Solving Box-Constrained Integer Least Squares Problems

Xiao-Wen Chang, Qing Han

***Abstract*— A box-constrained integer least squares problem (BILS) arises from several wireless communications applications. Solving a BILS problem usually has two stages: reduction (or preprocessing) and search. This paper presents a reduction algorithm and a search algorithm. Unlike the typical reduction algorithms, which use only the information of the lattice generator matrix, the new reduction algorithm also uses the information of the given input vector and the box constraint and is very effective for search. The new search algorithm overcomes some shortcomings of the existing search algorithms and gives some other improvement. Simulation results indicate the combination of the new reduction algorithm and the new search algorithm can be much more efficient than the existing algorithms, in particular when the least square residual is large.**

***Index Terms*— Integer least squares, lattice, MIMO channels, detection, decoding. reduction, search.**

I. Introduction

GIVEN a real m-vector $\boldsymbol{y}$ and a real $m \times n$ matrix $\boldsymbol{A}$ with full column rank, one wants to solve the minimization problem

$$\min_{\boldsymbol{x}\in\mathbb{Z}^n} \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2, \tag{1}$$

where $\mathbb{Z}^n$ denotes the set of all integer n-vectors. We refer to (1) as the *integer least squares (ILS) problem*. In the lattice theory, $\boldsymbol{A}$ is called the generator matrix of the lattice $\mathcal{L}(\boldsymbol{A}) = \{\boldsymbol{A}\boldsymbol{x} : \boldsymbol{x} \in \mathbb{Z}^n\}$, $\boldsymbol{y}$ is called the input vector, and (1) is referred to as a *closest-point problem*, since it is to find a point in the lattice which is closest to the given input point $\boldsymbol{y}$. In channel coding, the ILS problem is referred to as *decoding*. The ILS problem may arise from many applications, such as communications, cryptograph, lattice design, Monte Carlo second-moment estimation, radar imaging, and global positioning systems etc, see, e.g., [1], [2], [3] and references therein. It is well know that the ILS problem (1) is NP-hard [4].

In some wireless communications applications, the integer vector $\boldsymbol{x}$ is constrained to a box (see, e.g., [5]):

$$\mathcal{B} = \{\boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u},\ \boldsymbol{l} \in \mathbb{Z}^n,\ \boldsymbol{u} \in \mathbb{Z}^n\}. \tag{2}$$

Then one wants to solve

$$\min_{\boldsymbol{x}\in\mathcal{B}} \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2. \tag{3}$$

This work was supported by NSERC of Canada grant RGPIN217191-03 and it was done when the second author was an M.Sc. student under the first author's supervision

X.-W. Chang is with the School of Computer Science, McGill University, Montreal, QC H3A 2A7, Canada (email: chang@cs.mcgill.ca).

Q. Han is currently with SolVision Inc., Boucherville, QC J4B 1E6, Canada (email: aeris.han@solvision.net).

We refer to (3) as the *box-constrained integer least squares (BILS) problem*.

Any typical method for solving the ILS problem (1) or the BILS problem (3) has two stages: reduction (or preprocessing) and search. An excellent survey on the search methods for solving (1) can be found in the semi-tutorial paper [1], which also mentions typical reduction methods. In [1], an efficient search algorithm based on the Schnorr-Euchner enumeration strategy (see [6]) was proposed for solving the ILS problem (1). The algorithm was then modified in [7] to solve the BILS problem (3) by taking the box constraint (2) into account. In [5], two search algorithms based on the Phost enumeration strategy (see [8], [9] and [10]) and Schnorr-Euchner enumeration strategy, respectively, were proposed for solving the BILS problem (3). As in [1] for solving the ILS problem (1), it was found in [5] that the Schnorr-Euchner strategy is usually more efficient than the Phost strategy for solving the BILS problem (3). To make the search process easier and more efficient, the reduction stage is needed to transform $\boldsymbol{A}$ to an upper triangular matrix by an orthogonal transformation. The key part of a reduction algorithm is to reorder the columns of $\boldsymbol{A}$. Different ordering may have significant effect on the search speed. In [5], three reduction strategies were introduced. In [11], an algorithm was proposed for finding a suboptimal solution of the ILS problem (1) and it can be used for the reduction purpose. All these reduction strategies use only the information of the generator matrix $\boldsymbol{A}$.

Since computing the optimal solution to (3) may become time-prohibitive when the least squares residual is large or the dimension of the problem is large (see, e.g., [12]), recently some algorithms have been proposed to compute a suboptimal solution by using convex optimization methods or semidefinite programming techniques, see, e.g., [13]–[15]. But computing a suboptimal solution is beyond the scope of this paper and will not be discussed further.

The goal of this paper is to present a faster algorithm for solving the general BILS problem (3). We do not consider any specific application. Specifically, we will propose a new reduction algorithm, which uses all information provided in the BILS problem (3) and is much more effective than the reduction strategies given in [5] and [11], and a new search algorithm, which gives a few modifications of the Schnorr-Euchner strategy based algorithms given in [5] and [7].

The rest of the paper is organized as follows. In Section II, we review the reduction algorithms given in [16] and [5] and the Schnorr-Euchner strategy based search algorithms presented in [5] and [7]. In Section III, we present a new reduction algorithm and a new search algorithm. Section IV

gives simulation results to show the advantages of our new algorithms. Finally a summary is given in Section V.

The following notation is used in the paper. $\mathbb{R}^{m\times n}$ and $\mathbb{R}^n$ denotes the set of all real $m\times n$ matrices and the set of all real n-vectors. $\mathbb{Z}^{m\times n}$ and $\mathbb{Z}^n$ denotes the set of all integer $m\times n$ matrices and the set of all integer n-vectors. Bold upper case letters and bold lower case letters are used to denote matrices and vectors, respectively. A vector $\boldsymbol{a}$ is a column vector and a vector $\boldsymbol{a}^T$ is a row vector. The $n\times n$ identity matrix is denoted by $\boldsymbol{I}$ or $\boldsymbol{I}_n$. MATLAB notation is used to denote a submatrix. $[\boldsymbol{v}_1;\boldsymbol{v}_2]$ denotes a column vector whose top part is a vector $\boldsymbol{v}_1$ and whose bottom part is a vector $\boldsymbol{v}_2$. For a real scalar z, we use $\lfloor z\rceil$ to denote its nearest integer. If there is a tie, $\lfloor z\rceil$ denotes the one with smaller magnitude. The operation $\text{sign}(z)$ returns -1 if $z\le 0$ and 1 if $z>0$.

II. Overview of some existing algorithms

We will briefly review some recent algorithms for solving the BILS problem (3). In Section II-A, we give a general reduction process and introduce two permutation strategies proposed in [16] and [5], respectively, and suggest another strategy which was proposed in [11] to find a suboptimal solution of the ILS problem (1) for reduction. In Section II-B, we introduce the two Schnorr-Euchner strategy based search algorithms presented in [5] and [7], respectively, and give some comments.

A. Reduction

A reduction process transforms the matrix $\boldsymbol{A}$ to an upper triangular matrix, which has good properties to make the search process more efficient. For solving the BILS problem (3), this can be accomplished by the QR decomposition of $\boldsymbol{A}$ with column pivoting:

$$\boldsymbol{AP}=[\boldsymbol{Q}_1,\boldsymbol{Q}_2]\begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{bmatrix}=\boldsymbol{Q}_1\boldsymbol{R} \tag{4}$$

where $\boldsymbol{P}\in\mathbb{Z}^{n\times n}$ is a permutation matrix, $[\underset{n}{\boldsymbol{Q}_1},\ \underset{m-n}{\boldsymbol{Q}_2}]\in\mathbb{R}^{m\times m}$ is orthogonal, and $\boldsymbol{R}\in\mathbb{R}^{n\times n}$ is nonsingular upper triangular. The QR decomposition can be computed by using Householder transformations, Givens rotations, or Gram-Schmidt orthogonalization (which gives only $\boldsymbol{Q}_1$ not $\boldsymbol{Q}$), see, e.g., [20, Sec 2.4]. The main difference between different reduction strategies in the literature is the permutation matrix $\boldsymbol{P}$.

With the QR decomposition (4), we have

$$\|\boldsymbol{y}-\boldsymbol{Ax}\|_2^2=\|\boldsymbol{Q}_1^T\boldsymbol{y}-\boldsymbol{RP}^T\boldsymbol{x}\|_2^2+\|\boldsymbol{Q}_2^T\boldsymbol{y}\|_2^2. \tag{5}$$

Define

$$\bar{\boldsymbol{y}}=\boldsymbol{Q}_1^T\boldsymbol{y},\quad \boldsymbol{z}=\boldsymbol{P}^T\boldsymbol{x},\quad \bar{\boldsymbol{l}}=\boldsymbol{P}^T\boldsymbol{l},\quad \bar{\boldsymbol{u}}=\boldsymbol{P}^T\boldsymbol{u}.$$

Then from (5) we see that the original BILS problem (3) is equivalent to the reduced one:

$$\min_{\boldsymbol{z}\in\bar{\mathcal{B}}}\|\bar{\boldsymbol{y}}-\boldsymbol{Rz}\|_2^2 \tag{6}$$

where

$$\bar{\mathcal{B}}=\{\boldsymbol{z}\in\mathbb{Z}^n:\ \bar{\boldsymbol{l}}\le\boldsymbol{z}\le\bar{\boldsymbol{u}},\ \bar{\boldsymbol{l}}\in\mathbb{Z}^n,\ \bar{\boldsymbol{u}}\in\mathbb{Z}^n\}.$$

Then the search process tries to solve (6). Note that if $\hat{\boldsymbol{z}}$ is the solution to (6), then $\hat{\boldsymbol{x}}=\boldsymbol{P}\hat{\boldsymbol{z}}$ is the solution to (3).

