A tree-search algorithm for ML decoding in underdetermined MIMO systems

Gianmarco Romano [#1], Francesco Palmieri [#2], Pierluigi Salvo Rossi [#3], Davide Mattera [*4]

[#] *Dipartimento di Ingegneria dell'Informazione, Seconda Università di Napoli*
via Roma 29, 81031 Aversa, ITALY
[1] gianmarco.romano@unina2.it
[2] francesco.palmieri@unina2.it
[3] pierluigi.salvorossi@unina2.it

[*] *Dipartimento di Ingegneria Biomedica, Elettronica e delle Telecomunicazioni, Università di Napoli "Federico II",*
via Claudio 21, 80125 Napoli, ITALY
[4] mattera@unina.it

***Abstract*—It is well known that Maximum Likelihood (ML) detection for multiantenna and/or multiuser systems has complexity that grows exponentially with the number of antennas and/or users. A number of suboptimal algorithms has been developed in the past that present an acceptable computational complexity and good approximations of the optimal solution. In this paper we propose a tree-search algorithm that provides the *exact* ML solution with lower computational complexity than that required by an exhaustive search of minimum distance. Also a two-stage tree-search algorithm is presented based on the idea that the ML solution is in the set of equilibrium points of a Hopfield Neural Networks (HNN). The two algorithms work without any modification both in underloaded and overloaded (underdetermined) systems. Numerical simulations show that improvements, in terms of computational complexity measured as the average number of required sum and/or products, are encouraging.**

I. Introduction

Given an N-dimensional observed vector

$$\mathbf{y} = \mathbf{G}\mathbf{x} + \mathbf{w} = \mathbf{g}_1 x_1 + \mathbf{g}_2 x_2 + ... + \mathbf{g}_M x_M + \mathbf{w}, \qquad (1)$$

where $\mathbf{w}$ is Gaussian noise, $x_j \in \{-1, 1\}$, $j = 1, ..., M$ and $\mathbf{g}_j$, $j = 1, ..., M$ are users' signatures, the problem of optimal decoding $\mathbf{x}$ from $\mathbf{y}$, i.e.

$$\mathbf{x}_{ML} = \text{argmin}_{\mathbf{x}\in\{-1,1\}} \|\mathbf{y} - \mathbf{G}\mathbf{x}\|_2 \qquad (2)$$

is known to be exponentially complex, as the worst-case computational cost grows exponentially in the number of users [1]. Eq. (1) is a general model that can represent a variety of communication systems, as for example CDMA and MIMO systems. A number of suboptimal algorithm have been developed as low-complexity alternatives to the ML decoding. Optimal and approximately-optimal solutions are available without prohibitive computational cost via branch and bound techniques [2], sphere decoding [3], lattice-based sub-optimal approaches [4] and other tree-search algorithms as the A* algorithm [5]. These algorithms perform well for underloaded systems, i.e. when the number of users M is less then the signal space dimension N, and their use in overloaded or underdetermined systems, i.e. when $M > N$, is not always possible. Several algorithms have been developed to tackle the problem of optimal and sub-optimal decoding of underdetermined systems. In [6] an extension of the sphere-decoding is proposed based on a geometrical condition. In [7] an efficient tree-search algorithm for underdetermined system is presented (see also references therein).

A different approach to ML decoding is represented by the use of Hopfield Neural Networks (HNN) as proposed in [8], [9]. In [8] it has been shown that the ML solution can be obtained through dynamic update of the discrete-time approximation of the equation of motion of neurons. That equation may present limit cycles problems when the update is done in parallel. Furthermore the update rule may not provide the ML solution since in many cases solution is not unique and may be a local minimum. In order to prevent the updating rule to enter in a limit cycle and to force the dynamic update through increasing likelihood in [10] a modified HNN approach to ML decoding is proposed, leading to a family of likelihood ascent sub-optimal detectors (LAS). These algorithms are sub-optimal and can approach optimal performances under specific conditions.

