SIAM J. Matrix Anal. Appl.
Vol. 31, No. 3, pp. 1071–1089

SOLVING ELLIPSOID-CONSTRAINED INTEGER LEAST SQUARES PROBLEMS[*]

XIAO-WEN CHANG[†] AND GENE H. GOLUB[‡]

Abstract. A new method is proposed to solve an ellipsoid-constrained integer least squares (EILS) problem arising in communications. In this method, the LLL reduction, which is cast as a QRZ factorization of a matrix, is used to transform the original EILS problem to a reduced EILS problem, and then a search algorithm is proposed to solve the reduced EILS problem. Simulation results indicate the new method can be much more computationally efficient than the existing method. The method is extended to solve a more general EILS problem.

Key words. integer least squares, ellipsoidal constraint, LLL reduction, QRZ factorization, generalized QRZ factorization, search

AMS subject classifications. 15A23, 65F20, 68P30, 90C10

DOI. 10.1137/060660680

1. Introduction.

Let the set of all real $m \times n$ matrices be denoted by $\mathbb{R}^{m\times n}$ and the set of all real n-vectors by $\mathbb{R}^n$. Similarly $\mathbb{Z}^{m\times n}$ and $\mathbb{Z}^n$ denote the set of all integer $m \times n$ matrices and the set of all integer n-vectors, respectively.

Given a vector $\boldsymbol{y} \in \mathbb{R}^m$ and a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with full column rank, the minimization problem

$$\min_{\boldsymbol{x}\in\mathbb{Z}^n} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \tag{1.1}$$

is referred to 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 (1.1) is referred to as a *closest point problem* or a *closest vector problem*; and in channel coding, (1.1) is referred to as a *decoding problem*; see, e.g., [1]. The ILS problem may arise from several applications, such as communications, cryptograph, lattice design, Monte Carlo second-moment estimation, radar imaging, and global navigation satellite systems, etc.; see, e.g., [1], [12], [18] and references therein. It was shown in [19] that it is NP-hard. This means all known algorithms for solving (1.1) have exponential complexity.

A common approach to solving the ILS problem (1.1) and other constrained ILS problems (see later), which can be referred to as the enumeration approach, usually has two stages: reduction (or preprocessing) and search. The typical reduction strategy for solving (1.1) is the well-known Lenstra–Lenstra–Lovász (LLL) reduction [14]. An excellent survey on the search algorthms for solving (1.1) can be found in [1], which provides an efficient implementation of the Schnorr–Euchner search strategy [17].

In some communications applications, the integer vector $\boldsymbol{x}$ is constrained to a box and one wants to solve (see, e.g., [3] and [8])

$$\min_{\boldsymbol{x}\in\mathcal{B}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2, \quad \mathcal{B} = \{\boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u},\ \boldsymbol{l} \in \mathbb{Z}^n,\ \boldsymbol{u} \in \mathbb{Z}^n\}. \tag{1.2}$$

[*]Received by the editors May 24, 2006; accepted for publication (in revised form) by H. Park May 28, 2009; published electronically August 28, 2009.
http://www.siam.org/journals/simax/31-3/66068.html

[†]School of Computer Science, McGill University, Montreal, QC H3A 2A7, Canada (chang@cs.mcgill.ca). This author's work was supported by NSERC of Canada grant RGPIN217191-03.

[‡]This author is deceased. Former address: Department of Computer Science, Stanford University, Stanford, CA 94305-9025. This author's work was supported in part by NSF grant QAACT.

We refer to this problem as a *box-constrained integer least squares (BILS) problem.* In [3], the implementation of the Schnorr–Euchner search strategy given in [1] was directly extended to solve the BILS problem (1.2). In [8], two search algorithms based on the Phost search strategy (see [16], [20], and [21]) and the Schnorr–Euchner search strategy, respectively, were also proposed for solving (1.2). In addition, [8] introduced some reduction (or preprocessing) strategies. As in [1] for solving the ILS problem (1.1), it was found in [8] that the Schnorr–Euchner strategy is usually more efficient than the Phost strategy for solving the BILS problem (1.2).

In some communications applications, the integer vector $\boldsymbol{x}$ is subject to the following constraint (see, e.g., [8]):

$$\mathcal{E} = \{\boldsymbol{x} \in \mathbb{Z}^n : \|\boldsymbol{A}\boldsymbol{x}\|_2^2 \leq \alpha^2\}, \tag{1.3}$$

where α is a given constant. The constraint (1.3) is a hyperellipsoid and will be referred to as the constraint ellipsoid in this paper. Then the ILS problem becomes

$$\min_{\boldsymbol{x} \in \mathcal{E}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2. \tag{1.4}$$

We refer to (1.4) as an *ellipsoid-constrained integer least squares (EILS) problem.* It appears that [8] is the first paper to show how to solve this problem. It suggests modifying the methods for solving the BILS problem (1.2) presented in [8] to solve (1.4). Notice that (1.4) is a special case of the following more general EILS problem:

$$\min_{\boldsymbol{x} \in \mathcal{E}_g} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2, \quad \mathcal{E}_g = \{\boldsymbol{x} \in \mathbb{Z}^n : \|\boldsymbol{w} - \boldsymbol{B}\boldsymbol{x}\|_2^2 \leq \alpha^2\}, \tag{1.5}$$

where $\boldsymbol{B} \in \mathbb{R}^{p \times n}$ is a given matrix with full column rank and $\boldsymbol{w} \in \mathbb{R}^p$ is a given vector.

The main goal of this paper is to present a faster method to solve the EILS problem (1.4). Specifically, we will propose to use the LLL reduction for preprocessing, which can be much more effective than the reduction strategies given in [8], and propose a search algorithm to deal with the ellipsoidal constraint (1.3). The combination of the reduction algorithm and the search algorithm offers a significant decrease in computational cost compared to the method given in [8]. The method will then be extended to solve the more general EILS problem (1.5).

ILS problems can be regarded as special cases of quadratic integer programming or nonlinear integer programming, and thus relevant methods for some integer programming problems can be applied to solve ILS problems. For example, we can use the well-known TOMLAB/CPLEX package or TOMLAB/MINLP package to solve the EILS problem (1.4). However, our preliminary numerical experiments indicated that our new method implemented in MATLAB is faster than TOMLAB/CPLEX and much faster than TOMLAB/MINLP. It would be interesting to give comprehensive comparisons of the enumeration approach for ILS problems and the existing approaches for integer programming problems which can be applied to ILS problems. But this is beyond the scope of this paper, and we leave this for a future investigation.

The rest of the paper is organized as follows. In section 2, we introduce the LLL reduction, our implementation of the LLL reduction, the so-called V-BLAST reduction strategy (see [8] and [10]), and the Schnorr–Euchner strategy based search algorithm given in [8] for solving the BILS problem (1.2). In section 3, we review the method described in [8] for solving the EILS problem (1.4). In section 4, for solving (1.4), we first propose to use the LLL reduction strategy to do the reduction and then present a new search algorithm. Then this method is extended to solve the more general EILS problem (1.5) in section 5. Section 6 gives simulation results to show the advantages of our new method. Finally, a summary is given in section 7.

