Stopping Rule-Based Iterative Tree Search for Low-Complexity Detection in MIMO Systems

Abhay Kumar Sah and A. K. Chaturvedi, Senior Member, IEEE

Abstract—Breadth first tree search (BFTS) algorithms are known to provide a close to maximum likelihood (quasi-ML) solution at a low-complexity if the received sequence is detected in the right sequence order. However, finding the right sequence order has an exponential overhead. In view of this, we propose to repeatedly apply a BFTS algorithm to all sequence orders. Since it will test all the orders, it is expected to achieve quasi-ML performance. However, this will increase the complexity because of redundant iterations. The complexity can be reduced if we can stop the iterations as soon as a quasi-ML solution is achieved. For this, we propose two stopping rules, one relies on a constellation based heuristic and the other one uses the distribution of ML cost. It is found that their complexity curves have a cross-over point. Thus, a combination of the two rules provides a quasi-ML error performance at a low-complexity for uncoded as well as coded systems. We further show that the proposed stopping rule can reduce the complexity of depth first tree search algorithms also. Last, for large MIMO systems, compared with existing algorithms, it is found to be exceptionally better in terms of both error performance and complexity.

Index Terms—Multiple-input and multiple-output (MIMO), K-best, IKSD, large MIMO

I. INTRODUCTION

BREADTH first tree search (BFTS) [1] algorithms such as K-best [2], [3] and improved K-best sphere decoder (IKSD) [4] are known to provide a good combination of error performance and complexity for detection in MIMO systems. The rational of K-best is to select K best paths (in terms of Euclidean distance) at each level of the tree moving in a forward direction and finally at the end select the best among these K path. If K is chosen to be sufficiently large, it can achieve close to maximum likelihood (quasi-ML) symbol error rate (SER). However, the computational complexity increases with K, which in turn leads to a high computational complexity. IKSD improves upon K-best by selecting some more paths, the cost of which are within a range (say Δ) of the cost of the Kth path. This leads to a reduction in the required value of K as compared to K-best and is shown to be efficient in terms of computational complexity without any loss in error performance.

On a somewhat different note, the performance of K-best and IKSD algorithms are known to be sensitive to the order in which the symbols are detected [5], [6]. In fact, if the symbols are detected in the right order, they can provide quasi-ML performance at smaller values of K and Δ leading to much lower complexity. The problem of finding the right order has been addressed in the literature and there exist several channel ordering schemes [7], [8]. However, the optimal channel ordering scheme has exponential overhead while sub-optimal schemes lead to a loss in SER performance [7].

In this paper, we seek to lower the complexity of K-best and IKSD without compromising their error performance. For this we propose to apply IKSD iteratively (i.e. repeatedly) for all sequence orders and then select the one which gives the minimum Euclidean cost. Since all possible orders would have been tested, this process should lead to a quasi-ML performance except that the complexity will again become high. We propose to cut down the complexity by doing two things. Firstly, since we will be testing over all possible sequence orders, the values of K and Δ can be chosen to be low, which will lead to a lower complexity per iteration. Secondly, we propose to stop the iterations as soon as a quasi-ML performance is achieved. This can be done, if we can devise a rule to identify the iteration, at which the iterative process can be stopped.

Ideally, a stopping rule should allow the iterations to continue until a quasi-ML solution is achieved, and as soon as it is achieved, the process should terminate. In addition, the overhead of computing the stopping rule itself should be low. Motivated by these requirements we propose two stopping rules. The first rule is a constellation based heuristic. It is based on the premise that it would be reasonable to stop, when the Euclidean cost is less than half of the minimum Euclidean distance between all possible pairs of transmit vectors, which differ at only one place. Here, the only overhead is computation of the minimum Euclidean distance. This is not only simple to compute but also, once computed, it is used for the detection of all vectors which are received during a given channel realization.

The second rule is based on the cumulative distribution of squared norm of the error vector and has zero overhead because it need not be computed online. The computational complexity of the algorithm is a function of SNR in both cases. It turns out that for lower SNR’s, the second rule has
lower complexity while it is the other way around for higher
SNR’s. In other words, when their respective complexities are
plotted as a function of SNR, there is a cross-over point. Hence
using a combination of the two stopping rules, a quasi-ML
performance can be obtained at a complexity which is the
lower of the two.

We have also examined the performance of this
stopping rule for two depth first tree search (DFTS)
algorithm [1], [9], [10]. Specifically, we applied it over
statistical pruning sphere decoder (SPSD) [11] and a sphere
decoder (SD) which uses channel ordering [8]. We found that
it is able to cut the redundant search in this context too.

Lastly, the ability to provide a quasi-ML solution at
low-complexity, makes the proposed iterative detection an
attractive candidate for large antenna systems. Using simula-
tions we show that for a 32 × 32 MIMO system with 4-QAM
it outperforms the existing algorithms for large MIMO such
as [12]–[16], not only in terms of error performance but also
in terms of complexity.

The paper is organized as follows: Section III proposes
the iterative detection scheme while Section IV proposes the
two stopping rules. In Section V, we propose a composite
version of the two rules. The iterative detection technique
with composite stopping rule is simulated and compared with
K-best [3], IKSD [4] and SPSD [11] in Section VI. Section VII presents the benefits of proposed schemes for
DFTS algorithms and for large antenna systems. We conclude
the paper in Section VIII.

II. PRELIMINARIES

A. System Model

We consider a single-user MIMO system having $N_t$ transmit
antennas and $N_r$ receive antennas ($N_t \leq N_r$). It is common
to represent the input-output relation by the following mathe-
matical model

$$Y = HX + N,$$

where $Y$ is $(N_r \times 1)$ received vector, $H$ is $(N_r \times N_t)$
channel matrix, $N = (n_1, n_2, \ldots, n_N)^T$ is $(N_t \times 1)$ i.i.d.
additive white Gaussian noise (AWGN) vector with each $n_i \in
\mathcal{CN}(0, \sigma^2)$ and $X = \left[\begin{array}{c} x_1, x_2, \ldots, x_{N_t} \end{array}\right]^T$ is $(N_t \times 1)$ trans-
mitted vector with each $x_i \in \Omega = \{\Omega_1, \ldots, \Omega_M\}$. Here $\Omega$
is a set of complex constellation symbols such as M-QAM
with $E[XH] = 1$ and minimum distance $d_{\text{min}}$. The $E_s/N_0$
of MIMO system is given by $\frac{E_s}{N_0} = \frac{1}{\sigma^2}$.