We propose two new algorithms for ML decoding based on a tree-search that gives *optimal* solutions for both underloaded and overloaded systems. Both algorithms are based on a dominance condition, derived from distance computation but with lower computational complexity, that can be checked at each node of the tree to reduce the number of visited nodes and then of the paths on the tree. The second algorithm is based not only on the dominance condition, but also on the use of the update rule for a HNN as a necessary condition for the optimal ML solution in order to reduce the number of surviving paths. We show that both algorithms present lower computational complexity with respect to that required by an exhaustive minimum distance search.

The paper is organized as follows: in Sec. II we present the two algorithms for ML decoding; in Sec. III we analyze the computational complexity of both algorithms, measured in terms of sums and/or products, and show simulation results; some concluding remarks are given in Sec. IV.

II. Minimum Distance Decoding

Consider the problem (2) and suppose we divide the users in two sets a and b with M_a and M_b users respectively ($M_a + M_b = M$)

$$\mathbf{G} = [\mathbf{G}_a \mathbf{G}_b], \quad \mathbf{x}^T = \left[\mathbf{x}_a^T \mathbf{x}_b^T\right] \tag{3}$$

The squared distance can be written as

$$\begin{aligned}\|\mathbf{y} - \mathbf{G}\mathbf{x}\|_2 = \mathbf{y}^T\mathbf{y} + \mathbf{x}_a^T\mathbf{R}_a\mathbf{x}_a - 2\mathbf{y}^T\mathbf{G}_a\mathbf{x}_a \\ + 2\mathbf{x}_a^T\mathbf{R}_{ab}\mathbf{x}_b + \mathbf{x}_b^T\mathbf{R}_b\mathbf{x}_b - 2\mathbf{y}^T\mathbf{G}_b\mathbf{x}_b,\end{aligned} \tag{4}$$

where

$$\mathbf{R}_a = \mathbf{G}_a^T\mathbf{G}_a, \quad \mathbf{R}_b = \mathbf{G}_b^T\mathbf{G}_b, \quad \mathbf{R}_{ab} = \mathbf{G}_a^T\mathbf{G}_b. \tag{5}$$

Now suppose we select only one user, say user j, in set a, ($M_a = 1$). The squared distance can be written as

$$\begin{aligned}||\mathbf{y} - \mathbf{G}\mathbf{x}||_2 = \mathbf{y}^T\mathbf{y} + x_j^2 r_{jj} - 2\mathbf{y}^T\mathbf{g}_j x_j \\ + 2x_j\mathbf{R}_{jb}\mathbf{x}_b + \mathbf{x}_b^T\mathbf{R}_b\mathbf{x}_b - 2\mathbf{y}^T\mathbf{G}_b\mathbf{x}_b.\end{aligned} \tag{6}$$

The decision rule is: $x_j = 1$ if

$$-2\mathbf{y}^T\mathbf{g}_j + 2\mathbf{R}_{jb}\mathbf{x}_b > 2\mathbf{y}^T\mathbf{g}_j - 2\mathbf{R}_{jb}\mathbf{x}_b \tag{7}$$

or

$$-\mathbf{y}^T\mathbf{g}_j + \mathbf{R}_{jb}\mathbf{x}_b < 0. \tag{8}$$

Decision on x_j will not depend on $\mathbf{x}_b$ if

$$|\mathbf{y}^T\mathbf{g}_j| > \sup_{\mathbf{x}_b \in \{-1,1\}^{M-1}} |\mathbf{R}_{jb}\mathbf{x}_b|, \tag{9}$$

or

$$|\mathbf{y}^T\mathbf{g}_j| > \sum_{i \neq j} |r_{ji}|. \tag{10}$$

We say that, on observation $\mathbf{y}$, user j is *dominant* over his multi-user interference.

The condition (10) represents a sufficient condition for an optimal decision to be made, as the multiuser interference is so small that does not affect the decision on the j-th user, i.e. the j-th user dominates the multiuser interference. Such condition depends on the correlation among users' signatures and also on the received signal. When this condition is not satisfied no decision can be made ad then no conclusions can be drawn on user j.

