An Efficient Tree Search Decoder with Column Reordering for Underdetermined MIMO Systems

Xiao-Wen Chang
School of Computer Science
McGill University
Montreal, Quebec, Canada H3A 2A7

Xiaohua Yang
School of Computer Science
McGill University
Montreal, Quebec, Canada H3A 2A7

***Abstract*—An efficient tree search decoder for underdetermined MIMO systems is presented. The decoder employs a novel column reordering strategy for the channel matrix in the reduction process, which can significantly reduce the computational cost. Simulation results show that this new algorithm is much more efficient than current approaches for a square constellation higher than 4QAM.**

I. Introduction

In Gaussian multi-input multi-output (MIMO) linear flat-fading channel systems, the relation between the received signal vector and the transmit signal vector can be written as a complex linear system

$$\boldsymbol{y}_c = \boldsymbol{H}_c\boldsymbol{x}_c + \boldsymbol{v}_c \tag{1}$$

where $\boldsymbol{H}_c \in \mathbb{C}^{N_r\times N_t}$ represents the flat-fading channel with N_t transmitter antennas and N_r receiver antennas, and the elements of $\boldsymbol{H}_c$ are complex i.i.d Gaussian variables with (normalized) distribution $CN(0,1)$, $\boldsymbol{v}_c \in \mathbb{C}^{N_r}$ is the white Gaussian noise vector with distribution $CN(\boldsymbol{0}, 2\sigma^2\boldsymbol{I})$, and $\boldsymbol{x}_c \in \mathbb{C}^{N_t}$ is the unknown signal vector and its elements are odd numbers in the finite set $\mathcal{X}_c(k) = \{k_1 + k_2 j : k_1, k_2 = \pm 1, \pm 3, \cdots, \pm(2^k-3), \pm(2^k-1)\}$ where $j^2 = -1$. Note that $k = 1, 2, 3$ corresponds to QPSK (i.e., 4QAM), 16QAM, 64QAM constellations, respectively.

To avoid complex operations, we first transform (1) to the following real linear system

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{v} \tag{2}$$

$$\boldsymbol{y} = \begin{bmatrix}\boldsymbol{y}_c^R\\ \boldsymbol{y}_c^I\end{bmatrix}, \boldsymbol{H} = \begin{bmatrix}\boldsymbol{H}_c^R & -\boldsymbol{H}_c^I\\ \boldsymbol{H}_c^I & \boldsymbol{H}_c^R\end{bmatrix}, \boldsymbol{x} = \begin{bmatrix}\boldsymbol{x}_c^R\\ \boldsymbol{x}_c^I\end{bmatrix}, \boldsymbol{v} = \begin{bmatrix}\boldsymbol{v}_c^R\\ \boldsymbol{v}_c^I\end{bmatrix}$$

where $\boldsymbol{A}_c^R$ and $\boldsymbol{A}_c^I$ denote the real part and image part of a complex matrix or vector $\boldsymbol{A}_c$, respectively. Obviously, $\boldsymbol{H} \in \mathbb{R}^{m\times n}$ with $m \triangleq 2N_r$, $n \triangleq 2N_t$ and $h_{ij} \sim N(0, 1/2)$, $\boldsymbol{v} \sim N(0, \sigma^2\boldsymbol{I}_m)$, and $\boldsymbol{x} \in \mathcal{X}(k)^n \triangleq \mathcal{X}(k)\times\cdots\times\mathcal{X}(k)$ with

$$\mathcal{X}(k) \triangleq \{\pm 1, \pm 3, \cdots, \pm(2^k-3), \pm(2^k-1)\}. \tag{3}$$

In order to estimate $\boldsymbol{x}_c \in \mathbb{C}^{N_t}$ in (1) or $\boldsymbol{x} \in \mathbb{R}^n$ in (2), one solves the following minimization problem

$$\min_{\boldsymbol{x}\in\mathcal{X}(k)^n} \|\boldsymbol{y} - \boldsymbol{H}\boldsymbol{x}\|_2^2. \tag{4}$$

