrix}$$

We then get an equivalent system to (1)

$$\begin{aligned} \boldsymbol{y^c_{n_r \cdot T}} &= \boldsymbol{H^c_{1,n_r \cdot T \times n_t \cdot T}} \cdot \boldsymbol{\phi^c_{n_t \cdot T \times n_t \cdot T}} \cdot \boldsymbol{x^c_{n_t \cdot T}} + \boldsymbol{w^c_{n_r \cdot T}} \\ &= \boldsymbol{H'^c_{n_r \cdot T \times n_t \cdot T}} \cdot \boldsymbol{x^c_{n_t \cdot T}} + \boldsymbol{w^c_{n_r \cdot T}} \end{aligned} \quad (5)$$

Then, the separation of the real and imaginary parts is applied on the former equation as in (2), and the coded system is therefore given by

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{H'_{n \times n}} \cdot \boldsymbol{x_n} + \boldsymbol{w_n} \\ &= \boldsymbol{H} \cdot \boldsymbol{x} + \boldsymbol{w} \end{aligned} \quad (6)$$

where we define by $\boldsymbol{H} = \boldsymbol{H'_{n \times n}}$ the equivalent lattice generator matrix of dimension $n = 2{n_t}^2$.

C. Decoding scheme

Under the assumption of a perfect knowledge of the channel state information at the receiver, the ML decoding rule is given by

$$\hat{\boldsymbol{x}} = arg \left(\min_{x \in \mathbb{Z}^n or\, x \in (QAM)^n} \|y - \boldsymbol{H} \cdot \boldsymbol{x}\|^2 \right) \quad (7)$$

The obtained system model can then be decoded by several decoders such as the sphere decoder and the Schnorr-Euchner algorithm [3][4] or the Stack decoder which are basically tree-search algorithms.

More generally, to apply tree-search, we need first to expose the tree structure. A QR or a Cholesky decomposition can be then applied on the lattice generator matrix $\boldsymbol{H}$. These two methods are quite equivalent, however the QR is more complex than the Cholesky decomposition but it is more stable numerically [13]. So, we apply the QR decomposition, the system can be written as

$$\boldsymbol{y} = \boldsymbol{Q} \cdot \boldsymbol{R} \cdot \boldsymbol{x} + \boldsymbol{w} \quad (8)$$

where $\boldsymbol{Q}$ is an orthogonal matrix and $\boldsymbol{R}$ an upper triangular one. The multiplication of both sides of (8) by the transpose of $\boldsymbol{Q}$ does not change the decoding problem and we get

$$y_1 = \boldsymbol{Q^\dagger} \cdot \boldsymbol{y} = \boldsymbol{R} \cdot \boldsymbol{x} + \boldsymbol{Q^\dagger} \cdot \boldsymbol{w}$$

$\boldsymbol{R}$ is now the equivalent lattice matrix. Exploiting the upper triangular form of $\boldsymbol{R}$, one can solve the decoding problem using a tree search algorithm and the ML solution to look for is then given by

$$\hat{\boldsymbol{x}} = arg \left(\min_{x \in \mathbb{Z}^n or\, x \in (QAM)^n} \|\boldsymbol{y_1} - \boldsymbol{R} \cdot \boldsymbol{x}\|^2 \right) \quad (9)$$

Throughout this paper, we consider a tree rooted at a fictive node x_{root}. The node at level k is denoted by the vector $x^{(k)} = (x_n, x_{n-1}, ..., x_k)$ where $x_j, j = 1, \ldots, n$ are the components of $\boldsymbol{x}$. Moreover, the branches of the tree at level k define all the possible values that can be taken by x_k, and each node $x^{(k)}$ is associated with the squared distance

$$f(x^{(k)}) = \sum_{i=k}^{n} f_i(x_i) \quad (10)$$

where $f_i(x_i) = \left| y_{1i} - \sum_{j=i}^{n} r_{i,j} x_j \right|^2$. We call $f(x^{(k)})$ the cost function of the node $x^{(k)}$. It represents the "sub-distance" between the received and the transmitted signal at the level k. The tree search consists in exploring the tree nodes in order to find the path $(x_n, x_{n-1}, ..., x_1)$ with the least cost as shown in the figure 1.