Consider now the case where some of the users have been decoded, or simply have been fixed. If we denote with $\mathcal{U}_d$ the set of the already known users, the dominance condition can be extended as

$$\left|\mathbf{g}_j^T\mathbf{y} - \sum_{k \in \mathcal{U}_d} \mathbf{g}_j^T\mathbf{g}_k x_k\right| > \sum_{i \neq j,\, i \notin \mathcal{U}_d} |r_{ji}|, \tag{11}$$

i.e. we can cancel out the multiuser interference contributed by the already known bits. Condition (11) generalizes the condition (10). If some user j is not dominant over his multi-user interference, it may happen that it is conditionally dominant, giving rise to a subset of all possible solutions.

A. Tree-search algorithm

Condition (11) can be used for searching the ML solution on a tree, where each leaf of the tree represents a possible solution. In the traditional minimum distance algorithm all leaves are checked, and since their number is exponential in the number of users, the algorithm has an exponential complexity. In order to reduce the computational complexity a reduction in the number of the leaf to be checked can be achieved based on some criteria. The sphere decoding algorithm for example only considers leaves corresponding to points in a sphere, that is equivalent to cut those branches in the tree whose distance is greater than the radius of the sphere. However, different criteria for a reduction of possible solutions can be employed. The idea that we propose is to use the conditional dominance condition (11) at each node of the tree. Suppose that we have fixed a user decoding order, then at each node we can check whether the condition (11) is satisfied or not. If the condition is satisfied then a decision on the corresponding user can be made and only one of the two branches that departs from that node is selected, and half of child nodes can be cut.

An example is shown in Fig. (1). We have a tree corresponding to an overloaded system with 5 users, and then with 32 possible transmitted bit vectors; at each node a branch on the right represents a +1 and a branch on the left a -1. The node pointed by the arrow corresponds to the dominance condition

$$\left|\mathbf{g}_3^T\mathbf{y} - \mathbf{g}_3^T\mathbf{g}_2 x_2 - \mathbf{g}_3^T\mathbf{g}_1 x_1\right|_{x_1=1, x_2=1} > \left|\mathbf{g}_3^T\mathbf{g}_4\right| + \left|\mathbf{g}_3^T\mathbf{g}_5\right| \tag{12}$$

Since such condition is satisfied in our example, a decision on user 3 can be made: $x_3 = -1$, if $x_1 = 1$ and $x_2 = 1$. Note that at this point there is no need to visit children of the node (1,1,1). At the end we obtain a set of possible ML solutions, as shown in Fig. (1), where only 7 out of 32 paths survive.

The ML solution can now be searched only over the reduced set of possible solutions. The final step of the algorithm consists of computation of the minimum distance among the surviving paths on the tree and the received signal, providing the exact ML solution. In most cases the algorithm offers substantial improvement on exhaustive ML decoding, as the number of surviving paths is greatly reduced. Since the algorithm is based on a dominance condition we call it *king algorithm.*

The algorithm does not require any matrix inversion as in similar tree-search based algorithms and can be employed unmodified both in underloaded and overloaded systems. It is worth noting that the algorithm returns always the *exact* ML solution, and not a sub-optimal solution as done in other similar algorithm, such as sphere decoding and any other tree search algorithm based on an heuristic metric. Since the BER obtained is the same of the minimum distance algorithm, the only difference between the two is measured in terms of computational complexity.