We refer to (4) as a box constrained integer least squares (ILS) problem. When $\boldsymbol{H}$ has full column rank, a regular sphere decoder can be employed to solve (4), see, e.g., [1] and [2]. In this paper, we consider the case that $m < n$ and rank$(\boldsymbol{H}) = m$. One such application is a multiple-antenna communication system which has more transmitting antennas than receiving antennas. To find the optimal solution of this problem, Damen et al [3] proposed the first so-called generalized sphere decoding (GSD) algorithm. Its basic idea is to partition $\boldsymbol{x}$ in (4) into two subvectors $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$. For each candidate for $\boldsymbol{x}^{(2)}$, it solves a determined sub-ILS problem to find $\boldsymbol{x}^{(1)}$. After all candidates for $\boldsymbol{x}^{(2)}$ are enumerated, an optimal solution can be found. Later, Dayal and Varanasi [4] proposed another GSD algorithm, to be called Algorithm DV in this paper for convenience, which can significantly reduce the computational complexity by partitioning the candidate set for $\boldsymbol{x}^{(2)}$ into disjoint ordered subsets. Recently, Yang et al [5] proposed a so-called double-layer sphere decoder, which will be referred to as Algorithm YLH in this paper. Its basic idea is to apply an outer layer sphere decoder (SD) to get possible candidates for $\boldsymbol{x}^{(2)}$. When $\boldsymbol{x}^{(2)}$ is determined, an inner layer SD is employed to get $\boldsymbol{x}^{(1)}$. Algorithm YLH is usually faster than Algorithm DV. More recently, Chang and Yang [6] proposed a recursive GSD algorithm, to be called Algorithm CY, which modified Algorithm DV and incorporated a column reordering strategy in reduction, and is (much) faster than Algorithms DV and YLH. All the above algorithms mainly consider how to generate a sequence of determined sub-ILS problems. In [7], Cui and Tellambura proposed a different approach, which transforms the underdetermined problem (4) to an equivalent overdetermined problem so that a regular SD algorithm can then be applied. We refer to the corresponding algorithm as Algorithm CT. All the above algorithms were presented for $k = 1$ in (3). The case $k \geq 2$ was transformed into the case $k = 1$, leading to a new ILS problem with larger dimensions and those algorithms become less efficient. Note that in some literature, see, e.g., [8], only a sub-optimal solution to (4) is found for computational efficiency. This will not be discussed in this paper.

In this paper, an efficient tree search decoder (TSD) will be presented to find the optimal solution for the underdetermined problem (4). We integrate the two search processes in Algorithm YLH into one process seamlessly, which is a depth-first tree search algorithm, and present a novel column reordering (CR) strategy for the reduction process to make the search

1930-529X/07/$25.00 © 2007 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2007 proceedings.

process more efficient. Our new algorithm (to be referred to as TSD-CR) does not handle the cases $k = 1$ and $k \geq 2$ separately, and so does not enlarge the dimensions of the ILS problem when $k \geq 2$. Another advantage is that our algorithm can easily handle a general box constraint.

II. Tree Search Decoder With Column Reordering

To solve (4), first we reduce it to a new problem (this process is sometimes called preprocessing). Let $N \triangleq n-m+1$. We compute the QR decomposition:

$$\boldsymbol{H} = \boldsymbol{Q}\boldsymbol{R}, \quad \boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_1 & \boldsymbol{R}_2 \\ \boldsymbol{0} & \boldsymbol{r}^T \end{bmatrix} \begin{matrix} m-1 \\ 1 \end{matrix} \tag{5}$$
$$\qquad\qquad\quad m-1 \quad N$$

where $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ is orthogonal, $\boldsymbol{R} \in \mathbb{R}^{m\times n}$ is an upper trapezoidal matrix, i.e., $\boldsymbol{R}_1 \in \mathbb{R}^{(m-1)\times(m-1)}$ is upper triangular. With $\bar{\boldsymbol{y}} \triangleq \boldsymbol{Q}^T\boldsymbol{y}$, the problem (4) is transformed to

$$\min_{\boldsymbol{x}\in\mathcal{X}(k)^n} \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{x}\|_2^2. \tag{6}$$

To solve (6), we find a constant β such that

$$\|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{x}\|_2^2 < \beta^2. \tag{7}$$

One effective strategy to choose β was proposed in Chang and Yang [6], which solves a box constrained real least squares problem, rounds the solution to an integer vector $\boldsymbol{x}_0$ in $\mathcal{X}(k)^n$, and then takes $\beta = \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{x}_0\|_2$. We use this strategy in this paper. Note that (7) is a hyper-ellipsoid in terms of $\boldsymbol{x}$, and a hyper-sphere in terms of $\boldsymbol{R}\boldsymbol{x}$. If there is no integer point in the hyper-ellipsoid (7), then $\boldsymbol{x}_0$ is the solution to (6).

Then we start a search process to find the solution in the hyper-ellipsoid (7). In the following we first present a tree search process for solving the reduced problem (6) and then propose a column reordering strategy in the reduction process.

A. Tree search process

We rewrite (7) as

$$\sum_{i=1}^{m}(\bar{y}_i - \sum_{j=i}^{n} r_{ij}x_j)^2 < \beta^2. \tag{8}$$

To simplify notation, define $c_i \triangleq (\bar{y}_i - \sum_{j=i+1}^{n} r_{ij}x_j)/r_{ii}$ for $i = m:-1:1$. Note that c_i depends on $x_{i+1}, x_{i+2}, \cdots, x_n$. Then it is easy to see that the inequality (8) is equivalent to a set of inequalities:

$$(\bar{y}_m - \sum_{j=m}^{n} r_{mj}x_j)^2 < \beta^2 \tag{9}$$

$$r_{jj}^2(x_j - c_j)^2 < \beta^2 - \sum_{i=j+1}^{m} r_{ii}^2(x_i - c_i)^2 \tag{10}$$

for $j = m-1:-1:1$. Thus for $j = m-1:-1:1$,

$$x_j \in \mathcal{X}_j \triangleq \mathcal{X}(k) \cap (\lambda_j, \mu_j) \tag{11}$$

$$\begin{aligned} \lambda_j &\triangleq c_j - \sqrt{\beta^2 - \textstyle\sum_{i=j+1}^{m} r_{ii}^2(x_i - c_i)^2}/|r_{jj}| \\ \mu_j &\triangleq c_j + \sqrt{\beta^2 - \textstyle\sum_{i=j+1}^{m} r_{ii}^2(x_i - c_i)^2}/|r_{jj}|. \end{aligned}$$

