An Efficient Optimal Algorithm for Integer-Forcing Linear MIMO Receivers Design

Jinming Wen*, Lanping Li†, Xiaohu Tang†, Wai Ho Mow‡, and Chintha Tellambura*

*Department of Electrical and Computer Engineering, University of Alberta,
Edmonton T6G 2V4, Canada (e-mail: jinming1@ualberta.ca, chintha@ece.ualberta.ca)
†Information Security and National Computing Grid Laboratory, Southwest Jiaotong University,
Chengdu 610031, China, (e-mail: lanping@my.swjtu.edu.cn, xhutang@swjtu.edu.cn)
‡Department of Electronic and Computer Engineering, HKUST, Hong Kong (e-mail: eewhmow@ust.hk)

***Abstract*—The integer-forcing (IF) linear multiple-input and multiple-output (MIMO) receiver is a recently proposed suboptimal receiver which nearly reaches the performance of the optimal maximum likelihood receiver for the entire signal-to-noise ratio (SNR) range and achieves the optimal diversity multiplexing tradeoff for the standard MIMO channel with no coding across transmit antennas in the high SNR regime. The optimal integer coefficient matrix $\boldsymbol{A}^{\star} \in \mathbb{Z}^{N_t \times N_t}$ for IF maximizes the total achievable rate, where N_t is the column dimension of the channel matrix. To obtain $\boldsymbol{A}^{\star}$, a successive minima problem (SMP) on an N_t-dimensional lattice that is suspected to be NP-hard needs to be solved. In this paper, an efficient exact algorithm for the SMP is proposed. For efficiency, our algorithm first uses the LLL reduction to reduce the SMP. Then, different from existing SMP algorithms which form the transformed $\boldsymbol{A}^{\star}$ column by column in N_t iterations, it first initializes with a suboptimal matrix which is the $N_t \times N_t$ identity matrix with certain column permutations that guarantee this suboptimal matrix is a good initial solution of the reduced SMP. The suboptimal matrix is then updated, by utilizing the integer vectors obtained by employing an improved Schnorr-Euchner search algorithm to search the candidate integer vectors within a certain hyper-ellipsoid, via a novel and efficient algorithm until the transformed $\boldsymbol{A}^{\star}$ is obtained in only one iteration. Finally, the algorithm returns the matrix obtained by left multiplying the solution of the reduced SMP with the unimodular matrix that is generated by the LLL reduction. Simulation results show the optimality of our novel algorithm and indicates that the new one is much more efficient than existing optimal algorithms.**

***Index Terms*—Integer-forcing linear receiver, sphere decoding, successive minima problem, achievable rate.**

I. Introduction

Due to the exponential growth of mobile traffic and subscribers globally, current and future wireless systems require ongoing improvements in capacity, quality and coverage. Multiple-input multiple-output (MIMO) technology uses multiple antenna arrays both in transmitters and receivers and thus exploits the space dimension to improve wireless capacity and reliability. However, to actually realize these gains, optimal or near optimal receiver designs are critical. The maximum likelihood (ML) receiver is optimal and achieves the highest data rate and smallest error probability. However, its complexity is exponential in the number of antennas. Consequently, zero-forcing (ZF), successive interference cancellation (SIC) and minimum mean square error (MMSE) receivers have been developed, which achieve low-complexity, albeit with a performance loss. Although lattice reductions (such as LLL reduction [1]) usually improve their performance (see, e.g., [2], [3], [4]), performance losses can be significant, especially in the low signal-to-noise ratio (SNR) regime.

To overcome these issues, a new linear receiver structure, called integer-forcing (IF), has been proposed by Zhan *et al.* [5] [6]. It exploits the fact that any integer linear combination of lattice points is still a lattice point. Based on this insight, it decodes integer combinations of transmit data, which are digitally solved for the original data. Since the IF linear receiver can equalize the channel to any full-rank integer matrix $\boldsymbol{A}$, it can be optimized over the choice of $\boldsymbol{A}$ to minimize the noise amplification. It has been shown that the linear IF receiver nearly reaches the performance of the optimal joint ML receiver for the entire SNR range.

Wei *et al.* [7] developed an algorithm for $\boldsymbol{A}^{\star}$, which is the optimal $\boldsymbol{A}$ that maximizes the achievable rate, by solving a successive minima problem (SMP). Since this algorithm does not preprocess with a lattice reduction, the choice of initial radius, which is used to create a set Ω of candidate vectors for $\boldsymbol{A}^{\star}$, is sub-optimal, thus this algorithm is slow. Recently, another exact algorithm for $\boldsymbol{A}^{\star}$ by solving a SMP was proposed in [8]. Although this algorithm is generally more efficient than the algorithm developed in [7], a highly efficient faster algorithm is still desirable.

In this paper, we will design an novel efficient exact algorithm for the SMP. For efficiency, the algorithm first uses the LLL reduction [1] to reduce the SMP. Then, different from the algorithms in [9] and [8] which form the transformed $\boldsymbol{A}^{\star} \in \mathbb{Z}^{N_t \times N_t}$ column by column in N_t iterations, where N_t is the column dimension of the channel matrix, our algorithm initializes with a suboptimal matrix which is the $N_t \times N_t$ identity matrix with certain column permutations that ensure the suboptimal matrix is a good initial solution of the reduced

The work is partially supported by the National High Technology Research and Development Program of China (863 Program) under Grant 2015AA01A710, the University Grants Committee of Hong Kong under Project AoE/E-02/08.

SMP. The suboptimal matrix is then updated by a novel algorithm which uses an improved Schnorr-Euchner search algorithm [10] to search for candidates of the columns of $\boldsymbol{A}^\star$ and uses a novel and efficient algorithm to update the suboptimal matrix until the transformed $\boldsymbol{A}^\star$ is obtained. Finally, the algorithm returns the matrix obtained by left multiplying the solution of the reduced SMP with the unimodular matrix that is generated with the LLL reduction.

