FIXED-COMPLEXITY VARIANTS OF THE EFFECTIVE LLL ALGORITHM WITH GREEDY CONVERGENCE FOR MIMO DETECTION

Qingsong Wen and Xiaoli Ma

School of ECE, Georgia Institute of Technology, Atlanta, GA 30332

ABSTRACT

Effective Lenstra-Lenstra-Lovász (ELLL) algorithm is a common low-complexity lattice reduction (LR) technique adopted in LR-aided successive interference cancellation (SIC) multiple-input multiple-output (MIMO) detectors. However, the original ELLL algorithm is undesirable for hardware implementation since the number and the sequence of ELLL iterations are not deterministic. To address these issues, some fixed-complexity ELLL (fcELLL) algorithms have recently been proposed. In this paper, we propose two new fcELLL algorithms to further improve the efficiency of existing fcELLL algorithms by eliminating unnecessary iterations, such that the fcELLL algorithm exhibits faster convergence and lower computational complexity. Compared to the existing fcELLL algorithms, simulations show that the proposed fcELLL algorithms can save up to $39\% - 48\%$ complexity for 8×8 MIMO systems without performance loss.

***Index Terms*—** Effective LLL, lattice reduction, MIMO detection, successive interference cancellation.

1. INTRODUCTION

Multiple-input multiple-output (MIMO) [1] has become one of the key techniques to enhance the spectral efficiency and data rate in modern wireless systems. However, it is hard to design high performance MIMO detectors with low complexity. It is well-known that maximum likelihood (ML) detector provides optimal error performance, but it exhibits exponential complexity with respect to the number of transmit antennas even implemented with the efficient sphere decoding algorithm [2]. To reduce the complexity, linear detectors and successive interference cancellation (SIC) detectors can be adopted, but these detectors suffer from degraded error performance compared to the ML detector due to diversity loss [3].

Recently, lattice reduction (LR) aided SIC detectors have attracted lots of interests due to the high performance and moderate complexity [4]. Among them, the Lenstra-Lenstra-Lovász (LLL) [5] and effective LLL (ELLL) [6] algorithms have been commonly adopted due to their polynomial complexity on average [7]. One drawback of the LLL/ELLL algorithm is the variable complexity due to non-deterministic iterations, which is not desirable for hardware implementation. To address it, some fixed-complexity LLL/ELLL (fcLLL/fcELLL) algorithms are developed [8, 9, 10], which have similar complexity as the LLL/ELLL algorithms, but with fixed number of iterations as well as the predefined sequence of iterations. Here we focus on the ELLL/fcELLL algorithms, since they have less complexity while maintaining the same bit error rate (BER) performance compared with the LLL/fcLLL counterparts in the LR-aided SIC detectors [10, 11].

In this paper, we propose two novel fcELLL algorithms with greedy convergence which can adopt any predefined sequence of iterations of the existing fcLLL/fcELLL algorithms. First, we investigate the ELLL's termination characteristics. Then, motivated by the investigation, we design two schemes to select the ELLL iterations in the proposed two fcELLL algorithms, respectively, such that the column swap occurs at each iteration for greedy convergence and low complexity. Two types of column swap flag are designed, which are used not only to track the column swap but also to decide the algorithm's termination. Lastly, simulations show that the proposed two fcELLL algorithms exhibit faster convergence and lower complexity than the ELLL and existing fcELLL algorithms.

Notations: $(\cdot)^*$ and $(\cdot)^H$ denote conjugate and Hermitian transpose, respectively. Boldface upper- and lower-case letters indicate matrices and column vectors, respectively. $\boldsymbol{I}_N$ denotes the $N \times N$ identity matrix. $m\!:\!n$ denotes all integers from m to n. $\lfloor a \rceil$ indicates rounding to the nearest integer of a. The real and imaginary parts of a complex number are represented as $\Re[\cdot]$ and $\Im[\cdot]$, respectively.