Detection of transmitted symbol vector $\hat{X}$, using the knowl-
edge of received vector $Y$ and channel matrix $H$, is a key
problem in MIMO systems. This can be done by finding the
actual transmitted symbol vector $X$ among the all possible
$M^{N_t}$ transmit vectors which is nearest to the received signal
vector $Y$ for the given channel matrix $H$. Utilizing QR
decomposition, this can be mathematically stated as

$$\hat{X} = \arg\min_{X \in \Omega^{N_t}} \|Z - RX\|_2^2,$$

and well known as ML detection. Here, $Z = Q^HY$ is $(N_t \times 1)$
equivalent received vector, $Q$ is $(N_r \times N_t)$ orthogonal matrix
and $R$ is a $(N_r \times N_t)$ upper triangular matrix obtained using QR
decomposition of $H$. $\| \cdot \|$ represents $L_2$-norm. In the sequel
we referred the Euclidean cost associated with the detected
vector $\hat{X}$ simply as cost.

There exist various algorithms which provide quasi-ML
solutions to the problem described in (2). These are broadly
classified into two classes of tree search algorithms i.e. DFTS
and BFTS algorithms. For achieving a quasi-ML solution, the
computational complexity of DFTS algorithm varies with SNR
while it is constant for the BFTS algorithm. Thus, depending
upon the SNR, a BFTS algorithm may require higher (or
lower) complexity than the DFTS algorithm. Motivated by this
we propose a low-complexity variant of the BFTS algorithm in
the next section. Before proceeding to our proposed solution,
let us briefly discuss two popular variants of the BFTS

B. Review of BFTS Algorithms

K-best algorithm [2], [3] searches for the minimum cost
vector over a set of $K$ candidate solution vectors. This set of
candidate solution vectors is found from the sub-problems in
(2) by expanding it into a sum of $N_t$ terms and solving it
for $K$ best possible solutions instead of an unique solution
as in (2), sequentially. Let us express one such intermediate
sub-problem as

$$\{x_i, \ldots, x_{N_t}, x_{i+1}, \ldots, x_{N_t}\} = \arg\min_{x_i \in \Omega} \sum_{j=1}^{N_t} |z_j - \sum_{k=j}^{N_t} r_{j,k}x_k|^2,$$

where $r_{j,k}$ represents the $(j,k)$th element of $R$ and $i$ varies
from $N_t$ to 1. Here, $\arg\min_k$ returns the $K$ intermediate
vectors $[x_1, x_{i+1}, \ldots, x_{N_t}]$ for which the objective function
is minimum. As $i$ varies from $N_t$ to 1, there are $N_t$ such
sub-problems. These sub-problems are solved one by one starting
from $i = N_t$. As a result, $K$ symbols are selected from $\Omega$.
Given the knowledge of $K$ best symbols, the search is repeated
for $i = N_t - 1$. This means now the search is restricted to
$MK$ solution vectors only and again $K$ solution vectors are
selected. Repeating the same procedure by decrementation of
the value of $i$ by one, we arrive at $i = 1$, where the best among
these $K$ candidate solution vectors is declared as the final
solution.

The key idea of IKSD [4] is to choose $K + \lambda$ symbols at
each layer instead of selecting strictly $K$ symbols, and the value of
$\lambda$ is decided by the symbol vectors which are within
a certain margin ($\Delta$) of the $K$th symbol vector. For deciding
the value of $\Delta$ three rules viz. fixed threshold, normalized
threshold and adaptive threshold have been proposed in [4].
These help in reducing the required value of $K$ compared to
K-best and hence save computational complexity.

Since K-best and IKSD both are sequential detection tech-
niques, the error in symbol detection at lower layers propagates
to upper layers. Hence to avoid error propagation, the symbols
need to be detected in a right order [5]. Detection in a
right order not only avoids error propagation but also saves
complexity because the value of $K$ required to achieve the
quasi-ML solution reduces. However, finding a right order is
computationally expensive [7]. In view of this, an iterative BFTS algorithm is proposed in the next section.

III. PROPOSED ITERATIVE DETECTION

Let us denote the order in which the symbols will be detected as a sequence, say \( (x_{N_t-i+1}, \cdots, x_N, x_1, \cdots, x_{N_t-i}) \) for a given \( i \), where \( i = 0, 1, \cdots, (N_t - 1) \). Then we can express (1) using this order and a shifted version of \( \mathbf{H} \) as

\[
Y = \mathbf{H}' X^i + N,
\]

where

\[
\mathbf{H}' = \begin{bmatrix}
h_{1,N_t-i+1} & \cdots & h_{1,N_t} & h_{1,1} & \cdots & h_{1,N_t-i} \\
h_{2,N_t-i+1} & \cdots & h_{2,N_t} & h_{2,1} & \cdots & h_{2,N_t-i} \\
\vdots & \ddots & \vdots & \ddots & \ddots & \vdots \\
h_{N_t,N_t-i+1} & \cdots & h_{N_t,N_t} & h_{N_t,1} & \cdots & h_{N_t,N_t-i}
\end{bmatrix}
\]

and \( X^i = \frac{1}{\sqrt{N_t}} [x_{N_t-i+1}, \cdots, x_N, x_1, \cdots, x_{N_t-i}]^T \). Using QR decomposition, \( \mathbf{H}' \) can be decomposed as \( \mathbf{H}' = \mathbf{Q}' \mathbf{R}' \) and (4) can be expressed as

\[
Z^i = \mathbf{R}' X^i + \tilde{N}^i,
\]

where \( Z^i = \mathbf{Q}' H Y \) and \((j,k)\)th element of \( \mathbf{R}' \) is defined as

\[
[R]_{j,k}' = \begin{cases} 
  r_{j,k}', & j \leq k, \quad 1 \leq j, \quad k \leq N_t \\
  0, & j > k.
\end{cases}
\]