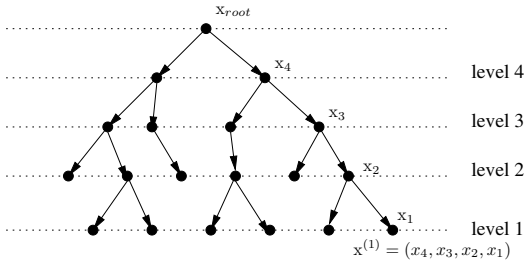


Figure 1. Example of a tree structure with a dimension n=4

III. Overview of tree search-algorithms: the Stack decoder and the QRD-Stack decoder

We recall here the main ideas of these two decoders as existing in the literature.

A. Stack decoder

The Stack decoder belongs to the class of sequential decoders that were originally proposed to decode binary trellis codes [14]. To describe the Stack algorithm, let us suppose that the decoder is started at a fictive root node x_{root}. The Stack algorithm generates all the child nodes of x_{root}. We call a child node the successor node which leads to the next level of the tree. For example, by using finite constellations such as a $16-QAM$ constellation, each tree node belongs to the set $I_c = \{-3, -1, +1, +3\}$, then we have a $4-ary$ tree with 4 nodes at each level. More generally, for a $q-QAM$, the nodes to consider are in $I_c = \left\{\pm 1, \pm 3, \ldots, \pm\sqrt{q}-1\right\}$.

The algorithm computes then the respective costs of those nodes according to the cost function in (10) and stores them in an increasing order in the stack, so that the top node of the stack is the one having the least cost. Afterward, the algorithm generates the children of the top node, computes their costs, places them in the stack and removes the top node being just extended. The algorithm reorders again the stack, generates the children nodes of the current top node, and so on. The algorithm terminates when a leaf node ($i = 1$) is found on the top of the stack.

As explained above, the Stack algorithm conducts a best-first-search strategy, so that the solution provided is the one with the least metric which corresponds to the ML solution. The different steps of the Stack algorithm can be described as:

1) Start from the root node with zero accumulative metric, and $i = n$ (the first detection stage).
2) Extend branch on top of stack to all possible nodes and remove parent branch.
3) Order resulting branches based on their metrics.
4) If the last detection stage is reached ($i = 1$), go to step 6 otherwise continue.
5) Move to the next stage ($i = i - 1$), and go to step 2.
6) Order survival branches based on their accumulative metrics and retain the best branch.

B. QRD-Stack algorithm

The QRD-Stack algorithm performs the same algorithm than the Stack one however it retains only a pre-defined number of branches at each level, the M best ones which have the M smallest metrics, instead of all the possible branches [7]. As a result, the number of visited nodes of the algorithm becomes deterministic for fixed problem size and more reduced than the first one especially for high constellation sizes like 64-QAM and 256-QAM.

Nevertheless, the nodes retained may not lead to the shortest path but discarded by the algorithm. Consequently, the ML solution is not reached. So, this algorithm allows to limit the number of the generated nodes but at the price of a performance loss. We reformulate here the QRD-Stack algorithm as following steps:

1) Start from the root node with zero accumulative metric, and $i = n$ (the first detection stage).
2) Extend branch on top of stack to all possible nodes and remove parent branch.
3) Order resulting branches based on their metrics, retain the M best branches with the least metrics.
4) If the last detection stage is reached ($i = 1$), go to step 6 otherwise continue.
5) Move to the next stage ($i = i - 1$), and go to step 2.
6) Order survival branches based on their accumulative metrics and retain the best branch as the solution of the QRD-Stack algorithm.

IV. The proposed $M-$Stack decoder

From the two former algorithms, we can see that they can be used only if we use finite constellations. The principle is to enumerate the nodes belonging to the constellation which contains a finite number of components.

However, considering a lattice, the symbols are defined in an infinite field $\mathbb{Z}^n$, each component $x_i \in \mathbb{Z}$ which leads to an infinite tree structure. So, it is impossible to enumerate an infinite number of nodes which makes it impossible to use these algorithms. In the following, we propose the $M-$Stack algorithm which allows to resolve this problem.

A. Lattice decoding

In order to apply the Stack decoder, we should search for the solution in a finite region $\Lambda \subset \mathbb{Z}^n$ and discard the rest. Unfortunately, the truncation of the tree will affect the performances. In fact, the transmitted symbol may be out of the search region and then the solution is discarded by the algorithm. The main challenge is then how to choose the optimal region Λ.