For solving the ILS problem (1), an often used reduction algorithm is the well-known LLL algorithm [17], which computes (4), but $\boldsymbol{P}$ is a unimodular matrix, i.e., $\boldsymbol{P}$ is an integer matrix with $|\det(\boldsymbol{P})|=1$. For the BILS problem (3), a general unimodular transformation will make the box constraint $\mathcal{B}$ very complicated. So the LLL reduction is usually not used for solving the BILS problem in the literature. Here we introduce a few typical reduction strategies suitable for solving (3).

To make the search process more efficient, a reduction algorithm usually strives for

$$|r_{11}|\le|r_{22}|\le\cdots\le|r_{nn}|.$$

Note that this may not be achievable. The justification for this order can be found in [1] which considers solving the unconstrained problem (1).

In [16], $\boldsymbol{P}$ was chosen such that the columns of the permuted $\boldsymbol{A}$ are arranged in a nondecreasing order in terms of the 2-norms.

In [5], the so-called vertical Bell Labs layered space-time (V-BLAST) optical detection ordering given in [18] was proposed for permutations. The permutation strategy (to be called V-BLAST) determines the columns of the permuted $\boldsymbol{A}$ from the last to the first. Let $\mathcal{J}_k$ denote the set of column indices for the not yet chosen columns when the k-th column of the permuted $\boldsymbol{A}$ is to be determined ($k=n,n-1,\ldots,1$). This strategy chooses the $p(k)$-th column of the original matrix $\boldsymbol{A}$ as the k-th column of the permuted matrix $\boldsymbol{A}$ we seek:

$$p(k)=\arg\max_{j\in\mathcal{J}_k}\boldsymbol{a}_j^T\Big[\boldsymbol{I}-\boldsymbol{A}_{k,j}(\boldsymbol{A}_{k,j}^T\boldsymbol{A}_{k,j})^{-1}\boldsymbol{A}_{k,j}^T\Big]\boldsymbol{a}_j \tag{7}$$

where $\boldsymbol{a}_j$ is the j-th column of $\boldsymbol{A}$ and $\boldsymbol{A}_{k,j}$ is the $m\times(k-1)$ matrix formed by the columns $\boldsymbol{a}_i$ with $i\in\mathcal{J}_k-\{j\}$.

We can easily show that $\boldsymbol{a}_j^T\Big[\boldsymbol{I}-\boldsymbol{A}_{k,j}(\boldsymbol{A}_{k,j}^T\boldsymbol{A}_{k,j})^{-1}\boldsymbol{A}_{k,j}^T\Big]\boldsymbol{a}_j$ in (7) is the square of the Euclidean distance from $\boldsymbol{a}_j$ to the space spanned by the columns of $\boldsymbol{A}_{k,j}$. Note that in the QR decomposition (4), $|r_{kk}|$ is the orthogonal distance from the k-th column of $\boldsymbol{AP}$ to the space spanned by the first $k-1$ columns of $\boldsymbol{AP}$. Thus $\boldsymbol{a}_{p(k)}$ is the column which makes $|r_{kk}|$ maximum over all the not yet chosen columns when we determine the k-th column of the permuted $\boldsymbol{A}$ for $k=m,m-1,\ldots,1$. For finding the permutations, we can design an efficient algorithm, which will become obvious after we give a new reduction algorithm in Section III-A.

Numerical simulations given in [5] show that the second strategy above is more effective than the first one. In [5], another reduction strategy called V-BLAST MMSE-DFE was introduced. But it uses some statistical information of $\boldsymbol{y}$ and $\boldsymbol{x}$ and is for computing a sub-optimal solution to (3). Our simulations indicated that this strategy is not as effective as the second strategy when they are used for finding the optimal solution to (3). For these reasons, it will not be introduced here.

In [11], the so called *sorted QR decomposition* (SQRD) algorithm was proposed for decoding the same codes as what the V-BLAST algorithms decode. It is used to find a

suboptimal solution (the Babai integer point, see later) to the ILS problem (1). We can use it for the reduction purpose. In contrast to the V-BLAST strategy, this SQRD strategy determines the columns of the permuted $\boldsymbol{A}$ we seek from the first to the last by using the modified Gram-Schmidt method. In the k-th step of the modified Gram-Schmidt method, the k-th column of the permuted $\boldsymbol{A}$ we seek is chosen from the remaining $n-k+1$ columns of $\boldsymbol{A}$ such that r_{kk} is smallest (for $k=1,2,\ldots,n$).

Algorithm SQRD

Input: The generator matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with full column rank, the input vector $\boldsymbol{y} \in \mathbb{R}^m$, the lower bound vector $\boldsymbol{l} \in \mathbb{Z}^n$, and the upper bound vector $\boldsymbol{u} \in \mathbb{Z}^n$.

Output: The reduced upper triangular matrix $\boldsymbol{R} \in \mathbb{R}^{n\times n}$, the permutation matrix $\boldsymbol{P} \in \mathbb{Z}^{n\times n}$, the vector $\bar{\boldsymbol{y}} \in \mathbb{R}^n$, the permuted lower bound vector $\bar{\boldsymbol{l}}$, and the permuted upper bound vector $\bar{\boldsymbol{u}}$.

(Initialization) $\boldsymbol{Q}_1 := \boldsymbol{A}$, $\boldsymbol{R} := \boldsymbol{0}$, $\boldsymbol{P} := \boldsymbol{I}_n$, $\bar{\boldsymbol{l}} := \boldsymbol{l}$, $\bar{\boldsymbol{u}} := \boldsymbol{u}$.
for $k = 1,\ldots,n$
  $p := \arg\min_{j\in\{k,\ldots,n\}} \|\boldsymbol{q}_j\|_2$.
  if $p \neq k$, **then**
    interchange columns k and p of $\boldsymbol{Q}_1$, $\boldsymbol{R}$ and $\boldsymbol{P}$
    interchange entries k and p of $\bar{\boldsymbol{l}}$ and $\bar{\boldsymbol{u}}$
  end
  $r_{kk} := \|\boldsymbol{q}_k\|_2$
  $\boldsymbol{q}_k := \boldsymbol{q}_k / r_{kk}$
  for $j = k+1 : n$
    $r_{kj} := \boldsymbol{q}_k^T \boldsymbol{q}_j$
    $\boldsymbol{q}_j := \boldsymbol{q}_j - r_{kj}\boldsymbol{q}_k$
  end
end
$\bar{\boldsymbol{y}} := \boldsymbol{Q}_1^T \boldsymbol{y}$

Note that in a practical implementation, we need only a vector rather than the matrix $\boldsymbol{P}$ to store the permutation information. There is a numerical stability problem with the above algorithm—the $\boldsymbol{Q}_1$ factor may lose orthogonality when $\boldsymbol{A}$ is ill-conditioned (see, e.g., [20, Sec 2.4]). We can easily overcome this problem by using Householder transformations instead of the modified Gram-Schmidt orthogonalization to compute the QR decomposition with the same permutation strategy. The simulations given in [11] indicated that the Babai integer point (see later) as an estimate of the concerned parameter vector obtained by the SQRD strategy is slightly less accurate than the Babai integer point obtained by the V-BLAST strategy. However, SQRD is computationally more efficient than V-BLAST. Algorithm SQRD requires $2mn^2$ flops (see [19, p232]) and note that computing all $\|\boldsymbol{q}_j\|_2$ can be done in $O(mn)$ flops, cf. [19, p240]), while an implementation of V-BLAST requires more flops, see III-A.

B. Search

We first introduce the ideas of the search algorithm for the unconstrained problem (i.e., $\vec{\mathcal{B}}$ in (6) is replaced by $\mathbb{Z}^n$), which leads to the search algorithm given in [5] for the constrained problem by some modifications.

Suppose the solution of (6) satisfies the following bound

$$\|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|^2 < \beta \tag{8}$$

or equivalently

$$\sum_{k=1}^{n} \Big(\bar{y}_k - \sum_{j=k}^{n} r_{kj} z_j\Big)^2 < \beta. \tag{9}$$

If we are only aware of an upper bound on $\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2$, rather than (8), we can easily obtain (8) by using the equality $\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2 = \|\bar{\boldsymbol{y}}-\boldsymbol{R}\boldsymbol{z}\|_2^2 + \|\boldsymbol{Q}_2^T\boldsymbol{y}\|_2^2$. The inequality (8) stands for a hyper-ellipsoid in $\mathbb{R}^n$. The search process is to seek the integer point within this hyper-ellipsoid which minimize the left hand side of (8). Since (8) can also be regarded as a hyper-sphere in terms of $\boldsymbol{w}$, where $\boldsymbol{w} = \boldsymbol{R}\boldsymbol{z}$ and all $\boldsymbol{w}$ forms the lattice set generated by $\boldsymbol{R}$, finding the optimal lattice point in the hyper-sphere is referred to as sphere decoding in communications.