Before ending this section, we introduce some notation to be used in this paper. Bold uppercase letters and bold lowercase letters are used to denote matrices and vectors, respectively. The kth column of the identity matrix $\boldsymbol{I}$ is denoted by $\boldsymbol{e}_k$. MATLAB notation is used to denote a submatrix. For a real vector $\boldsymbol{a} = (a_i) \in \mathbb{R}^n$, $\lfloor \boldsymbol{a} \rceil$ denotes the vector whose ith entry is the nearest integer to a_i (if there is a tie, we choose the one with smaller magnitude) for $i = 1 : n$. Similarly, $\lfloor \boldsymbol{a} \rfloor$ and $\lceil \boldsymbol{a} \rceil$ denote the vectors whose ith entries are $\lfloor a_i \rfloor$ and $\lceil a_i \rceil$, respecively, for $i = 1 : n$. Operation $\text{sign}(z)$ returns -1 if $z \leq 0$ and 1 if $z > 0$. For a random vector $\boldsymbol{v}$ which is normally distributed with mean $\boldsymbol{u}$ and covariance $\boldsymbol{\Sigma}$, we denote it by $\boldsymbol{v} \sim N(\boldsymbol{u}, \boldsymbol{\Sigma})$.

2. Preliminary.

In this section we present the ideas for solving the ILS problem (1.1) and the BILS problem (1.2). These ideas will be used for developing algorithms to solve the EILS problem (1.4). In section 2.1, we introduce the well-known LLL reduction presented in [14] and a reduction strategy proposed in [8]. In section 2.2, we introduce the Schnorr–Euchner strategy based search algorithms presented in [8] for solving the BILS problem.

2.1. Reduction.

A reduction process transforms a given ILS problem into a new ILS problem, and its essential part is to transform the matrix $\boldsymbol{A}$ into an upper triangular matrix, which has good properties to make the later search process efficient. For solving the ordinary ILS problem (1.1), we can describe the transformations on $\boldsymbol{A}$ as the following matrix factorization:

$$\boldsymbol{Q}^T \boldsymbol{A}\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{bmatrix} \quad \text{or} \quad \boldsymbol{A} = \boldsymbol{Q}_1^T \boldsymbol{R}\boldsymbol{Z}^{-1}, \tag{2.1}$$

where $\boldsymbol{Q} = [\underset{n}{\boldsymbol{Q}_1}, \underset{m-n}{\boldsymbol{Q}_2}\,] \in \mathbb{R}^{m \times m}$ is orthogonal, $\boldsymbol{R} \in \mathbb{R}^{n \times n}$ is nonsingular upper triangular, and $\boldsymbol{Z} \in \mathbb{Z}^{n \times n}$ is unimodular (i.e., $\boldsymbol{Z}$ is an integer matrix with $|\det(\boldsymbol{Z})| = 1$, so $\boldsymbol{Z}^{-1}$ is an integer matrix). We refer to (2.1) as a *QRZ factorization.* Without loss of generality, we assume that the diagonal entries of $\boldsymbol{R}$ are positive in this paper. Then, with (2.1), we have

$$\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 = \|\boldsymbol{Q}_1^T \boldsymbol{y} - \boldsymbol{R}\boldsymbol{Z}^{-1}\boldsymbol{x}\|_2^2 + \|\boldsymbol{Q}_2^T \boldsymbol{y}\|_2^2. \tag{2.2}$$

Defining

$$\bar{\boldsymbol{y}} \triangleq \boldsymbol{Q}_1^T \boldsymbol{y}, \qquad \boldsymbol{z} \triangleq \boldsymbol{Z}^{-1}\boldsymbol{x}, \tag{2.3}$$

we see that the ILS problem (1.1) is reduced to

$$\min_{\boldsymbol{z} \in \mathbb{Z}^n} \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2. \tag{2.4}$$

If $\hat{\boldsymbol{z}}$ is the solution to the above reduced ILS problem, then $\hat{\boldsymbol{x}} = \boldsymbol{Z}\hat{\boldsymbol{z}}$ is the solution to the original ILS problem (1.1).

The typical reduction for solving the ILS problem (1.1) is the LLL reduction (see [14] and [13]). In the LLL reduction, the upper triangular matrix $\boldsymbol{R}$ satisfies the following criteria:

$$|r_{k-1,j}| \leq \frac{1}{2} r_{k-1,k-1}, \quad r_{k-1,k-1} \leq \delta \sqrt{r_{k-1,k}^2 + r_{kk}^2}, \qquad j = k : n, \quad k = 2 : n, \tag{2.5}$$

where δ is a parameter satisfying $1 \leq \delta < 2$. In this paper, we take $\delta = 1$ in (2.5). Then from (2.5) we can easily obtain the following relation among the diagonal entries of $\boldsymbol{R}$:

$$r_{k-1,k-1} \leq \frac{2}{\sqrt{3}} r_{kk}, \qquad k = 2 : n.$$

These properties of $\boldsymbol{R}$ can make a typical search process more efficient; see [1] and [15]. The original LLL reduction was not described in terms of matrices. Casting the LLL reduction as a matrix factorization (2.1) explicitly will help in designing a good algorithm to compute it.

In the original LLL reduction algorithm presented in [14], the Gram-Schmidt orthogonalization was used. But, for numerical stability, we will use Householder transformations and Givens rotations (see, e.g., [2, section 2.4] and [11, Chapter 5]). In our implementation, two types of unimodular matrices are used. One is of integer Gauss matrices, which are used to pursue the first criterion in (2.5); and the other is of permutation matrices, which are used to pursue the second criterion in (2.5). In the following, we introduce these two types of unimodular matrices and the corresponding operations when they are applied to the R factor of the QR factorization of $\boldsymbol{A}$.

Integer Gauss transformations (IGTs). An integer matrix is called an IGT if it has the following form:

$$\boldsymbol{Z}_{ij} = \boldsymbol{I} - \zeta_{ij} \boldsymbol{e}_i \boldsymbol{e}_j^T, \qquad i \neq j, \quad \zeta_{ij} \text{ is an integer.}$$

It is easy to show that $\boldsymbol{Z}_{ij}$ is unimodular and $\boldsymbol{Z}_{ij}^{-1} = \boldsymbol{I} + \zeta_{ij} \boldsymbol{e}_i \boldsymbol{e}_j^T$. If we apply $\boldsymbol{Z}_{ij}$ $(i < j)$ to $\boldsymbol{R}$ from the right, we have

$$\bar{\boldsymbol{R}} \triangleq \boldsymbol{R} \boldsymbol{Z}_{ij} = \boldsymbol{R} - \zeta_{ij} \boldsymbol{R} \boldsymbol{e}_i \boldsymbol{e}_j^T.$$

Thus $\bar{\boldsymbol{R}}$ is the same as $\boldsymbol{R}$, except that $\bar{r}_{kj} = r_{kj} - \zeta_{ij} r_{ki}$ for $k = 1 : i$. To meet the first criterion in (2.5), we take $\zeta_{ij} = \lfloor r_{ij}/r_{ii} \rceil$, ensuring $|\bar{r}_{ij}| \leq \frac{1}{2}|\bar{r}_{ii}|$. Note that if we did not have the requirement that ζ_{ij} is an integer number, we could take $\zeta_{ij} = r_{ij}/r_{ii}$ to make $\bar{r}_{ij} = 0$—this is what Gaussian elimination does (see, e.g., [11, section 3.2]).

