IMPLEMENTATION ISSUES OF A LIST SPHERE DECODER

Jin Lee[*], Sungchung Park[**], Yuping Zhang[***], Keshab K. Parhi[***] and Sin-Chong Park[*]

Information and Communications University[*]
Korea Advanced Institute of Science and Technology[**]
University of Minnesota[***]

ABSTRACT

Since finding the nearest point in a lattice for multi-input multi-output (MIMO) channels is NP-hard, simplified algorithms such as sphere decoder (SD) have been proposed. List sphere decoder (LSD), which is a modified version of SD, allows soft information to be extracted for channel decoding and iterative detection/decoding. In this paper, recently proposed efficient methods for reducing the computational complexity of SD and LSD with depth-first tree searching are summarized. Numerous simulations have been carried out and comparison has been made based on the average number of processing cycles. We also present two efficient schemes which can decrease hardware complexity without significant performance degradation, restricted list updating in LSD and restricted node storing at each tree level.

1. INTRODUCTION

Enormous researches on MIMO techniques have been shown in the past decade. The optimal MIMO detector, maximum likelihood (ML) detector, is infeasible because of prohibitively high complexity when a large number of antennas are used together with high-order modulation [1]. Efficient algorithms such as sphere decoder (SD) have been proposed for reducing computational complexity of ML detector. VLSI implementation of SD has been studied in several papers. For example, in [1] and [2], efficient implementations of depth-first SD are presented with almost no performance penalty.

The iterative processing technique, applied in Turbo and LDPC codes significantly improve the performance of coded MIMO systems [3]. MIMO detector should be able to generate soft reliability information to be used for consecutive outer soft-input/soft-output (SISO) decoder. In SD, only the ML solution is extracted. For generating soft information, a certain number of lattice points including the ML solution are required. Thus, SD can be modified to be a list sphere decoder, which finds a list of most likely points, and the soft information can be generated based on corresponding distances of these points [3]. In [1]-[4], some novel techniques which can reduce the computational complexity of SD or LSD are presented for depth-first tree searching, for example, complex/real valued operation, radius update with Schnorr-Euchner enumeration, storage pruning and antenna ordering. In this paper, we summarize these techniques and describe the advantages and disadvantages of each technique.

In this paper, we also present two efficient schemes which can decrease hardware complexity without significant performance degradation, restricted list updating in LSD and restricted node storing at each tree level. In the case of LSD, a list of the candidates for generating soft information is required and the list is updated whenever the leaf nodes of the tree are evaluated. Since the implementation of full list updating results in extremely-high complexity, we propose a restricted list updating scheme. SD or LSD using depth-first tree searching needs to store intermediate results. We also present a restricted node storing scheme, which significantly reduces the size of the storage with negligible performance loss.

This paper is organized as follows. Section 2 briefly presents SD algorithm and the generation of the soft information. In Section 3, the efficient methods for reducing the number of processing cycles are summarized. In section 4, the proposed restricted list updating scheme and restricted node storing scheme are presented, and we conclude the paper in section 5.

2. PROBLEM DESCRIPTION

Consider the following generic model:

$$\underline{y} = H\underline{s} + \underline{n}, \tag{1}$$

where $\underline{s}$ is $N_t \times 1$ transmitted vector symbol of which the i-th entry, s_i, is a complex symbol chosen from a constellation with 2^{M_c} possible signal points, C. Here s_i is given by

$$s_i = map\left(\underline{x}^{(m)}\right), \quad m = 1, \cdots, N_t,$$

where $\underline{x}^{(m)}$ is $M_c \times 1$ coded bit vector. $\underline{y}$ is $N_r \times 1$ received vector symbol, H is $N_r \times N_t$ i.i.d. zero-mean unit

variance complex Gaussian matrix that is known perfectly to the receiver and $\underline{n}$ is $N_r \times 1$ additive noise.