Define

$$c_n = \bar{y}_n / r_{nn}, \quad c_k = (\bar{y}_k - \sum_{j=k+1}^{n} r_{kj} z_j)/r_{kk}, \tag{10}$$

for $k = n-1,\ldots,1$. Note that c_k depends on z_{k+1}, z_{k+2}, …, z_n. Then (9) can be rewritten as

$$\sum_{k=1}^{n} r_{kk}^2 (z_k - c_k)^2 < \beta, \tag{11}$$

which implies a set of inequalities:

$$\text{level } n: \ r_{nn}^2 (z_n - c_n)^2 < \beta, \tag{12}$$

$$\vdots$$

$$\text{level } k: \ r_{k,k}^2 (z_k - c_k)^2 < \beta - \sum_{i=k+1}^{n} r_{ii}^2 (z_i - c_i)^2, \tag{13}$$

$$\vdots$$

$$\text{level } 1: \ r_{11}^2 (z_1 - c_1)^2 < \beta - \sum_{i=2}^{n} r_{ii}^2 (z_i - c_i)^2. \tag{14}$$

Based on the above inequalities, a search process using the Schnorr and Euchner enumeration strategy can start. First at level n, choose $z_n = \lfloor c_n \rceil$. If it does not satisfy the inequality (12), no any other integer will satisfy the inequality, so the optimal solution of (1) is outside the hyper-ellipsoid and we have to increase the value of β (this will not happen if the initial bound β is large enough, see some choices introduced at the end of Section III-B). If it satisfies the inequality, we proceed to level $n-1$. At this level, we compute c_{n-1} by (10) and choose $z_{n-1} = \lfloor c_{n-1} \rceil$. If z_{n-1} does not satisfy the inequality (13) with $k = n-1$, then we move back to level n and choose z_n to be the second nearest integer to c_n, and so on; otherwise, we proceed to level $n-2$. We continue this procedure until we reach the level 1 and obtain an integer point $\hat{\boldsymbol{z}}$. We store this point and update the bound β by taking $\beta = \|\bar{\boldsymbol{y}} - \boldsymbol{R}\hat{\boldsymbol{z}}\|^2$. Then we start the search process again to try to find a better integer point. First we move up to level 2 to update the value of z_2 by choosing z_2 to be the next nearest integer to c_2 ("next" means "next to $\hat{z}_2$"). If it satisfies the inequality at level 2, we move down to level 1 to update the

value of z_1 (note that z_2 has just been updated and $z_3, \ldots, z_n$ are the same as those corresponding entries of $\hat{\boldsymbol{z}}$), otherwise we move up to level 3 to update the value of z_3, and so on. Finally, when we fail to find a new value for z_n to satisfy the inequality (12), the search process stops and the latest found integer point is the optimal solution we seek. If the initial bound β is set to be ∞, then we refer to the first found integer point $\hat{\boldsymbol{z}}$ as the Babai integer point (note that the lattice point $\boldsymbol{R}\hat{\boldsymbol{z}}$ is called the Babai point in [1]).

To solve the BILS problem (3), the box constraint in (6) has to be considered during the search. The following search algorithm presented in Damen, El Gamal, and Caire (DEC) [5] (with minor changes) shows how the modification can be done to take the box constraint into account.

Algorithm DEC

Input: The nonsingular upper triangular matrix $\boldsymbol{R} \in \mathbb{R}^{n\times n}$, the vector $\bar{\boldsymbol{y}} \in \mathbb{R}^n$, the lower bound vector $\bar{\boldsymbol{l}} \in \mathbb{Z}^n$, the upper bound vector $\bar{\boldsymbol{u}} \in \mathbb{Z}^n$, and the initial hyper-ellipsoid bound β.
Output: The solution $\hat{\boldsymbol{z}} \in \mathbb{Z}^n$ to the BILS problem (6).

1) (Initialization) Set $k := n$ and $T_k := 0$
2) Compute $c_k := (\bar{y}_k - \sum_{j=k+1}^{n} r_{kj} z_j)/r_{kk}$, $z_k := \lfloor c_k \rceil$, $\Delta_k := \text{sign}(c_k - z_k)$
3) **if** $T_k + r_{kk}^2 (z_k - c_k)^2 \geq \beta$, **then**
   Go to Step 4 // we are not inside the ellipsoid
   else if $z_k \notin [\bar{l}_k, \bar{u}_k]$, **then**
   Go to Step 6 // we are inside the ellipsoid
   // but outside the box constraint
   else if $k > 1$, **then**
   Compute $T_{k-1} := T_k + r_{kk}^2 (z_k - c_k)^2$
   Set $k := k - 1$, go to Step 2
   else Go to Step 5 // $k = 1$
   end
4) **if** $k = n$, **then**
   Terminate
   else Set $k := k + 1$, go to Step 6
   end
5) (A valid point is found)
   Compute $\beta := T_1 + r_{11}^2 (z_1 - c_1)^2$
   Set $\hat{\boldsymbol{z}} := \boldsymbol{z}$ and $k := k + 1$
6) (Enumeration at level k)
   Compute $z_k := z_k + \Delta_k$, $\Delta_k := -\Delta_k - \text{sign}(\Delta_k)$
   Go to Step 3.

Suppose that in Step 3 the inequality $T_k + r_{kk}^2 (z_k - c_k)^2 < \beta$ holds, then

$$z_k \in \left[\left\lceil c_k - \sqrt{\beta - T_k}/|r_{kk}| \right\rceil, \ \left\lfloor c_k + \sqrt{\beta - T_k}/|r_{kk}| \right\rfloor \right].$$

If the above interval is large, but there is small overlap or no overlap between it and the interval $[\bar{l}_k, \bar{u}_k]$, then a lot of invalid integers are enumerated in Step 6. This shortcoming is avoided in another Schnorr-Euchner strategy based algorithm proposed in Boutros, Gresset, Brunel and Fossorier (BGBF) [7], which is a direct extension of the search algorithm proposed in [1] to deal with the box constraint. Their algorithm is for a nonsingular generator matrix $\boldsymbol{A}$ and the upper triangular $\boldsymbol{R}$ is obtained from the Cholesky decomposition of $\boldsymbol{A}^T\boldsymbol{A}$. From the least squares theory (see, e.g., [20, Sec 2.2]), obtaining $\boldsymbol{R}$ in this way may cause some numerical difficulties when $\boldsymbol{A}$ is ill conditioned. Thus we just apply the BGBF algorithm to the reduced problem (6), which was obtained by the QR decomposition. In the BGBF algorithm, the diagonal entries of $\boldsymbol{R}$ are assumed to be positive. If any diagonal entry of the $\boldsymbol{R}$ obtained by the QR decomposition is negative, we can multiply the corresponding row of $\boldsymbol{R}$ by -1 and the corresponding column of $\boldsymbol{Q}$ by -1, leading to the new $\boldsymbol{Q}$ and $\boldsymbol{R}$ factors as we desire. In the original BGBF algorithm, the bounds for all integer parameters are identical, i.e., $l_1 = \cdots = l_n$ and $u_1 = \cdots = u_n$. But we can easily extend the algorithm to deal with the more general constraint $\mathcal{B}$ given in (2). The following is the description of the BGBF algorithm (with slight modifications).

Algorithm BGBF

Input: The upper triangular matrix $\boldsymbol{R} \in \mathbb{R}^{n\times n}$ with positive diagonal entries, the vector $\bar{\boldsymbol{y}} \in \mathbb{R}^n$, the lower bound vector $\bar{\boldsymbol{l}} \in \mathbb{Z}^n$, the upper bound vector $\bar{\boldsymbol{u}} \in \mathbb{Z}^n$, the initial hyper-ellipsoid bound β.
Output: The solution $\hat{\boldsymbol{z}} \in \mathbb{Z}^n$ to the BILS problem (6).

1) (Initialization) Set $k := n$, $T_k := 0$
   Compute $\boldsymbol{H} := \boldsymbol{R}^{-1}$, $\boldsymbol{s}_k := \boldsymbol{H}\bar{\boldsymbol{y}}$,
   $z_k := \lfloor s_{kk} \rceil$, $z_k := \max(z_k, \bar{l}_k)$, $z_k := \min(z_k, \bar{u}_k)$,
   $\rho_k := (s_{kk} - z_k)/h_{kk}$, $\Delta_k := \text{sign}(\rho_k)$
2) Compute $T := T_k + \rho_k^2$
   if $T < \beta$ and $k \neq 1$, **then**
   Go to Step 3 // we are inside the ellipsoid
   else Go to Step 4 // we are not inside the ellipsoid
   end
3) **for** $i = 1 : k - 1$
   Compute $s_{i,k-1} := s_{ik} - \rho_k h_{ik}$
   // s_{ij} is the i-th entry of $\boldsymbol{s}_j$
   end
   Set $T_{k-1} := T$, $k := k - 1$
   Compute $z_k := \lfloor s_{kk} \rceil$,
   $z_k := \max(z_k, \bar{l}_k)$, $z_k := \min(z_k, \bar{u}_k)$,
   $\rho_k := (s_{kk} - z_k)/h_{kk}$, $\Delta_k := \text{sign}(\rho_k)$
   Go to Step 2
4) **if** $T < \beta$, **then** // $k = 1$
   Set $\beta := T$, $\hat{\boldsymbol{z}} := \boldsymbol{z}$ // a valid point is found
   else if $k = n$, **then**
   Terminate
   else $k := k + 1$
   end
5) (Enumeration at level k)
   Compute $z_k := z_k + \Delta_k$.
   if $z_k < \bar{l}_k$ **or** $z_k > \bar{u}_k$, **then**
   Compute $\Delta_k := -\Delta_k - \text{sign}(\Delta_k)$, $z_k := z_k + \Delta_k$
   end
   if $z_k < \bar{l}_k$ **or** $z_k > \bar{u}_k$, **then**
   Go to Step 4
   end
   Compute $\rho_k := (s_{kk} - z_k)/h_{kk}$,
   $\Delta_k := -\Delta_k - \text{sign}(\Delta_k)$
   Go to Step 2

Here we give a few comments on this algorithm. There is a minor inefficiency problem with Step 4. If $T < \beta$ is true, then from Step 2, we can conclude that k must be equal to 1, thus it should be increased to $k+1$ immediately after setting $\hat{\boldsymbol{z}} = \boldsymbol{z}$, since at level 1 the equality in (14) holds for $\hat{\boldsymbol{z}}$ and we can not find any other integer for z_1 to satisfy the inequality.