In the following, we will use a strategy, which is similar to the one presented in [5], to determine $x_n, x_{n-1}, \ldots, x_m$ based on (9) and use the search algorithm given in [2] to determine $x_{m-1}, x_{m-2}, \ldots, x_1$ based on (10). These two processes are integrated to one process seamlessly.

Let $\mathcal{I}^+ \triangleq \{j \mid r_{mj} \geq 0, m \leq j \leq n\}$, $\mathcal{I}^- \triangleq \{j \mid r_{mj} < 0, m \leq j \leq n\}$. Then (9) is equivalent to

$$\bar{y}_m - \beta < \sum_{j\in\mathcal{I}^+} r_{mj}x_j + \sum_{j\in\mathcal{I}^-} r_{mj}x_j < \bar{y}_m + \beta. \tag{12}$$

For each x_j ($j = m:n$), define the following transformation:

$$\bar{x}_j \triangleq \begin{cases} \frac{2^k-1+x_j}{2}, & j \in \mathcal{I}^+ \\ \frac{2^k-1-x_j}{2}, & j \in \mathcal{I}^- \end{cases} \tag{13}$$

so that

$$\bar{x}_j \in \bar{\mathcal{X}}(k) \triangleq \{0, 1, \cdots, 2^k - 1\}. \tag{14}$$

Define $\alpha \triangleq \bar{y}_m + (2^k - 1)\sum_{j=m}^{n} |r_{mj}|$, then (12) becomes

$$\alpha - \beta < \sum_{j=m}^{n} 2|r_{mj}|\bar{x}_j < \alpha + \beta. \tag{15}$$

For the time being, assume $r_{mj} \neq 0$ for $j = m:n$, then from (14) and (15), it is easy to verify that for $j = n:-1:m$

$$\bar{x}_j \in \bar{\mathcal{X}}_j \triangleq \bar{\mathcal{X}}(k) \cap (\bar{\lambda}_j, \bar{\mu}_j) \tag{16}$$

$$\bar{\lambda}_j \triangleq \frac{\alpha - \beta - 2\sum_{i=j+1}^{n} |r_{mi}|\bar{x}_i - 2(2^k-1)\sum_{i=m}^{j-1} |r_{mi}|}{2|r_{mj}|} \tag{17}$$

$$\bar{\mu}_j \triangleq (\alpha + \beta - 2\sum_{i=j+1}^{n} |r_{mi}|\bar{x}_i)/(2|r_{mj}|). \tag{18}$$

Based on (16) and (11), our search process can start. The search algorithm we propose here is a depth-first tree search. We determine $\bar{x}_j$ (or x_j) at level j. First at level n, choose $\bar{x}_n = \lfloor \frac{\alpha}{2|r_{mn}|} \rceil|_{\bar{\mathcal{X}}_n}$, which denotes the nearest integer in the set $\bar{\mathcal{X}}_n$ to $\frac{\alpha}{2|r_{mn}|}$. Then we proceed to level $n-1$. At this level, we compute the set $\bar{\mathcal{X}}_{n-1}$. If $\bar{\mathcal{X}}_{n-1}$ is empty, then we move back to level n and choose $\bar{x}_n$ to be the next nearest integer to $\frac{\alpha}{2|r_{mn}|}$ in the set $\bar{\mathcal{X}}_n$ and go to level $n-1$ again. Otherwise, we choose $\bar{x}_{n-1} = \lfloor \frac{\alpha - 2|r_{mn}|\bar{x}_n}{2|r_{m,n-1}|} \rceil|_{\bar{\mathcal{X}}_{n-1}}$. In general, when we go to level j from level $j+1$ where $j \geq m$, the first chosen integer for $\bar{x}_j$ is $\lfloor \frac{\alpha - 2\sum_{i=j+1}^{n} |r_{mi}|\bar{x}_i}{2|r_{mj}|} \rceil|_{\bar{\mathcal{X}}_j}$. Continue the procedure until we reach level m and choose a valid integer for $\bar{x}_m$. At this point we obtain $\bar{\boldsymbol{x}}^{(2)} \triangleq [\bar{x}_m, \cdots, \bar{x}_n]^T$. From (13) we can transform $\bar{\boldsymbol{x}}^{(2)}$ to the original integer vector $\boldsymbol{x}^{(2)} \triangleq [x_m, \cdots, x_n]^T$. Then we go to level $m-1$ and compute the set $\mathcal{X}_{m-1}$ (see (11)). If $\mathcal{X}_{m-1}$ is empty, we go back to level m. Otherwise, choose $x_{m-1} = \lfloor c_{m-1} \rceil|_{\mathcal{X}_{m-1}}$, and proceed to level $m-2$. Continue this process and finally we reach level 1 and choose a valid integer for x_1, resulting in $\boldsymbol{x}^{(1)} \triangleq [x_1, \ldots, x_{m-1}]^T$. Thus $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^{(1)} \\ \boldsymbol{x}^{(2)} \end{bmatrix}$ is the first integer point we have found by the search process. Then we update β by taking $\beta = \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{x}\|_2$. Now we seek an integer point within the new hyper-ellipsoid by updating $\boldsymbol{x}$. We move up to level 2 and try to choose x_2 to be