2.1. Sphere decoding algorithm

SD algorithm can be divided into two parts: the calculation of a partial Euclidian distance (PED) and the control of tree traversal. The following inequality expresses the sphere constraint :

$$\left\| \underline{y} - H\underline{s} \right\|^2 < r^2 . \tag{2}$$

The inequality in (2) can be decomposed as follows

$$\left\| \hat{\underline{y}} - R\underline{s} \right\|^2 = \sum_{i=1}^{N_t} \left\| \hat{y}_i - \sum_{j=i}^{N_t} r_{i,j} s_j \right\|^2 , \tag{3}$$

where $\hat{\underline{y}} = Q^H \underline{y}$, Q and R are calculated from QR decomposition of H , and $r_{i,j}$ is the element of the upper triangular matrix R . A PED $T_i\left(\underline{s}^i\right)$ can be rewritten as

$$T_i\left(\underline{s}^i\right) = T_{i+1}\left(\underline{s}^{i+1}\right) + \left\| \left(\hat{y}_i - \sum_{j=i}^{N_t} r_{i,j} s_j \right) - r_{i,i} s_i \right\|^2 . \tag{4}$$

An efficient depth-first tree searching can be carried out by calculating the PEDs of all children nodes of a mother node in one cycle. Fig. 1 illustrates tree searching and pruning shown in SD. The number inside each circle (node) represents the corresponding PED. In this paper, tree level *i* is presented in descending order. The boxes in the figure indicate the nodes calculated at same time. Dark gray nodes are pruned in the tree, since the PEDs of their mother nodes have exceeded the radius *r*.

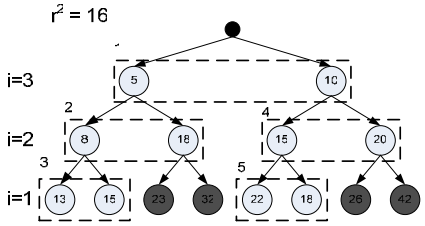

Figure 1. Example of tree traversal

2.2. Generation of soft information and list sphere decoder

In [3], iterative detection and decoding algorithms are presented. For soft channel decoding or iterative detection/decoding, the soft information of each coded bit is required to be calculated.

The log-likelihood ratio (LLR) value of a posteriori probability is given as

$$L_D\left(x_k \mid \underline{y}\right) = L_A\left(x_k\right) + L_E\left(x_k \mid \underline{y}\right), \quad k = 0, \cdots, N_t M_c - 1 \tag{5}$$

where x_k is the *k*-th coded bit, $L_A\left(x_k\right)$ is a priori LLR value and $L_E\left(x_k \mid \underline{y}\right)$ is an extrinsic LLR value. $L_A\left(x_k\right)$ is provided by the channel decoder, and $L_E\left(x_k \mid \underline{y}\right)$ is calculated and then sent to the channel decoder from LSD. Computational complexity of $L_E\left(x_k \mid \underline{y}\right)$ is exponential in the length of the coded bit vector $\underline{x} = \left\{ \underline{x}^{(m)} \right\}_{m=1}^{N_t} = \left\{ x_k \right\}_{k=0}^{N_t M_c - 1}$ or the number of symbols in the constellation C [3].

A simple modification to the SD helps us to compute $L_D\left(x_k \mid \underline{y}\right)$. To be specific, $L_D\left(x_k \mid \underline{y}\right)$ is calculated as

$$L_D\left(x_k \mid \underline{y}\right) = \max_{x_k = 1} \left(-N_0^{-1} \left\| \underline{y} - H\underline{s} \right\|^2 + \sum_{j=1}^{N_t \log_2 M_c} L_A\left(x_j\right) \right)$$

$$- \max_{x_k = 1} \left(-N_0^{-1} \left\| \underline{y} - H\underline{s} \right\|^2 + \sum_{j=1}^{N_t \log_2 M_c} L_A\left(x_j\right) \right), \tag{6}$$

where N_0 is the single-sided noise power spectral density.