Note that Algorithm BGBF, a direct extension of the search algorithm presented in [1], works with $\boldsymbol{H}$, the inverse of $\boldsymbol{R}$, rather than $\boldsymbol{R}$ itself. It is not difficult to show that s_{kk} and ρ_k^2 in Algorithm BGBF are identical to c_k and $r_{kk}^2(z_k - c_k)^2$ in Algorithm DCE, respectively. But working with $\boldsymbol{R}$ as in Algorithm DEC is more nature and is easier to follow than working with $\boldsymbol{H}$. Also from the numerical stability point of view, $\boldsymbol{H}$ may have large rounding errors if $\boldsymbol{R}$ is ill conditioned. Thus it is better to work with $\boldsymbol{R}$.

In Step 1 and in Step 3, the last value of z_k may be the lower bound $\bar{l}_k$ or the upper bound $\bar{u}_k$. In either case, in Step 4 the algorithm enumerates some integers outside the interval $[\bar{l}_k, \bar{u}_k]$. For instance, when $\lfloor s_{kk} \rceil > \bar{u}_k$ in Step 1 or Step 3, the offset variable Δ_k is 1 since $\rho_k = (s_{kk} - \bar{u}_k)/h_{kk} > 0$. This means the next value of z_k to be enumerated is $z_k := z_k + 1 = \bar{u}_k + 1$, followed by $\bar{u}_k - 1, \bar{u}_k + 2, \ldots$. For efficiency, however, only the integers within the interval should be considered in enumeration, although this shortcoming does not seem serious.

III. New algorithms

We first present a new reduction algorithm which uses all available information in Section III-A, and then present a new search algorithm.

A. Reduction

The three strategies given in Section II-B use only the generator matrix $\boldsymbol{A}$ and do not use the input vector $\boldsymbol{y}$ and the box constraint. But the search speed appears to depend on all of the above information. In this section we propose a new reduction strategy which use all available information.

In the search process, at level i we have the inequality

$$r_{ii}^2(z_i - c_i)^2 < \beta - \sum_{k=i+1}^{n} r_{kk}^2(z_k - c_k)^2. \qquad (15)$$

When the right hand side of (15) is fixed, we require that $|r_{ii}|$ be as large as possible to reduce the search range of z_i for $i = n, n-1, \ldots, 1$. This gives some motivation for the V-BLAST reduction strategy introduced in Section II-A. However, on the other hand, to reduce the search range of z_i, we want the right hand side of (15) to be small, i.e., each $r_{kk}^2(z_k - c_k)^2$ should be large. This motivates us to propose a new reduction strategy. The basic idea is to determine the permutations of $\boldsymbol{A}$ such that $|r_{kk}(z_k - c_k)|$ is as large as possible for $k = n, n-1, \ldots 1$, and also simultaneously take the requirement that $|r_{kk}|$ should be as large as possible into account.

The question now is how to choose c_k and z_k in $|r_{kk}(z_k - c_k)|$ when we determine the permutations. First suppose c_k is known. It appears sensible to choose z_k to be the first nearest integer on the constrained interval $[\bar{l}_k, \bar{u}_k]$ to c_k, since the resulting point $\boldsymbol{z}$ will be the first integer point (i.e, the Babai integer point for the constrained case) to be obtained by a Schnorr and Euchner based search algorithm when the initial bound $\beta = \infty$. But this choice may cause a problem. Note that the first nearest integer can be very close to or even equal to c_k. Thus even if $|r_{kk}|$ is very large, $|r_{kk}(z_k - c_k)|$ can be very small or even zero, i.e., a column of $\boldsymbol{R}$ (or equivalently a column of $\boldsymbol{A}$) corresponding to a large $|r_{kk}|$ may not be chosen as the kth column, while a column corresponding to a small $|r_{kk}|$ may be chosen as the kth column. Since $|r_{11}r_{22}\cdots r_{nn}|$ is fixed (note that $|r_{11}r_{22}\cdots r_{nn}| = |\det(\boldsymbol{R})| = \det^{1/2}(\boldsymbol{A}^T\boldsymbol{A})$), the above choice could result small $|r_{ii}|$ for a large index i and large $|r_{ii}|$ for a small index i, not complying with our requirement (see the last sentence of the previous paragraph). Thus we propose to choose z_k to be the second nearest integer on $[\bar{l}_k, \bar{u}_k]$ to c_k to avoid the above problem. For this choice, $|z_k - c_k|$ is always larger than 0.5, so if $|r_{kk}|$ is large, $|r_{kk}(z_k - c_k)|$ is large too. On the other hand, $|z_k - c_k|$ is usually not very large for this choice, thus if $|r_{kk}(z_k - c_k)|$ is large, $|r_{kk}|$ is usually large too. In other words, the above choice comply with our requirement (again see the last sentence of the previous paragraph). Our simulations showed that the above choice for z_k is a little more effective than some other slightly different choices, e.g., taking z_k to be the first nearest integer on $[\bar{l}_k, \bar{u}_k]$ to c_k if it makes $|z_k - c_k| \geq 1$ or the second nearest integer if the first nearest integer makes $|z_k - c_k| < 1$. For c_k in $|r_{kk}(z_k - c_k)|$, we obtain it by using the formula given in (10), where each z_j is chosen to be the first nearest integer on $[\bar{l}_j, \bar{u}_j]$ to c_j for $j = n, n-1, \ldots, k+1$ — this is a choice used in a Schnorr and Euchner based search process if the initial bound $\beta = \infty$. In the following we will show how to efficiently compute the reduction.

We first compute the QR decomposition of $\boldsymbol{A}$ (with no permutations) by Householder transformations and compute $\bar{\boldsymbol{y}} = \boldsymbol{Q}_1^T\boldsymbol{y}$. Note that we do not need to explicitly form and store the Q-factor of the QR decomposition in our computations. Then we obtain the corresponding $|r_{nn}(z_n - c_n)|$, where $c_n = y_n/r_{nn}$ and z_n is the second nearest integer in $[\bar{l}_n, \bar{u}_n]$ to c_n. In order to determine the last column of the permuted $\boldsymbol{A}$ (or equivalently the last column of the permuted $\boldsymbol{R}$) we seek, we interchange the last column of $\boldsymbol{R}$ with its j-th column for $j = 1, 2, \ldots, n-1$. After each interchange, we compute the QR decomposition of the new $\boldsymbol{R}$. We now show how to do this in an efficient way. After we interchange the j-th column and the last column of $\boldsymbol{R}$, we obtain (for $n = 6$ and $j = 3$)

$$\begin{bmatrix} \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & \times & \times & \times & \\ & & \times & & \times & \\ & & \times & & & \end{bmatrix}.$$

Then we use $n-j$ Givens rotations $\boldsymbol{G}_{n-1,n}$, $\boldsymbol{G}_{n-2,n-1}$, …, $\boldsymbol{G}_{j,j+1}$ to eliminate the last $n-j$ elements in the j-th column of the permuted $\boldsymbol{R}$, where

$$\boldsymbol{G}_{i,i+1} = \begin{bmatrix} \boldsymbol{I}_{i-1} & & & \\ & c & s & \\ & -s & c & \\ & & & \boldsymbol{I}_{n-i-1} \end{bmatrix}, \quad c^2 + s^2 = 1.$$