Yet, the triangular form of the lattice basis $\boldsymbol{R}$ reminds us the Schnorr-Euchner enumeration strategy [15]. The key idea

of this algorithm is to view the lattice as a superposition of n hyperplanes and to start the search by projecting the received vector on the nearest hyperplane (level n in the tree). The resulting point is then recursively projected on the following $n-1$ hyperplanes. The point found is the "babai point", it corresponds to the ZF-DFE solution [8]. Then, the Schnorr-Euchner algorithm consists in generating the nodes by zigzagging around the babai point using the depth-first-search strategy.

In the proposed $M-$Stack algorithm, we were inspired by the Schnorr-Euchner principle, however the search strategy and the construction of the tree are quite different.

In fact, the algorithm starts by projecting the vector y_1 on the n^{th} hyperplane. The resulting point z_n that corresponds to the n^{th} component of the babai point is then considered as the first node generated in the tree-level n:

$$z_n = \left\lceil \frac{y_{1n}}{r_{n,n}} \right\rfloor \tag{11}$$

where $\lceil x \rfloor$ rounds x to the nearest integer. Then, the algorithm enumerates M nodes x_n of the tree, centered at z_n by zigzagging around it as follows:

$$x_n \in \{z_n, z_n \pm 1, z_n \pm 2, z_n \pm 3, ...z_n \pm (M-1)/2\} \tag{12}$$

So that, at the level n, we get exactly M nodes. Then, the $M-$Stack algorithm calculates their metrics and stores them in the stack. As in the conventional Stack algorithm, the top will be extended. First, the top is projected on the hyperplane $n-1$, we obtain the point z_{n-1} as follows:

$$z_{n-1} = \left\lceil \frac{y_{1n-1} - r_{n-1,n} x_n}{r_{n-1,n-1}} \right\rfloor$$

(We note that z_{n-1} does not correspond necessarily to the $(n-1)^{th}$ component of the babai point as the top node x_n may be different of z_n). The M nodes x_{n-1} to consider at level $n-1$ are computed as the same manner as in (12), then they will be stored in the stack and the top removed. If the top is at level i, it is projected on the hyperplane i and so on. The algorithm stops when the leaf node is reached.

In the figure 2, we represent an example of tree construction where $n = 4$ and $M = 3$.

For the ease of understanding, the $M-$Stack can be summarized as follows:

1) Start from the root node with zero accumulative metric, and $i = n$ (the first detection stage). $x = top = []$.
2) $z_i = \left\lceil \left(y_{1i} - \sum_{j=i}^{n} r_{i,j} x_j \right) / r_{i,i} \right\rfloor$. Enumerate M nodes around z_i.
3) Order resulting branches based on their metrics, $x = top$.
4) If the last detection stage is reached ($i = 1$), go to step 6 otherwise continue.
5) Move to the next stage ($i = i - 1$), and go to step 2.
6) Order survival branches based on their accumulative metrics and retain the best branch as the solution of the $M-$Stack algorithm.

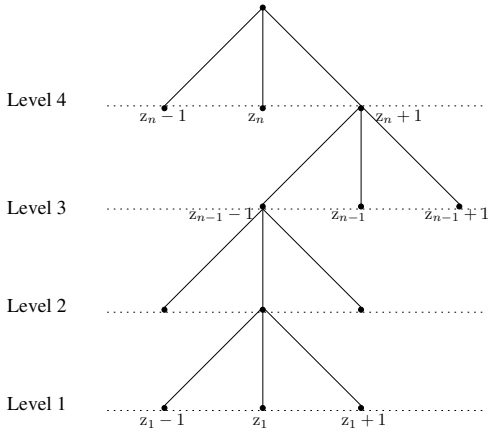


Figure 2. Example of a tree construction with a dimension $n = 4$ and $M = 3$

By applying this algorithm, we get a finite tree and so the Stack algorithm can be performed. However, the ML solution is not guaranteed to be included in the considered tree. To reach it, we should enlarge the search region. Meanwhile, that implies to have a denser tree which leads to a more complex decoding task.