The rest of the paper is organized as follows. In Section II, we introduce the coefficient matrix design problem for IF receivers. We propose our new optimal algorithm in Section III. A comparative performance evaluation of the proposed and existing algorithms is proposed in Section IV. Finally, conclusions are given in Section V.

Notation. Let $\boldsymbol{a}_i$ be the i-th column of matrix $\boldsymbol{A}$ and $\boldsymbol{A}_{[1,i]}$ be the submatrix of $\boldsymbol{A}$ formed by its first i columns. Let $\lfloor \boldsymbol{x} \rceil$ denote the nearest integer vector of $\boldsymbol{x}$, i.e., each entry of $\boldsymbol{x}$ is rounded to its nearest integer (if there is a tie, the one with smaller magnitude is chosen).

II. Problem Statement

In this paper, we consider a slow fading channel model where the channel remains unchanged over the entire block length. Since a complex MIMO system can be readily transformed to an equivalent real system, without loss of generality, we consider the real-valued channel model only.

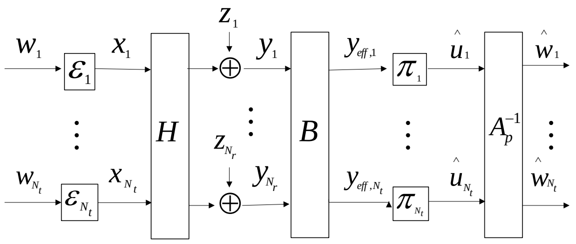

Fig. 1. The block diagram of an IF MIMO system

In the IF MIMO system (see Figure 1), the m-th transmitter antenna is equipped with a lattice encoder ε_m, which maps the length-k message $\boldsymbol{w}_m$ into a length-n lattice codeword $\boldsymbol{x}_m \in \mathbb{R}^n$, i.e.,

$$\varepsilon_m : \mathbb{F}_p^k \to \mathbb{R}^n, \quad \boldsymbol{w}_m \to \boldsymbol{x}_m,$$

where the entries of $\boldsymbol{w}_m$ are independent and uniformly distributed over a prime-size finite field $\mathbb{F}_p = \{0, 1, \cdots, p-1\}$, i.e.,

$$\boldsymbol{w}_m \in \mathbb{F}_p^k, \quad m = 1, 2, \cdots, N_t.$$

All transmit antennas employ the same lattice code. Each codeword satisfies the power constraint:

$$\frac{1}{n} \|\boldsymbol{x}_m\|^2 \leq P.$$

Let $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n] \in \mathbb{R}^{N_t \times n}$, then the received signal $\boldsymbol{Y} \in \mathbb{R}^{N_r \times n}$ is given by

$$\boldsymbol{Y} = \boldsymbol{H}\boldsymbol{X} + \boldsymbol{Z}$$

where $\boldsymbol{H} \in \mathbb{R}^{N_r \times N_t}$ is the channel matrix, and $\boldsymbol{Z} \in \mathbb{R}^{N_r \times n}$ is the noise matrix. All the elements of both $\boldsymbol{H}$ and $\boldsymbol{Z}$ independent and identically follow the standard Gaussian distribution $\mathcal{N}(0, 1)$.

The receiver aims to find a coefficient matrix $\boldsymbol{A} \in \mathbb{Z}^{N_t \times N_t}$ and a filter matrix $\boldsymbol{B} \in \mathbb{R}^{N_t \times N_r}$. With the matrix $\boldsymbol{B}$, the received message $\boldsymbol{Y}$ will be projected to the more effective received vector for further decoding. The m-th filter outputs

$$\boldsymbol{y}_{\text{eff},m} = \boldsymbol{b}_m^T \boldsymbol{Y} = \boldsymbol{a}_m^T \boldsymbol{X} + (\boldsymbol{b}_m^T \boldsymbol{H} - \boldsymbol{a}_m^T)\boldsymbol{X} + \boldsymbol{b}_m^T \boldsymbol{Z},$$

where

$$\boldsymbol{z}_{\text{eff},m} = (\boldsymbol{b}_m^T \boldsymbol{H} - \boldsymbol{a}_m^T)\boldsymbol{X} + \boldsymbol{b}_m^T \boldsymbol{Z}$$

is the effective noise, $\boldsymbol{a}_m^T$ and $\boldsymbol{b}_m^T$ respectively denote the m-th rows of $\boldsymbol{A}$ and $\boldsymbol{B}$.

The receiver recovers the original N_t messages $\boldsymbol{W} = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_{N_t}]^T$ by decoding $\boldsymbol{Y} = [\boldsymbol{y}_{eff,1}, \cdots, \boldsymbol{y}_{eff,N_t}]^T$ in parallel based on the algebraic structure of lattice codes, i.e., the integer combination of lattice codewords is still a codeword. In this way, we first recover the linear equation $\boldsymbol{u}_m = \left[\boldsymbol{a}_m^T \boldsymbol{W}\right] \mod p$ from the $\boldsymbol{y}_{eff,m}$ at one decoder, i.e.,

$$\pi_m : \mathbb{R}^n \to \mathbb{F}_p^k, \quad \boldsymbol{y}_{\text{eff},m} \to \hat{\boldsymbol{u}}_m.$$

The original messages can be recovered free of error as long as all the lattice equations are correctly detected, i.e.,

$$[\hat{\boldsymbol{w}}_1, \cdots, \hat{\boldsymbol{w}}_{N_t}]^T = \boldsymbol{A}_p^{-1} [\hat{\boldsymbol{u}}_1, \cdots, \hat{\boldsymbol{u}}_{N_t}]^T,$$

where $\boldsymbol{A}_p = [\boldsymbol{A}] \mod p$ is full-rank over $\mathbb{Z}_p$.

Hence, the design of IF receiver is the construction of a full rank IF matrix $\boldsymbol{A} \in \mathbb{Z}^{N_t \times N_t}$ such that the achievable rate is maximized. For more details, see [6] and [11].