However, these Givens rotations make the subdiagonal entries $r_{n,n-1}$, $r_{n-1,n-2}$, ..., $r_{j+2,j+1}$ nonzero, respectively. To make these subdiagonal entries zero, we apply $n-j-1$ Givens rotations $\boldsymbol{G}'_{j+1,j+2}$, $\boldsymbol{G}'_{j+2,j+3}$, ..., $\boldsymbol{G}'_{n-1,n}$, leading to an updated upper triangular $\boldsymbol{R}$. When we update $\boldsymbol{R}$ by a Givens rotation, we simultaneously update the vector $\bar{\boldsymbol{y}}$ by the same Givens rotation. Then we can obtain the corresponding $|r_{nn}(z_n - c_n)|$. If this quantity is larger than the largest $|r_{nn}(z_n - c_n)|$ obtained in the previous $j-1$ permutations, we keep the current $\boldsymbol{R}$ (and $\bar{\boldsymbol{y}}$ as well). Therefore, finally after $n-1$ permutations, an updated upper triangular $\boldsymbol{R}$ and an updated vector $\bar{\boldsymbol{y}}$ corresponding to the largest $|r_{nn}(z_n - c_n)|$ is obtained and the last column of the permuted $\boldsymbol{A}$ we seek is determined. We then obtain $\hat{z}_n$, the first nearest integer on the constrained interval $[\bar{l}_n, \bar{u}_n]$ to c_n, which will be used in later computations. Now we show how to determine the kth column of the permuted $\boldsymbol{A}$ we seek (for $k = n-1, n-2, \ldots, 2$). Based on the $\boldsymbol{R}$ we obtained in the previous step, we compute $|r_{k,k}(z_k - c_k)|$, where c_k (see (10)) can be computed in an efficient way (see later) and z_k is the second nearest integer on $[\bar{l}_k, \bar{u}_k]$ to c_k. Then we interchange columns j and k of $\boldsymbol{R}(1:k, 1:k)$ for $j = 1, 2, \cdots, k-1$. As before, after each interchange we apply Givens rotations to bring $\boldsymbol{R}(1:k, 1:k)$ back to an upper triangular matrix. In order to compute the new value of $|r_{k,k}(z_k - c_k)|$, we still need to obtain c_k and z_k (which is determined immediately after c_k is obtained). Note that the expression of c_k in (10) involves $\bar{y}_k$ and $r_{k,k+1}, \ldots, r_{kn}$, which could be obtained if we apply those Givens rotations (which bring $\boldsymbol{R}(1:k, 1:k)$ back to an upper triangular matrix) to the vector $\bar{\boldsymbol{y}}(1:k)$ and the matrix $\boldsymbol{R}(1:k, k+1:n)$. However, updating $\boldsymbol{R}(1:k, k+1:n)$ for each column permutation is expensive and unnecessary. A more efficient way is to apply those Givens rotations to the vector $\hat{\boldsymbol{y}}(1:k) \triangleq \bar{\boldsymbol{y}}(1:k) - \boldsymbol{R}(1:k, k+1:n)[\hat{z}_{k+1}, \hat{z}_{k+2}, \ldots, \hat{z}_n]^T$ (this vector can also be computed in an efficient way, see our later algorithm). Notice that the kth entry of the updated vector $\hat{\boldsymbol{y}}(1:k)$ is equal to $c_k r_{kk}$ (see (10) again), so that c_k can be obtained. If the value of $|r_{k,k}(z_k - c_k)|$ is larger than the largest obtained in the previous $j-1$ permutations, we keep the current $\boldsymbol{R}(1:k, 1:k)$, $\hat{\boldsymbol{y}}(1:k)$, Givens rotations and c_k. Finally after $k-1$ column permutations, we obtain an updated upper triangular $\boldsymbol{R}(1:k, 1:k)$ which corresponds to the largest $|r_{kk}(z_k - c_k)|$, and the kth column of the permuted $\boldsymbol{A}$ we seek is determined. At this point, we apply those corresponding Givens rotations to $\bar{\boldsymbol{y}}(1, k)$ and $\boldsymbol{R}(1:k, k+1:n)$ and find $\hat{z}_k$, the first nearest integer on $[\bar{l}_k, \bar{u}_k]$ to c_k. After this, we move to the next step to determine column $k-1$ of the permuted $\boldsymbol{A}$. Note that when the reduction process is finished, the Babai integer point $\hat{\boldsymbol{z}} = [\hat{z}_1, \hat{z}_2, \ldots, \hat{z}_n]^T$ is obtained as well.

Algorithm REDUCTION

Input: The generator matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with full column rank, the input vector $\boldsymbol{y} \in \mathbb{R}^m$, the lower bound vector $\boldsymbol{l} \in \mathbb{Z}^n$, and the upper bound vector $\boldsymbol{u} \in \mathbb{Z}^n$.

Output: The reduced upper triangular matrix $\boldsymbol{R} \in \mathbb{R}^{n \times n}$, the permutation matrix $\boldsymbol{P} \in \mathbb{Z}^{n \times n}$, the vector $\bar{\boldsymbol{y}} \in \mathbb{R}^n$, the permuted lower bound vector $\bar{\boldsymbol{l}}$, the permuted upper bound vector $\bar{\boldsymbol{u}}$, and the Babai integer point $\hat{\boldsymbol{z}} \in \mathbb{Z}^n$.

Compute $[\boldsymbol{Q}_1, \boldsymbol{Q}_2]^T \boldsymbol{A} = \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{bmatrix}$ by the Householder transformations and simultaneously compute $[\boldsymbol{Q}_1, \boldsymbol{Q}_2]^T \boldsymbol{y}$ and set $\bar{\boldsymbol{y}} := \boldsymbol{Q}_1^T \boldsymbol{y}$
Set $\boldsymbol{P} := \boldsymbol{I}_n$, $\bar{\boldsymbol{l}} := \boldsymbol{l}$, $\bar{\boldsymbol{u}} := \boldsymbol{u}$, $\hat{\boldsymbol{z}} = [\,]$, and $\hat{\boldsymbol{y}} := \bar{\boldsymbol{y}}$
for $k = n : -1 : 2$
  if $k < n$
    // for computing c_k and later c_j ($j < k$) use
    Compute $\hat{\boldsymbol{y}}(1:k) := \hat{\boldsymbol{y}}(1:k) - \boldsymbol{R}(1:k, k+1)\hat{z}_{k+1}$
  end
  Compute $c_k := \hat{\boldsymbol{y}}(k)/r_{kk}$
  Set z_k to be the second nearest integer on $[\bar{l}_k, \bar{u}_k]$ to c_k
  Compute $\alpha := |r_{kk}(z_k - c_k)|$ and set $p := k$
  Set $\boldsymbol{R}_{\text{tmp}} := \boldsymbol{R}(1:k, 1:k)$, $\boldsymbol{y}_{\text{tmp}} := \hat{\boldsymbol{y}}(1:k)$
  for $j = 1 : k-1$
    Set $\boldsymbol{R}' := \boldsymbol{R}_{\text{tmp}}$, $\boldsymbol{y}' := \boldsymbol{y}_{\text{tmp}}$ //temporary variables
    Interchange columns of j and k of $\boldsymbol{R}'$ and transform it to an upper triangular matrix by Givens rotations $\boldsymbol{G}_{k-1,k}, \ldots, \boldsymbol{G}_{j,j+1}$ and $\boldsymbol{G}'_{j+1,j+2}, \ldots, \boldsymbol{G}'_{k-1,k}$
    Compute $\boldsymbol{y}' := \boldsymbol{G}'_{k-1,k} \cdots \boldsymbol{G}'_{j+1,j+2} \boldsymbol{G}_{j,j+1} \cdots \boldsymbol{G}_{k-1,k} \boldsymbol{y}'$
    Compute $c'_k := \boldsymbol{y}'(k)/r'_{kk}$
    Set z'_k to be the 2nd nearest integer on $[\bar{l}_j, \bar{u}_j]$ to c'_k
    Compute $\alpha' = |r'_{kk}(z'_k - c'_k)|$
    if $\alpha' > \alpha$, **then**
      Set $\alpha := \alpha'$, $p := j$, $\boldsymbol{R}(1:k, 1:k) := \boldsymbol{R}'$, $\hat{\boldsymbol{y}}(1:k) := \boldsymbol{y}'$, $c_k := c'_k$
      Store the Givens rotations $\boldsymbol{G}_{k-1,k}, \ldots, \boldsymbol{G}_{j,j+1}$ and $\boldsymbol{G}'_{j+1,j+2}, \ldots, \boldsymbol{G}'_{k-1,k}$
    end
  end
  if $p \neq k$, **then**
    Interchange columns p and k of $\boldsymbol{P}$, entries p and k of $\bar{\boldsymbol{l}}$ and $\bar{\boldsymbol{u}}$
    Compute $\bar{\boldsymbol{y}}(1:k) := \boldsymbol{G}'_{k-1,k} \cdots \boldsymbol{G}'_{p+1,p+2} \times \boldsymbol{G}_{p,p+1} \cdots \boldsymbol{G}_{k-1,k} \bar{\boldsymbol{y}}(1:k)$
    Compute $\boldsymbol{R}(1:k, k+1:n) := \boldsymbol{G}'_{k-1,k} \cdots \boldsymbol{G}'_{p+1,p+2} \times \boldsymbol{G}_{p,p+1} \cdots \boldsymbol{G}_{k-1,k} \boldsymbol{R}(1:k, k+1:n)$
  end
  Set $\hat{z}_k$ to be the 1st nearest integer on $[\bar{l}_k, \bar{u}_k]$ to c_k and $\hat{\boldsymbol{z}} := [\hat{z}_k; \hat{\boldsymbol{z}}]$
end
Compute $\hat{\boldsymbol{y}}(1) := \hat{\boldsymbol{y}}(1) - \boldsymbol{R}(1,2)\hat{z}_2$
Compute $c_1 := \hat{\boldsymbol{y}}(1)/r_{11}$
Set $\hat{z}_1$ to be the first nearest integer on $[\bar{l}_1, \bar{u}_1]$ to c_1 and $\hat{\boldsymbol{z}} := [\hat{z}_1; \hat{\boldsymbol{z}}]$

Note that in practical implementation, we need only a vector rather than the matrix $\boldsymbol{P}$ to store the permutation information. Now we discuss the computational cost of Algorithm REDUCTION. Computing the QR decomposition of $\boldsymbol{A}$ and $\bar{\boldsymbol{y}}$ at the beginning requires about $2n^2(m - n/3)$ flops (see [19, p225]). In the outer for loop, for each k, the cost of the steps before the inner for loop is negligible, the cost of the inner for loop is about $2k^3$ flops (note that for each j, the cost is about $6(k-j)^2$

flops), the cost of the steps after the inner for loop is at most $12k(n-k)$ flops. Thus the total cost of the outer for loop is $\sum_{k=2}^{n-1}[2k^3+12k(n-k)] \approx n^4/2$ flops. The cost of the steps after the outer for loop is negligible. Therefore the total cost of the algorithm is about $2mn^2+n^4/2$ flops.