B. Decoding finite constellations

We propose here to adapt the $M-$Stack to the case of finite constellations. In this case, the algorithm should simply check if the generated branches belong to the constellation interval I_c. Then, the $M-$Stack algorithm can be modified as follows:

1) Start from the root node with zero accumulative metric, and $i = n$ (the first detection stage). Put $x = top = []$.
2) $z_i = \left\lceil \left(y_{1i} - \sum_{j=i}^{n} r_{i,j} x_j \right) / r_{i,i} \right\rfloor$. Enumerate M ((nodes around z_i) $\cap I_c$). ($\lceil x \rfloor$ is the demodulation of the float x in the considered QAM constellation)
3) Order resulting branches based on their metrics. Put $x = top$.
4) If the last detection stage is reached ($i = 1$), go to step 6 otherwise continue.
5) Move to the next stage ($i = i - 1$), and go to step 2.
6) Order survival branches based on their accumulative metrics and retain the best branch as the solution.

In this algorithm, we have considered a fixed number of nodes generated at each stage (M=constant). One can think

to use large numbers M_i for first levels and small M_i for the last ones. This choice may be efficient due to the problem of error propagation in the tree search.

In the next section, the performances of the proposed $M-$Stack decoder will be evaluated.

V. Simulation results

In this section, we give some simulation results obtained by considering a MIMO system using SM by means of Monte-Carlo simulations.

In the figure 3.a, we plot the symbol error rate (SER) as a function of the signal to noise ratio (SNR) given in dB scale for a 2×2 MIMO system in the case of a lattice defined in $\mathbb{Z}^n$, when performing the $M-$Stack algorithm. The different curves show that as well as we increase the search region (*i.e* we consider larger M) the performances are closer to ML, however the complexities are more and more important as shown in the figure 3.b. The complexity is computed as the total number of multiplications needed to decode one symbol.

Therefore, a compromise may be done and this decoding algorithm can be of great interest. In fact, at the start of the algorithm, one only needs to choose the performance-complexity trade-off to reach, to define the appropriate M to consider.

In the figure 4.a, we evaluate the $M-$Stack algorithm in the case of using QAM constellations. We consider a 4×4 MIMO system using 16-QAM constellation and we compare it to the QRD-Stack and the conventional Stack one. For $M = 4$, the three algorithms generate all the constellation nodes $(-3, -1, +1, +3)$. In this case, the ML solution is not discarded and the three algorithms give exact ML performance. The QRD-Stack and the $M-$Stack are then equivalent to the conventional Stack algorithm and consequently they exhibit the same complexity.

For $M \leq 4$, the conventional Stack generates more nodes than the others. Therefore, the QRD-Stack and the $M-$Stack give less complexity than it. However, they exhibit sub-optimal performances since the ML solution is not among the M retained nodes. The performance decreases as well as M decreases, however the complexity is more and more reduced as reported in the figure 4.b. Besides, we can see for all algorithms that the complexity decreases rapidly with the increase of SNR.

Furthermore, we can see that for the same number of generated nodes (M is the same for both algorithms), the QRD-Stack presents better performances than the $M-$Stack and the gap in performances is more and more important than M decreases. This is due to the fact that the QRD-Stack algorithm retains better nodes than the $M-$Stack one. This can be also seen in the figure 5 while using a 64-QAM constellation.