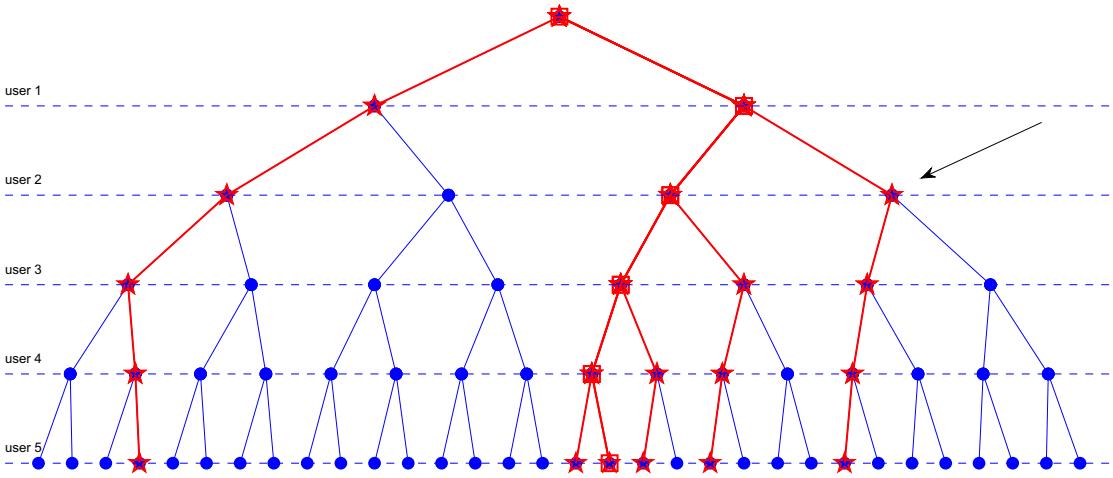

Figure 1. Tree-search algorithm for a system with $N = 2$ and $M = 5$ users and average SNR of 10 dB. The transmitted bit vector is $\mathbf{x} = (1, -1, -1, -1, 1)^T$. Paths with star nodes are provided by the algorithm. The final steps needs to compute 6 distances and get the minimum.

B. Two-stages tree-search algorithm

One more stage, that may further reduce the set of surviving paths in the tree-search algorithm, can be added. In [8] it has been shown that, for a CDMA system, a discrete Hopfield Neural Networks (HNN), simply made up of matched filters, may find in many cases the ML solution by updating the discrete-time approximation of the equation of motion of neurons

$$\mathbf{b}(n+1) = \text{sign}\left(\mathbf{G}^T\mathbf{y} - \left(\mathbf{G}^T\mathbf{G} - \mathbf{E}\right)\mathbf{b}\,(n)\right), \qquad (13)$$

where $\mathbf{E}$ is a diagonal matrix whose j-th diagonal element is $\mathbf{g}_j^T\mathbf{g}_j$. The update stops when some fixed point is reached. Eq. (13) may provide several solutions (equilibrium points), that represent local minima that does not necessarely minimize the distance from the received signal. The specific equilibrium point resulting by the dynamic update rule depends on the initial condition, that in this case coincides with the estimate of the conventional detector. Only when the set of equilibrium points contains one element the update rule provides surely the ML solution. In general the number of equilibrium points is not known in advance.

In any case the set of equilibrium points must contain the ML solution and therefore the optimal solution must satisfy the eq. (13). We can then use such a condition as necessary condition for the ML solution. Since in most cases the number of surviving paths in the tree-search algorithm is greater than the number of equilibrium points the additional stage can check whether each surviving path satisfies the following condition

$$\mathbf{b} = \text{sign}\left(\mathbf{G}^T\mathbf{y} - \left(\mathbf{G}^T\mathbf{G} - \mathbf{E}\right)\mathbf{b}\right) \qquad (14)$$

and therefore discard those points that are not equilibrium points.

Fig. 2 shows equilibrium points for the same received signal in Fig. 1. As shown in the figures, in general the set of decoded points in the tree-search algorithm represents a super-set of the equilibrium points, which is the best one could obtain when no distance metric is employed.

III. Computational Complexity

Since the proposed algorithms provide the exact ML solution, performances are measured in terms of computational complexity to be compared to those obtained with the exhaustive search of minimum distance. We consider as performance figure the average number of sums and/or products as function of the number of users.

The computational cost of tree-search algorithm is given by the sum of two contributions

$$C_{TS} = C_{DC} + C_D. \qquad (15)$$

C_{DC} is the cost due to the search on the tree and is proportional to the number of visited nodes; C_D represents the exponential cost of the computation of the minimum distance, and grows with the number of surviving paths.