1930-529X/07/$25.00 © 2007 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2007 proceedings.

the next nearest integer to c_2 in the set $\mathcal{X}_2$ (which has been updated by using the new β). Note that in this tree search process, if we succeed in finding a valid integer at a level, then we move to the next lower level, otherwise we move to the next higher level. Finally, when we fail to find a new valid integer for $\bar{x}_n$ at level n, the search process terminates and the latest found integer point is the optimal solution we seek.

In theory it is possible that in (15) some $r_{mj} = 0$, $m \le j \le n$. In this case each integer in the set $\bar{\mathcal{X}}(k)$ (see (14)) can be a candidate for $\bar{x}_j$, and one is chosen each time when the algorithm is at level j. Specifically, if we go to level j from level $j+1$, we choose $\bar{x}_j = 0$, and when we move to level j from level $j-1$, we choose $\bar{x}_j$ to be the next smallest integer in $\bar{\mathcal{X}}(k)$ ("next" means next to the one chosen last time). This tends to take more search time compared with the normal case $r_{mj} \neq 0$. Fortunately, this rarely occurs in practice.

Now we make a remark about the main differences between Algorithm YLH [5] and the above tree search process. In the former, when $k \ge 2$, a binary representation $\bar{x}_j = \sum_{i=0}^{k-1} 2^i \bar{x}_{ij}$, $\bar{x}_{ij} \in \{0,1\}$ was used to transform the inequality (15) into a new one, which has unknown binary numbers $\bar{x}_{ij}$. Then a search process based on a sphere decoding algorithm was applied to find $[\bar{x}_{0,m}, \ldots, \bar{x}_{k-1,m}, \ldots, \bar{x}_{0,n}, \ldots, \bar{x}_{k-1,n}]^T$, which is then transformed back to $\boldsymbol{x}^{(2)} = [x_m, \ldots, x_n]^T$. This approach increases the number of the levels of the tree, while it decreases the constraint interval for each level. From the tree search point of view, this makes search less efficient, see, e.g., [9, Chaps 12 and 13] for explanations. To obtain $\boldsymbol{x}^{(1)}$, like other existing algorithms, Algorithm YLH suggests to use a regular SD to solve

$$\min_{\boldsymbol{x}^{(1)} \in \mathcal{X}(k)^{(m-1)}} \|(\boldsymbol{y}(1\!:\!m-1) - \boldsymbol{R}_2 \boldsymbol{x}^{(2)}) - \boldsymbol{R}_1 \boldsymbol{x}^{(1)}\|_2. \quad (19)$$

Notice that our approach integrates the two search processes for $\boldsymbol{x}^{(1)}$ and $\boldsymbol{x}^{(2)}$ into one tree search process.

Our tree search process can easily handle the general box constraint $\boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}$ where $\boldsymbol{l}$ and $\boldsymbol{u}$ are integer vectors, while the existing algorithms we mentioned before cannot. Essentially what we need to do is to replace the transformation (13) by

$$\bar{x}_j \triangleq \begin{cases} -l_j + x_j, & j \in \mathcal{I}^+ \\ u_j - x_j, & j \in \mathcal{I}^- \end{cases}$$

so that $\bar{x}_j \in \{0, 1, \ldots, u_j - l_j\}$.

B. Column reordering strategy

From (16) and (11) we find that the efficiency of the tree search process greatly depends on the upper trapezoidal matrix $\boldsymbol{R}$. Note that different ordering of the columns of the channel matrix $\boldsymbol{H}$ leads to different $\boldsymbol{R}$. To make the tree search process given in Sec. II-A more efficient, we will find a good permutation matrix $\boldsymbol{P}$ in the QR decomposition $\boldsymbol{HP} = \boldsymbol{QR}$ (cf. (5)). It is easy to verify that if $\hat{\boldsymbol{x}}$ is the solution to (6) with $\boldsymbol{R}$ defined above, then $\boldsymbol{P}\hat{\boldsymbol{x}}$ is the solution to (4).