Permutations. In order to pursue the second criterion in (2.5), permutations are needed in the reduction process. Suppose in the upper triangular $\boldsymbol{R}$,

$$r_{k-1,k-1} > \sqrt{r_{k-1,k}^2 + r_{kk}^2}; \tag{2.6}$$

then we interchange columns $k-1$ and k of $\boldsymbol{R}$. The upper triangular structure is no longer maintained, but we can apply a Givens rotation to $\boldsymbol{R}$ from the left to bring $\boldsymbol{R}$ back to an upper triangular form again. After the Givens rotation, the second inequality in (2.5) with $\delta = 1$ holds.

Our implementation of the LLL reduction first computes the QR factorization of $\boldsymbol{A}$ by Householder transformations and then works on the upper triangular $\boldsymbol{R}$ from left to right. For the kth column of $\boldsymbol{R}$, the algorithm first uses IGTs to make $|r_{i,k}| \leq r_{i,i}/2$ for $i = k-1 : -1 : 1$ (cf. the first inequality in (2.5)) and then checks if it is necessary to permute columns $k-1$ and k according to the criterion (2.6). If (2.6) is satisfied, it performs the permutation, applies the corresponding Givens rotation, and moves back to column $k-1$; otherwise it moves to column $k+1$. The description of our implementation of the LLL reduction is as follows.

```
Algorithm LLL
Input: The full column rank matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and the vector $\boldsymbol{y} \in \mathbb{R}^m$.
Output: The reduced upper triangular matrix $\boldsymbol{R} \in \mathbb{R}^{n\times n}$, the unimodular matrix
$\boldsymbol{Z} \in \mathbb{Z}^{n\times n}$, and the vector $\bar{\boldsymbol{y}} \in \mathbb{R}^n$.
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 $\boldsymbol{Z} := \boldsymbol{I}_n$, $k = 2$
while $k \le n$
    for $i = k-1 : -1 : 1$
        apply the IGT $\boldsymbol{Z}_{ik}$ to $\boldsymbol{R}$ from the right such that $|r_{ik}| \le r_{ii}/2$,
        update $\boldsymbol{Z}$, i.e., $\boldsymbol{Z} := \boldsymbol{Z}\boldsymbol{Z}_{ik}$
    end
    if $r_{k-1,k-1} > \sqrt{r_{k-1,k}^2 + r_{k,k}^2}$, then
        interchange columns $k-1$ and $k$ of $\boldsymbol{R}$ and transform $\boldsymbol{R}$ to
            an upper triangular matrix by a Givens rotation,
        interchange columns of $k-1$ and $k$ of $\boldsymbol{Z}$,
        apply the same Givens rotation to $\bar{\boldsymbol{y}}$
        if $k > 2$, then
            $k = k-1$
        end
    else
        $k = k+1$
    end
end
```

For the BILS problem (1.2), using the LLL reduction would bring difficulties in handling the box constraint (notice that $\boldsymbol{z}$ in (2.3) would be subject to a non-box constraint). But if the unimodular matrix $\boldsymbol{Z}$ in the QRZ factorization (2.1) is a permutation matrix, i.e., the QRZ factorization is actually a QR factorization with column pivoting (or column reordering), then there is not any difficulty in applying the factorization. Define

$$\bar{\boldsymbol{l}} \triangleq \boldsymbol{Z}^T \boldsymbol{l}, \quad \bar{\boldsymbol{u}} \triangleq \boldsymbol{Z}^T \boldsymbol{u}. \tag{2.7}$$

Then, using (2.1) (where $\boldsymbol{Z}$ is now a permutation matrix, so $\boldsymbol{Z}^{-1} = \boldsymbol{Z}^T$), we see that with (2.3) and (2.7) the original BILS problem (1.2) is equivalent to the reduced one:

$$\min_{\boldsymbol{z}\in\bar{\mathcal{B}}} \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2, \quad \bar{\mathcal{B}} = \{\boldsymbol{z} \in \mathbb{Z}^n : \bar{\boldsymbol{l}} \le \boldsymbol{z} \le \bar{\boldsymbol{u}},\ \bar{\boldsymbol{l}} \in \mathbb{Z}^n,\ \bar{\boldsymbol{u}} \in \mathbb{Z}^n\}. \tag{2.8}$$

In [8], the so-called vertical Bell Labs layered space-time (V-BLAST) optical detection ordering given in [10] was proposed for column reordering, and it is more effective than other column reordering strategies given in [8]. This 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 indexes for the not-yet-chosen columns when the kth column of the permuted $\boldsymbol{A}$ is to be determined ($k = n\!:\!-1\!:\!1$). This strategy chooses the $p(k)$th column of the original matrix $\boldsymbol{A}$ as the kth column of the permuted matrix $\boldsymbol{A}$ we seek:

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

where $\boldsymbol{a}_j$ is the jth column of $\boldsymbol{A}$, and $\boldsymbol{A}_{k,j}$ is the $m \times (k-1)$ matrix formed by the columns $\boldsymbol{a}_i$ with $i \in \mathcal{J}_k - \{j\}$. It is easy to see that $\boldsymbol{a}_j^T[\boldsymbol{I} - \boldsymbol{A}_{k,j}(\boldsymbol{A}_{k,j}^T\boldsymbol{A}_{k,j})^{-1}\boldsymbol{A}_{k,j}^T]\boldsymbol{a}_j$ is the squared Euclidean distance from $\boldsymbol{a}_j$ to the space spanned by the columns of $\boldsymbol{A}_{k,j}$. For an efficient algorithm to compute the QR factorization with this pivoting strategy, see [6].

2.2. Search. We first introduce the ideas of the Schnorr–Euchner enumerating strategy based search algorithm for the ILS problem (1.1), which leads to the search algorithm given in [8] for the BILS problem by some modifications.

Suppose the solution of (2.4) satisfies the bound

$$\|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2 < \beta^2 \tag{2.9}$$

or equivalently

$$\sum_{k=1}^{n}\Big(\bar{y}_k - \sum_{j=k+1}^{n} r_{kj}z_j - r_{kk}z_k\Big)^2 < \beta^2 \tag{2.10}$$

for some β. This is also a hyperellipsoid. The search process is to seek the integer point within this hyperellipsoid which makes the left-hand side of (2.9) smallest. We refer to this hyperellipsoid as the search ellipsoid to distinguish it from the constraint ellipsoid.