2. PRELIMINARIES

2.1. System Model and LR-aided MIMO Detection

Consider a common flat-fading spatial multiplexing MIMO system with N_t transmit and N_r receive antennas as

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{s} + \boldsymbol{w}, \tag{1}$$

where $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_{N_t}]$ is an $N_r \times N_t$ channel matrix, $\boldsymbol{s} = [s_1, s_2, \cdots, s_{N_t}]^T$ is the transmitted signal vector from the QAM constellation set $\mathcal{S}$ whose real and imaginary parts are integers from the set $\{-\sqrt{\mathcal{M}}+1, \ldots, -1, 1, \ldots, \sqrt{\mathcal{M}}-1\}$ with $\mathcal{M}$ being the constellation size, $\boldsymbol{w} = [w_1, w_2, \cdots, w_{N_r}]^T$ is the additive white Gaussian noise vector with zero mean and covariance matrix $E[\boldsymbol{w}\boldsymbol{w}^{\mathcal{H}}] = \sigma_w^2 \boldsymbol{I}_{N_r}$, and $\boldsymbol{y} = [y_1, y_2, \cdots, y_{N_r}]^T$ is the received signal vector. For simplicity, we assume that equal power is adopted at each antenna with $E[\boldsymbol{s}\boldsymbol{s}^{\mathcal{H}}] = \sigma_s^2 \boldsymbol{I}_{N_t}$, and $\boldsymbol{H}$ is unknown at the transmitter but known at the receiver.

The complex lattice $\mathcal{L}$ from channel matrix $\boldsymbol{H}$ is defined as

$$\mathcal{L}(\boldsymbol{H}) = \left\{ \boldsymbol{v} \,\middle|\, \boldsymbol{v} = \sum_{i=1}^{N_t} c_i \boldsymbol{h}_i,\ c_i \in \mathbb{Z}_j \right\}, \tag{2}$$

where $\mathbb{Z}_j = \{a + bj | a, b \in \mathbb{Z}\}$ represents the Gaussian integer ring, and the column vectors $\boldsymbol{h}_i$ of the matrix $\boldsymbol{H}$ represent a basis of the lattice. Given a basis $\boldsymbol{H}$, LR algorithms produce a shorter and near-orthogonal basis as $\tilde{\boldsymbol{H}} = \boldsymbol{H}\boldsymbol{T}$, where $\boldsymbol{T}$ is a unimodular matrix consisted of Gaussian integers with determinant ± 1 or $\pm j$.

To incorporate LR into MIMO detection, we rewrite the system model by applying scaling and shifting on $\boldsymbol{s}$ such that $\boldsymbol{H}\boldsymbol{s}$ can be viewed as a lattice point with $\boldsymbol{H}$ as its basis, i.e.,

$$\begin{aligned} \boldsymbol{y}' &= \frac{\boldsymbol{y} + \boldsymbol{H}\boldsymbol{1}(1+j)}{2} = \boldsymbol{H}\boldsymbol{T}\boldsymbol{T}^{-1}\frac{\boldsymbol{s} + \boldsymbol{1}(1+j)}{2} + \frac{1}{2}\boldsymbol{w} \\ &= \tilde{\boldsymbol{H}}\boldsymbol{z} + \frac{1}{2}\boldsymbol{w}, \end{aligned} \tag{3}$$

This material is based on work supported by NSF ECCS-1202286.

Table 1. The ELLL Algorithm

Input: $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{P}$ (after QR/SQRD/MMSE-SQRD: $\boldsymbol{HP} = \boldsymbol{QR}$)		
Output: $\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{R}}, \boldsymbol{T}$		
1:	Initialization: $\tilde{\boldsymbol{Q}} = \boldsymbol{Q}, \tilde{\boldsymbol{R}} = \boldsymbol{R}, \boldsymbol{T} = \boldsymbol{P}, \delta \in (1/2, 1]$	
2:	$k = 2$	
3:	**while** $k \leq N_t$	
4:	$u = \lceil \tilde{R}_{k-1,k} / \tilde{R}_{k-1,k-1} \rfloor$	***effective size reduction***
5:	**if** $u \neq 0$	
6:	$\tilde{\boldsymbol{R}}_{1:k-1,k} = \tilde{\boldsymbol{R}}_{1:k-1,k} - u\tilde{\boldsymbol{R}}_{1:k-1,k-1}$	
7:	$\boldsymbol{T}_{:,k} = \boldsymbol{T}_{:,k} - u\boldsymbol{T}_{:,n}$	
8:	**end**	
9:	**if** $\delta \lvert \tilde{R}_{k-1,k-1} \rvert^2 > \lvert \tilde{R}_{k,k} \rvert^2 + \lvert \tilde{R}_{k-1,k} \rvert^2$	***Lovasz condition***
10:	swap columns $k-1$ and k in $\tilde{\boldsymbol{R}}$ and $\tilde{\boldsymbol{T}}$	***column swap***
11:	$\boldsymbol{\Theta} = \begin{bmatrix} \alpha^* & \beta \\ -\beta & \alpha \end{bmatrix}$ with $\alpha = \frac{\tilde{R}_{k-1,k-1}}{\lvert\lvert \tilde{\boldsymbol{R}}_{k-1:k,k-1} \rvert\rvert}$, $\beta = \frac{\tilde{R}_{k,k-1}}{\lvert\lvert \tilde{\boldsymbol{R}}_{k-1:k,k-1} \rvert\rvert}$	
12:	$\tilde{\boldsymbol{R}}_{k-1:k,k-1:N_t} = \boldsymbol{\Theta} \tilde{\boldsymbol{R}}_{k-1:k,k-1:N_t}$	
13:	$\tilde{\boldsymbol{Q}}_{:,k-1:k} = \tilde{\boldsymbol{Q}}_{:,k-1:k} \boldsymbol{\Theta}^{\mathcal{H}}$	
14:	$k = \max(k-1, 2)$	
15:	**else**	
16:	$k = k + 1$	
17:	**end**	
18:	**end**	