By [6], at the m-th decoder π_m, the achievable rate is,

$$\boldsymbol{R}_m = \frac{1}{2} \log^+ \left(\frac{P}{P\|\boldsymbol{H}^T \boldsymbol{b}_m - \boldsymbol{a}_m\|_2^2 + \|\boldsymbol{b}_m\|_2^2} \right),$$

where $\log^+(x) \triangleq \max(\log(x), 0)$. Moreover,

$$\boldsymbol{b}_m^T = \boldsymbol{a}_m^T \boldsymbol{H}^T (\boldsymbol{H}\boldsymbol{H}^T + \boldsymbol{I}/P)^{-1}$$

and the achievable rate is

$$\boldsymbol{R}_m = \frac{1}{2} \log^+ \left(\frac{1}{\boldsymbol{a}_m^T \boldsymbol{G}\, \boldsymbol{a}_m} \right),$$

where

$$\boldsymbol{G} = \boldsymbol{I} - \boldsymbol{H}^T (\boldsymbol{H}\boldsymbol{H}^T + \boldsymbol{I}/P)^{-1} \boldsymbol{H}. \tag{1}$$

Furthermore, the total achievable rate is

$$\boldsymbol{R}_{total} = N_t \min \{\boldsymbol{R}_1, \boldsymbol{R}_2, \cdots, \boldsymbol{R}_{N_t}\}. \tag{2}$$

In this paper, we want to find $\boldsymbol{A}^\star$ to maximize the total achievable rate. Equivalently, we need to solve the following optimization problem,

$$\boldsymbol{A}^\star = \arg \min_{\substack{\boldsymbol{A} \in \mathbb{Z}^{N_t \times N_t} \\ \det(\boldsymbol{A}) \neq 0}} \max_{1 \leq m \leq N_t} \boldsymbol{a}_m^T \boldsymbol{G}\, \boldsymbol{a}_m, \tag{3}$$

where $\boldsymbol{G}$ is defined in (1).

III. A NOVEL OPTIMAL ALGORITHM FOR (3)

In this section, we propose a novel algorithm for (3).

A. Preliminaries of lattices

Let $\boldsymbol{G}$ in (1) have the following Cholesky factorization:

$$\boldsymbol{G} = \boldsymbol{R}^T \boldsymbol{R}, \tag{4}$$

where $\boldsymbol{R} \in \mathbb{R}^{N_t \times N_t}$ is an upper triangular matrix. Then, (3) can be transformed to:

$$\boldsymbol{A}^\star = \arg \min_{\substack{\boldsymbol{A} \in \mathbb{Z}^{N_t \times N_t} \\ \det(\boldsymbol{A}) \neq 0}} \max_{1 \leq m \leq N_t} \|\boldsymbol{R}\,\boldsymbol{a}_m\|_2. \tag{5}$$

To solve (5), it is equivalent to find a nonsingular matrix $\boldsymbol{A}^\star = [\boldsymbol{a}_1^\star, \ldots, \boldsymbol{a}_{N_t}^\star] \in \mathbb{Z}^{N_t \times N_t}$ such that $\max\limits_{1 \leq m \leq N_t} \|\boldsymbol{R}\,\boldsymbol{a}_m^\star\|_2$ is as small as possible. In lattice theory, this value is called the N_t-th successive minimum of lattice $\mathcal{L}(\boldsymbol{R}) = \{\boldsymbol{R}\boldsymbol{a} | \boldsymbol{a} \in \mathbb{Z}^{N_t}\}$. More generally, the k-th ($1 \leq k \leq N_t$) successive minimum λ_k of $\mathcal{L}(\boldsymbol{R})$ is the smallest r such that the closed N_t-dimensional ball $\mathbb{B}(\boldsymbol{0}, r)$ of radius r centered at the origin contains k linearly independent lattice vectors.

Thus, to solve (5), it suffices to solve a SMP which is defined as:

Definition 1. *SMP: finding an invertible matrix* $\boldsymbol{A}^\star = [\boldsymbol{a}_1^\star, \ldots, \boldsymbol{a}_{N_t}^\star] \in \mathbb{Z}^{N_t \times N_t}$ *such that*

$$\|\boldsymbol{R}\boldsymbol{a}_i^\star\|_2 = \lambda_i, \quad i = 1, 2, \ldots, N_t.$$

For efficiency, we use the LLL reduction [1] to preprocess the SMP. Concretely, the LLL reduction reduces $\boldsymbol{R}$ in (4) to $\bar{\boldsymbol{R}}$ through

$$\bar{\boldsymbol{Q}}^T \boldsymbol{R}\boldsymbol{Z} = \bar{\boldsymbol{R}}, \tag{6}$$

where $\bar{\boldsymbol{Q}} \in \mathbb{R}^{N_t \times N_t}$ is orthogonal, $\boldsymbol{Z} \in \mathbb{Z}^{N_t \times N_t}$ is unimodular (i.e., $\boldsymbol{Z}$ also satisfies $|\det(\boldsymbol{Z})| = 1$), and $\bar{\boldsymbol{R}} \in \mathbb{R}^{N_t \times N_t}$ is upper triangular which satisfies

$$\begin{aligned} &|\bar{r}_{ik}| \leq \frac{1}{2}|\bar{r}_{ii}|, \quad i = 1, 2, \ldots, k-1, \\ &\delta\, \bar{r}_{k-1,k-1}^2 \leq \bar{r}_{k-1,k}^2 + \bar{r}_{kk}^2, \quad k = 2, 3, \ldots, N_t, \end{aligned}$$

where δ is a constant satisfying $1/4 < \delta \leq 1$. The matrix $\bar{\boldsymbol{R}}$ is said to be LLL reduced.

After using the LLL reduction (6), the SMP can be transformed to the following reduced SMP (RSMP):

Definition 2. *RSMP: finding an invertible integer matrix* $\boldsymbol{C}^\star = [\boldsymbol{c}_1^\star, \ldots, \boldsymbol{c}_n^\star] \in \mathbb{Z}^{N_t \times N_t}$ *such that*

$$\|\bar{\boldsymbol{R}}\boldsymbol{c}_i^\star\| = \lambda_i, \quad 1 \leq i \leq N_t.$$

Clearly, $\boldsymbol{A}^\star$ and $\boldsymbol{C}^\star$ satisfy $\boldsymbol{A}^\star = \boldsymbol{Z}\boldsymbol{C}^\star$.