Now, let us consider an iterative application of a tree search algorithm such as K-best [3] or IKSD [4] from iteration \( i = 0 \) to \( N_t - 1 \). The vector corresponding to the minimum cost over all the iterations will be taken as the detected vector. While this will give good error performance, the computational complexity will be high. This is because many of the iterations may not result in any improvement. In other words, there will be several redundant iterations. To address this issue, we propose to compute a threshold \( \alpha \) which will be used to stop the iterations as follows: if the cost associated with \( \hat{X}^i \) at \( i \)th iteration is less than \( \alpha \), we declare \( \hat{X}^i \) as the detected vector and no further iterations are carried out. The resulting algorithm is given in Algorithm-1, where \( qr\_dec(\mathbf{Q}, \mathbf{R}, Z) \) provides the QR-decomposition and the equivalent received vector for the current iteration using the QR-decomposition and the equivalent received vector of the previous iteration. Since this particular illustration of the proposed idea uses IKSD, we refer to it as Iterative IKSD (IIKSD). The same can be done for any BFTS algorithm.

It is important to point out here that in Algorithm-1, \( qr\_dec(\mathbf{Q}, \mathbf{R}, Z) \) is required to avoid a fresh QR decomposition at each iteration. Thus we compute the full QR decomposition only once i.e. at the very first iteration and use it in the subsequent iterations. We have provided a method for computing the QR decomposition of \( \mathbf{H}' \) using the QR decomposition of \( \mathbf{H}^{-1} \) in Algorithm-2. This updation process reduces the order of complexity of QR decomposition from \( O(N_t^2) \) to \( O(N_t^2) \). Here \( circh\_shift(\mathbf{R}, 1) \) denotes circular shift of the columns of \( \mathbf{R} \) from left to right by one unit, as in (5) for \( i = 1 \).

<table>
<thead>
<tr>
<th>Algorithm 1 Iterative IKSD (IIKSD) Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> ( \hat{Y}, \mathbf{H}, K, \Omega, \Delta, cost = \infty )</td>
</tr>
<tr>
<td><strong>Output:</strong> ( \hat{X} )</td>
</tr>
<tr>
<td><strong>Initialization</strong> ( i = 0 );</td>
</tr>
<tr>
<td>[ [\mathbf{Q}' \mathbf{R}' Z^i] = qr_dec(\mathbf{Q}, \mathbf{R}, Z) ];</td>
</tr>
<tr>
<td>while ( i &lt; N_t ) do</td>
</tr>
<tr>
<td>( C = 0 ) and start with level ( k = N_t );</td>
</tr>
<tr>
<td>while ( k \geq 1 ) do</td>
</tr>
<tr>
<td>( l = 1 );</td>
</tr>
<tr>
<td>for ( j \leftarrow 1 ) to length (( C )) do</td>
</tr>
<tr>
<td>( \tilde{c}_j = c_j +</td>
</tr>
<tr>
<td>( Z_j^i = Z_j^i - r_{j,k}^i x_1 \forall x_k \in \Omega );</td>
</tr>
<tr>
<td>( l = l + 1 );</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>Sort ( C ) in an ascending order and ( \tilde{Z}^i ) accordingly;</td>
</tr>
<tr>
<td>if length (( \tilde{C} )) ( \leq K ) then</td>
</tr>
<tr>
<td>Keep all the symbols in tree ( T );</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>Keep all the symbols which satisfy ( \tilde{c} \leq \tilde{c}_K + \Delta ) in Tree;</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>Replace ( C \leftarrow \tilde{C}, Z \leftarrow \tilde{Z} );</td>
</tr>
<tr>
<td>( k = k - 1 );</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>Return ( \hat{X}^i ) ← the first element in Tree;</td>
</tr>
<tr>
<td>if ( (|Z^i - \mathbf{R}' \hat{X}^i|^2 \leq \alpha) ) then</td>
</tr>
<tr>
<td>( \hat{X} = \text{cirk_shiff} (\hat{X}^i, N_t - i) ) and ( i = N_t );</td>
</tr>
<tr>
<td>else if ( (|Z^i - \mathbf{R}' \hat{X}^i|^2 &lt; cost) ) then</td>
</tr>
<tr>
<td>( cost = |Z^i - \mathbf{R}' \hat{X}^i|^2 );</td>
</tr>
<tr>
<td>Replace ( \mathbf{Q} \leftarrow \mathbf{Q}' ), ( \mathbf{R} \leftarrow \mathbf{R}' ) and ( Z \leftarrow Z^i );</td>
</tr>
<tr>
<td>( \hat{X} = \text{cirk_shiff} (\hat{X}^i, N_t - i) );</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>( i = i + 1 );</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

IV. STOPPING RULES

The error performance and computational complexity of the above proposed IIKSD depends on the choice of \( \alpha \). If the value of \( \alpha \) is high, it causes early termination of IIKSD and thus loss in error performance. While if we choose a small \( \alpha \), it will lead to redundant iterations and hence increase in complexity. This means we need to select the value of \( \alpha \) carefully. Ideally, \( \alpha \) should have two properties: i) easy to compute and ii) ability to stop IIKSD as soon as a quasi-ML solution is obtained. In this section we propose two rules to compute the value of \( \alpha \).