LSD generates the list for generating soft information. If the list is not full, LSD does not decrease radius and it only adds the searched points to list. On the other hand, if the list is full, it compares the searched points with the existing points with the largest distance in the list, and replaces this point if the searched points have smaller distance and updates the radius accordingly. The detailed description of calculation of soft information is beyond the scope of this paper.

3. RECENT IMPROVEMENTS OF IMPLEMENTATION EFFICIENCY

The average throughput of SD or LSD (in bits per second) is given by

$$E\left[Throughput\right] = \frac{N_t M_c}{E\left[D\right]} \times f_{clk} \tag{7}$$

$E\left[D\right]$ is the average number of processing cycles per received symbol vector and f_{clk} is the operating clock frequency determined by the critical path on the circuit implementation side. In this section we summarize the recent improvements of implementation efficiency, focusing on the reduction of $E\left[D\right]$.

3.1. Complex-valued vs. real-valued operation

In modern wireless communications, the symbol is generally expressed by complex values. However, the symbol can be decomposed into real values [1], [3]. The real-valued operation can reduce the complexity of PED calculation and the size of the storage for intermediate results, since the number of children becomes smaller than that of the complex-valued operation. Specifically, while the number of children calculated simultaneously is 2^{M_c} for complex-valued operation, only $2^{M_c/2}$ children need to be calculated for real-valued operation. For example, in 4x4 64QAM configuration, the storage unit for real-valued

operation has 56 entries, while the storage unit for complex-valued operation has 192 entries. Here, a single entry consists of the level of the node *i*, history of the node $\underline{s}^i$ and PED $T_i\left(\underline{s}^i\right)$ [2]. The drawback of real-valued operation is that the depth of tree becomes twice as high as that of complex-valued operation, resulting in increased $E[D]$.

3.2. Radius updating with Schnorr-Euchner enumeration

If the radius is too large, $E[D]$ becomes extremely high, making real-time operation impossible. On the other hand, if the radius is too small, even the ML solution cannot satisfy the sphere constraint in (2). Thus setting the appropriate radius is critical to successful implementation of SD and LSD. The authors in [2] and [4] presented radius updating methods that solve this problem. To be specific, the radius is initially set to infinity, and SD (or LSD) finds the first candidate vector (or a specific number of candidate vectors). Then, the radius is set to the distance of the first candidate vector and the maximum distance in the list, respectively. To make radius reduction more rapid, Schnorr-Euchner (SE) enumeration can be utilized for selecting next mother node. With SE enumeration, the next mother node is the currently calculated child node which has the minimum PED. For implementing SE enumeration, additional logic for finding the minimum PED is required, but it can be effectively implemented by the method provided in [1]. Though the complexity increases, SE enumeration can significantly reduce $E[D]$ since the optimum radius can be obtained much earlier.

3.3. Storage pruning

In the middle of tree searching, the level of the node *i*, history of the node $\underline{s}^i$ and PED $T_i\left(\underline{s}^i\right)$ should be stored as intermediate calculations. In [2], the multi-port stack is employed for storage while the cache is used in [1]. Note that the cache mentioned here is somewhat different from that of the microprocessor architecture. After updating the radius, some PEDs in the storage may be larger than the radius. These PEDs should be removed from the storage. Otherwise, redundant operations occur, resulting in increased number of processing cycles. In [4], the predictive pruning of each tree level is proposed. The threshold for predictive pruning can be extracted through the simulations. Either pruning or predictive pruning can be performed with the same number of comparators as the number of entries of the storage, $\left(N_t-1\right)2^{M_c}$.