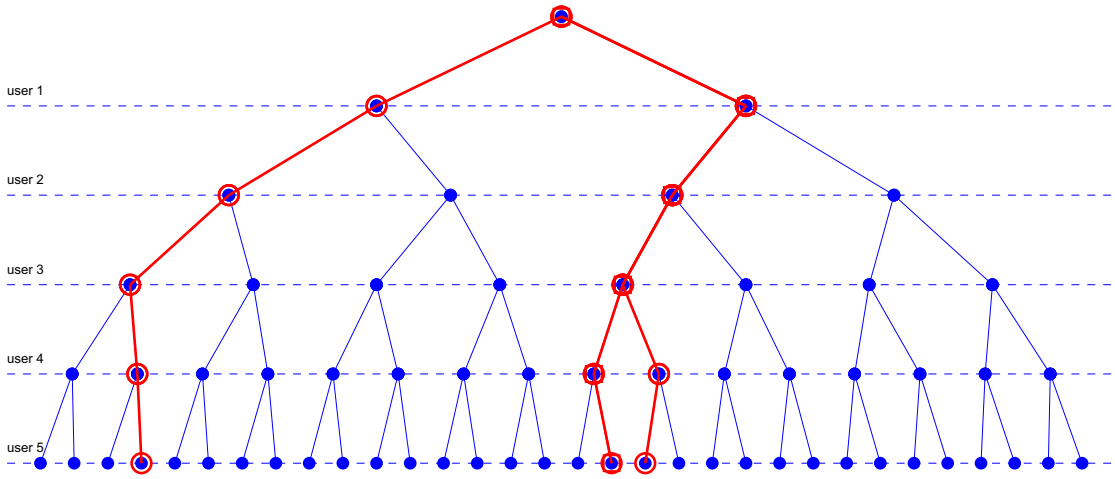

Figure 2. Two-stages tree-search algorithm for an overloaded system with $N = 2$ and $M = 5$ users and average SNR of 10 dB. The transmitted bit vector is $\mathbf{x} = (1, -1, -1, -1, 1)^T$. Paths with circled nodes are equilibrium points. The received signal is the same of the Fig. (1). Note the reduced number of paths that need to be considered for the final step that computes the minimum distance.

The computational cost due to the tree-search stage of the algorithm can be expressed as

$$C_{DC} = \sum_{i=1}^{M} d_i N_i \tag{16}$$

where d_i is the number of sum and/or products needed for dominance condition computations at the tree level i, and N_i is the number of visited nodes at level i. Since we cannot predict N_i, because it depends on the received signal (given the channel matrix), the overall evaluation of computational cost needs to be averaged over several transmitted signals. Given a number N_s of surviving paths on the tree, the cost of computation of euclidean distances is written as

$$C_D = dN_s, \tag{17}$$

where d is the number of sum and/or products for computation of a single euclidean distance.

The key advantage of the algorithm is the expected great reduction of the number of the visited nodes and then of surviving paths. In the best-case scenario, if in every visited node the dominance condition is satisfied, the algorithm returns a unique solution that corresponds to the ML solution. In this case only M nodes are visited and the overall computational cost is due only to C_{DC}. In general the number of visited nodes is greater than M because the condition (10) is not always satisfied. Since when there is no dominance at one node no final decision on the user j can be made and no branches can be cut, in the worst-case scenario no dominant user is found and no path can be discarded and no reduction in computational complexity can be obtained. In this case we have the same computational cost of the exhaustive search of minimum distance. Therefore the efficiency of the algorithm depends on the existence of dominant users and conditionally dominant users.

The two-stage tree-search algorithm presents an additional step that adds up a term that takes into account the average number of sum and/or products needed to check which of the N_s surviving paths are equilibrium points and that can be expressed as

$$C_E = d_e N_s \tag{18}$$

where d_e is the cost of checking the equilibrium condition (14). Even though now the overall cost is given by the sum of three contributions

$$C_{2S} = C_{DC} + C_E + C_D, \tag{19}$$

the increase due to C_E is might be compensated by the reduction of the number of surviving paths for which distances have to be computed.