A. Cost Based Rule

In the first rule, we propose to use a constellation based heuristic. This takes the value of \( \alpha \) as half of the minimum Euclidean distance between any two possible transmit vectors,
Algorithm 2 qr\_dec(Q, R, Z): QR-Decomposition

Input: Q, R, Z
Output: \( Q, R, Z \)

Initialization: \( i = N_t \);
while \( i > 1 \) do
  \( g = \frac{r_{t,i}}{r_{t,i-1}} \);
  \( j = N_t \);
  while \( j \geq 1 \) do
    \( q_{j,i-1} = q_{j,i-1} + g \cdot q_{j,i} \);
    if \( j \geq i - 1 \) then
      \( r_{i,j} = r_{i,j} - g \cdot r_{i-1,j} \);
    end
    \( j = j - 1 \);
  end
  \( z_{i-1} = z_{i-1} + g \cdot z_i \);
  \( i = i - 1 \);
end
\( R = \text{circ\_shift}(R, 1) \), \( \tilde{Q} = Q \) and \( \tilde{Z} = Z \);
return \( \tilde{Q}, \tilde{R} \) and \( \tilde{Z} \);

after they are multiplied with the given channel matrix \( H \). The motivation for this is explained in the next paragraph.

Let us denote \( X_1 \) as a possible transmit vector and for a given \( H \), let \( Y_1 \) denote an observation vector for which the ML solution is \( X_1 \). The cost corresponding to this ML solution is given by \( \|Y_1 -HX_1\|^2 \). Let \( X_2 \) be another transmit vector such that \( HX_2 \) is one of the vectors in the set of vectors nearest to \( HX_1 \). It can be seen that more the distance between \( Y_1 \) and \( HX_1 \), more will be the cost. The farthest value of \( Y_1 \) for which the ML solution will be the mid point of \( HX_1 \) & \( HX_2 \) i.e. \( \frac{1}{2}H(X_1 + X_2) \). The cost for this \( Y_1 \) will be \( \|\frac{1}{2}H(X_1 - X_2)\|^2 \). Now if we choose \( α \) to be such that the cost is not greater than this, we can expect no loss in error performance with respect to the ML solution whenever \( X_1 \) is the ML solution. However, since we do not know the set of vectors nearest to \( HX_1 \) finding such an \( α \) is a non trivial problem because the cost \( \|\frac{1}{2}H(X_1 - X_2)\|^2 \) will have to be minimized over all possible \( X_1 \) and \( X_2 \). Therefore, as a compromise between ease of computation and a reasonable error performance, we consider only those vectors for \( X_2 \) which differ at only one element from \( X_1 \), say the \( i \)th element. It may be noted that this does not necessarily imply that \( X_2 \) is nearest to \( X_1 \). Since only the \( i \)th element of \( X_1 \) & \( X_2 \) differ, \( (X_1 - X_2) \) will have all zero entries except at the \( i \)th place, where it is equal to \( \frac{d_{\min}}{\sqrt{N_t}} \). Thus for a given \( i \), the minimum cost will be \( \left( \frac{d_{\min}}{\sqrt{N_t}} \right)^2 \|H_i\|^2 \), where \( H_i \) denotes the \( i \)th column of \( H \). Hence minimizing over all \( i \), we propose our stopping rule to be the one where \( α \) should be

\[ α = \frac{1}{N_t} \left( \frac{d_{\min}}{2} \right)^2 \min_i \|H_i\|^2. \]  

One can notice that the above expression requires computation of the norm of the columns of \( H \), which is easy to compute and has a low overhead in terms of complexity. The actual overhead is even lesser because once computed, it can be used for the detection of all the vectors, which are received during a given channel realization. Since the expression for \( α \) in (7) is arrived at using a heuristic based on Euclidean cost, we call the stopping rule based on \( α \) as the cost based rule.

B. Distribution Based Rule

In the above rule, we chose \( α \) using the numerical value of \( \|Y -HX\|^2 \). The other way would be decide the value of \( α \) according to the distribution of \( \|Y -HX\|^2 \).

1) Distribution of Cost: The distribution of \( \|Y -HX\|^2 \) is equivalent to \( \|N\|^2 = \sum_{i=1}^{N_t} |n_i|^2 \) where \( n_i \sim CN(0, \sigma^2) \). This can be expressed as a sum of \( 2N_r \) real Gaussian random variables with mean 0 and variance \( \sigma^2/2 \). We know that the squared sum of \( CN(0,1) \) follows a standard chi-square distribution. Let \( t = \frac{\sigma^2}{\sigma^2/2} \left( \sum_{i=1}^{N_t} |n_i|^2 + \sum_{i=1}^{N_r} |\gamma_1|^2 \right) \) denote the standard chi square distributed random variable, then we can express the cost \( \|N\|^2 \) as \( \sigma^2 t/2 \). Hence, the distribution of cost is given by

\[ P(\text{cost} \leq α) = P \left( \frac{\sigma^2}{2} T \leq α \right) = FT \left( \frac{2α}{\sigma^2}, 2N_r \right), \]

where \( FT(t, N_r) = \gamma \left( \frac{N_r}{t}, \frac{1}{2} \right) / \Gamma \left( \frac{N_r}{2} \right) \). Here \( γ \) and \( Γ \) are incomplete gamma function and gamma function respectively.

If \( p \) denotes \( P(\text{cost} \leq α) \), we can set \( α \) as follows

\[ α = \frac{\sigma^2}{2} FT^{-1}(p, 2N_r). \]

The above expression for \( α \) is derived from the distribution of the Euclidean cost, which in turn is related to the distribution of noise. Hence we refer to it as distribution based rule. Here, we would like to point out that the online computational overhead is zero. This is because \( α \) in (9) can be computed offline and need not be computed online.

2) Judicious Choice of \( p \): The performance of IIKSD depends on the value of threshold \( α \) and in this case \( α \) is a function of \( p \) (9). In Fig. 1 we have plotted \( α \) as a function of \( p \) for a fixed value of \( N_t \) and \( \sigma^2 \). From the figure we can observe that initially \( α \) increases linearly and after the dashed vertical line it increases exponentially. A higher value of \( p \) leads to a higher \( α \), which in turn leads to earlier termination of iterations and hence, lower complexity but loss in SER performance. Thus \( p \) can serve as a tuning parameter to get the right combination of SER performance and complexity. From Fig. 1 a judicious choice of \( p \) could be the point up to which \( α \) increases linearly i.e. \( p = 0.8 \).