We will determine how to choose $n-m+1$ columns from matrix $\boldsymbol{H}$ as the right part of the permuted $\boldsymbol{H}$. The remaining $m-1$ columns form the left part of the permuted $\boldsymbol{H}$. Then we will determine how to order the columns of these two parts, respectively.

First assume that the two parts of $\boldsymbol{H}$ have been determined and we have a permuted $\boldsymbol{H}$ and the corresponding upper trapezoidal matrix $\boldsymbol{R}$. In the following we show how to reorder the last $n-m+1$ columns of $\boldsymbol{R}$, or equivalently, the last $n-m+1$ columns of the permuted $\boldsymbol{H}$. At level j, where $m \le j \le n$, any integer number in $\bar{\mathcal{X}}_j$ (see (16)) is a possible candidate for $\bar{x}_j$. To make the search process efficient, the number of integers in $\bar{\mathcal{X}}_j$ should be as small as possible. This motivates the following reordering strategy. Define

$$\begin{aligned} L_j \triangleq & \min\{2^k - 1, \lfloor \bar{\mu}_j \rfloor + \text{sign}(\bar{\mu}_j - \lfloor \bar{\mu}_j \rfloor) - 1\} \\ & - \max\{0, \lceil \bar{\lambda}_j \rceil - \text{sign}(\lceil \bar{\lambda}_j \rceil - \bar{\lambda}_j) + 1\} + 1 \end{aligned} \quad (20)$$

where $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ denote the floor and ceiling operations, respectively. If $L_j \le 0$, then $\bar{\mathcal{X}}_j$ is empty, otherwise L_j is the number of integers in $\bar{\mathcal{X}}_j$. Suppose we have determined the last $n-j$ columns of $\boldsymbol{R}$, now we want to determine column j, $j \ge m$. First we compute L_j corresponding to the current column j of $\boldsymbol{R}$. Notice that $\bar{\lambda}_j$ and $\bar{\mu}_j$ depend on $\bar{x}_p$ for $p = n, n-1, \cdots, j+1$, see (17)-(18). In computing them, we take $\bar{x}_p = \lfloor \frac{\alpha - 2\sum_{l=p+1}^{n} |r_{ml}| \bar{x}_l}{2|r_{mp}|} \rceil |_{\bar{\mathcal{X}}_p}$, which is the first integer we take for $\bar{x}_p$ at level p in the search process. Then we interchange columns j and i of $\boldsymbol{R}$ for $i = m : j-1$. After each interchange, we compute the corresponding L_j. Then the column corresponding to the smallest L_j is chosen to be the $j-$th column of $\boldsymbol{R}$. If the smallest value is nonpositive, we stop this process. Later we will show how to reorder the remaining columns. Otherwise we go to level $j-1$.

Algorithm 2.1 Reorder part of columns of $\boldsymbol{R}(:, m:n)$

Input: upper trapezoidal matrix $\boldsymbol{R} \in \mathbb{R}^{m \times n}$, $\bar{y}_m$, β.
Output: permuted $\boldsymbol{R}$, $n \times n$ permutation $\boldsymbol{P}$, variable $index$ for which $\bar{\mathcal{X}}_{index}$ is empty, variable $Prod \triangleq \prod_{j=index+1}^{n} L_j$.
function: $[\boldsymbol{R}, \boldsymbol{P}, index, Prod] = \text{REORDER}(\boldsymbol{R}, \bar{y}_m, \beta)$
Set $\boldsymbol{P} = \boldsymbol{I}_n$, $index = m-1$, $Prod = 1$, $j = n$
while $j \ge m$
  Compute L_j according to (20), set $p = j$
  for $i = m : j-1$
    Set $\boldsymbol{R}' = \boldsymbol{R}$, and interchange columns i and j of $\boldsymbol{R}'$, then compute the corresponding L_i'
    if $L_i' < L_j$
      Set $p = i$, $L_j = L_i'$
    end
  end
  if $p \neq j$
    Interchange columns p and j of $\boldsymbol{P}$ and $\boldsymbol{R}$
  end
  if $L_j \le 0$
    Set $index = j$, and break the while loop
  end
  Compute $\bar{x}_j = \lfloor \frac{\alpha - 2\sum_{l=i+1}^{n} |r_{ml}| \bar{x}_l}{2|r_{mj}|} \rceil |_{\bar{\mathcal{X}}_j}$
  Compute $Prod = Prod * L_j$, $j = j-1$
end

We could make the above algorithm more computationally efficient, but for clarity, we will not do it here.

1930-529X/07/$25.00 © 2007 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2007 proceedings.

If the output $index > m+1$, we can reorder the remaining columns $m, m+1, \ldots, index-1$ of $\boldsymbol{R}$ in the following way. From (20), (17) and (18) we find that a larger $|r_{mj}|$ leads to a smaller L_j if $\bar{\lambda}_j \leq 0$. Our simulations indicated that usually $\bar{\lambda}_j \leq 0$ if we have $\bar{y}_m \leq 0$. The latter can be guaranteed, since if $\bar{y}_m > 0$, we can simply multiply $\bar{y}_m$ and the last row of $\boldsymbol{R}$ by -1. So we reorder columns $m, m+1, \ldots, index-1$ of $\boldsymbol{R}$ such that $|r_{mm}| \leq |r_{m,m+1}| \leq \cdots \leq |r_{m,index-1}|$.