3.4. Antenna ordering

SD and LSD have been developed based on the upper triangular matrix manipulation, so the tree searching can be applied to solve the problem. Similar to the ordering of V-BLAST, it is more efficient to reduce $E[D]$ if more reliable transmitted symbols are placed at the higher level of the tree. The antenna order can be determined according to the norm of the column of H since the index of the column of H is the index of the transmit antenna. A larger norm represents higher reliability. For this operation, additional computation units for calculating the norms of the columns and additional logics for re-ordering the sequence are required.

3.5. Simulation results

Fig. 2 shows the processing cycles averaged over 65000 channel realizations at different received SNRs. Five different configurations in Table I are considered here. Four transmit antennas, four receive antennas and 64-QAM are assumed. The number of candidate is assumed to be 16. Also, the one-node-per-cycle architecture in [1] is assumed, i.e., the calculation of PED can be performed within a single cycle.

	Options
Configuration 1	Radius updating with Schnorr-Euchner enumeration Real-valued operation
Configuration 2	Radius updating with Schnorr-Euchner enumeration Complex-valued operation
Configuration 3	Radius updating with Schnorr-Euchner enumeration Real-valued operation, Storage pruing
Configuration 4	Radius updating with Schnorr-Euchner enumeration Complex-valued operation, Storage pruing
Configuration 5	Radius updating with Schnorr-Euchner enumeration Complex-valued operation Storage pruing, Antenna ordering

Table I. Description of simulation configurations

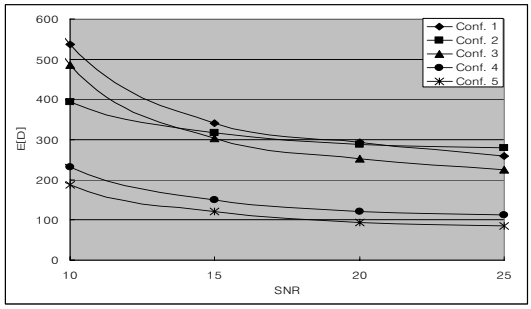

Figure 2. Average number of processing cycles of LSD at 4x4 64QAM

Fig. 2 shows that complex-valued operation with SE enumeration and storage pruning perform the tree searching almost twice faster than real-valued operation. Furthermore, the antenna ordering can reduce $E[D]$ by about 30 cycles.

4. PROPOSED IMPROVEMENTS

4.1. Restricted list updating

In a LSD, a list unit plays a role of list generation. After the calculation of PEDs for all the children nodes of a

mother node, if leaf nodes are reached, the PEDs move to the list unit, which will generate a new list from N_{cd} candidates in the current list and new survived leaf nodes. For generating new list, a sorting for $N_{cd} + 2^{M_c}$ values is required. If N_{cd} and M_c are quite large (for example, 128 candidates and 64-QAM) the sorting unit requires prohibitively high implementation complexity. According to our simulation results, 16 candidates are enough for 4x4 64-QAM LDPC-coded MIMO system [5]. In this paper, we propose a restricted list updating scheme which only considers a few leaf nodes to update the list. Our simulation results show that, even though only 4 leaf nodes with the smallest PEDs are considered in the list updating, the performance degradation is quite trivial, as shown in Fig. 5. The 4 leaf nodes can be easily extracted from the expanded constellation, as shown in [6]. It follows that sorting of only 20 values is required for the list updating. The numbers of outer iterations for iterative detection and decoding and the number of inner iterations for LDPC decoding are indicated in the parentheses in Fig. 5.