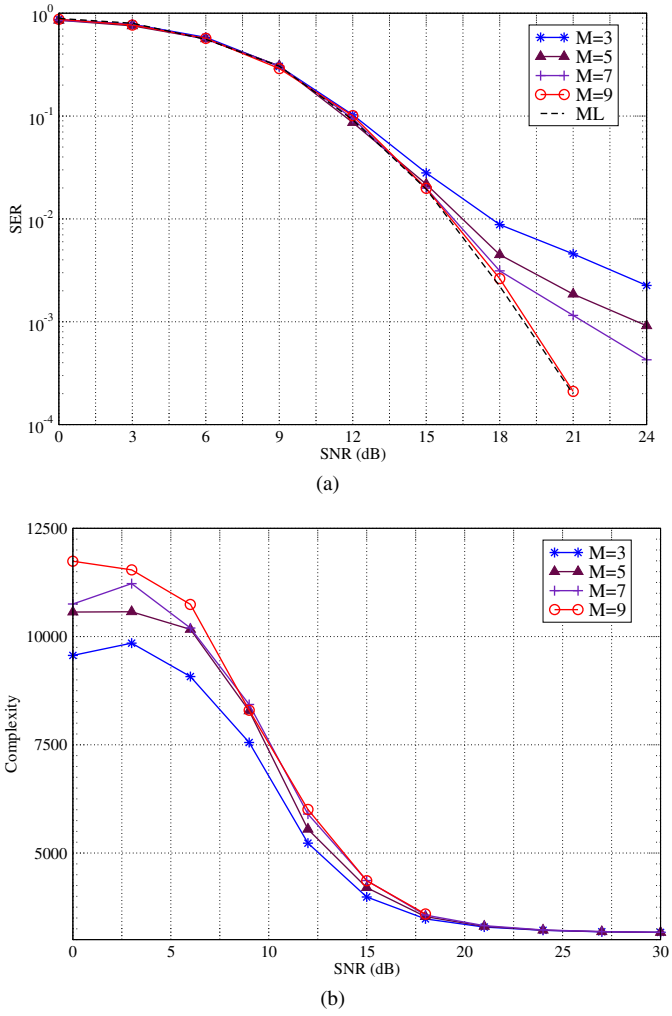
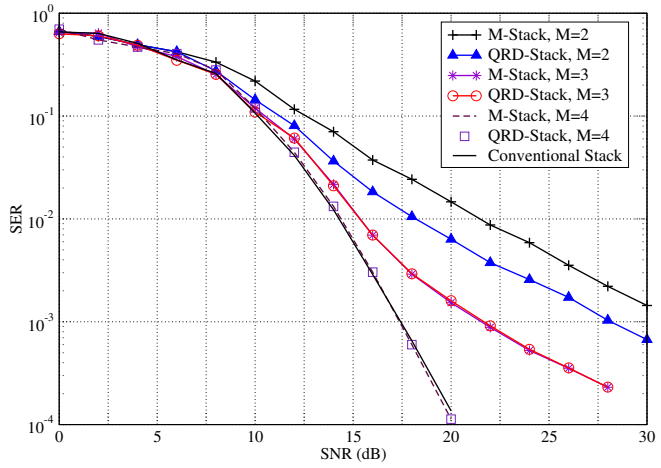


Figure 3. Performance and complexity of the $M-$Stack decoder for a 2×2 MIMO system with SM

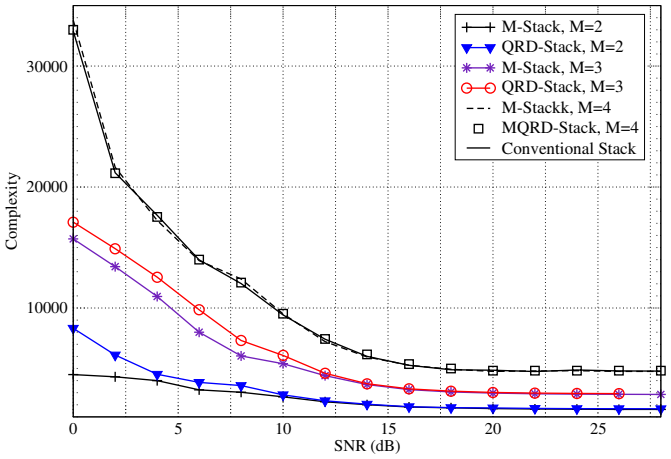
However, for a sufficiently high value of M, the $M-$Stack gives the same performance as the QRD-Stack, as we can see in the figure 4.a for $M = 3$. But in all cases, the $M-$Stack offers a lower complexity. This can be explained by the fact that the QRD-Stack starts by calculating all the constellation nodes, which presents an additional computational complexity, then it retains the M best ones. Whereas the $M-$Stack only needs to compute the point z_i and generates directly the M nodes by zigzagging around it. So, the $M-$Stack decoder represents an interesting alternative to the QRD-Stack algorithm especially for high values of M.

VI. Conclusion

In this paper, we have introduced the $M-$Stack decoder for detection in a MIMO communication system. The main



(a)



(b)

Figure 4. Performance and complexity of the $M-$Stack decoder for a 4×4 MIMO system with SM using 16-QAM constellation

advantage of this decoder is that it is suitable for decoding both lattice and finite constellations. While the existing decoders, like the QRD-Stack and the conventional Stack, only decode finite constellations. The performances of this algorithm for various M values were also presented and compared to both QRD-Stack and conventional Stack decoders. It was shown that these ones reach ML performance at high values of M. However, the $M-$Stack is shown to achieve such performance with less computational complexity. This complexity gain is more and more important as we increase the constellation size ans the system dimensions.