Note that it is easy to modify Algorithm REDUCTION to give an efficient implementation for the V-BLAST reduction strategy (see Section II-A). The resulting algorithm costs less than Algorithm REDUCTION since it does not need to compute c_k, but costs the same order of flops. Thus these two algorithms have almost the same efficiency and are less efficient than Algorithm SQRD (see Section II-A) which costs only $2mn^2$ flops. However, for solving a general BILS problem, the search process has exponential computational complexity and dominates the cost of the whole algorithm, and the efficiency of the search process rather than the efficiency of the reduction process is crucial for the efficiency of the whole algorithm. We will see our new reduction strategy makes the search process much more efficient than V-BLAST and SQRD. Having said that, we want to point out that one still wants to make the implementation of a reduction strategy efficient, since for some applications one just wants to find a Baba integer point, which does not need to do a search. If one just wants to find a Babai integer point, our new reduction strategy may still be more preferable than SQRD and V-BLAST, since our simulations indicated that the Babai integer point obtained by the new reduction strategy is usually much more closer to the ILS solution than the Babai integer point obtained by either SQRD or V-BLAST.

B. Search

In Section II-B, we introduced two Schnorr-Euchner strategy based search algorithms: Algorithm DEC and Algorithm BGBF. In this section, we will provide a new search algorithm which uses the advantages of those two algorithms, but avoids their drawbacks. To improve the search speed, the new search algorithm will also use a new bound which is as least as tight as that in (13) at each level k.

To avoid enumerating integers beyond the box constraint at each level, which can occur in Step 4 of Algorithm BGBF, we will introduce two flag variables *ubound*$_k$ and *lbound*$_k$ in our new search algorithm to indicate whether the enumeration has reached the lower bound and the upper bound of the box constraint at each level k, respectively. Using these two variables, our new algorithm will always enumerate the integers within the box constraint at each level.

The complexity of a search algorithm highly depends on the size of the search region. If the bound of the search region at each level is not tight enough, plenty of time might be wasted by moving up and down between different levels before we obtain the result eventually. In the following, we will make use of the box constraint to derive a new upper bound on $r_{kk}^2(z_k-c_k)^2$.

First we derive a lower bound d_k for each $(\bar{y}_k - \sum_{j=k}^n r_{kj}z_j)^2$ or $r_{kk}^2(z_k-c_k)^2$. Since $\bar{l}_k \le z_k \le \bar{u}_k$, we have

$$\min(r_{kj}\bar{l}_k, r_{kj}\bar{u}_k) \le r_{kj}z_j \le \max(r_{kj}\bar{l}_k, r_{kj}\bar{u}_k).$$

Then it follows that

$$\begin{aligned} &\bar{y}_k - \sum_{j=k}^n \max(r_{kj}\bar{l}_k, r_{kj}\bar{u}_k) \\ \le\ &\bar{y}_k - \sum_{j=k}^n r_{kj}z_j \le \bar{y}_k - \sum_{j=k}^n \min(r_{kj}\bar{l}_k, r_{kj}\bar{u}_k). \end{aligned} \tag{16}$$

If the lower bound and the upper bound in (16) have the same sign, i.e.,

$$\begin{aligned} &\text{sign}[\bar{y}_k - \sum_{j=k}^n \min(r_{kj}\bar{l}_k, r_{kj}\bar{u}_k)] \\ =\ &\text{sign}[\bar{y}_k - \sum_{j=k}^n \max(r_{kj}\bar{l}_k, r_{kj}\bar{u}_k)] \end{aligned} \tag{17}$$

then obviously we have

$$(\bar{y}_k - \sum_{j=k}^n r_{kj}z_j)^2 \ge d_k$$

where $d_k = \min\big\{[\bar{y}_k - \sum_{j=k}^n \min(r_{kj}\bar{l}_k, r_{kj}\bar{u}_k)]^2, [\bar{y}_k - \sum_{j=k}^n \max(r_{kj}\bar{l}_k, r_{kj}\bar{u}_k)]^2\big\}$. In other cases, we take $d_k = 0$. Now we derive an upper bound on $r_{kk}^2(z_k-c_k)^2$. As in Algorithm DEC in Section II-B, we denote

$$T_n = 0, \quad T_k = \sum_{i=k+1}^n r_{ii}^2(z_i-c_i)^2, \quad k = 1,\ldots,n-1.$$

Then we have

$$\begin{aligned} &\sum_{i=1}^n r_{ii}^2(z_i-c_i)^2 \\ =\ &\sum_{i=1}^{k-1}\Big(\bar{y}_i - \sum_{j=i}^n r_{ij}z_j\Big)^2 + r_{kk}^2(z_k-c_k)^2 + T_k \\ \ge\ &\sum_{i=1}^{k-1} d_i + r_{kk}^2(z_k-c_k)^2 + T_k. \end{aligned} \tag{18}$$

Define

$$\delta_1 = 0, \quad \delta_k = \sum_{i=1}^{k-1} d_i, \quad k = 2,\ldots,n.$$

Then if the bound (11) holds, i.e., the BILS solution is within the hyper-ellipsoid, we have from (18) that

$$r_{kk}^2(z_k-c_k)^2 < \beta - \delta_k - T_k. \tag{19}$$

Obviously the upper bound in (19) is at least as tight as that in (13). This new bound is used in the following new search algorithm.

Algorithm SEARCH

Input: The nonsingular upper triangular matrix $\boldsymbol{R} \in \mathbb{R}^{n\times n}$, the vector $\bar{\boldsymbol{y}} \in \mathbb{R}^n$, the lower bound vector $\bar{\boldsymbol{l}} \in \mathbb{Z}^n$, the upper bound vector $\bar{\boldsymbol{u}} \in \mathbb{Z}^n$, and the initial hyper-ellipsoid bound β.
Output: The solution $\hat{\boldsymbol{z}} \in \mathbb{Z}^n$ to the BILS problem (6).

1) (Initialization) Set $k := n$ and $T_k := 0$, compute δ_i for $i = 1,\ldots,n$
2) Compute $c_k := (\bar{y}_k - \sum_{j=k+1}^n r_{kj}z_j)/r_{kk}$, $z_k := \lfloor c_k \rceil$

Set $lbound_k := 0$ and $ubound_k := 0$
if $z_k \le \bar{l}_k$, **then**
Set $z_k := \bar{l}_k$, $lbound_k := 1$ and $\Delta_k := 1$
else if $z_k \ge \bar{u}_k$, **then**
Set $z_k := \bar{u}_k$, $ubound_k := 1$ and $\Delta_k := -1$
else // no boundary of the constraint is reached
Set $\Delta_k := \text{sign}(c_k - z_k)$
end

3) **if** $\delta_k + T_k + r_{kk}^2(z_k - c_k)^2 \ge \beta$, **then**
Go to Step 4 // we are not inside the ellipsoid
else if $k > 1$, **then**
Compute $T_{k-1} := T_k + r_{kk}^2(z_k - c_k)^2$,
Set $k := k - 1$, go to Step 2
else // $k = 1$ and a valid point is found
Compute $\beta := T_1 + r_{11}^2(z_1 - c_1)^2$,
Set $\hat{\boldsymbol{z}} := \boldsymbol{z}$ and $k := k + 1$, go to Step 5
end

4) **if** $k = n$, **then**
Terminate
else Set $k := k + 1$
end

5) (Enumeration of level k)
if $ubound_k = 1$ **and** $lbound_k = 1$, **then**
Go to Step 4 // no integer is available at this level
end
Set $z_k := z_k + \Delta_k$
if $z_k = \bar{l}_k$, **then**
Set $lbound_k := 1$
Compute $\Delta_k := -\Delta_k - \text{sign}(\Delta_k)$
else if $z_k = \bar{u}_k$, **then**
Set $ubound_k := 1$
Compute $\Delta_k := -\Delta_k - \text{sign}(\Delta_k)$
else if $lbound_k = 1$, **then**
Set $\Delta_k := 1$
else if $ubound_k = 1$, **then**
Set $\Delta_k := -1$
else Compute $\Delta_k := -\Delta_k - \text{sign}(\Delta_k)$
end
Go to Step 3

We give a remark about choosing the initial hyper-ellipsoid bound β. The first strategy, which is a typical one used in the literature, is to choose $\beta = \infty$. But since Algorithm REDUCTION generates the Babai integer point $\hat{\boldsymbol{z}}$, we can take $\beta = \|\bar{\boldsymbol{y}} - \boldsymbol{R}\hat{\boldsymbol{z}}\|_2^2$ instead. This second strategy makes Algorithm SEARCH a little faster than the first strategy. The third strategy is that we solve the box-constrained *real* LS problem

$$\min_{\boldsymbol{z}\in\mathbb{R}^n,\, \bar{\boldsymbol{l}}\le\boldsymbol{z}\le\bar{\boldsymbol{u}}} \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2$$

to obtain the solution $\hat{\boldsymbol{z}}_R \in \mathbb{R}^n$. Then we take $\beta = \|\bar{\boldsymbol{y}} - \boldsymbol{R}\lfloor\hat{\boldsymbol{z}}_R\rceil\|_2^2$. The box-constrained *real* LS problem can be solved by an active method (see, e.g., [20, Sec 5.2.4]) or the gradient projection method (see, e.g, [21, Sec 16.6]). All these strategies guarantee that the solution to (6) can be found in the search region, otherwise the Babai integer point $\hat{\boldsymbol{z}}$ is the solution for the second strategy and $\lfloor\hat{\boldsymbol{z}}_R\rceil$ is the solution for the third strategy. Our numerical simulation results indicate that for solving the reduced BILS problem (6) obtained by the V-BLAST reduction strategy or the SQRD strategy both Algorithm DEC and Algorithm DGBF are much faster by using the third strategy than by using the first strategy. However, we also found from our simulations that if the reduced BILS problem (6) is obtained by Algorithm REDUCTION, using the third strategy is more or less as effective as using the second strategy for any of the three search algorithms. The reason we found is that the residual $\|\bar{\boldsymbol{y}} - \boldsymbol{R}\lfloor\hat{\boldsymbol{z}}_R\rceil\|_2$ (which is independent of the reduction strategy) is more or less the same as the residual $\|\bar{\boldsymbol{y}} - \boldsymbol{R}\hat{\boldsymbol{z}}\|_2$, i.e., the size of the initial search region for the third strategy is more or less the same as that for the second strategy for our simulation examples.