In the tree search process, the empty set $\bar{\mathcal{X}}_{index}$ means the search process has to move to level $index+1$. In other words, we cut off a branch at level $index$ in the search tree. When the value of $index$ is larger, the branch we can cut off is larger, and the search process is likely to be more efficient. Therefore, we would like to get a larger $index$ by reordering some columns of $\boldsymbol{R}$.

Now we describe the entire reduction process, which involves column reordering and QR decompositions. (1) *Find the first QR decomposition*. Compute the Householder QR decomposition of $\boldsymbol{H}$ with minimum column pivoting: $\boldsymbol{HP} = \boldsymbol{QR}$, where $\boldsymbol{P}$ is a permutation matrix. In the j-th step of this process ($j = 1:m$), we do a column permutation so that $|r_{jj}|$ is the smallest positive number we can achieve. This is similar to (but not the same as) the so-called sorted QR decomposition for overdetermined $\boldsymbol{H}$ (see [10] and [2]). This will tend to make $|r_{mj}|$ ($j = m+1:n$) larger, since, roughly speaking, m smaller columns of $\boldsymbol{H}$ (in terms of the 2-norm) have been moved to the left. Larger $|r_{mj}|$ tends to make μ_j smaller (see (18)). We mentioned before that λ_j is usually nonpositive. Hence, from (20), we see that larger $|r_{mj}|$ is likely to lead to smaller L_j. This is what we pursue. Then we compute $\bar{\boldsymbol{y}} = \boldsymbol{Q}^T\boldsymbol{y}$. In the reduction process we always keep $\bar{y}_m$ nonpositive. (2) *Determine the last $n-m+1$ columns*. For $j = 1:m$, we interchange columns j and m of $\boldsymbol{R}$ (if $j = m$, actually there is no interchange). After each interchange, we compute the new QR decomposition by using Givens rotations efficiently (note that the permuted $\boldsymbol{R}$ has structure). When we apply a Givens rotation to $\boldsymbol{R}$, we always simultaneously update $\bar{\boldsymbol{y}}$ by the same Givens rotation. Then we apply Algorithm 2.1 to the new $\boldsymbol{R}$ to reorder (part of) its last $n-m+1$ columns and obtain the corresponding $index$ and $Prod$. Finally from the m values of $index$, we find the largest one $index_{\max}$ and then the corresponding ordering. If there is more than one ordering giving the same largest $index$, we choose the one which gives the smallest $Prod$, a product of numbers of candidates at levels higher than $index$. Then we order columns $m, m-1, \ldots, index-1$ by the method we mentioned before. Now the last $n-m+1$ columns of the permuted $\boldsymbol{H}$ have been completely determined and so has the last row of $\boldsymbol{R}$. (3) *Reorder the first $m-1$ columns*. We use the tree search process to get the first $\boldsymbol{x}^{(2)}$. Note that the best $\boldsymbol{x}^{(1)}$ is the solution to the determined ILS problem (19). We employ the column reordering strategy given in [2] to (19) to reorder the columns of $\boldsymbol{R}_1$ (or the first $m-1$ columns of $\boldsymbol{R}$).

Algorithm 2.2 Reduction

Input: channel matrix $\boldsymbol{H} \in \mathbb{R}^{m\times n}$, $\boldsymbol{y} \in \mathbb{R}^m$, β.

Output: permutation $\boldsymbol{P} \in \mathbb{Z}^{n\times n}$, upper trapezoidal $\boldsymbol{R} \in \mathbb{R}^{m\times n}$ and $\bar{\boldsymbol{y}} \in \mathbb{R}^m$.