where $\mathbf{1}$ is the $N_t \times 1$ vector of ones, $\tilde{\boldsymbol{H}} = \boldsymbol{HT}$ is the reduced basis, and $\boldsymbol{z}$ is the vector in the lattice-reduced domain. Then, the low-complexity SIC or MMSE-SIC detectors are performed according to (3) to obtain the estimated $\boldsymbol{z}$ as $\hat{\boldsymbol{z}}$. Finally, the estimation of the transmitted symbols $\boldsymbol{s}$ are computed as

$$\hat{\boldsymbol{s}} = \mathcal{Q}\left[2\boldsymbol{T}\hat{\boldsymbol{z}} - \mathbf{1}(1+j)\right], \tag{4}$$

where $\mathcal{Q}(\cdot)$ is the symbol-wise quantizer to the nearest point in the constellation set $\mathcal{S}$.

2.2. Effective LLL and the Fixed-Complexity Variants

The complex ELLL version with QR decomposition as preprocessing part is summarized in Table 1, where the preprocessing part can be sorted QR decomposition (SQRD) or MMSE-SQRD [12] to reduce ELLL's iterations. The ELLL contains a while loop which repeatedly excuses three steps: *1) effective size reduction* at Lines 4-8; *2) Lovász condition evaluation* at Line 9; and *3) column swap* at Lines 10-13 based on the Lovász condition. Since the main complexity comes from effective size reduction and column swap, we refer one-time execution of these two parts as one ELLL iteration for later comparison among ELLL variants. Each ELLL iteration corresponds to a specific value of the control variable $k \in [2, N_t]$ in the while loop. Note that by replacing the effective size reduction with size reduction in ELLL, the LLL algorithm (see Table I in [13]) is obtained with higher complexity than the ELLL algorithm.

When the control variable k reaches $N_t{+}1$, the ELLL algorithm terminates with a shorter and closer to orthogonal basis called ELLL-reduced basis [6] defined as follows.

Definition 1 (ELLL-reduced basis): Let $\tilde{\boldsymbol{H}} = \boldsymbol{HT} = \tilde{\boldsymbol{Q}}\tilde{\boldsymbol{R}}$ be the QR decomposition of the reduced basis from matrix $\boldsymbol{H}$. $\tilde{\boldsymbol{H}}$ is defined as an ELLL-reduced basis if it satisfies

$$\lvert \Re[\tilde{R}_{k-1,k}] \rvert \leq \frac{1}{2} \lvert \tilde{R}_{k-1,k-1} \rvert, \lvert \Im[\tilde{R}_{k-1,k}] \rvert \leq \frac{1}{2} \lvert \tilde{R}_{k-1,k-1} \rvert, \tag{5}$$

$$\delta \lvert \tilde{R}_{k-1,k-1} \rvert^2 \leq \lvert \tilde{R}_{k-1,k} \rvert^2 + \lvert \tilde{R}_{k,k} \rvert^2, \;\; 1/2 < \delta \leq 1, \tag{6}$$

where $2 \leq k \leq N_t$. The (5) is called effective size reduction condition, and the (6) is called Lovász condition where the parameter δ is selected for performance-complexity tradeoff (larger δ results in better performance but with higher computational complexity).