Define

$$b_n \triangleq 0, \qquad b_k \triangleq \sum_{j=k+1}^{n} r_{kj}z_j, \quad k = n-1:-1:1, \tag{2.11}$$

$$c_k \triangleq (\bar{y}_k - b_k)/r_{kk}, \quad k = n:-1:1, \tag{2.12}$$

$$t_n \triangleq 0, \quad t_{k-1} \triangleq \sum_{i=k}^{n} r_{ii}^2(z_i - c_i)^2 = t_k + r_{kk}^2(z_k - c_k)^2, \quad k = n:-1:2.$$

Note that b_k and c_k depend on $z_{k+1}, z_{k+2}, \ldots, z_n$. Then (2.10) can be rewritten as

$$\sum_{k=1}^{n} r_{kk}^2(z_k - c_k)^2 < \beta^2, \tag{2.13}$$

which is equivalent to a set of inequalities:

$$\text{level } n:\ r_{nn}^2(z_n - c_n)^2 < \beta^2 = \beta^2 - t_n, \tag{2.14}$$

$$\vdots$$

$$\text{level } k:\ r_{k,k}^2(z_k - c_k)^2 < \beta^2 - t_k, \tag{2.15}$$

$$\vdots$$

$$\text{level } 1:\ r_{11}^2(z_1 - c_1)^2 < \beta^2 - t_1. \tag{2.16}$$

Based on the above inequalities, a search process using the Schnorr–Euchner enumeration strategy can start. First, at level n, choose $z_n = \lfloor c_n \rceil$. If it does not satisfy the inequality (2.14), no other integer will satisfy the inequality, so the optimal solution of (1.1) is outside the search ellipsoid and we have to increase the value of β. If it satisfies the inequality, we proceed to level $n-1$. At this level, we compute c_{n-1} by (2.12) and choose $z_{n-1} = \lfloor c_{n-1} \rceil$. If z_{n-1} does not satisfy the inequality (2.15) 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 $n-2$. Continue this procedure until we reach level 1 and obtain a valid integer for z_1. At this moment, we obtain the first integer point denoted by $\hat{\boldsymbol{z}} = [\hat{z}_1, \ldots, \hat{z}_n]^T$ within the search ellipsoid. Then we update the search ellipsoid by taking $\beta^2 = \|\bar{\boldsymbol{y}} - \boldsymbol{R}\hat{\boldsymbol{z}}\|_2^2$ and search for a better integer point within the new search ellipsoid. The idea is to try to update $\hat{\boldsymbol{z}}$. At level 1, we cannot find any interger z_1 to satisfy the inequality in (2.16) (note that β has been updated). So 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 (note that the most up-to-date nearest integer to c_2 is $\hat{z}_2$). 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 (2.14), the search process stops and the latest found integer point is the optimal solution we seek. Notice that during the search process the search ellipsoid shrinks each time when a new integer point is found. This is crucial to the efficiency of the search process. For the initial bound β, a common strategy is to set it to be infinity.

To solve the BILS problem (1.2), the box constraint in (2.8) has to be considered during the search process. The following search algorithm presented in Damen, El Gamal, and Caire (DEC) [8] (with minor changes) shows how the modification can be done to take the box constraint into account.

$\boxed{\text{ALGORITHM DEC-BILS}}$

Input: The nonsingular 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 $\bar{\boldsymbol{u}} \in \mathbb{Z}^n$, and the initial search ellipsoid bound β.
Output: The solution $\hat{\boldsymbol{z}} \in \mathbb{Z}^n$ to the BILS problem (2.8).

```
1. (Initialization) Set $k := n$, $b_k := 0$, and $t_k := 0$
2. Compute $b_k := \sum_{j=k+1}^{n} r_{kj} z_j$ if $k < n$, $c_k := (\bar{y}_k - b_k)/r_{kk}$, $z_k := \lfloor c_k \rceil$,
   $\Delta_k := \text{sign}(c_k - z_k)$
3. **if** $t_k + r_{kk}^2(z_k - c_k)^2 \ge \beta^2$, **then**
         go to step 4    // we are not inside the search ellipsoid
   **else if** $z_k \notin [\bar{l}_k, \bar{u}_k]$, **then**
         go to step 6    // we are inside the search 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 $\beta^2 := t_1 + r_{11}^2(z_1 - c_1)^2$, 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.
```

The above algorithm has a shortcoming. Suppose that in step 3 the inequality $t_k + r_{kk}^2(z_k - c_k)^2 < \beta^2$ holds; then $z_k \in [\lceil c_k - \sqrt{\beta^2 - t_k}/r_{kk} \rceil, \lfloor c_k + \sqrt{\beta^2 - t_k}/r_{kk} \rfloor]$. If this interval is large (e.g., when β is large) but there is a 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.

3. Overview of the method of Damen, El Gamal, and Caire for solving (1.4).

In this section we mainly introduce the method of Damen, El Gamal, and Caire outlined in [8] for solving the EILS problem (1.4). For convenience, their method is referred to as the DEC method.

The idea of the DEC method is to extend the method for solving the BILS problem (1.2) given in [8] to the EILS problem (1.4). It first restricts $\boldsymbol{x}$ to a box:

$$\mathcal{X} = \{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{\mu} \le \boldsymbol{x} \le \boldsymbol{\nu}\}, \tag{3.1}$$

which is the smallest hyperrectangle including the constrained ellipsoid region and whose edges are parallel to the axes of the coordinate system. Then the following BILS problem can be formed:

$$\min_{\boldsymbol{x} \in \mathcal{B}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2, \tag{3.2}$$

where

$$\mathcal{B} = \{\boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}\}, \quad \boldsymbol{l} = \lceil \boldsymbol{\mu} \rceil, \quad \boldsymbol{u} = \lfloor \boldsymbol{\nu} \rfloor. \tag{3.3}$$

This BILS problem can be solved using the method presented in [8] (see the previous section). However, (3.2) is different from the EILS problem (1.4), since there might be some $\boldsymbol{x} \in \mathcal{B}$ but $\|\boldsymbol{A}\boldsymbol{x}\|_2^2 > \alpha^2$. Damen, El Gamal, and Caire [8] suggested that Algorithm DEC-BILS be modified to handle this problem (after the reduction process). The main idea is that when an integer point $\hat{\boldsymbol{x}}$ is found, the algorithm tests if $\|\boldsymbol{A}\hat{\boldsymbol{x}}\|_2^2 \le \alpha^2$. If yes, it stores $\hat{\boldsymbol{x}}$, updates the ellipsoid bound β^2, and then searches for the next integer point. Otherwise it tries to find a new integer point without updating β^2. Later, we will give the details of the modification, which were not given in [8].

The paper [8] did not say how to find $\mathcal{B}$ in (3.3). Here we present an efficient method to find $\bar{\mathcal{B}}$, which is transformed from $\mathcal{B}$ after the reduction process. From $\bar{\mathcal{B}}$ we could obtain $\mathcal{B}$, but this is not necessary. The DEC method for solving the EILS problem (1.4) implies that the reduction process should be realized by the QRZ factorization (2.1), where $\boldsymbol{Z}$ is a permutation matrix, such as that obtained by the V-BLAST pivoting strategy (see section 2.1). Thus, with (2.1), (2.2), and (2.3), the original EILS problem (1.4) is transformed to

$$\min_{\bar{\mathcal{E}}} \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2, \quad \bar{\mathcal{E}} = \{\boldsymbol{z} \in \mathbb{Z}^n : \|\boldsymbol{R}\boldsymbol{z}\|_2^2 \le \alpha^2\}, \tag{3.4}$$

and the box constraint (3.3) is transformed to (cf. (2.7) and (2.8))

$$\bar{\mathcal{B}} = \{\boldsymbol{z} \in \mathbb{Z}^n : \bar{\boldsymbol{l}} \le \boldsymbol{z} \le \bar{\boldsymbol{u}}\}, \tag{3.5}$$

where with $\bar{\boldsymbol{\mu}} \triangleq \boldsymbol{Z}^T\boldsymbol{\mu}$ and $\bar{\boldsymbol{\nu}} \triangleq \boldsymbol{Z}^T\boldsymbol{\nu}$,

$$\bar{\boldsymbol{l}} \triangleq \boldsymbol{Z}^T\boldsymbol{l} = \boldsymbol{Z}^T\lceil\boldsymbol{\mu}\rceil = \lceil\boldsymbol{Z}^T\boldsymbol{\mu}\rceil = \lceil\bar{\boldsymbol{\mu}}\rceil, \quad \bar{\boldsymbol{u}} \triangleq \boldsymbol{Z}^T\boldsymbol{u} = \boldsymbol{Z}^T\lfloor\boldsymbol{\nu}\rfloor = \lfloor\boldsymbol{Z}^T\boldsymbol{\nu}\rfloor = \lfloor\bar{\boldsymbol{\nu}}\rfloor. \tag{3.6}$$