IV. Simulations

In this section, we compare the performance of the proposed algorithms given in Section III with other existing algorithms given in Section II by computer simulations. All our computations were performed in MATLAB 6.5 on a sun4u sparc SUNW, Ultra-60 with 2048MB memory running SunOS 5.8.

A. Setup

In our simulations, the elements of the generator matrices $\boldsymbol{A}$ were drawn from an i.i.d. zero-mean, unit variance Gaussian distribution. Without lose of generality, we took $\boldsymbol{A}$ to be a square matrix. The input vector $\boldsymbol{y}$ was constructed as follows

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{v} \tag{20}$$

where each entry of $\boldsymbol{x}$ was generated according to a uniform distribution over a specific interval, and the elements of the noise vector $\boldsymbol{v}$ were drawn from an i.i.d zero-mean Gaussian distribution. Each interval of the box constraint was set to the same interval over which each entry of $\boldsymbol{x}$ was generated. Note that flat-fading multiple-input multiple-output (MIMO) systems in communications are usually assumed to have the same form as (20) and the assumptions about $\boldsymbol{A}$ and $\boldsymbol{v}$ are also the same, except that $\boldsymbol{A}$, $\boldsymbol{v}$ and $\boldsymbol{x}$ as well are complex, see e.g., [7] and [22]. Note that such a complex system can easily be transformed into a real system, so that our algorithms can be applied directly, so does the merit of our algorithms. We also tested other types of matrix $\boldsymbol{A}$ and got similar results.

B. Comparison of the reduction strategies

To show the effectiveness of Algorithm REDUCTION, we compared it with the V-BLAST strategy and the SQRD strategy. In our simulations, each entry of the integer vector $\boldsymbol{x}$ was generated uniformly over the interval $[0, 3]$ and the noise vector $\boldsymbol{v} \sim N(0, \sigma^2\boldsymbol{I})$ for $\sigma = 0.1, 1, 10$. We took dimension $n = 10, \ldots, 40$ and performed 200 runs for each case. Algorithm SEARCH was applied in the search process for all the three reduction strategies and the initial bound β was set to infinity. The results for the average search time (in seconds) are given in Fig. 1–Fig. 3, corresponding to the three different σ.

From the simulations results, we observe that Algorithm REDUCTION is (much) more effective than the V-BLAST

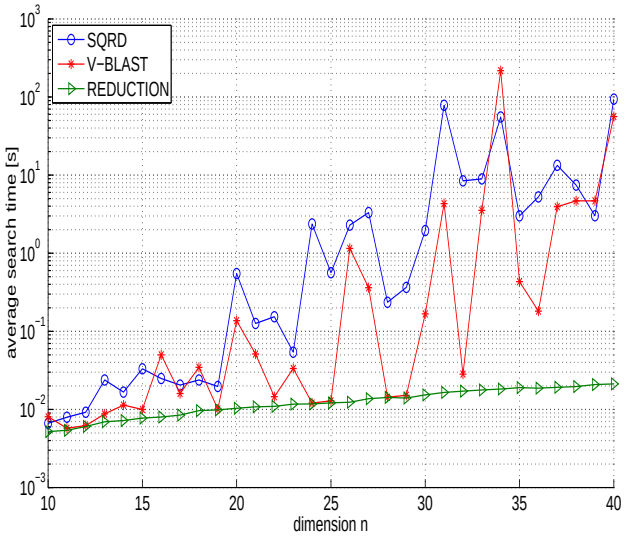

Fig. 1. Average search time vs dimension, $\boldsymbol{v} \sim N(\boldsymbol{0}, 0.1^2\boldsymbol{I})$.

Fig. 2. Average search time vs dimension, $\boldsymbol{v} \sim N(\boldsymbol{0}, \boldsymbol{I})$.

and SQRD strategies, while V-BLAST is a little more effectiv than SQRD. The advantage of Algorithm REDUCTION ove the other two becomes more significant as the dimension n increases and as the noise $\boldsymbol{v}$ increases (this usually means tha the LS residual becomes larger).

In Fig. 1 which corresponds a small noise, we observe tha when the dimension is small, the search process is very fas for all the three reduction strategies. But when the dimensio n increases, the search time for the V-BLAST strategy o the SQRD strategy can increase rapidly, while the searcl time for Algorithm REDUCTION increases very slowly. I is interesting to note that unlike the latter, the former may b very unstable when n increases.

Fig. 2 and Fig. 3 show that as the noise increases, the searcl time becomes more and more significant, and the effectivenes of Algorithm REDUCTION becomes increasingly evident. Fo example, for $n = 20$, when $\sigma = 1$, Algorithm REDUCTIO makes the search process 8 times faster than the V-BLAS strategy and the SQRD strategy, and when $\sigma = 10$ the improvement increases to more than 10 000 times.

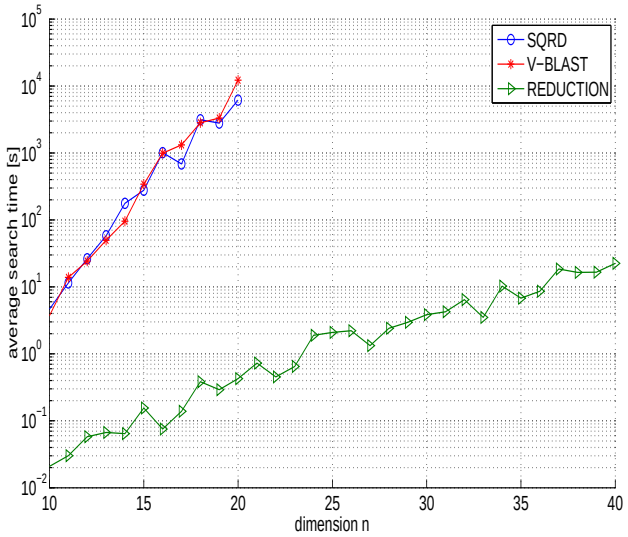

Fig. 3. Average search time vs dimension, $\boldsymbol{v} \sim N(\boldsymbol{0}, 10^2\boldsymbol{I})$.

C. Comparison of the search algorithms

In Fig. 4 and Fig. 5, we compare the average search time of three search algorithms: Algorithm DEC, Algorithm BGBF, and Algorithm SEARCH. In the simulations, each entry of the vector $\boldsymbol{x}$ was uniformly distributed on $[0, 3]$ and the noise vector $\boldsymbol{v} \sim N(0, \sigma^2\boldsymbol{I})$ for $\sigma = 0.1, 10$. We took dimension $n = 10, \ldots, 40$ and performed 200 runs for each case. For all the search algorithms, Algorithm REDUCTION was first used for reduction and the initial bound β was set to infinity.

From the results we observe that when the noise is small, there is no significant difference among the three search algorithms and all algorithms are relatively fast. An explanation is as follows: for Algorithm SEARCH, when the noise is small, the lower bound and the upper bound in (16) are likely to have different signs, i.e., (17) is not likley to hold, so that the new upper bound in (19) is likely to be equal to that in (13) which is used in Algorithms DEC and BGBF. In addition, when the noise is small, the Babai integer point is close to the optimal solution, which has a small LS residual. Thus the ellipsoid bound β defined by the Babai integer point is small, and it is unlikely that the enumeration of Algorithms DEC and BGBF goes far beyond the box constraint, i.e., the drawback of Algorithms DEC and BGBF (which is not serious in Algorithm BGBF in any case) we mentioned in Section II-B is rarely a problem. Therefore all the three search algorithms have more or less the same search speed. When the noise is large, we see Algorithm SEARCH is faster than Algorithm BGBF, which is faster than Algorithm DEC for all tested dimensions. For example, in Fig. 5, when $n = 40$, Algorithm DEC and Algorithm BGBF take about 134 and 43

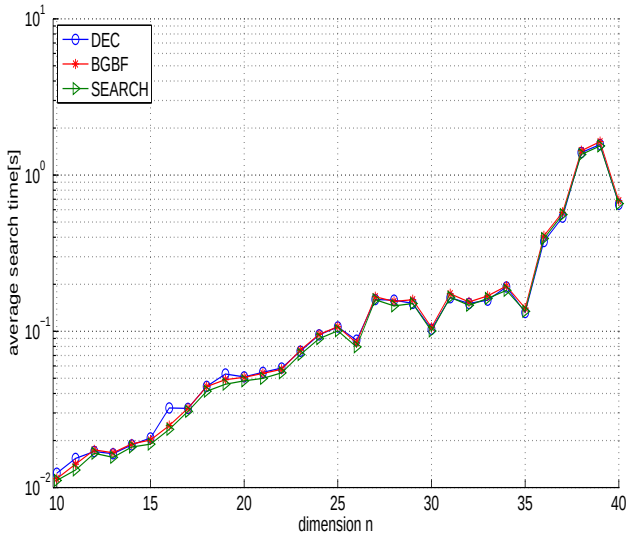

Fig. 4. Average search time vs dimension, $\boldsymbol{v} \sim N(\boldsymbol{0}, 0.5^2\boldsymbol{I})$.