We have performed some simulations in order to evaluate the computational complexity of both tree-search and two-stage tree-search algorithms. We have measured the computational complexity in terms of the number of products

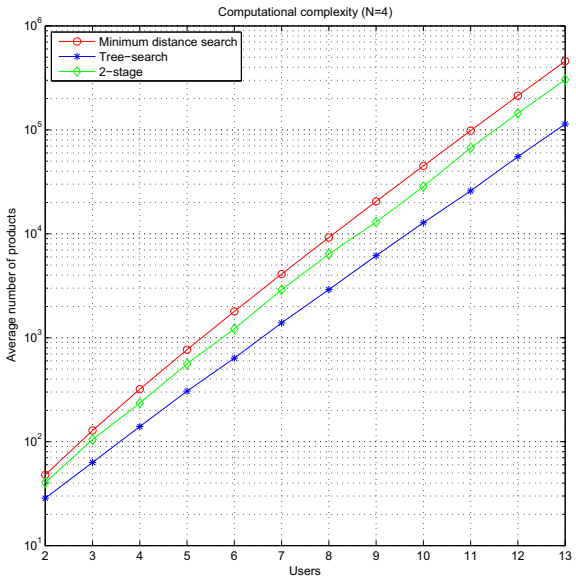

Figure 3. Average number of products for minimum distance and tree search algorithm. A system with signal space dimension $N = 4$ has been considered.

needed averaged over several transmitted signals and we have considered a system with $N = 4$ and increasing number of users M. In Fig. 3 results are shown for the tree-search algorithm and the two-stage tree-search algorithm compared to the exhaustive search that computes all distances.

Both algorithms present better performances than the exhaustive search. In particular the tree-search algorithm results to be better than the two-stage tree-search.

This is because the computational cost of checking the equilibrium condition is comparable with the distance computation. The two-stage algorithm might have better performaces than the tree-search only when N is very large and the system is underloaded.

IV. Conclusions

We have proposed two tree-search algorithms that provide the exact ML solution with reduced computational complexity with respect to the exhaustive minimum distance search. The algorithms are based on the dominance condition and on the HNN equilibrium condition, that provide simple conditions for a reduction of the number of points for which compute the euclidean distance. Numerical simultations have confirmed that their computational complexity measured in terms of products is better than the ML algorithm.

References

[1] S. Verdú, *Multiuser Detection*. Cambrige University Press, 1998.

[2] J. Luo, K. Pattipati, P. Willett, and G. Levchuk, "Fast optimal and suboptimal any-time algorithms for CDMA multiuser detection based on branch and bound," *IEEE Transactions on Communications*, vol. 52, pp. 632–642, Apr. 2004.

[3] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Transactions on Information Theory*, vol. 45, no. 5, pp. 1639–1642, 1999.

[4] W. H. Mow, "Universal lattice decoding: principle and recent advances," *Wireless Communications and Mobile Computing*, vol. 3, no. 5, pp. 553–569, 2003.

[5] L. Ekroot and S. Dolinar, "A* decoding of block codes," *IEEE Transactions on Communications*, vol. 44, pp. 1052–1056, Sept. 1996.

[6] K.-K. Wong and A. Paulraj, "Efficient near maximum-likelihood detection for underdetermined mimo antenna systems using a geometrical approach," *EURASIP Journal on Wireless Communications and Networking*, 2007.

[7] X.-W. Chang and X. Yang, "An efficient tree search decoder with column reordering for underdetermined MIMO systems," in *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pp. 4375–4379, Nov. 2007.

[8] G. I. Kechriotis and E. S. Manolakos, "Hopfield neural network implementation of the optimal CDMAmultiuser detector," *IEEE Transactions on Neural Networks*, vol. 7, pp. 131–141, Jan. 1996.

[9] G. Kechriotis and E. S. Manolakos, "A hybrid digital signal processing-neural network CDMA multiuserdetection scheme," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 43, pp. 96–104, Feb. 1996.

[10] Y. Sun, "A family of likelihood ascent search multiuser detectors: an upper bound of bit error rate and a lower bound of asymptotic multiuser efficiency," *to appear in IEEE Transactions on Communications*, 2009.