One issue of the ELLL algorithm is that the number of ELLL iterations is not deterministic, even infinite in worst cases [7]. To solve it, the fcELLL algorithm [10] with fixed maximum number of ELLL iterations is developed. Another desirable property associated fcELLL is that the sequence of ELLL iterations (i.e., the k sequence) is also fixed and predefined, which is different from the original ELLL algorithm where the value of k can be decreased or increased depending on Lovász condition (see Lines 9, 14, and 16 in Table 1). These properties make fcELLL easier in hardware implementation.

By applying different types of sequence of ELLL iterations, we can obtain the following fcELLL variants.

- *Sequential fcELLL [10]:* it adopts the sequential sequence [8] which repeats a super-iteration composed of integers from 2 to N_t.
- *Even-odd fcELLL:* it adopts the even-odd sequence [14] which repeats a super-iteration composed of even integers from 2 to N_t followed by odd integers from 3 to N_t.
- *Incremental fcELLL:* it adopts the incremental sequence [9] composed of initial stage followed by repeat stage, where the initial stage comprises alternate reversed even odd sequence with incremental length, and the repeat stage comprises the repetition of the last $N_t - 1$ integers in the initial stage.

3. PROPOSED TWO GREEDY FCELLL ALGORITHMS

3.1. Motivation

The primary motivation of the proposed fcELLL algorithms is to exploit the termination characteristics of the ELLL algorithm. To understand it, we provide the following definition of LLL potential.

Definition 2 (LLL Potential [5, 10]): a positive real number depending on the diagonal elements of $\tilde{\boldsymbol{R}}$ matrix is the LLL potential

$$D \overset{\text{def}}{=} \prod_{i=1}^{N_t-1} d_i = \prod_{i=1}^{N_t-1} \lvert \tilde{R}_{i,i} \rvert^{2(N_t-i)}, \tag{7}$$

where $d_i = \det^2(\mathcal{L}_i) = \prod_{k=1}^{i} \lvert \tilde{R}_{k,k} \rvert^2$, and $\mathcal{L}_i$ is the sub-lattice spanned by column vectors $\tilde{\boldsymbol{q}}_1, \ldots, \tilde{\boldsymbol{q}}_i$ of matrix $\tilde{\boldsymbol{Q}}$ from $\tilde{\boldsymbol{H}} = \tilde{\boldsymbol{Q}}\tilde{\boldsymbol{R}}$.

During the execution of ELLL algorithm, the LLL potential D is monotonically decreasing and there exists a lower bound depending on the lattice $\mathcal{L}(\tilde{\boldsymbol{H}})$ [5]. By checking the three steps of each ELLL iteration as in Table 1, we conclude that both effective size reduction and Lovász condition evaluation do not affect the LLL potential D. The D only decreases when the Lovász condition is not satisfied in an ELLL iteration that leads to the occurrence of column swap. For greedy convergence and low complexity, our aim is to modify the structure of fcELLL algorithm such that column swap always occurs in each ELLL iteration in order to decrease the LLL potential D as rapidly as possible. Similar idea can be found in the greedy diagonal reduction (GDR) algorithm [15], which does not fix the number of iterations.

3.2. Proposed Greedy fcELLL Algorithm I and II

The proposed two fcELLL algorithms are summarised in Tables 2 and 3, respectively, where N_{max} is the predefined maximum number of ELLL iterations, and $kSeq$ is the predefined k sequence for deterministic ELLL iterations. Any k sequence of the existing fcELLL algorithms can be used in the proposed two algorithms. For now,

Table 2. Proposed Greedy fcELLL Algorithm I

Input: $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{P}$ (after QR/SQRD/MMSE-SQRD: $\boldsymbol{HP} = \boldsymbol{QR}$)
Output: $\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{R}}, \boldsymbol{T}$