In the next section we compare the complexity of the two stopping rules and then combine them to obtain a composite stopping rule.

V. COMPOSITE STOPPING RULE

The computational complexity of IIKSD depends on the number of iterations which in turn depends on the choice of \( α \). Let us analyze the expected number of iterations performed by IIKSD for a given \( α \). We denote the probability of the event that the algorithm stops at \( i \)th iteration as \( P(e_i) \), which
can be expressed as a function of the distribution of cost i.e. $P(\text{cost} \leq \alpha)$ given by (8) as follows

$$P(e_i) = P(\text{cost} > \alpha)^{i-1} P(\text{cost} \leq \alpha),$$

where the first term denotes the probability that it does not stop till $i$th iteration while the second term denotes the probability that it stops at $i$th iteration. For $i = N_t$, the second term is equal to one because there will be no any further iterations. Thus from the definition of expectation, we can express the expected number of iterations as the sum of the product of number of iterations and $P(e_i)$ over all possible iterations. This can be mathematically expressed as

$$E[i] = N_t P(\text{cost} > \alpha)^{N_t-1} + \sum_{i=1}^{N_t-1} i P(\text{cost} > \alpha)^{i-1} \times P(\text{cost} \leq \alpha).$$

Let us denote the $\alpha$ obtained from (7) as $\alpha_{\text{cost}}$ and the $\alpha$ obtained from (9) as $\alpha_{\text{dist}}$. The expected number of iterations for $\alpha_{\text{cost}}$ and $\alpha_{\text{dist}}$ (taking $p = 0.8$) are given in Table-I for a 4 x 4 MIMO system with 16-QAM modulation. It can be observed that at low SNR the expected number of iterations is lower for distribution based rule while at high SNR it is lower for cost based rule. This motivates us to combine the two rules to obtain a stopping rule which is better than both.

1) Construction: First let us define a cross-over point $\xi$ as the $E_i/N_0$ at which the complexities of the cost based rule and the distribution based rule are equal. This can be obtained by equating both the $\alpha$ and using (7), (9) and the fact that $E_i/N_0 = \frac{1}{\sigma^2_i}$. Thus we get

$$F_T \left( \frac{d_{\min}^2 \min_i \|H_i\|^2 E_s}{2N_tN_0}, 2N_r \right) = p.$$

Solving (12) for $E_i/N_0$ by using the expression for chi-square distribution, the expression for $\xi$ is obtained as

$$\xi = \frac{4N_t\gamma^{-1}(p\Gamma(N_t), N_r)}{d_{\min}^2 \min_i \|H_i\|^2 E_s}.$$

where $\gamma^{-1}$ is inverse incomplete gamma function.

2) Composite Stopping Rule: IIKSD is stopped when cost (2) of the solution obtained at any iteration is less than $\alpha$, where $\alpha$ is decided by the cost based rule when $E_i/N_0 \geq \xi$ and by the distribution based rule when $E_i/N_0 \leq \xi$.

VI. SIMULATION RESULTS

In this section we present the simulation results for the proposed algorithm applied over uncoded as well as coded MIMO systems. The results are compared with some recent tree search algorithms such as IKSD [4] and SPSD [11], and with the well established tree search algorithm K-best [3]. For tree search algorithms, it is common to measure their computational complexity in terms of the average number of nodes visited. For a moment, if we forget about the performance and focus only on the complexity then one can say that the algorithm which traverses lesser number of nodes in the tree is better. Now, the question is how much lower we can go i.e. the minimum number of nodes to be visited by any tree search algorithm irrespective of whether it belongs to BFTS or DFTS [1], [10]. In the next subsection we compute this minimum number of nodes for BFTS as well as DFTS algorithms and show that this lower bound is same for both the class of algorithms. Later this lower bound is used to compare different algorithms.

A. Minimum Number of Nodes Required

For BFTS this can be computed as follows: at each antenna at least one node has to be selected and for the equivalent real system model [3], there are $2N_t$ numbers of antennas and $\sqrt{M}$ choices at each antenna i.e. the minimum number of nodes required for this class of algorithms is $2\sqrt{M}N_t$. This holds for DFTS also. For understanding this, let us consider the best scenario where we get the ML solution in the first depth itself. It means that by visiting the first $2N_t$ nodes, we will have the ML solution. Next, the algorithm visits rest of the $\sqrt{M} - 1$ nodes at least once at each antenna, starting from the $N_t$th antenna, and sequentially moving on to each antenna one by one. Hence, the total number of nodes visited by this class of algorithms is sum of the first $2N_t$ nodes and the rest $2N_t(\sqrt{M} - 1)$ nodes, which yields the same result as BFTS. Hence, the minimum number of nodes required to be visited by a tree search algorithm is $2\sqrt{M}N_t$.

B. Comparison of the Three Stopping Rules

Using the equivalent real system model [3], the three stopping rules have been simulated for the proposed IIKSD for a 4 x 4 MIMO system with 4-QAM modulation assuming channel conditions as flat Rayleigh faded. To obtain near ML performance, the simulation parameters were taken to
be $K = 1$ and $\Delta = 0.5/8$.\textsuperscript{1} In Fig. 2, SER performance and the average number of nodes visited by all the three stopping rules have been plotted respectively. We can see that the composite stopping rule has the lowest complexity and it is worth comparing the average number of nodes visited with the lower bound for this class of algorithms i.e. $2\sqrt{MN_t}$. For the simulated case, the lower bound comes out to be 16 and interestingly, the composite stopping rule achieves it at high $E_s/N_0$ (pink dotted line).