In the following, we will focus on how to solve the RSMP to obtain $\boldsymbol{C}^\star$.

B. Preliminaries of the novel algorithm

Since our new algorithm starts with a suboptimal matrix $\boldsymbol{C}$ and updates it during the process of a sphere decoding, an efficient algorithm to update $\boldsymbol{C}$ are provided in this subsection. We begin with introducing the following theorem.

Theorem 1. *Let* $\boldsymbol{C} \in \mathbb{R}^{n \times n}$ *be an arbitrary invertible matrix and* $\boldsymbol{c} \in \mathbb{R}^n$ *be an arbitrary nonzero vector such that* $\widetilde{\boldsymbol{C}}_{[1,i+1]}$ *is full column rank for some* i *with* $0 \leq i \leq n-1$, *where*

$$\widetilde{\boldsymbol{C}} = \begin{bmatrix} \boldsymbol{c}_1 & \ldots & \boldsymbol{c}_i & \boldsymbol{c} & \boldsymbol{c}_{i+1} & \ldots & \boldsymbol{c}_n \end{bmatrix}. \tag{7}$$

Then there exists at least one j *with* $i+2 \leq j \leq n+1$ *such that* $\widetilde{\boldsymbol{C}}_{[\backslash j]}$ *is also invertible, where* $\widetilde{\boldsymbol{C}}_{[\backslash j]}$ *is the matrix obtained by removing* $\widetilde{\boldsymbol{c}}_j$ *from* $\widetilde{\boldsymbol{C}}$.

Due to the limitation of space, the proof of Theorem 1 is omitted. Interesting readers are referred to [12].

As will be seen in the next subsection, to efficiently solve the RSMP, we need to develop a fast algorithm for the following problem: for any given nonsingular matrix $\boldsymbol{C} \in \mathbb{R}^{n \times n}$ and nonzero vector $\boldsymbol{c} \in \mathbb{R}^n$ that satisfy

$$\|\boldsymbol{c}_1\|_2 \leq \|\boldsymbol{c}_2\|_2 \leq \ldots \leq \|\boldsymbol{c}_n\|_2 \text{ and } \|\boldsymbol{c}\|_2 < \|\boldsymbol{c}_n\|_2, \tag{8}$$

we need to get an invertible matrix $\bar{\boldsymbol{C}} \in \mathbb{R}^{n \times n}$, whose columns are chosen from $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n$ and $\boldsymbol{c}$, such that $\|\bar{\boldsymbol{c}}_i\|_2$ are as small as possible for all $1 \leq i \leq n$ and

$$\|\bar{\boldsymbol{c}}_1\|_2 \leq \|\bar{\boldsymbol{c}}_2\|_2 \leq \ldots \leq \|\bar{\boldsymbol{c}}_n\|_2. \tag{9}$$

We first show the problem is well-defined.

By (8), one can see that there exists i with $0 \leq i \leq n-1$ such that $\|\boldsymbol{c}_i\|_2 \leq \|\boldsymbol{c}\|_2 < \|\boldsymbol{c}_{i+1}\|_2$ (note that if $i = 0$, it means $\|\boldsymbol{c}\|_2 < \|\boldsymbol{c}_1\|_2$). Then, one can see that

$$\|\widetilde{\boldsymbol{c}}_1\|_2 \leq \|\widetilde{\boldsymbol{c}}_2\|_2 \leq \ldots \leq \|\widetilde{\boldsymbol{c}}_{n+1}\|_2, \tag{10}$$

where $\widetilde{\boldsymbol{C}}$ is defined in (7). Hence, the question is equivalent to find the largest j with $1 \leq j \leq n+1$ such that $\widetilde{\boldsymbol{C}}_{[\backslash j]}$ is invertible. Specifically, after finding j, set $\bar{\boldsymbol{C}} = \widetilde{\boldsymbol{C}}_{[\backslash j]}$, then $\|\bar{\boldsymbol{c}}_i\|_2$ are as small as possible for all $1 \leq i \leq n$. Moreover, by (10), (9) holds.

If $\widetilde{\boldsymbol{C}}_{[1,i+1]}$ is not full column rank, then $j = i+1$, i.e., $\boldsymbol{c}$ should be removed from $\widetilde{\boldsymbol{C}}$, and the resulting matrix is $\boldsymbol{C}$ which is invertible by assumption. If $\widetilde{\boldsymbol{C}}_{[1,i+1]}$ is full column rank, then by Theorem 1, there exists at least one j with $i+2 \leq j \leq n+1$ such that $\widetilde{\boldsymbol{C}}_{[\backslash j]}$ is invertible. Thus, j exists no matter whether $\widetilde{\boldsymbol{C}}_{[1,i+1]}$ is full column rank or not. Hence, the aforementioned problem is well-defined.

By the above analysis, a natural method to find the desire j is to check whether $\widetilde{\boldsymbol{C}}_{[\backslash j]}$ is invertible for $j = n+1, n, \ldots, i+1$ until finding an invertible matrix. Clearly, this approach works, but the main drawback of this method is its worst complexity is $\mathcal{O}(n^4)$ flops which is too high.

In the following, we introduce a method which can find j in $\mathcal{O}(n^3)$ flops. By the above analysis, $\bar{\boldsymbol{C}} = \widetilde{\boldsymbol{C}}_{[\backslash(i+1)]}$ if $\widetilde{\boldsymbol{C}}_{[1,i+1]}$ is not full column rank. Thus, in the sequel, we only consider the case that $\widetilde{\boldsymbol{C}}_{[1,i+1]}$ is full column rank. We begin with introducing the following theorem.