Obviously $\bar{\mathcal{X}} \triangleq \{\boldsymbol{z} \in \mathbb{R}^n : \bar{\boldsymbol{\mu}} \le \boldsymbol{z} \le \bar{\boldsymbol{\nu}}\}$, which is transformed from $\mathcal{X}$ in (3.1) by the permutation matrix $\boldsymbol{Z}^T$, is the smallest hyperrectangle including the constraint ellipsoid $\|\boldsymbol{Rz}\|_2 \le \alpha$, with edges parallel to the axes of the $\boldsymbol{z}$-coordinate system (note that $\boldsymbol{Z}$ is a permutation matrix). Therefore, for $k = 1 : n$,

$$\bar{\mu}_k = \min_{\boldsymbol{z}\in\mathbb{R}^n} \boldsymbol{e}_k^T\boldsymbol{z} \quad \text{subject to } \|\boldsymbol{Rz}\|_2 \le \alpha, \tag{3.7}$$

$$\bar{\nu}_k = \max_{\boldsymbol{z}\in\mathbb{R}^n} \boldsymbol{e}_k^T\boldsymbol{z} \quad \text{subject to } \|\boldsymbol{Rz}\|_2 \le \alpha. \tag{3.8}$$

We first consider solving (3.8). Let $\boldsymbol{p} \triangleq \boldsymbol{Rz}$. Then (3.8) becomes

$$\bar{\nu}_k = \max_{\boldsymbol{p}} \boldsymbol{e}_k^T\boldsymbol{R}^{-1}\boldsymbol{p} \quad \text{subject to } \|\boldsymbol{p}\|_2 \le \alpha. \tag{3.9}$$

By the Cauchy–Schwarz inequality (see, e.g., [11, p. 53]), we have

$$\boldsymbol{e}_k^T\boldsymbol{R}^{-1}\boldsymbol{p} \le \|\boldsymbol{R}^{-T}\boldsymbol{e}_k\|_2\|\boldsymbol{p}\|_2 \le \|\boldsymbol{R}^{-T}\boldsymbol{e}_k\|_2\alpha. \tag{3.10}$$

Here the first inequality becomes an equality if and only if $\boldsymbol{p} = c\boldsymbol{R}^{-T}\boldsymbol{e}_k$ for a nonnegative scalar c, while the second inequality becomes an equality if and only if $\|\boldsymbol{p}\|_2 = \alpha$. Thus the optimal $\boldsymbol{p}$ for (3.9) satisfies $\boldsymbol{p} = \alpha\boldsymbol{R}^{-T}\boldsymbol{e}_k/\|\boldsymbol{R}^{-T}\boldsymbol{e}_k\|_2$, and from (3.9)

$$\bar{\nu}_k = \alpha\|\boldsymbol{R}^{-T}\boldsymbol{e}_k\|_2,$$

where $\boldsymbol{q} \triangleq \boldsymbol{R}^{-T}\boldsymbol{e}_k$ can easily be obtained by solving $\boldsymbol{R}^T\boldsymbol{q} = \boldsymbol{e}_k$. After finding $\bar{\nu}_k$, we immediately obtain $\bar{\mu}_k$ for $k = 1 : n$, since it is easy to observe that

$$\bar{\mu}_k = -\bar{\nu}_k = -\alpha\|\boldsymbol{R}^{-T}\boldsymbol{e}_k\|_2.$$

Then immediately we obtain $\bar{\boldsymbol{l}}$ and $\bar{\boldsymbol{u}}$ in (3.5) by (3.6).

Now we give details of the modification of Algorithm DEC-BILS, resulting in Algorithm DEC-EILS.

$\boxed{\text{ALGORITHM DEC-EILS}}$

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 $\bar{\boldsymbol{u}} \in \mathbb{Z}^n$, the initial search ellipsoid bound β, and the constraint ellipsoid bound α.
Output: The solution $\hat{\boldsymbol{z}} \in \mathbb{Z}^n$ to the EILS problem (3.4).

```
In Algorithm DEC-BILS, replace step 3 with
if $t_k + r_{kk}^2(z_k - c_k)^2 \ge \beta^2$, then
    go to step 4    // we are not inside the search ellipsoid
else if $z_k \notin [l_k, u_k]$, then
    go to step 6    // we are inside the search 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 if $\|\boldsymbol{Rz}\|_2^2 > \alpha^2$, then
    go to step 6    // $k = 1$, we are inside the constraint box
                       but outside the constraint ellipsoid
else go to step 5    // $k = 1$
end
```

Note that when an integer point $\hat{\boldsymbol{z}}$ is found, Algorithm DEC-EILS checks whether it is valid by testing $\|\boldsymbol{R}\hat{\boldsymbol{z}}\|_2^2 \le \alpha^2$. If so, it stores the current $\hat{\boldsymbol{z}}$, updates the bound β^2 by taking $\beta^2 = \|\bar{\boldsymbol{y}} - \boldsymbol{R}\hat{\boldsymbol{z}}\|_2^2$, and moves up to level 2 to continue the search process; otherwise it continues the search at level 1. Besides the shortcoming which is inherited from Algorithm DEC-BILS, Algorithm DEC-EILS may waste time in enumerating integer points which are inside the constraint box $\bar{\mathcal{B}}$ but outside the constraint ellipsoid $\bar{\mathcal{E}}$, especially when the initial value of β is large. If the initial value of β is small and the solution is inside the initial search ellipsoid, then this shortcoming will be less serious since the set of the above integer points (notice that they are inside the search ellipsoid) will be small.