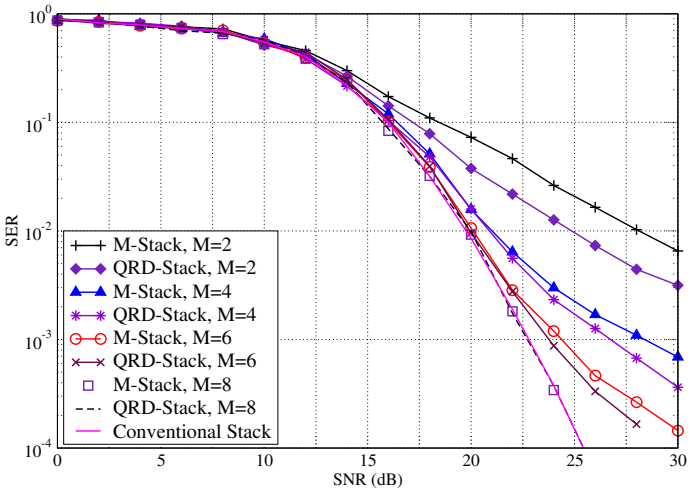


Figure 5. Performance of the $M-$Stack decoder for a 4×4 MIMO system with SM using 64-QAM constellation

REFERENCES

[1] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: an architecture for realizing very high data rates over the rich-scattering wireless channel," Bell Labs, Lucent Technologies, Crawford Hill Laboratory 791 Holmdel-Keyport RD., Holmdel, NJ07733.

[2] M. O. Damen, A. Chkeif, and J.-C. Belfiore, "Lattice code decoder for space-time codes," *IEEE Communications Letters*, vol. 4, pp. 161–163, May 2000.

[3] J. Boutros and E. Viterbo, "A universal lattice code decoder for fading channels," *IEEE Transactions On Information Theory*, vol. 45, pp. 1639–1642, July 1999.

[4] C. Schnorr and M. Euchner, "Lattice basis reduction: improved practical algorithms and solving subset sum problems," *Mathematical Programming*, vol. 66, pp. 181–199, September 1994.

[5] A. D. Murugan, H. ElGamal, M. O. Damen, and G. Caire, "A unified framework for tree search decoding: rediscovering the sequential decoder," *IEEE Transactions On Information Theory*, vol. 52, pp. 933–953, March 2006.

[6] F. Jelinek, "Fast sequential decoding algorithm using a stack," *IBM J. Res. Develop.*, vol. 13, pp. 675–685, November 1969.

[7] W. Chin, ""QRD Based Tree Search Data Detection for MIMO Communication Systems"," *Vehicular Technology Conference*, vol. 3, pp. 1624–1627, June 2005.

[8] M. O. Damen, H. ElGamal, and G. Caire, "On maximum likelihood detection and the search for the closest lattice point," *IEEE Transactions on Information Theory*, pp. 2389–2402, October 2003.

[9] B. Hassibi and B. M. Hochwald, "Linear dispersion codes," *Proceedings of the IEEE International Conference on Information Theory*, p. 325, June 24-29 2001.

[10] H. ElGamal and M. O. Damen, "Universal space-time coding," *IEEE Transactions On Information Theory*, May 2003.

[11] J.-C. Belfiore, G. Rekaya, and E. Viterbo, "The Golden Code: a 2x2 full rate space-time code with non-vanishing determinants," *IEEE Transactions on Information Theory*, vol. 51, pp. 1596–1600, November 2004.

[12] F. Oggier, G. Rekaya, J.-C. Belfiore, and E. Viterbo, "Perfect space-time block codes," *IEEE Transctions on Information Theory*, vol. 52, pp. 3885–3902, September 2006.

[13] "Numerical Recipes," *http://www.nr.com*.

[14] R. Johannesson and K. S. Zigangirov, "Fundamentals of convolutional coding," *IEEE series on digital and mobile communication*, pp. 269–315, 1999.

[15] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Transactions On Information Theory*, vol. 48, pp. 2201–2214, August 2002.