1:	Initialize: $\tilde{\boldsymbol{Q}} = \boldsymbol{Q},\ \tilde{\boldsymbol{R}} = \boldsymbol{R},\ \boldsymbol{T} = \boldsymbol{P},\ N_{max},\ kSeq$ $possibleCS = ones(1, N_t + 1)$
2:	$n_{iter} = 1, kSeq_{idx} = 1$
3:	**while** $(n_{iter} \le N_{max})$ && $(sum(possibleCS(2{:}N_t)) \neq 0)$
4:	$\quad k = kSeq(kSeq_{idx})$
5:	$\quad$ **if** $possibleCS(k) == 1$
6:	$\quad\quad possibleCS(k) = 0$
7:	$\quad\quad$ **if** $CScheck(k, \delta, \tilde{\boldsymbol{R}})$
8:	$\quad\quad\quad$ Execute ***effective size reduction*** (Lines 4-8 of Table 1)
9:	$\quad\quad\quad$ Execute ***column swap*** (Lines 10-13 of Table 1)
10:	$\quad\quad\quad possibleCS(k-1 : k+1) = [1,\ 1,\ 1]$
11:	$\quad\quad\quad n_{iter} = n_{iter} + 1$
12:	$\quad\quad$ **end**
13:	$\quad$ **end**
14:	$\quad kSeq_{idx} = kSeq_{idx} + 1$
15:	**end**
16:	**function** $flag = CScheck(k, \delta, \tilde{\boldsymbol{R}})$
17:	$\quad u = \lceil \tilde{R}_{k-1,k} / \tilde{R}_{k-1,k-1} \rfloor$
18:	$\quad \tilde{R}'_{k-1,k} = \tilde{R}_{k-1,k} - u\tilde{R}_{k-1,k-1}$
19:	$\quad flag = (\delta|\tilde{R}_{k-1,k-1}|^2 > |\tilde{R}_{k,k}|^2 + |\tilde{R}'_{k-1,k}|^2)$

we adopt the incremental sequence in $kSeq$, since it exhibits better performance than others as shown in [9].

The main idea of the proposed fcELLL algorithm-I is to check whether column swap will occur before each ELLL iteration (Line 7 of Table 2) when following the predefined k sequence, such that only the ELLL iteration with column swap is executed each time. To facilitate it, we adopt an $(N_t + 1)$-bit flag $possibleCS$ similar as that in [9] to track the column swap, which is defined as

$$possibleCS(k) \overset{\text{def}}{=} \begin{cases} 1, & \text{column swap might occur at } k, \\ 0, & \text{column swap will not occur at } k, \end{cases}$$

where bit-2 to bit-N_t of $possibleCS$ correspond to indices k of ELLL iterations from 2 to N_t, while the bit-1 and bit-$(N_t + 1)$ are just nil bits for the simplified operation in Line 10 of Table 2. Each $possibleCS(k)$ is initialized as one before the fcELLL's execution because of no prior at first. If $possibleCS(k)$ is one, then column swap is checked (Lines 7, and 16-19 of Table 2) where the Lovász condition is evaluated (Line 19 of Table 2) to check if there is column swap; if $possibleCS(k)$ is zero, the algorithm proceeds to the next ELLL iteration. Once an ELLL iteration with index k is performed, the column swap of k and $k \pm 1$ would be affected as shown in [9], and thus the corresponding three bits of $possibleCS$ are updated as ones (Line 10 of Table 2). Note that different from ELLL, the proposed fcELLL evaluates Lovász condition before the effective size reduction (Lines 7 and 8 of Table 2). Therefore, $\tilde{R}_{k-1,k}$ is size reduced as $\tilde{R}'_{k-1,k}$ for Lovász condition evaluation when checking column swap (Lines 18-19 of Table 2). The proposed fcELLL algorithm-I terminates either the maximum number of iterations N_{max} is achieved or $possibleCS(k)$ is zero for k from 2 to N_t.

The main idea of the proposed fcELLL algorithm-II is to record whether column swap will occur for each k before ELLL iterations (Lines 2-4 of Table 3), and update the record at the end of each ELLL iteration (Lines 14-16 of Table 3). We design an N_t-bit flag $CSflag$ similar as the aforementioned $possibleCS$ to record and track the column swap. Since whether column swap of each k occurs or not is known from $CSflag$, we just select the ELLL iteration

Table 3. Proposed Greedy fcELLL Algorithm II

Input: $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{P}$ (after QR/SQRD/MMSE-SQRD: $\boldsymbol{HP} = \boldsymbol{QR}$)
Output: $\tilde{\boldsymbol{Q}}, \tilde{\boldsymbol{R}}, \boldsymbol{T}$