Unlike Algorithm DEC-BILS for solving the BILS problem, a finite initial value for β is necessary to make Algorithm DEC-EILS work. Note that only the first branch in step 3 (i.e., we are not inside the search ellipsoid) leads to step 4 in which the search process moves up to level $k+1$. If β is infinity, the inequality $t_k + r_{kk}^2(z_k - c_k)^2 \geq \beta^2$ will never be satisfied. However, it is possible that, after $z_n, \dots, z_2$ have been determined, the constraint $\|\boldsymbol{Rz}\|_2^2 \leq \alpha^2$ may never be satisfied for any integer value of z_1; then we cannot obtain a valid integer point. In this situation, we should move to level 2 and try the next value for z_2. But we cannot move up since the only move-up branch will never be satisfied until we get a finite value for β. Hence, the initial β has to be finite. If it is set to be too small, the search algorithm has to be restarted again and again until a valid integer point is found. For solving the BILS problem (1.2), one choice for the initial β suggested by Damen, Chkeif, and Belfiore in [9] (see also [4]) is to use Roger's upper bound on the covering radius of the lattice generated by $\boldsymbol{A}$ (or by $\boldsymbol{R}$) (see [7, Chapter 2])

$$\beta = (n \log n + n \log\log n + 5n)^{1/n} \times \frac{|\det(\boldsymbol{R})|^{1/n}}{V_n^{1/n}}, \tag{3.11}$$

where V_n is the volume of a sphere of radius 1 in the real space $\mathbb{R}^n$. If the algorithm did not find a valid integer point with the current β, it increases β to $\sqrt{\beta^2+1}$ and restarts the search process. This is a shortcoming with such an initial choice of β.

4. A new method for solving the EILS problem (1.4).

In this section we propose a new method to solve the EILS problem (1.4). In the reduction process we use the QRZ factorization obtained by the LLL reduction, and in the search process the enumerated integer points always satisfy the ellipsoidal constraint in (1.3) or the equivalent one in (3.4).

Note that in the reduction process if we use the QRZ factorization (2.1) where $\boldsymbol{Z}$ is a general unimodular matrix, the original EILS problem (1.4) can still be transformed to (3.4). So unlike the DEC method which tries to solve the BILS problem (3.2), where the $\boldsymbol{Z}$ matrix in the QRZ factorization of $\boldsymbol{A}$ is a permutation matrix, we can apply the LLL reduction in computing the QRZ factorization in solving (1.4). This can make the search process more efficient.

Now we consider designing a search algorithm. Suppose that the solution of (3.4) satisfies the bound $\|\bar{\boldsymbol{y}} - \boldsymbol{Rz}\|_2^2 < \beta^2$ for some β. Recall from section 2.2 that when we search an integer point $\boldsymbol{z}$, after the last $n-k$ entries $z_n, z_{n-1}, \dots, z_{k+1}$ have been fixed, we want to determine z_k at level k. Note that z_k has to satisfy the inequality (2.15). In the following we derive the other constraint on z_k imposed by the ellipsoidal constraint (1.3).

Rewrite the ellipsoidal constraint $\|\boldsymbol{Rz}\|_2^2 \leq \alpha^2$ into the following form:

$$\sum_{k=1}^{n} \Big(r_{kk} z_k + \sum_{j=k+1}^{n} r_{kj} z_j \Big)^2 \leq \alpha^2. \tag{4.1}$$

With b_k defined in (2.11), we define

$$s_n \triangleq \alpha^2, \quad s_{k-1} \triangleq \alpha^2 - \sum_{i=k}^{n} \left(r_{ii} z_i + \sum_{j=i+1}^{n} r_{ij} z_j \right)^2 = s_k - (r_{kk} z_k + b_k)^2, \quad k = n : -1 : 2.$$

It is easy to see that (4.1) is equivalent to

$$(r_{kk}z_k + b_k)^2 \le s_k, \quad k = n : -1 : 1.$$

Thus z_k is restricted to the following interval:

$$\bar{l}_k \le z_k \le \bar{u}_k, \quad \bar{l}_k \triangleq \left\lceil \frac{-\sqrt{s_k} - b_k}{r_{kk}} \right\rceil, \quad \bar{u}_k \triangleq \left\lfloor \frac{\sqrt{s_k} - b_k}{r_{kk}} \right\rfloor, \quad k = n : -1 : 1. \tag{4.2}$$

These are the constraints at all levels imposed by the ellipsoidal constraint in (3.4). Note that, unlike the interval constraints in the BILS problem, here $\bar{l}_k$ and $\bar{u}_k$ are not constants, since s_k and b_k depend on the values of $z_n, z_{n-1}, \dots, z_{k+1}$. Hence, we have to compute $\bar{l}_k$ and $\bar{u}_k$ while moving from level $k+1$ down to level k. If $\bar{l}_k > \bar{u}_k$, then we cannot find a satisfactory integer z_k at level k, and we have to move up to level $k+1$. Based on the above discussion, we propose the following search algorithm.

Algorithm SEARCH-EILS

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 initial search ellipsoid bound β, and the constraint ellipsoid bound α.
Output: The solution $\boldsymbol{z} \in \mathbb{Z}^n$ to the EILS problem (3.4).

```
1. (Initialization) Set $k := n$, $b_k = 0$, $s_k := \alpha^2$, and $t_k := 0$.
2. Set $lbound_k := 0$ and $ubound_k := 0$
   Compute $\bar{l}_k := \left\lceil \frac{-\sqrt{s_k}-b_k}{r_{kk}} \right\rceil$, $\bar{u}_k := \left\lfloor \frac{\sqrt{s_k}-b_k}{r_{kk}} \right\rfloor$
   **If** $\bar{u}_k < \bar{l}_k$, **then** go to step 4, **end**
   **If** $\bar{u}_k = \bar{l}_k$, **then** set $lbound_k := 1$ and $ubound_k := 1$, **end**
   Compute $c_k := (\bar{y}_k - b_k)/r_{kk}$, $z_k := \lfloor c_k \rceil$
   **if** $z_k \le \bar{l}_k$, **then**
       $z_k := \bar{l}_k$, set $lbound_k := 1$ and $\Delta_k := 1$
   **else if** $z_k \ge \bar{u}_k$, **then**
       $z_k := \bar{u}_k$, set $ubound_k := 1$ and $\Delta_k := -1$
   **else**   // no bound of the constraint is reached
       set $\Delta_k := \mathrm{sign}(c_k - z_k)$
   **end**
3. **if** $t_k + r_{kk}^2(z_k - c_k)^2 \ge \beta^2$, **then**
       go to step 4   // we are not inside the search ellipsoid
   **else if** $k > 1$, **then**
       compute $b_{k-1} := \sum_{j=k}^{n} r_{k-1,j}z_j$, $t_{k-1} := t_k + r_{kk}^2(z_k - c_k)^2$,
       $s_{k-1} := s_k - (r_{kk}z_k + b_k)^2$, set $k := k-1$, go to step 2
   **else**   // $k = 1$ and a valid point is found
       compute $\beta^2 := t_1 + r_{11}^2(z_1 - c_1)^2$, 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 at 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 $\Delta_k := -\Delta_k - \mathrm{sign}(\Delta_k)$
   **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
```

In this new search algorithm, all enumerated integer points satisfy the ellipsoid constraint (3.4). So it avoids the main problem with Algorithm DEC-EILS we mentioned in the paragraph following the description of Algorithm DEC-EILS in section 3. Furthermore, it does not have the shortcoming that Algorithm DEC-EILS inherited from Algorithm DEC-BILS (see the last paragraph of section 2.2).

Unlike Algorithm DEC-EILS in which the initial search ellipsoid bound β has to be a finite number, in the new algorithm β can be set to be infinity, which is usually a good choice. This avoids the possible problem with Algorithm DEC-EILS we mentioned in the second to the last paragraph in section 3.