C. Comparison With Other Detectors

To show the utility of IIKSD with composite stopping rule, we now compare it with the best reported algorithms in the class of BFTS and DFTS algorithms i.e. IKSD [4] and SPSD [11] respectively. The simulations have been done for the same scenario as in [4] and [11] respectively. Thus, an $8 \times 8$ MIMO system with 16-QAM modulation has been considered and the simulation parameters for IKSD have been taken to be $K = 4$ and $\Delta = 0.75/16$, same as in [4] and for SPSD geometrical pruning rule is considered with $K_0 = 3$, same as in [11]. To achieve the same SER performance as IKSD, the values for IIKSD have been taken to be $K = 1$ and $\Delta = 0.25/16$. The results obtained with these parameters have been plotted in Fig. 3. From the figure we can say that IKSD, IIKSD and SPSD have identical SER performance which is close to that of sphere decoder (SD). Here we consider SD as a benchmark instead of ML. This is because the simulation for ML for such a large system is computationally demanding and SD is well known for providing ML performance at a much lower complexity. Further, IIKSD consistently visited lesser number of nodes than IKSD as well as SPSD. One can observe from the figure that the difference between the complexities of these algorithms is not constant. In particular, the difference between the complexities of IIKSD and IKSD is minimum at $E_s/N_0 = 17.5$ dB and maximum at $E_s/N_0 = 25$ dB. At these points, on an average IIKSD visited only 300 and 123 nodes in comparison to 472 and 497 nodes of IKSD respectively. This means that for the simulated system, the proposed IIKSD requires 36% to 75% less computations than IKSD. Comparing with SPSD one can notice that IIKSD visits lesser number of nodes and the difference varies between 300 nodes ($E_s/N_0 = 0$ dB) to 460 nodes ($E_s/N_0 = 25$ dB). In fact, one can also observe that the number of nodes visited by IIKSD approaches the lower bound of 64 at high $E_s/N_0$. Since IIKSD provides quasi-ML error performance and visits lesser number of nodes than others, this makes it a good choice for low-complexity detection in uncoded MIMO system. For the sake of completeness, we have also compared IIKSD with the K-best algorithm [3] and two channel ordering schemes (COS) suggested in [7] and [8]. For simplicity, we have named them as COS-1 and COS-2 respectively. It can be observed that K-best ($K = 32$) and COS-2 have almost the same SER performance as IIKSD while at the same time K-best requires 6-15 times more computations and COS-2 requires significantly higher complexity. On the contrary, COS-1 visits less number of nodes as compared to IIKSD but at the cost of a much inferior SER performance.

Next we examine the performance for a higher constellation i.e. 64-QAM for a $6 \times 6$ MIMO system. Just like the previous cases, we have chosen the simulation parameters such that they achieve quasi-ML (i.e. SD) performance. Thus, $K = 1$ and $\Delta = 0.2/12$ for IIKSD, $K = 2$ and $\Delta = 0.25/12$ for IKSD and $K = 64$ for K-best have been taken. The simulation results are shown in Fig. 4. From the figure one can observe trends similar to previous case. The gap between the number of nodes visited by IIKSD and IKSD is closest at 25 dB. At this point IIKSD visits 831 nodes in comparison to 1059 nodes of IKSD, which leads to a minimum of 21.5% savings in complexity.

D. Complexity Comparison of IIKSD and IKSD

Theoretically, the complexity of K-best is known to be of the order of $O(KMN_t^2)$. Extending it to IKSD, we can simply replace $K$ by $K_{avg}$ where $K_{avg}$ denotes the average value of $K$ which depends on both $K$ and $\Delta$. For $\Delta = 0$\textsuperscript{1}The factor in the denominator is equal to $2N_t$ for an equivalent real system model.
TABLE II

<table>
<thead>
<tr>
<th>Type of Detector</th>
<th>8 × 8 MIMO 16-QAM</th>
<th>6 × 6 MIMO 64-QAM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10dB</td>
<td>15dB</td>
</tr>
<tr>
<td>IIKSD</td>
<td>239/240</td>
<td>391/393</td>
</tr>
<tr>
<td>IKS</td>
<td>544/544</td>
<td>606/606</td>
</tr>
<tr>
<td>K-best (K=94)</td>
<td>184/184</td>
<td>216/216</td>
</tr>
<tr>
<td>SPSS</td>
<td>159/159</td>
<td>212/212</td>
</tr>
<tr>
<td>COS-1</td>
<td>597/597</td>
<td>629/629</td>
</tr>
<tr>
<td>COS-1 (L=0)</td>
<td>170/170</td>
<td>170/170</td>
</tr>
<tr>
<td>COS-2 (b)</td>
<td>252/252</td>
<td>252/252</td>
</tr>
<tr>
<td>COS-2 (L=0)</td>
<td>170/170</td>
<td>170/170</td>
</tr>
</tbody>
</table>

E. Arithmetic Operations

Most works in the literature [4], [8], [11] use the number of nodes visited or the number of arithmetic operations to compare the complexities of algorithms. Since we have compared the complexities of various algorithms on the basis of average nodes visited and have also established the superiority of IIKSD over IKSD, it will be interesting to investigate the gains in terms of the total number of arithmetic operations.

In Table II we have shown the number of additions, multiplications and the number of parallel computations (discussed in the next section) as well. For a fair comparison, the numbers shown include the complexity of pre-processing operations. For the convenience of the reader, we show the number for the proposed IIKSD and also the number closest to it from amongst all the other algorithms in bold face. From the table we can observe that for an 8 × 8 system with 16-QAM modulation and for a 6 × 6 system with 64-QAM modulation, compared to all the other algorithms IIKSD requires significantly lower number of additions and multiplications throughout the SNR range. In the case of 8 × 8 and 6 × 6 systems, IIKSD is found to be closest to IKSD, with respect to which the reduction in complexity is at least 35% and 27% respectively. These observations match with the conclusions arrived at from Fig. 3 and Fig. 4, except for COS-1 which requires a higher number of additions and multiplications. This is because the pre-processing required to find the optimal channel order in COS-1 has significant complexity which has not been accounted for in Fig. 3 and Fig. 4.