Theorem 2. *Let $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ be an arbitrary invertible matrix and $\boldsymbol{c} \in \mathbb{R}^n$ be an arbitrary nonzero vector such that $\widetilde{\boldsymbol{C}}_{[1,i+1]}$ (see (7)) is full column rank for some i with $0 \leq i \leq n-1$. If there exists some j with $i+2 \leq j \leq n$ such that $\widetilde{\boldsymbol{C}}_{[1,j]}$ is not full column rank, then $\widetilde{\boldsymbol{C}}_{[\backslash j]}$ is invertible.*

Due to the limitation of space, the proof of Theorem 2 is omitted. Interesting readers are referred to [12].

The following theorem shows how to find the desire j.

Theorem 3. *Let $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ be an arbitrary invertible matrix and $\boldsymbol{c} \in \mathbb{R}^n$ be an arbitrary nonzero vector such that $\widetilde{\boldsymbol{C}}_{[1,i+1]}$ (see (7)) is full column rank for some i with $0 \leq i \leq n-1$. Suppose that j is the smallest integer with $i+2 \leq j \leq n$ such that $\widetilde{\boldsymbol{C}}_{[1,j]}$ is not full column rank, then j is the largest integer with $j \leq n$ such that $\widetilde{\boldsymbol{C}}_{[\backslash j]}$ is invertible.*

Proof: By Theorem 2, $\widetilde{\boldsymbol{C}}_{[\backslash j]}$ is invertible. In the following, we show that $\widetilde{\boldsymbol{C}}_{[\backslash k]}$ is not invertible for any k with $j < k \leq n+1$ by contradiction.

Suppose that there exists k with $j < k \leq n+1$ such that $\widetilde{\boldsymbol{C}}_{[\backslash k]}$ is invertible. Then $\widetilde{\boldsymbol{C}}_{[1,j]}$ is full column rank which contradicts the assumption. Thus, j is the largest integer with $j \leq n$ such that $\widetilde{\boldsymbol{C}}_{[\backslash j]}$ is invertible. ■

Remark 1. *In Theorem 3, we assumed $j \leq n$ which is because if $\widetilde{\boldsymbol{C}}_{[1,j]}$ is full column rank for all j with $i+2 \leq j \leq n$, then $\widetilde{\boldsymbol{C}}_{[\backslash(n+1)]}$ is invertible, leading to $j = n+1$.*

Based on Theorem 3, an algorithm to find j is described in Algorithm 1. Since the complexity of the second step of Algorithm 1, which dominates the whole algorithm, is $\mathcal{O}(n^3)$ flops via Gaussian elimination [13, p.44], its total complexity is $\mathcal{O}(n^3)$ flops.

Algorithm 1 Efficient algorithm for updating $\boldsymbol{C}$

Input: A full column rank matrix $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ and a nonzero vector $\boldsymbol{c} \in \mathbb{R}^n$ that satisfy (8).

Output: An invertible matrix $\bar{\boldsymbol{C}} \in \mathbb{R}^{n\times n}$, whose columns are chosen from $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_n$ and $\boldsymbol{c}$, such that (9) holds and $\|\bar{\boldsymbol{c}}_i\|_2$ are as small as possible for all $1 \leq i \leq n$.

1: Find i and form $\widetilde{\boldsymbol{C}}$ (see (7)) such that it satisfies (10);
2: Reduce $\widetilde{\boldsymbol{C}}$ into its row echelon form by Gaussian elimination, for more details, see, e.g., [13, pp.40-41], and denote it by $\widetilde{\boldsymbol{R}}$.
3: Check whether $\widetilde{r}_{jj} = 0$ for $j = i+1, \ldots, n$ until finding a j such $\widetilde{r}_{jj} = 0$ if it exists, and set $\bar{\boldsymbol{C}} = \widetilde{\boldsymbol{C}}_{[\backslash j]}$. Otherwise, i.e., $\widetilde{r}_{jj} \neq 0$ for all $i+1 \leq j \leq n$, set $\bar{\boldsymbol{C}} = \widetilde{\boldsymbol{C}}_{[\backslash(n+1)]}$.

C. Schnorr-Euchner search algorithm

As our novel algorithm for the RSMP needs to use the integer vectors obtained by the improved Schnorr-Euchner search algorithm [10] to update the suboptimal solution $\boldsymbol{C}$, for self-contained, we introduce the Schnorr-Euchner search algorithm [14] for the following shortest vector problem (SVP)

$$\boldsymbol{c}^\star = \min_{\boldsymbol{c}\in\mathbb{Z}^{N_t}\backslash\{0\}} \|\bar{\boldsymbol{R}}\boldsymbol{c}\|_2^2. \tag{11}$$

More details on this algorithm are referred to [15], and its variants can be found in e.g., [16].

Suppose that $\boldsymbol{c}^\star$ satisfies the following hyper-ellipsoid constraint

$$\left\|\bar{\boldsymbol{R}}\boldsymbol{c}\right\|_2^2 < \beta^2, \tag{12}$$

where β is a given constant. Let

$$d_{N_t} = 0,\ d_i = -\frac{1}{\bar{r}_{ii}} \sum_{j=i+1}^{N_t} \bar{r}_{ij} c_j, \quad i = N_t - 1, \ldots, 1. \tag{13}$$

Then (12) can be transformed to

$$\sum_{i=1}^{N_t} \bar{r}_{ii}^2 (c_i - d_i)^2 < \beta^2$$

which is equivalent to

$$\bar{r}_{ii}^2 (c_i - d_i)^2 < \beta^2 - \sum_{j=i+1}^{N_t} \bar{r}_{jj}^2 (c_i - d_j)^2 \tag{14}$$

for $i = N_t, N_t - 1, \ldots, 1$ which is called as the level index, where $\sum_{j=N_t+1}^{N_t} \cdot = 0$.