1:	Initialize: $\tilde{\boldsymbol{Q}} = \boldsymbol{Q},\ \tilde{\boldsymbol{R}} = \boldsymbol{R},\ \boldsymbol{T} = \boldsymbol{P},\ N_{max},\ kSeq$ $CSflag = ones(1, N_t)$
2:	**for** $i = 2 : N_t$
3:	$\quad CSflag(i) = CScheck(i, \delta, \tilde{\boldsymbol{R}})$
4:	**end**
5:	$n_{iter} = 1,\ kSeq_{idx} = 1$
6:	**while** $(n_{iter} \le N_{max})$ && $(sum(CSflag(2{:}N_t)) \neq 0)$
7:	$\quad k = kSeq(kSeq_{idx})$
8:	$\quad$ **while** $CSflag(k) \neq 1$
9:	$\quad\quad kSeq_{idx} = kSeq_{idx} + 1$
10:	$\quad\quad k = kSeq(kSeq_{idx})$
11:	$\quad$ **end**
12:	$\quad$ Execute ***effective size reduction*** (Lines 4-8 of Table 1)
13:	$\quad$ Execute ***column swap*** (Lines 10-13 of Table 1)
14:	$\quad CSflag(k) = CScheck(k, \delta, \tilde{\boldsymbol{R}})$
15:	$\quad$ **if** $(k > 2)\ CSflag(k-1) = CScheck(k-1, \delta, \tilde{\boldsymbol{R}})$ **end**
16:	$\quad$ **if** $(k < N_t)\ CSflag(k+1) = CScheck(k+1, \delta, \tilde{\boldsymbol{R}})$ **end**
17:	$\quad n_{iter} = n_{iter} + 1,\ kSeq_{idx} = kSeq_{idx} + 1$
18:	**end**

with column swap when following the predefined k sequence (Lines 7-11 of Table 3). The same as the the proposed fcELLL algorithm-I, the ELLL iteration with index k affects the column swap of k and $k \pm 1$, and thus the corresponding three bits of $CSflag$ need to be updated (Lines 14-16 of Table 3). Since updating $CSflag$ involves the evaluation of Lovász condition, we only update $CSflag(k-1)$ if $K > 2$ and $CSflag(k+1)$ if $K < N_t$ to save complexity, which is different from $possibleCS$ (see Line 10 of Table 2 versus Lines 14-16 of Table 3). Note that $CSflag(k)$ is one indicates that column swap definitely occurs at ELLL iteration with index k instead of "might" occur as that in the $possibleCS(k)$. The proposed fcELLL algorithm-II terminates either the maximum number of iterations N_{max} is achieved or $CSflag(k)$ is zero for k from 2 to N_t.

Note that we can set $N_{max} = \infty$ in the proposed two fcELLL algorithms to obtain the versions without fixed-complexity. Then, either proposed fcELLL gets best-achievable performance with an ELLL-reduced basis after termination. Also note that the proposed fcELLL algorithms can be extended to fcLLL versions like [8, 9] by replacing the effective size reduction with size reduction or adding full size reduction at the end of the proposed fcELLL algorithms.

4. NUMERICAL RESULTS AND DISCUSSION

We compare BER, convergence, and complexity of the ELLL and different fcELLL variants (Sequential fcELLL [10], Even-odd fcELLL, Incremental fcELLL, and the proposed two fcELLLs) in LR-aided MMSE-SIC detectors. Note that the LLL and fcLLL variants (Sequential fcLLL [8], Even-odd fcELLL [14, 9], Incremental fcELLL [9], and the fcLLL versions of proposed two fcELLLs) are not considered, since they have the same BER but higher complexity than the corresponding ELLL and fcELLL counterparts in LR-aided MMSE-SIC detectors [10, 11]. In all LR algorithms, the MMSE-SQRD is selected as the preprocessing part since it reduces the number of iterations [12], and the parameter $\delta = 3/4$ is adopted for performance-complexity tradeoff [5]. The MIMO channels are flat Rayleigh fading whose entries are i.i.d. complex Gaussian variables with zero mean and unit variance. The BER is evaluated versus energy per bit to noise density defined as $E_b/N_0 = N_r\sigma_s^2/(\sigma_w^2 log_2\mathcal{M})$.

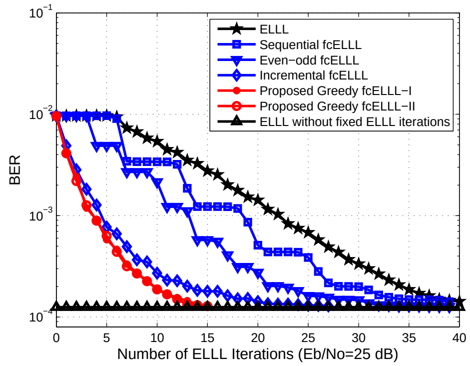

Fig. 1. BER versus number of ELLL iterations of different ELLL variants in LR-aided MMSE-SIC detectors for an 8×8 MIMO system using 64-QAM with $E_b/N_0 = 25$ dB.