F. Processing Latency

Next, we study the processing latency of the algorithms. For DFTS, it is directly proportional to the visited number of nodes while for BFTS, it is fixed assuming there is no constraint on the memory. Since the proposed IIKSD is somewhere in between these two algorithms (closer to BFTS), it is advisable to check the extent of parallelism in the algorithms [18]. Hence we have compared these algorithms on the basis of the total number of parallel operations in Table II. Here, by a parallel operation we mean a vector-vector operation, which is taken to be as one unit of computation.

It can be observed from the table that IIKSD requires the lowest number of parallel operations too for an
8 × 8 MIMO system with 16-QAM and for a 6 × 6 MIMO system with 64-QAM. These observations strengthen the conclusions made in Section VI-C that IIKSD requires a lower average number of nodes, lower number of additions, lower number of multiplications and lower number of parallel operations in comparison to the recently proposed IKSD [4] and SPSD [11].

G. CPS Required to Achieve Quasi-ML Performance

Now we examine the number of computations per symbol (CPS), in terms of total number of arithmetic operations i.e. (additions + multiplications) per symbol, required to achieve quasi-ML performance as a function of the number of antennas \( N_t = N_r \). We show the results for 16-QAM modulation, at an SNR of 10 dB, in Fig. 5. The simulation parameters for K-best, IKSD and IIKSD have been chosen such that all the algorithms achieve identical (quasi-ML) error performance. Thus for the 8 × 8 system we have taken the same parameters as used in Fig. 3. For the 4 × 4 system, we have taken \( K = 1 \) and \( \Delta = 0.2/8 \) for IIKSD, \( K = 2 \) and \( \Delta = 0.25/8 \) for IKSD same as [4] and \( K = 16 \) for K-best same as [3]. Similarly for the 12 × 12 system, we have chosen \( K = 3 \) and \( \Delta = 0.8/24 \) for IIKSD, \( K = 16 \) and \( \Delta = 0.9/24 \) for IKSD and \( K = 96 \) for K-best. From the figure it can be seen that the CPS for IIKSD is the lowest. Further, compared to IKSD its relative gain increases as the number of antennas increase. While evaluating the gains numerically, it may be noted that the results in Fig. 5 are plotted on a logarithmic scale. For example, at an SNR of 10 dB, for a 12 × 12 system with 16-QAM modulation IIKSD requires 1544 number of arithmetic operations while 5284 number of operations are required by IKSD. This translates to a 71% savings in complexity.

H. Extension to Coded Systems Using Soft Decoding

In this section, we evaluate the performance of IIKSD for coded MIMO systems. For soft detection in coded MIMO systems, we need log likelihood ratio (LLR) values to be computed over a candidate set. This candidate set can be generated for IIKSD as follows: in Algorithm-1 instead of returning the first element of the tree, we store all the elements in the candidate set. This set is updated in the subsequent iterations until the algorithm terminates. The final candidate set is used for iterative decoding as suggested in [19].

For evaluating the performance of coded systems, a rate 1/2 recursive systematic convolutional (RSC) code with memory 2 and a random interleaver of length 8192 bits has been considered for a 4 × 4 MIMO system with 16-QAM modulation. The convolutional code feedforward generator polynomial is \( 1 + D^2 \) and the feedback generator polynomial is \( 1 + D + D^2 \). The results obtained are compared in Fig. 6 with K-best and IKSD for the same simulation parameters as in [4]. Thus, we have taken \( K = 512 \) for K-best and \( K = 256, \Delta = 0.25/8 \) for IKSD. The results in Fig. 6 show that at any particular iteration of iterative decoding, IIKSD \( (K = 128, \Delta = 0.2/8) \), IKSD \( (K = 256, \Delta = 0.25/8) \) and K-best \( (K = 512) \) have almost the same error performance (K-best is slightly inferior) while they visit approximately \( 2.5 \times 10^3, 4.5 \times 10^3 \) and \( 7.5 \times 10^3 \) number of nodes respectively. Thus, we can see that IIKSD offers a lower complexity solution for coded systems as well.

VII. APPLICATIONS

The advantages of the proposed schemes can be extended to other classes of tree search algorithms and also to large antenna systems. In particular, the proposed stopping rules (refer Section IV and Section V) can be used to reduce the complexity of depth first tree search (DFTS) algorithms and the proposed IIKSD seems to be a good candidate for large antenna systems. In this section, we examine both these issues.

A. DFTS Algorithms

In this section we show that the utility of the composite stopping rule is not limited to BFTS algorithms and that it is useful for DFTS algorithms as well. In a DFTS algorithm, we select a node and then go to the depth of the tree. This is repeated for each node. In the process, whenever a better solution (in terms of cost) is achieved, we replace the old
solution with the current solution and the stopping rule is used as a tool for reducing redundant searches. For this, we need to modify the basic DFTS algorithm. Thus, whenever a better solution is achieved we compare the cost of the solution to \( \alpha \) (refer Section V). If it is lower than the threshold, we stop the search process and the best solution till then is declared as the final solution. The complete procedure is provided in Algorithm-3.

Based on the above, we apply the composite stopping rule on two different DFTS algorithms namely COS-2 [8] and SPSD [11]. On the basis of simulations, the error performance and the number of nodes visited are presented in Fig. 7 and compared with their original versions. It may be noted that the additional complexity because of the stopping rule, is negligible and that for DFTS algorithms, the number of nodes visited is a measure of their complexity. From the figure we can see that upto 15 dB, there is a significant reduction in the number of nodes visited for COS-2 and SPSD without any loss in SER performance. This shows that the benefit of the composite stopping rule can be extended to DFTS algorithms.