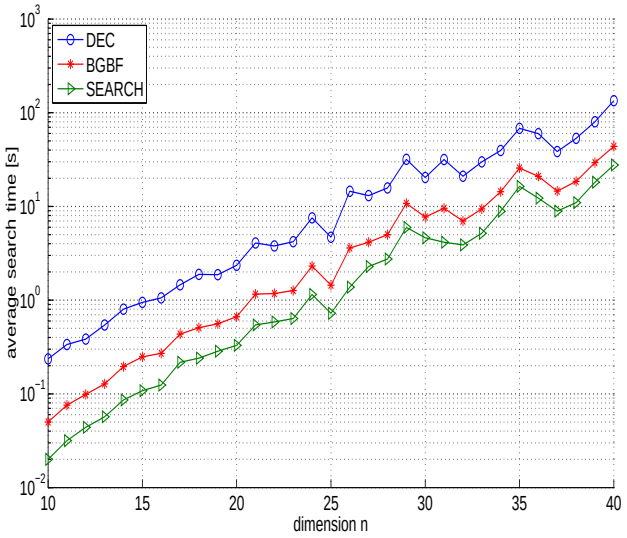

Fig. 5. Average search time vs dimension, $\boldsymbol{v} \sim N(\boldsymbol{0}, 10^2\boldsymbol{I})$.

seconds, respectively, while Algorithm SEARCH takes about 27 seconds. This phenomenon is what we expected in Section II-B.

D. Comparison of the Babai integer points

To compare the performance of the Babai integer points corresponding to Algorithm REDUCTION and the V-BLAST strategy, Fig. 6 gives the ratio between the β at the Babai integer point and the β at the optimal solution with different noises when the dimension $n = 8$ (note that $\beta = \beta(\boldsymbol{z}) = \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2$). In our simulations, each entry of $\boldsymbol{x}$ was generated uniformly on $[0, 3]$ and $[0, 63]$ separately, and the noise vector $\boldsymbol{v} \sim N(0, \sigma^2\boldsymbol{I})$ with $\sigma = 0.05 : 0.05 : 5$. For each case, 200 runs were performed. We see that the REDUCTION-Babai integer point is closer or much closer to the optimal solution

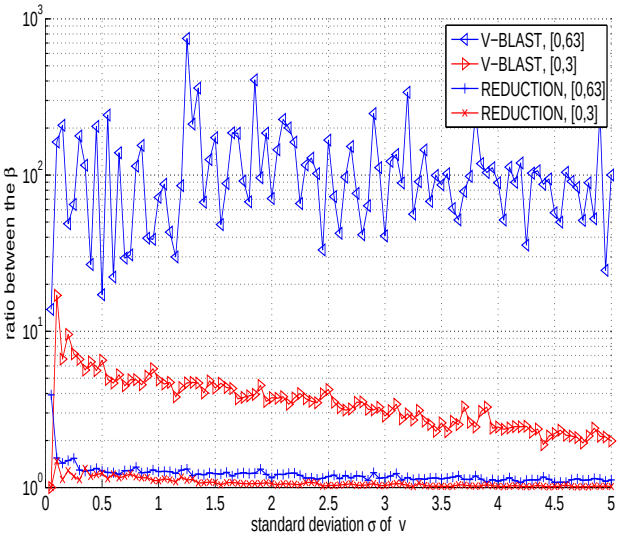

Fig. 6. Ratio between the β at the Babai integer point and the β at the ILS solution vs the standard deviation σ, $\boldsymbol{v} \sim N(\boldsymbol{0}, \sigma^2\boldsymbol{I})$, dimension $n = 8$.

for all cases, as compared to the V-BLAST-Babai integer point. We also see that for the two different box constraints, the V-BLAST-Babai integer point behaves very differently, while the REDUCTION-Babai integer point does not. For both reduction strategies (especially for V-BLAST), the smaller the box is, the better the Babai integer point performs.

V. Summary

We have proposed a new reduction algorithm and a new search algorithm for solving a general box-constrained integer least squares (BILS) problem. The reduction algorithm uses all available information of the BILS problem, i.e., the generator matrix, the input vector and the box constraint. Our simulations indicate it is much more effective than the V-BLAST and SQRD strategies, in particular when the LS residual is large. Theoretical research about the criterion to determine the permutations need to be further studied in the future. We also presented a new search algorithm to overcome some drawbacks of Algorithms DEC and BGBF. The new search algorithm also used the box constraint to reduce the search regions. Simulations showed that it performs better than Algorithms DEC and BGBF when the LS residual is large. It is our belief that all given information should be fully used to make a method efficient. Unlike many algorithms in the literature, we paid attentions to the efficient implementation of algorithms and took the numerical stability into account in designing algorithms. We hope a reader can implement our algorithms without difficulty.

Acknowledgment

We would like to thank Mohamed Oussama Damen and Ping Wang for helpful discussions with them and Martin Gander for his suggestions to improve the presentation. We are also grateful to the referees for their valuable suggestions and to one referee for providing some references.

REFERENCES

[1] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inform. Theory*, vol. 48, pp. 2201–2214, Aug. 2002.

[2] W.H. Mow, "Universal Lattice Decoding: Principle and Recent Advances", *Wireless Communications and Mobile Computing*, Special Issue on Coding and Its Applications in Wireless CDMA Systems, Vol.3, pp. 553-569, Aug. 2003.

[3] A. Hassibi and S. Boyd, "Integer parameter estimation in linear models with applications to GPS," *IEEE Trans. Signal Processing*, vol. 46, pp. 2938–2952, Nov. 1998.

[4] P. van Emde Boas, "Another NP-complete partition problem and the complexity of computing short vectors in a lattice," Mathematisch Institut, Amsterdam, The Netherlands, Tech. Rep. 81-04, Apr. 1981.

[5] M. O. Damen, H. El Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inform. Theory*, vol. 49, pp. 2389–2402, Oct. 2003.

[6] C. P. Schnorr and M. Euchner, "Lattice basis reduction: improved practical algorithms and solving subset sum problems," *Mathematical Programming*, vol. 66, pp. 181–199, 1994.

[7] J. Boutros, N. Gresset, L. Brunel, and M. Fossorier, "Soft-input soft-output lattice sphere decoder for linear channels," in *IEEE 2003 Global Communications Conference*, San Francisco, U.S.A., Dec. 2003.

[8] M. Pohst, "On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications," *ACM SIGSAM Bulletin*, vol. 15, pp. 37–44, Feb. 1981.

[9] E. Viterbo and E. Biglieri, "A universal lattice decoder," in *14-ème Colloque GRETSI*, Juan-Les-Pins, France, Sept. 1993.

[10] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1639–1642, July 1999.

[11] D. Wübben, R. Böhnke, J. Rinas, V. Kühn, and K. Kammeyer, "Efficient algorithm for decoding layered space-time codes," *IEEE Electronics Letters*, vol. 37, pp. 1348–1350, Oct. 2001.

[12] J.Jaldén and B. Ottersten, "On the Complexity of Sphere Decoding in Digital Communications," *IEEE Trans. on Signal Processing*,vol. 53, pp 1474-1484, April 2005.

[13] W.K. Ma, T.N. Davidson, K.M. Wong, Z.-Q. Luo, and P.C. Ching, "Quasi-maximum-likelihood multiuser detection using semi-definite relaxation," *IEEE Trans. Signal Processing*, vol. 50, pp. 912-922, April 2002.

[14] P. Tan, and L.K. Rasmussen, "The application of semidefinite programming for detection in CDMA," *IEEE Journal on Selected Areas in Communications*, vol. 19, pp. 1442–1449, Aug. 2001.

[15] M. Kisialiou and Z.-Q. Luo, "Performance Analysis of Quasi-Maximum-Likelihood Detector Based on Semi-Definite Programming," In Proc. of ICASSP, pp. 433-436, 2005.

[16] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Mathematics of Computation*, vol. 44, pp. 463–471, Apr. 1985.

[17] A. K. Lenstra, H. W. Lenstra, and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische Annalen*, vol. 261, pp. 515–534, 1982.

[18] G. J. Foscini, G. D. Golden, R. A. Valenzuela, and P. W. Wolniansky, "Simplified processing for high spectral efficiency wireless communication employing multi-element arrays," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 1841–1852, Nov. 1999.

[19] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore MD, third ed., 1996.

[20] Å. Björck, *Numerical Methods for Least Squares Problems*. Philadelphia: SIAM, 1996.

[21] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York: Springer-Verlag, 1999.

[22] M. O. Damen, A. Chkeif, J.-C. Belfiore, "Lattice Code Decoder for Space-Time Codes," *IEEE Commun. Lett.*, vol. 4, no.5, pp. 161–163, May 2000.

Xiao-Wen Chang received the B.S. and M.S. degrees in computational mathematics from Nanjing University, Nanjing, China, in 1986 and 1989, respectively, and the Ph.D. degree (Dean's Honor List) in computer science from McGill University, Montreal, Quebec, Canada, in 1997.

He is currently an Assocaite Professor of Computer Science at McGill University. His main research area is scientific computing with particular emphasis on matrix computations and applications. He has published over thirty papers in refereed journals. His current research interests include parameter estimation methods such as least squares, total least squares, integer least squares, robust estimation, and partially linear model estimation etc, and as well as their applications in communications, signal processing and the global navigation satellite systems.

Qing Han received the B.E. degree in Electrical Information Engineering in 2002 from Beijing University of Chemical Technology, Beijing, China, and received the M.S. degree (Dean's Honor List) in computer science in 2006 from McGill University, Montreal, Quebec, Canada. Her research interests are in the area of algorithm design, including algorithms for integer least squares estimation.

She is currently a vision developer in SolVision Inc., Boucherville, Quebec, Canada.