The Schnorr-Euchner search algorithm starts with $\beta = \infty$, and sets $c_i = \lfloor d_i \rceil$ (d_i are computed via (13)) for $i = N_t, N_t - 1, \ldots, 1$. Clearly, $\boldsymbol{c} = \boldsymbol{0}$ is obtained and (14) holds. Since $\boldsymbol{c}^\star \neq \boldsymbol{0}$, $\boldsymbol{c}$ should be updated. To be more specific, c_1 is set as the next closest integer to d_1. Since $\beta = \infty$, (14) with $i = 1$ holds. Thus, this updated $\boldsymbol{c}$ is stored and β is updated to $\beta = \|\boldsymbol{R}\boldsymbol{c}\|_2$. Then, the algorithm tries to update the latest found $\boldsymbol{c}$ by finding a new $\boldsymbol{c}$ satisfying (12). Since (14) with $i = 1$ is an equality for the current $\boldsymbol{c}$, c_1 only cannot be updated. Thus we try to update c_2 by setting it as the next closest integer to d_2. If it satisfies (14) with $i = 2$, we try to update c_1 by setting $c_1 = \lfloor d_1 \rceil$ (d_1 is computed via (13)) and then check whether (14) with $i = 1$ holds or not, and so on; Otherwise, we try to update c_3, and so on. Finally, when we are not able to find a new integer $\boldsymbol{c}$ such that (14) holds with $i = N_t$, the search process stops and outputs the latest $\boldsymbol{c}$, which is actually $\boldsymbol{c}^\star$ satisfying (11).

D. A novel algorithm for solving the SMP

In this subsection, we develop a novel and efficient algorithm for (5). We begin with presenting the algorithm for the RSMP by incorporating Algorithm 1 into an improved Schnorr-Euchner search algorithm [10].

The proposed algorithm for the RSMP is described as follows: we start with a suboptimal solution $\boldsymbol{C}$ which is the $N_t \times N_t$ identity matrix with some column permutations such that

$$\|\bar{\boldsymbol{R}}\boldsymbol{c}_1\|_2 \leq \|\bar{\boldsymbol{R}}\boldsymbol{c}_2\|_2 \leq \ldots \leq \|\bar{\boldsymbol{R}}\boldsymbol{c}_{N_t}\|_2, \tag{15}$$

and assume $\beta = \|\bar{\boldsymbol{R}}\boldsymbol{c}_{N_t}\|$. Then we modify the Schnorr-Euchner search algorithm to search the nonzero integer vectors $\boldsymbol{c}$ satisfying (12) to update $\boldsymbol{C}$. Specifically, whenever a zero vector $\boldsymbol{c}$ is obtained, we update $\boldsymbol{c}$ by setting c_1 as the next closest integer to d_1 to obtain a nonzero integer vector $\boldsymbol{c}$; and as long as a nonzero integer vector $\boldsymbol{c}$ is obtained (note that $\boldsymbol{c}$

satisfies $\|\bar{\boldsymbol{R}}\boldsymbol{c}\|_2 < \|\bar{\boldsymbol{R}}\boldsymbol{c}_{N_t}\|_2$), we use Algorithm 1 to update $\boldsymbol{C}$ (note that $\bar{\boldsymbol{R}}\boldsymbol{C}$ and $\bar{\boldsymbol{R}}\boldsymbol{c}$ are respectively viewed as $\boldsymbol{C}$ and $\boldsymbol{c}$) to another matrix which is also denoted as $\boldsymbol{C}$. Then, we set $\beta = \|\bar{\boldsymbol{R}}\boldsymbol{c}_{N_t}\|$. Note that β decreases only when the last column of $\boldsymbol{C}$ is changed. After this, we use the Schnorr-Euchner search algorithm [10] to update $\boldsymbol{c}$ and then update $\boldsymbol{C}$ with Algorithm 1. Finally, when $\boldsymbol{C}$ cannot be updated anymore and β cannot be decreased anymore, the search process stops and outputs $\boldsymbol{C}^\star$.

For efficiency, $\|\bar{\boldsymbol{R}}\boldsymbol{c}_i\|_2, 1 \leq i \leq N_t$, can be stored with a vector, say $\boldsymbol{p}$, and $\|\bar{\boldsymbol{R}}\boldsymbol{c}\|_2$ can be calculated while using the Schnorr-Euchner enumeration strategy. Then, whenever we need to update $\boldsymbol{C}$, we also update $\boldsymbol{p}$ instead of calculating $\|\bar{\boldsymbol{R}}\boldsymbol{c}_i\|_2$ for all $1 \leq i \leq N_t$.

Clearly, if $\boldsymbol{C}^\star$ is a solution to the RSMP, so is $\hat{\boldsymbol{C}}^\star$, where $\hat{\boldsymbol{C}}^\star = \boldsymbol{C}^\star$ except that $\hat{\boldsymbol{c}}_j^\star = -\boldsymbol{c}_j^\star$ for a $1 \leq j \leq N_t$. Thus, to further speed up the above process, the strategy proposed in [10] can be applied here. Specifically, only the nonzero integer vectors $\boldsymbol{c}$, satisfying $c_{N_t} \geq 0$ and $c_k \geq 0$ if $c_{k+1:N_t} = \boldsymbol{0}$, where $1 \leq k \leq N_t - 1$, are searched to update $\boldsymbol{C}$ in the above process. Note that only the former property of $\boldsymbol{c}$ is exploited in [8], whereas our strategy can prune more vectors while retaining optimality.

By the above analysis, the proposed algorithm for the RSMP can be summarized in Algorithm 2. Moveover, the algorithm for the problem (3) can be described in Algorithm 3.

Algorithm 2 A novel algorithm for the RSMP

Input: A nonsingular upper triangular $\bar{\boldsymbol{R}} \in \mathbb{R}^{N_t \times N_t}$.
Output: A solution $\boldsymbol{C}^\star$ to the RSMP, i.e.,

$$\|\bar{\boldsymbol{R}}\boldsymbol{c}_i^\star\|_2 = \lambda_i, \quad i = 1, 2, \ldots, N_t,$$

where λ_i is the i-th successive minimum of lattice $\mathcal{L}(\bar{\boldsymbol{R}})$.