### Algorithm 3 A DFTS Algorithm With Stopping Rule

**Input**: \( Y, H, \Omega, \alpha \)

**Output**: \( \hat{X} \)

**Initialization**

\( i = N_t, \text{cost} = \infty, \bar{c}_i = 0; \)

\([Q, R] \leftarrow \text{QR decomposition of } H \text{ and } Z = Q^H Y; \)

\( \hat{X} \leftarrow \text{DFTS}(Z, R, \Omega, \alpha, \text{cost}, \bar{c}_i, d, i); \)

**Function** DFTS\((Z, R, \Omega, \alpha, \text{cost}, \bar{c}_i, d, i)\)

for \( j \leftarrow 1 \) to length\((\Omega)\) do

\( c_j = |z_i - r_i, x_j|^2, \forall x_j \in \Omega; \)

end

Sort \( c_j \)'s in ascending order and keep only those symbols for which \( c_j < (\text{cost} - \bar{c}_i) \);

if \( c_j \leq (\text{cost} - \bar{c}_i) \) then

return \( \hat{X}, \text{cost}; \)

else

for \( k \leftarrow 1 \) to length\((C)\) do

\( \hat{x}_i = x_k; \)

\( \bar{c}_i \leftarrow \bar{c}_i + c_k; \)

if \( i = 1 \) then

if \( \bar{c}_i < \text{cost} \) then

\( \text{cost} \leftarrow \bar{c}_i; \)

return \( \hat{X}, \text{cost}; \)

else

\( \text{if } \text{cost} < \alpha \) then

\( \text{Terminate the algorithm}; \)

end

end

else

\( Z = Z - R_c, x_k; \)

Extend the tree \( T \) for all \( \Omega; \)

\( [\hat{X}, \text{cost}] \leftarrow \text{DFTS}(Z, R, \Omega, \alpha, \text{cost}, \bar{c}_i, i - 1); \)

end

end

end

Fig. 7. Bit error performance and average number of nodes visited per transmit vector for an \( 8 \times 8 \) MIMO system with 16-QAM modulation.

### B. Large MIMO Systems

Earlier, we have seen that in terms of CPS the relative gain offered by IIKSD compared to K-best [3], IKSD [4], COS-2 [8] and SPSD [11] increases as the number of antennas increase (refer Section VI-G). Motivated by this, we applied IIKSD on a large antenna system, specifically, on a \( 32 \times 32 \) MIMO system with \( 4\text{-QAM} \) modulation. However, simulating a \( 32 \times 32 \) system using K-best [3], IKSD [4], COS-2 [8] or SPSD [11] is quite difficult, because of their very high computational complexity, which is known to increase rapidly with the number of antennas (see Fig. 5). This difficulty has also been pointed out in [20] and [21] for the variants of SD and K-best. Therefore, we have compared IIKSD with some of the well-known low-complexity algorithms for large MIMO such as 1-LAS [12], MLAS [13] and RTS [14], [15] and also with a recent low-complexity sparsity boosted iterative linear (SBIL) detector [16]. The simulation parameters for IIKSD have been chosen such that there is no improvement in error performance for higher values of \( K \) and \( \Delta \). This leads to the choice of \( K = 24 \) and \( \Delta = 4/64 \). The resulting bit error performance is shown in Fig. 8. Due to the difficulty in obtaining the performance of near ML detectors (because of their very high computational complexity) for such large systems, the existing detectors for large MIMO systems are usually compared with the performance of a single antenna system over an additive white Gaussian noise (AWGN) channel [16]. Hence, we have also provided the performance for an AWGN channel in Fig. 8. From the figure it is clear that IIKSD is exceptionally better than all the others, even better than an AWGN channel. It may be noted that the AWGN performance is different from the ML performance of the MIMO system under consideration.

The algorithms have been compared in terms of computations per bit in Fig. 9. Interestingly, it turns out that except for SBIL, IIKSD has the lowest computations per bit. But since SBIL has the worst error performance, its lower complexity is not of much value. The exceptional gain of IIKSD in terms of error performance is not unexpected (since it achieves quasi-ML performance for smaller MIMO systems) but the gain
in terms of complexity is attractive and makes it a suitable candidate for large antenna systems also.

VIII. CONCLUSION

We propose an iterative scheme around the class of BFTS algorithms and two stopping rules for detection in MIMO systems. The stopping rules are based on a comparison of the cost of the detected vector with a threshold. The first rule is based on the Euclidean distance of two neighboring transmit vectors and makes it an attractive candidate for large antenna systems too. We have also applied the stopping rule to DFTS algorithms and found it to be effective in reducing their complexity.

REFERENCES

Abhay Kumar Sah (S’15) received the B.Tech. degree in Electronics and Communications Engineering from the Jaypee University of Information Technology, Waknaghat (H.P.), India, in 2010. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering, IIT Kanpur, Kanpur, India. His current research interests are in wireless communication and multi-antenna systems. He was a recipient of the TCS Research Scholarship and the Vice-Chancellor’s Gold Medal for outstanding performance in his bachelor’s degree.

A. K. Chaturvedi (S’91–M’96–SM’03) received the B.Tech., M.Tech., and Ph.D. degrees in Electrical Engineering from IIT Kanpur, Kanpur, India, in 1986, 1988, and 1995, respectively. He was with the Department of Electronics Engineering, IIT BHU, Varanasi, India, from 1994 to 1996. Subsequently, he joined the faculty of the Department of Electronics and Computer Engineering, IIT Roorkee, Roorkee, India. Since 1999, he has been with the Department of Electrical Engineering, IIT Kanpur. He was the Head of the Department from 2009 to 2010 and the Dean of Research and Development, IIT Kanpur, from 2011 to 2014. He is currently the Deputy Director of the Institute and the Coordinator of the BSNL-IITK Telecom Centre of Excellence, IIT Kanpur. His current research interests are in communication theory and wireless communication. He was a recipient of the Distinguished Teacher Award of IIT Kanpur and Tan Chin Tuan Fellowship of Nanyang Technological University, Singapore. He has actively participated in the activities of the Uttar Pradesh section of the IEEE. He has held several positions within the section and was also its Chair from 2004 to 2005.