5. Solving the general EILS problem (1.5).

In this section we extend the method given in the previous section to solve the general EILS problem (1.5). We have not seen any method in the literature specifically for solving (1.5).

In the reduction process we compute the following matrix factorization of $\boldsymbol{A}$ and $\boldsymbol{B}$ given in (1.5), to be referred to as the *generalized QRZ factorization*:

$$\boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{Z} = \begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{bmatrix}, \quad \hat{\boldsymbol{Q}}^T\boldsymbol{B}\boldsymbol{Z} = \begin{bmatrix}\hat{\boldsymbol{R}}\\ \boldsymbol{0}\end{bmatrix}, \tag{5.1}$$

where $\boldsymbol{Q} = [\underset{n}{\boldsymbol{Q}_1}, \underset{m-n}{\boldsymbol{Q}_2}] \in \mathbb{R}^{m\times m}$ and $\hat{\boldsymbol{Q}} = [\underset{p}{\hat{\boldsymbol{Q}}_1}, \underset{p-n}{\hat{\boldsymbol{Q}}_2}] \in \mathbb{R}^{p\times p}$ are orthogonal, $\boldsymbol{R} \in \mathbb{R}^{n\times n}$ and $\hat{\boldsymbol{R}} \in \mathbb{R}^{n\times n}$ are nonsingular upper triangular with positive diagonal entries, and $\boldsymbol{Z} \in \mathbb{Z}^{n\times n}$ is unimodular. Specifically, to make the later search process efficient, we first compute the LLL reduction of $\boldsymbol{A}$ to obtain the QRZ factorization of $\boldsymbol{A}$ and compute the QR factorization of $\boldsymbol{BZ}$ to obtain the QRZ factorization of $\boldsymbol{B}$. Define

$$\bar{\boldsymbol{y}} \triangleq \boldsymbol{Q}_1^T\boldsymbol{y}, \quad \hat{\boldsymbol{w}} \triangleq \hat{\boldsymbol{Q}}_1^T\boldsymbol{w}, \quad \boldsymbol{z} \triangleq \boldsymbol{Z}^{-1}\boldsymbol{x}, \quad \hat{\alpha}^2 \triangleq \alpha^2 - \|\hat{\boldsymbol{Q}}_2^T\boldsymbol{w}\|^2,$$

where we assume $\alpha^2 - \|\hat{\boldsymbol{Q}}_2^T\boldsymbol{w}\|^2 > 0$ (otherwise the solution does not exist). Then, with (5.1), the general EILS problem (1.5) is reduced to

$$\min_{\hat{\mathcal{E}}_g} \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2, \quad \hat{\mathcal{E}}_g = \{\boldsymbol{z} \in \mathbb{Z}^n : \|\hat{\boldsymbol{w}} - \hat{\boldsymbol{R}}\boldsymbol{z}\|_2^2 \le \hat{\alpha}^2\}. \tag{5.2}$$

For the reduced problem (5.2), we can extend Algorithm SEARCH-EILS to solve it. The ellipsoidal constraint $\|\hat{\boldsymbol{w}} - \hat{\boldsymbol{R}}\boldsymbol{z}\|_2^2 \le \hat{\alpha}^2$ can be rewritten as

$$\sum_{k=1}^{n}\left(\hat{r}_{kk}z_k + \sum_{j=k+1}^{n}\hat{r}_{kj}z_j - \hat{w}_k\right)^2 \le \hat{\alpha}^2. \tag{5.3}$$

Define

$$\hat{b}_k \triangleq \sum_{j=k+1}^{n}\hat{r}_{kj}z_j - \hat{w}_k,$$

$$\hat{s}_n \triangleq \hat{\alpha}^2, \quad \hat{s}_{k-1} \triangleq \hat{\alpha}^2 - \sum_{i=k}^{n}(\hat{r}_{ii}z_i + \hat{b}_i)^2 = \hat{s}_k - (\hat{r}_{kk}z_k + \hat{b}_k)^2, \quad k = n : -1 : 2.$$

Then (5.3) is equivalent to

$$(\hat{r}_{kk}z_k + \hat{b}_k)^2 \leq \hat{s}_k, \quad k = n : -1 : 1.$$

Thus z_k is restricted to the following interval:

$$(5.4) \quad \hat{l}_k \leq z_k \leq \hat{u}_k, \quad \hat{l}_k \triangleq \left\lceil \frac{-\sqrt{\hat{s}_k} - \hat{b}_k}{\hat{r}_{kk}} \right\rceil, \quad \hat{u}_k \triangleq \left\lfloor \frac{\sqrt{\hat{s}_k} - \hat{b}_k}{\hat{r}_{kk}} \right\rfloor, \quad k = n : -1 : 1.$$

With the above preparation, we can modify Algorithm SEARCH-EILS as follows. In step 1, set $\hat{b}_k = 0$ and replace $s_k := \alpha^2$ by $\hat{s}_k := \hat{\alpha}^2$. In step 2, replace $\bar{l}_k := \lceil \frac{-\sqrt{s_k}-b_k}{r_{kk}} \rceil$ by $\hat{l}_k := \lceil \frac{-\sqrt{\hat{s}_k}-\hat{b}_k}{\hat{r}_{kk}} \rceil$ and $\bar{u}_k := \lfloor \frac{\sqrt{s_k}-b_k}{r_{kk}} \rfloor$ by $\hat{u}_k := \lfloor \frac{\sqrt{\hat{s}_k}-\hat{b}_k}{\hat{r}_{kk}} \rfloor$. In step 3, if $k > 1$, compute $\hat{b}_{k-1} := \sum_{j=k}^{n} \hat{r}_{k-1,j} z_j - \hat{w}_{k-1}$ and replace $s_{k-1} := s_k - (r_{kk}z_k + b_k)^2$ by $\hat{s}_{k-1} := \hat{s}_k - (\hat{r}_{kk}z_k + \hat{b}_k)^2$.

6. Simulation results.

In this section we give simulation results to compare the efficiency of the DAE method and the efficiency of our new method for solving the EILS problem (1.4). In the DAE method, the V-BLAST column reordering strategy (see section 2.1) is used in reduction, Algorithm DEC-EILS is used for search, and the initial bound β is set to be Roger's upper bound (see (3.11)), which will be increased to $\sqrt{\beta^2 + 1}$ if no valid integer point is found in the search process, while, in our new method, Algorithm LLL is used for reduction, Algorithm SEARCH-EILS is used for search, and the initial bound β is set to be infinity. Our simulation results will show how these individual strategies affect the performance of the whole methods. We will also compare the overall performance of the two methods.

All our computations were performed in MATLAB 7.0 on a Pentium 4 3.20GHz machine with 1GB memory running Windows XP.