function: $[\boldsymbol{P}, \boldsymbol{R}, \bar{\boldsymbol{y}}] = \text{REDUCTION}(\boldsymbol{H}, \boldsymbol{y}, \beta)$
Compute $\boldsymbol{HP} = \boldsymbol{QR}$ with minimum pivoting
Set $\bar{\boldsymbol{y}} = \boldsymbol{Q}^T\boldsymbol{y}$, $index = 0$, $Prod = +\infty$
for $j = 1:m$
  Set $\boldsymbol{R}' = \boldsymbol{R}$, $\bar{\boldsymbol{y}}' = \bar{\boldsymbol{y}}$
  if $j \neq m$
    Swap columns j and m of $\boldsymbol{R}'$ & transform $\boldsymbol{R}'$ to an upper trapezoidal matrix by Givens rotations
    Apply the same Givens rotations to $\bar{\boldsymbol{y}}'$
  end
  if $\bar{\boldsymbol{y}}'(m) > 0$
    Set $\bar{\boldsymbol{y}}'(m) = -\bar{\boldsymbol{y}}'(m)$, $\boldsymbol{R}'(m,:) = -\boldsymbol{R}'(m,:)$
  end
  $[\boldsymbol{R}', \boldsymbol{P}', ind_{tmp}, Prod_{tmp}] = \text{REORDER}(\boldsymbol{R}', \bar{\boldsymbol{y}}'_m, \beta)$
  if $ind_{tmp} > index$
    Set $index = ind_{tmp}$, $p = j$, $\boldsymbol{R}_{tmp} = \boldsymbol{R}'$, $\bar{\boldsymbol{y}}_{tmp} = \bar{\boldsymbol{y}}'$, $\boldsymbol{P}_{tmp} = \boldsymbol{P}'$
  elseif $ind_{tmp} = index$
    if $Prod_{tmp} < Prod$
      Set $Prod = Prod_{tmp}$, $p = j$, $\boldsymbol{R}_{tmp} = \boldsymbol{R}'$, $\bar{\boldsymbol{y}}_{tmp} = \bar{\boldsymbol{y}}'$, $\boldsymbol{P}_{tmp} = \boldsymbol{P}'$
    end
  end
end
if $p \neq m$
  Swap columns p and m of $\boldsymbol{P}$
  Set $\boldsymbol{P} = \boldsymbol{P}\boldsymbol{P}_{tmp}$, $\boldsymbol{R} = \boldsymbol{R}_{tmp}$, $\bar{\boldsymbol{y}} = \bar{\boldsymbol{y}}_{tmp}$
end
if $index > m+1$
  Reorder the columns of $\boldsymbol{R}(:, m:index-1)$ such that $|r_{mm}| \leq |r_{m,m+1}| \leq \cdots \leq |r_{m,index-1}|$, and reorder the columns of $\boldsymbol{P}$ correspondingly
end
Use our tree search algorithm to find the first $\boldsymbol{x}^{(2)}$, apply the reduction strategy given in [2] to (19) to reorder the first $m-1$ columns of $\boldsymbol{R}$, leading to new trapezoid $\boldsymbol{R}$ and $\bar{\boldsymbol{y}}$, then reorder the columns of $\boldsymbol{P}$ correspondingly

Remark. Although the above reduction process costs more than the one which does not use column reordering, usually its cost is still negligible compared with that of the search process.

III. Simulation Results

In this section we compare the computational cost of our tree search decoder with column reordering, to be refereed to as Algorithm TSD-CR, with Algorithms YLH [5], CT [7] and CY [6]. All the simulations were run in MATLAB 7.0.

The channel matrix $\boldsymbol{H}$ and the receive vector $\boldsymbol{y}$ were generated according to (2), where $\boldsymbol{H} \in \mathbb{R}^{m\times n}$ with $m = 2N_r$ and $n = 2N_t$. In Algorithms YLH, CT and CY, the same regular sphere decoding algorithm (see [2]) with the V-BLAST column reordering strategy for reduction is employed for solving any overdetermined or determined box-constrained ILS problems. In CT, the parameter α is chosen to be the noise

1930-529X/07/$25.00 © 2007 IEEE
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE GLOBECOM 2007 proceedings.

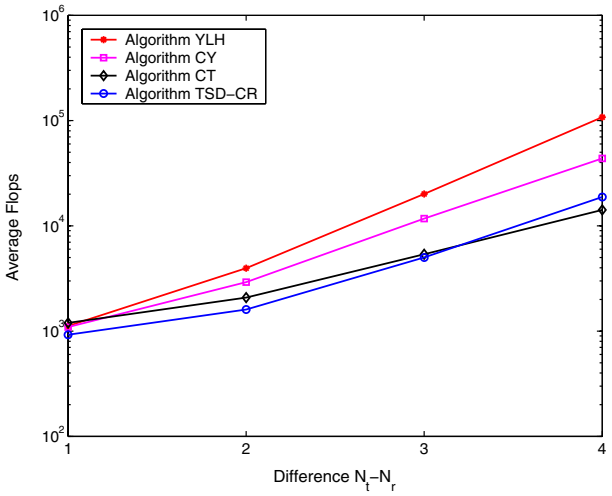

Fig. 1. Average flops vs $N_t - N_r$ (4QAM)

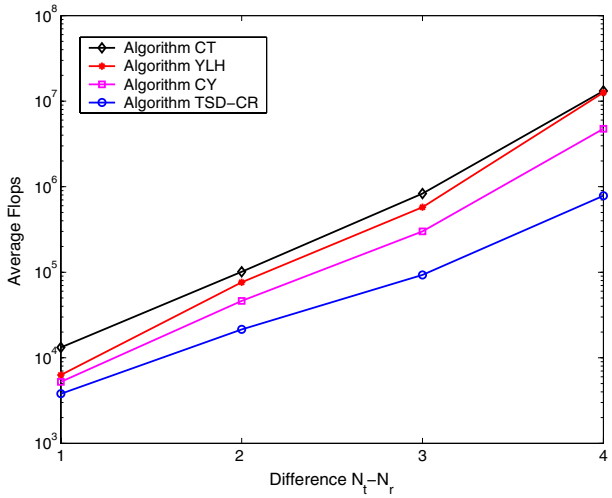

Fig. 2. Average flops vs $N_t - N_r$ (16QAM)

variance σ^2 (see [7]). The computational cost is measured by the number of flops. For each algorithm, only the flops of the search process are counted without considering the cost of the reduction process, which is relatively negligible based on our simulations. But due to limited space, the simulations for that are not given in this paper. For each case, we performed 100 runs and counted the average flops.