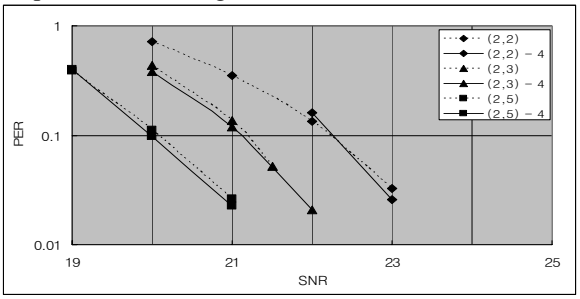

Figure 3. Performance for restricted list updating method

4.2. Restricted node storing

The well-known K-best SD employs restricted breadth-first tree searching so that the throughput is fixed. However, the performance of the K-best SD tends to be degraded. In the K-best SD, at most K nodes are retained at every level for computation of the next level. In this paper, we apply this concept to restricted node storing method. In more detail, the number of storing nodes is restricted to a small fraction, even though more children nodes may have smaller PED than the radius. Since the nodes of the higher tree level are more important than those of the lower level, different numbers of children nodes are stored according to the levels of the tree. Fig. 4 shows the performance degradation due to the restricted node storing method assuming 2 outer iterations and 3 inner iterations.

Fig. 5 shows the reduction of the amount of storage due to the restricted node storing method. In Fig.4 and Fig.5, the numbers of children nodes to be stored are indicated from the top level to the bottom level. For example, "64-16-4-4" represents 64 nodes at the top level and 4 nodes at the bottom level. According to Fig. 4 and Fig. 5, "64-8-4-4" shows a good tradeoff between storage amount and performance. To be specific, the amount of storage can be reduced by 60%, while performance degradation is less than 0.5dB.

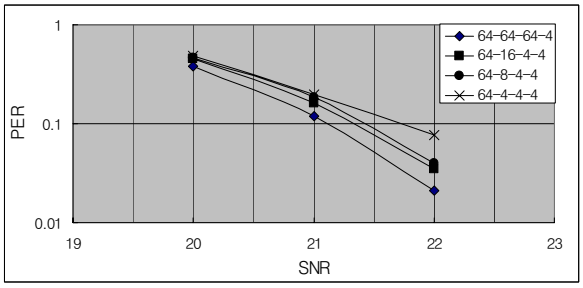

Figure 4. Performance for restricted node storing method

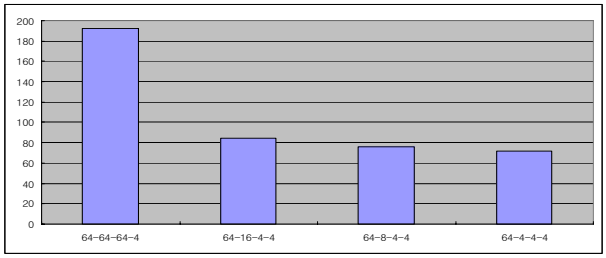

Figure 5. Number of entries of the storage in LSD

5. CONCLUSION

In this paper, we summarize several methods to improve the hardware efficiency, apply these method to LSD, and estimate the average number of processing cycles for 4x4 64-QAM. We also propose restricted list updating method and restricted node storing method that can reduce the hardware complexity while retaining the error performance.

6. REFERENCES

[1] A.Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Boelcskei., "VLSI implementation of MIMO detection using the sphere decoding algorithm," *IEEE J. Solid-State Circuits*, vol. 40, pp. 1566-1577, July 2005.

[2] D. Garrett, L. Davis, S. ten Brink, B. Hochwald, and G. Knagge, "Silicon complexity for maximum likelihood MIMO detection using spherical decoding," *IEEE J. Solid-State Circuits*, vol. 39, pp. 1544-1552, Sep. 2004.

[3] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, pp. 389-399, Mar. 2003.

[4] B. Widdup, G. Woodward, and G. Knagge, "A highly-parallel VLSI architecture for a list sphere detector," *IEEE Int. Conf. Commun. 2004*, vol. 5, pp 2720-2725, June 2004.

[5] J. Lee and S. Park, "Performance evaluation of various LDPC coded MIMO-OFDM receiver architectures," *Int. Tech. Conf. on Circuts /Systems, Computers and Communications 2005*, pp. 1157-1158, June 2005.

[6] J. Lee, S. Park and S. Park, "A pipelined VLSI architecture for a list sphere decoder," *Int. Symp. on Circuits and Systems, 2006*, May 2006, to be presented.