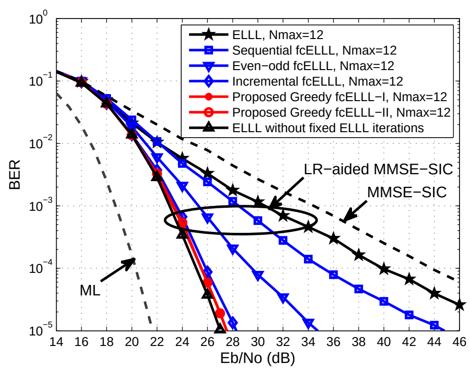

Fig. 2. BER versus E_b/N_0 of different ELLL variants in LR-aided MMSE-SIC detectors for an 8×8 MIMO system using 64-QAM.

4.1. Performance of ELLL Variants with Fixed Complexity

Firstly, we consider the fixed-complexity cases, i.e., the maximum number of the ELLL iterations is fixed in each LR algorithm.

Fig. 1 shows the uncoded BER performance of different LR algorithms versus the number of ELLL iterations in LR-aided MMSE-SIC detectors for an 8×8 MIMO system using 64-QAM. The best BER from ELLL algorithm without fixed number of iterations is also provided as a performance bound. It can be seen that all fcELLL algorithms converge faster than the ELLL algorithm, and the proposed two fcELLL algorithms converge fastest among all LR algorithms. The proposed two algorithms only need around 12 ELLL iterations to achieve near the best BER performance bound.

Fig. 2 depicts BER results versus E_b/N_0 when the maximum number of ELLL iterations is selected as 12. Compared to the Incremental fcELLL, the Even-odd fcELLL, and the Sequential fcELLL algorithms, the BER gains of the proposed two fcELLL algorithms at BER=10^{-5} are 0.8 dB, 7.2 dB, and 17.3 dB, respectively.

4.2. Performance of ELLL Variants without Fixed Complexity

Secondly, we consider the cases without fixed complexity, i.e., no limitation for the number of ELLL iterations. Besides the aforementioned LR algorithms, we also consider the GDR algorithm [15] (another ELLL variant without fixed complexity) whose column swap also occurs at each ELLL iteration as our proposed two algorithms. Note that each LR here has the same BER results since each one generates an ELLL-reduced basis after termination.

Fig. 3 depicts the complementary cumulative distribution function (CCDF) of the ELLL iterations of different LR algorithms in an 8×8 MIMO systems, where the noise used in MMSE-SQRD is randomly generated so that the SNR is uniformly distributed from 0 dB to 40 dB. It can be seen that the proposed two fcELLL and the GDR achieve the best performance among all the LR algorithms. Note that the proposed two fcELLL algorithms also enjoy the predefined deterministic sequence of ELLL iterations, while the sequence of ELLL iterations in GDR is not deterministic.

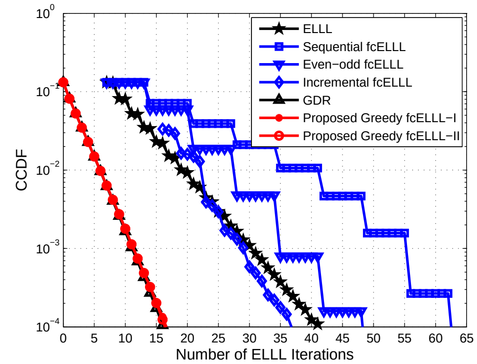

Fig. 3. CCDFs of ELLL iterations of different ELLL variants without fixed ELLL iterations in an 8×8 MIMO system.

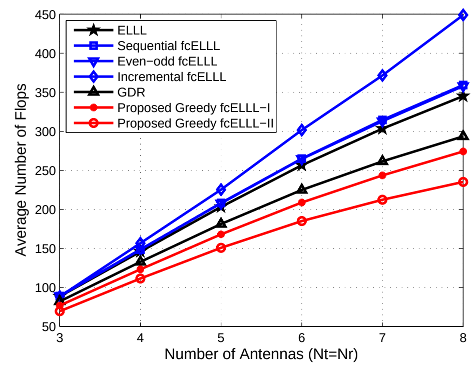

Fig. 4. Complexity comparisons of different ELLL variants without fixed ELLL iterations from 3×3 to 8×8 MIMO systems.