Figs. 1, 2 and 3 display the average flops of four algorithms versus N_t–N_r, for 4QAM, 16QAM and 64QAM, respectively. In all of these cases $\boldsymbol{v} \sim N(\mathbf{0}, 0.1^2 \boldsymbol{I}_{2N_r})$, $N_r = 5$ and N_t is a variable. For 4QAM, when $N_t - N_r \leq 3$, TSD-CR is the most efficient one among the four algorithms; when $N_t - N_r = 4$, CT is slightly more efficient than TSD-CR. When the number of constellation points increases, TSD-CR becomes more and more efficient compared with three other algorithms. For example, in Fig. 2, when $N_t - N_r = 4$, TSD-CR is about 13 times as fast as YLH on average; and in Fig. 3, when $N_t - N_r = 3$, TSD-CR is about 20 times as fast as YLH.

We found that when we decreased the variance of the noise $\boldsymbol{v}$, i.e., increased SNR, the cost for each of the four algorithms decreased, but TSD-CR was still the most efficient one for a square constellation higher than 4QAM. Due to limited space, we will not present those simulation results here.

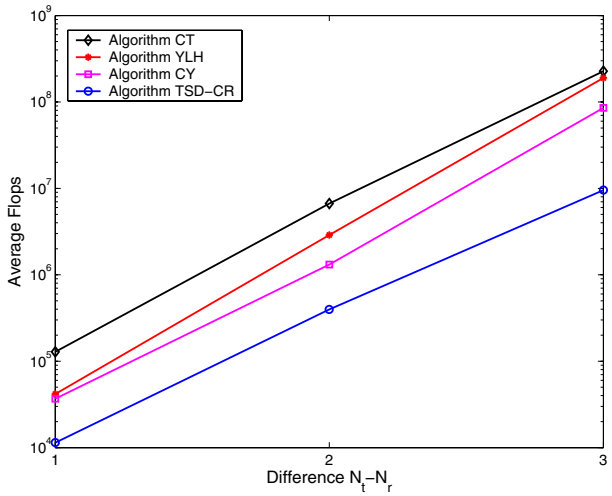

Fig. 3. Average flops vs $N_t - N_r$ (64QAM)

IV. Summary

We have presented a new efficient tree search decoder for the underdetermined MIMO systems. A column reordering strategy was proposed to make the search process more efficient. Numerical simulations indicated that this approach is (much) more efficient than current methods. Moreover, this new approach can handle general box-constraints easily.

References

[1] M. Damen, H. El Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.

[2] X.-W. Chang and Q. Han, "Solving box-constrained integer least squares problems," *IEEE Trans. Wireless Commun.*, in press.

[3] M. Damen, K. Abed-Meraim, and J. Belfiore, "Generalized sphere decoder for asymmetrical space-time communication architecture," *Electron. Lett.*, vol. 36, pp. 166–167, 2000.

[4] P. Dayal and M. Varanasi, "A fast generalized sphere decoder for optimum decoding of under-determined MIMO systems," in *41st Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Oct. 2003, pp. 1216–1225.

[5] Z. Yang, C. Liu, and J. He, "A new approach for fast generalized sphere decoding in MIMO systems," *IEEE Sig. Proc. Letters*, vol. 12, no. 1, pp. 41–44, 2005.

[6] X.-W. Chang and X. Yang, "A new fast generalized sphere decoding algorithm for underdetermined MIMO systems," in *23rd Biennial Symposium on Communications*, Kingston, Canada, May-June 2006, pp. 18–21.

[7] T. Cui and C. Tellambura, "An efficient generalized sphere decoder for rank-deficient MIMO systems," *IEEE Commun. Lett.*, vol. 9, no. 5, pp. 423–425, 2005.

[8] M.O. Damen, H. El Gamel, and G. Caire,"MMSE-GDFE lattice decoding for solving under-determined linear systems with integer unknowns," in *Proc. IEEE Int. Symp. Inform. Theory*, Chicago, USA, June-July 2004.

[9] T. H. Cormen, C. E. Leiserson, and C. Rivest, *Introduction to algorithms*. MIT Press, 2001.

[10] D. Wubben, R. Bohnke, J. Rinas, V. Kuhn, and K. Kammeyer,"Efficient algorithm for decoding layered space-time codes," *IEEE Electronics Letters*, vol. 37, pp. 1348-1350, Oct., 2001.