1) Set $k := N_t$, let $\boldsymbol{C}$ be the $N_t \times N_t$ identity matrix with some column permutations such that (15) holds, and let $\beta := \|\bar{\boldsymbol{R}}\boldsymbol{c}_n\|_2$.
2) Set $c_k := \lfloor d_k \rceil$ (where d_k is obtained by using (13)). Let $s_k := 1$ if $d_k \geq c_k$, otherwise let $s_k := -1$.
3) If (14) does not hold, then go to Step 4. Else if $k > 1$, set $k := k - 1$ and go to Step 2. Else, i.e., $k = 1$, go to Step 5.
4) If $k = N_t$, set $\boldsymbol{C}^\star := \boldsymbol{C}$ and terminate. Else, set $k := k + 1$ and go to Step 6.
5) If $\boldsymbol{c} \neq \boldsymbol{0}$, use Algorithm 1 to update $\boldsymbol{C}$, set $\beta := \|\bar{\boldsymbol{R}}\boldsymbol{c}_{N_t}\|_2$ and $k := k + 1$.
6) If $k = N_t$ or $\boldsymbol{c}_{k+1:N_t} = \boldsymbol{0}$, set $c_k := c_k + 1$, and go to Step 3. Otherwise, set $c_k := c_k + s_k$, $s_k := -s_k - \mathrm{sgn}(s_k)$ and go to Step 3.

IV. Numerical Results

This section presents simulation results to compare Algorithm 3 (denoted by "New Alg. ") with the two optimal algorithms in [7] and [8] (denoted by "WC" and "DKWZ", respectively) by using flat rayleigh fading channel. The average achievable rates and CPU time over 2000 random samples are reported. Specifically, for simplicity, we let $N_r = N_t$, and for any fixed N_t and P, we first generated 2000 $\boldsymbol{G}$'s according to (1). Then, we respectively used these algorithms to solve (3) for each generated $\boldsymbol{G}$, counted their CPU time and computed their achievable rates according to (2). Finally, we calculated their average CPU time and achievable rates.

Algorithm 3 A novel algorithm for (3)

Input: A symmetric positive definite matrix $\boldsymbol{G} \in \mathbb{R}^{N_t \times N_t}$.
Output: A solution $\boldsymbol{A}^\star$ to the problem (3).

1) Perform Cholesky factorization to $\boldsymbol{G}$ in (3) to get a nonsingular matrix $\boldsymbol{R}$ (see (4)).
2) Perform LLL reduction to $\boldsymbol{R}$ to get $\bar{\boldsymbol{R}}$ and $\boldsymbol{Z}$ (see (6)).
3) Getting $\boldsymbol{C}^\star$ by solving the RSMP with Algorithm 2.
4) Set $\boldsymbol{A}^\star := \boldsymbol{Z}\boldsymbol{C}^\star$.

TABLE I
Average CPU time in seconds over 2000 realizations with $N_t = 4$

P (dB) \ Alg.	WC	DKWZ	New Alg.
2	0.0039	0.0022	0.00073
4	0.0201	0.0023	0.00078
6	0.2426	0.0023	0.00078
8	0.1827	0.0023	0.00080
10	1.810	0.0023	0.00085
12	13.32	0.0022	0.00081
14	26.23	0.0023	0.00089
16	32.69	0.0023	0.00089

All of the simulations were performed on MATLAB 2016b on the same desktop computer with Intel(R) Xeon(R) CPU E5-1603 v4 working at 2.80 GHZ.

Figure 2 shows the average achievable rates for the three algorithms with $N_t = 2, 4$. Figure 3 shows the average CPU time for the three algorithms with $N_t = 2$. Since "WC" is time consuming when $N_t = 4$, to clearly see the average CPU time for these algorithms, we display them in Table I.

Figures 4 and 5 respectively show average achievable rates and CPU time for "New Alg. " and "DKWZ" with $P = 1, 10, 20$ dB and $N_t = 6 : 2 : 20$. We did not compare these two algorithms with "WC" because we found it is slow.

Figures 2 and 4 indicate that the average rates for the three algorithms are exactly the same, and this is because all of them are optimal algorithms for $\boldsymbol{A}^\star$.

From Table I, Figures 3 and 5, one can see that the new algorithm is always most efficient. Although Figure 3 indicates that "DKWZ" is slower than "WC" when $N_t = 2$, we found that the former is always faster than the latter when $N_t = 4 : 2 : 20$ (the case for $N_r = 4$ is showed in Table I).

V. Conclusion

In this paper, we developed a novel efficient exact SMP algorithm to find the optimal integer coefficient matrix $\boldsymbol{A}^\star$ for the IF linear receiver. Simulation results showed the optimality of the proposed algorithm and indicated that the proposed algorithms is much more efficient than existing optimal ones.