To approximately evaluate the computational complexity, the average floating-point operations (flops) are simulated, where the flops of different arithmetic operation are counted as: 1 for a real operation (i.e., addition, subtraction, multiplication, division, comparison, squaxre root, and absolute value), 2 for rounding a complex number, 6 for a complex multiplication, and 2 for a complex number multiplied or divided by a real number. Fig. 4 shows the average flops of different LR algorithms from 3×3 to 8×8 MIMO systems, where the noise in the MMSE-SQRD is the same as that in Fig. 3. It can be seen that the proposed two fcELLL algorithms exhibit lower complexity than other fcELLL algorithms as well as the ELLL and GDR algorithm. Compared to other LR algorithms, the proposed fcELLL algorithm I and algorithm II save around 7%-39% and 20%-48% complexity in the 8×8 MIMO systems, respectively.

5. CONCLUSION

In this paper, we propose two novel fcELLL algorithms with greedy convergence. The idea is to exploit the termination characteristics of the ELLL algorithm, such that column swap occurs at each iteration to speed up termination process. Compared to ELLL and existing fcELLL algorithms, the proposed two fcELLLs have quicker convergence and lower complexity without performance loss.

6. REFERENCES

[1] J. Mietzner, R. Schober, L. Lampe, W. H. Gerstacker, and P. A. Hoeher, “Multiple-antenna techniques for wireless communications - a comprehensive literature survey,” *IEEE Commu. Surveys & Tutorials*, vol. 11, no. 2, pp. 87–105, Second Quarter 2009.

[2] J. Jaldén and B. Ottersten, “On the complexity of sphere decoding in digital communications,” *IEEE Trans. Signal Process.*, vol. 53, no. 4, pp. 1474–1484, Apr. 2005.

[3] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[4] D. Wübben, D. Seethaler, J. Jaldén, and G. Matz, “Lattice reduction,” *IEEE Signal Process. Mag.*, vol. 28, no. 3, pp. 70–91, May 2011.

[5] A. K. Lenstra, H. W. Lenstra, and L. Lovász, “Factoring polynomials with rational coefficients,” *Math. Annalen*, vol. 261, no. 4, pp. 515–534, 1982.

[6] N. Howgrave-Graham, “Finding small roots of univariate modular equations revisited,” in *Proc. the 6th IMA Int. conf. on Crypt. and Coding (IMACC)*, Cirencester, UK, Dec. 1997, pp. 131–142.

[7] J. Jaldén, D. Seethaler, and G. Matz, “Worst-and average-case complexity of LLL lattice reduction in MIMO wireless systems,” in *Proc. IEEE Int. Conf. Acoust., Speech and Signal Process.(ICASSP)*, Las Vegas, NV, Mar. 2008, pp. 2685–2688.

[8] H. Vetter, V. Ponnampalam, M. Sandell, and P. A. Hoeher, “Fixed complexity LLL algorithm,” *IEEE Trans. Signal Process.*, vol. 57, no. 4, pp. 1634–1637, Apr. 2009.

[9] Q. Wen and X. Ma, “An Enhanced Fixed-Complexity LLL Algorithm for MIMO Detection,” in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Austin, TX, Dec. 2014, pp. 3231–3236.

[10] C. Ling, W. H. Mow, and N. Howgrave-Graham, “Reduced and Fixed-Complexity Variants of the LLL Algorithm for Communications,” *IEEE Trans. Commun.*, vol. 61, no. 3, pp. 1040–1050, Mar. 2013.

[11] B. Gestner, W. Zhang, X. Ma, and D. Anderson, “Lattice reduction for MIMO detection: from theoretical analysis to hardware realization,” *IEEE Trans. Circuits Syst. I*, vol. 58, no. 4, pp. 813–826, Apr. 2011.

[12] D. Wübben, R. Böhnke, V. Kühn, and K. D. Kammeyer, “Near-maximum-likelihood detection of MIMO systems using MMSE-based lattice reduction,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 2, Paris, France, Jun. 2004, pp. 798–802.

[13] X. Ma and W. Zhang, “Performance analysis for MIMO systems with lattice-reduction aided linear equalization,” *IEEE Trans. Commun.*, vol. 56, no. 2, pp. 309–318, Feb. 2008.

[14] G. Villard, “Parallel lattice basis reduction,” in *Proc. ACM Int. Symp. on Symbolic and Algebraic Computation (ISSAC)*, Berkeley, CA, Jul. 1992, pp. 269–277.

[15] W. Zhang, S. Qiao, and Y. Wei, “A diagonal lattice reduction algorithm for MIMO detection,” *IEEE Signal Process. Lett.*, vol. 19, no. 5, pp. 311–314, May 2012.