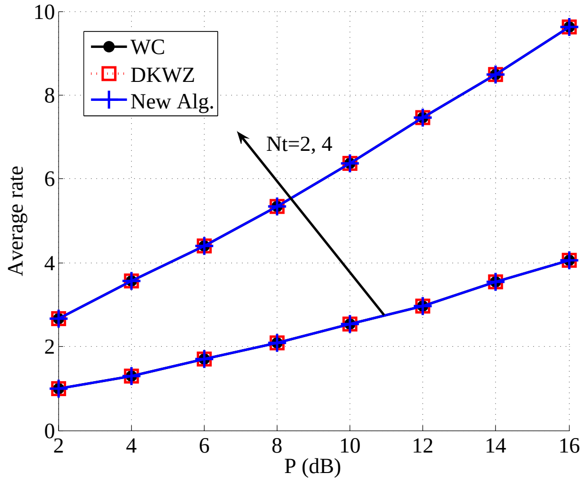

Fig. 2. Average rates over 2000 realizations for $N_t = 2, 4$

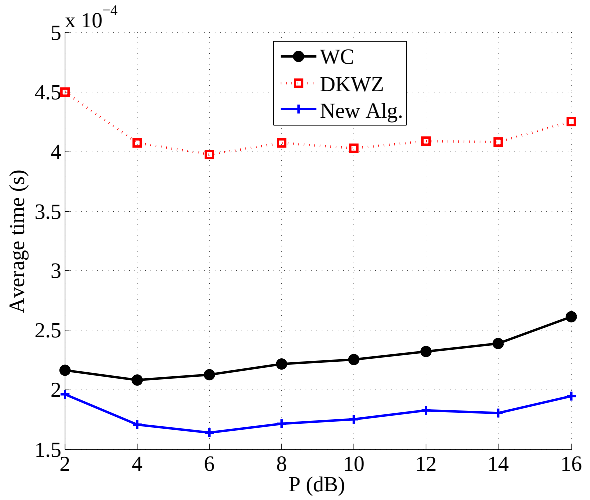

Fig. 3. Average CPU time in seconds over 2000 realizations for $N_t = 2$

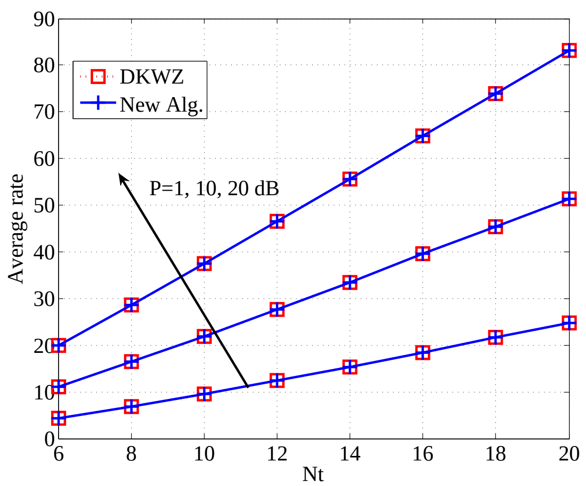

Fig. 4. Average rates over 2000 realizations for $P = 1, 10, 20$ dB

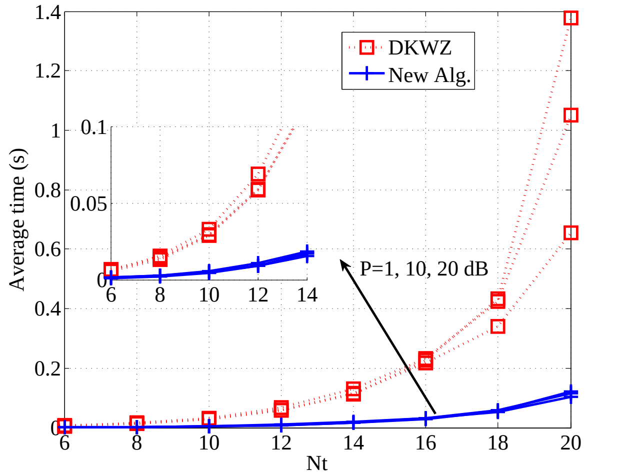

Fig. 5. Average CPU time in seconds over 2000 realizations for $P = 1, 10, 20$ dB

References

[1] A. Lenstra, H. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Math. Ann.*, vol. 261, no. 4, pp. 515–534, 1982.

[2] X.-W. Chang, J. Wen, and X. Xie, "Effects of the LLL reduction on the success probability of the babai point and on the complexity of sphere decoding," *IEEE Trans. Inf. Theory*, vol. 59, no. 8, pp. 4915–4926, 2013.

[3] J. Wen, C. Tong, and S. Bai, "Effects of some lattice reductions on the success probability of the zero-forcing decoder," *IEEE Commun. Lett.*, vol. 20, no. 10, p. 2031, 2016.

[4] J. Wen and X.-W. Chang, "The success probability of the Babai point estimator and the integer least squares estimator in box-constrained integer linear models," *IEEE Trans. Inf. Theory*, vol. 63, no. 1, pp. 631–648, 2017.

[5] J. Zhan, B. Nazer, U. Erez, and M. Gastpar, "Integer-forcing linear receivers: A new low-complexity MIMO architecture," in *2010 IEEE 72nd VTC 2010-Fall*. IEEE, 2010, pp. 1–5.

[6] ——, "Integer-forcing linear receivers," *IEEE Trans. Inf. Theory*, vol. 60, no. 12, pp. 7661–7685, 2014.

[7] L. Wei and W. Chen, "Integer-forcing linear receiver design over MIMO channels," in *2012 IEEE Global Communications Conference (GLOBECOM)*, 2012, pp. 3560–3565.

[8] L. Ding, K. Kansanen, Y. Wang, and J. Zhang, "Exact SMP algorithms for integer-forcing linear MIMO receivers," *IEEE Trans. Wireless Commun.*, vol. 14, no. 12, pp. 6955–6966, 2015.

[9] C. Feng, D. Silva, and F. R. Kschischang, "An algebraic approach to physical-layer network coding," *IEEE Trans. Inf. Theory*, vol. 59, no. 11, pp. 7576–7596, 2013.

[10] J. Wen and X.-W. Chang, "A KZ reduction algorithm," *submitted to IEEE Trans. Wireless Commun.*, 2017.

[11] O. Ordentlich and U. Erez, "Precoded integer-forcing universally achieves the MIMO capacity to within a constant gap," *IEEE Trans. Inf. Theory*, vol. 61, no. 1, pp. 323–340, 2015.

[12] J. Wen, L. Li, X. Tang, M. W. H., and C. Tellambura, "An exact algorithm for the optimal integer coefficient matrix in integer-forcing linear MIMO receivers," *arXiv:1610.06618*, 2016.

[13] S. Banerjee and A. Roy, *Linear algebra and matrix analysis for statistics*. CRC Press, 2014.

[14] C. P. Schnorr, "A hierarchy of polynomial time lattice basis reduction algorithms," *IEEE Trans. Commun.*, vol. 53, pp. 201–224, 1987.

[15] X.-W. Chang and Q. Han, "Solving box-constrained integer least squares problems," *IEEE Trans. Wireless Commun.*, vol. 7, no. 1, pp. 277–287, 2008.

[16] J. Wen, B. Zhou, W. H. Mow, and X.-W. Chang, "An efficient algorithm for optimally solving a shortest vector problem in compute-and-forward design," *IEEE Trans. Wireless Commun.*, vol. 15, no. 10, pp. 6541–6555, 2016.