6.1. Setup.

In our simulations, the generator matrices $\boldsymbol{A}$ were $n \times n$ matrices whose elements were drawn from independently and identically distributed zero-mean, unit variance Gaussian distributions (this is often the case in communications). The input vectors $\boldsymbol{y}$ were constructed based on the following linear model:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{v}, \quad \boldsymbol{v} \sim N(\boldsymbol{0}, \sigma^2 \boldsymbol{I}),$$

which is a typical model used in communications. To generate the vector $\boldsymbol{x}$, we first compute the smallest hyperrectangle $\mathcal{X}$ in (3.1) and then randomly pick one integer point inside $\mathcal{X}$ and check if it satisfies $\|\boldsymbol{A}\boldsymbol{x}\|_2^2 \leq \alpha^2$. If it is inside the constraint ellipsoid, we choose it as $\boldsymbol{x}$; otherwise we continue to pick another one. In all the figures to be given later, $n = 20 : 40$, and the cost for each n is the average CPU time (in seconds) over 1000 runs.

6.2. Comparison of the reduction strategies.

To show the effectiveness of the LLL reduction strategy, we compare it with the V-BLAST strategy. In our simulations, after either of the reduction strategies was used, Algorithm SEARCH-EILS was applied in the search process and the initial bound β was set to be infinity. The average search time and the average total time (including both the reduction time and the search time) for $\alpha = 0.5n$ and $\sigma = 0.1, 0.5$ are given in Figures 6.1 and 6.2.

From Figures 6.1 and 6.2, we observe that the LLL strategy is much more effective than the V-BLAST strategy for the search process. The advantage of the LLL strategy becomes more significant as the dimension n increases, especially when the noise is

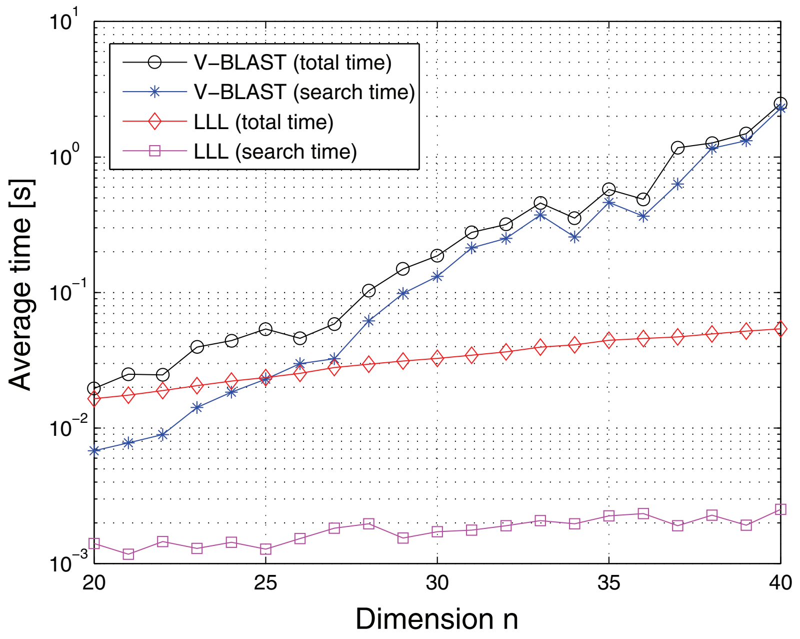

FIG. 6.1. *Average time versus dimension,* $\alpha = 0.5n$*,* $\sigma = 0.1$*.*

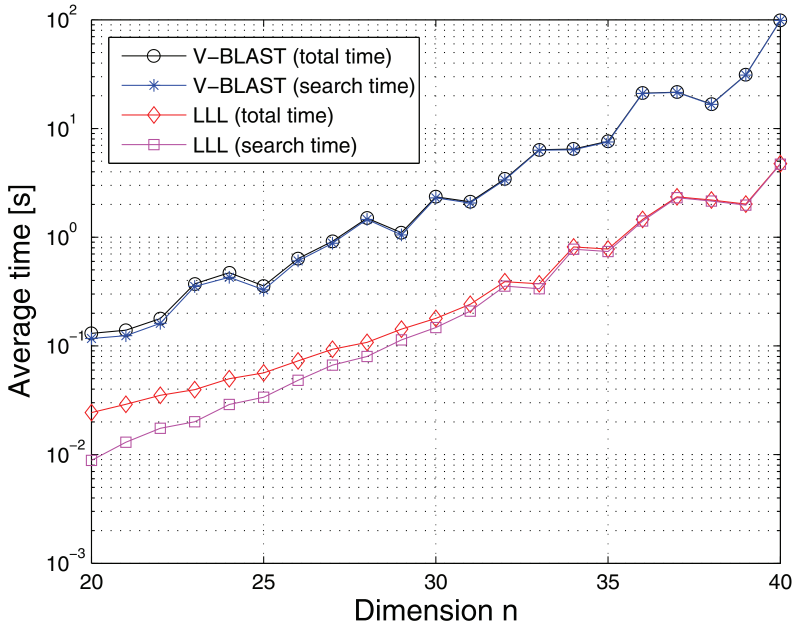

FIG. 6.2. *Average time versus dimension,* $\alpha = 0.5n$*,* $\sigma = 0.5$*.*

small. When the V-BLAST strategy is used, the search time dominates the cost of the whole algorithm for both $\sigma = 0.1$ and $\sigma = 0.5$. When the LLL strategy is used, the search time dominates the cost of the whole algorithm for $\sigma = 0.5$, but the reduction time dominates the cost of the whole algorithm for $\sigma = 0.1$.

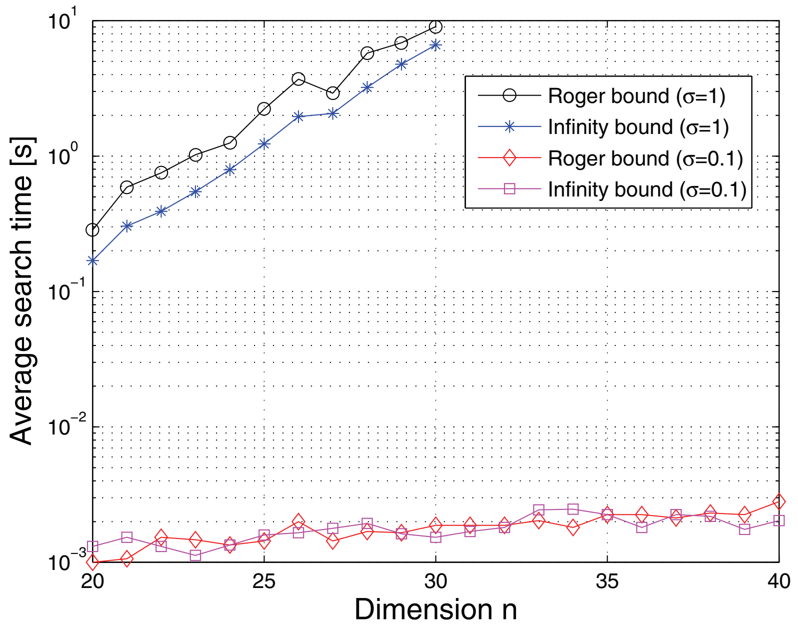

FIG. 6.3. *Average search time versus dimension, $\alpha = 0.5n$.*

6.3. Comparison of the initial choices of β. Recall that two different initial choices for the search ellipsoid bound β were mentioned in sections 3 and 4, respectively. One sets β to be Roger's upper bound (3.11), and if no valid integer point is found in the search process, update it by $\beta := \sqrt{\beta^2+1}$; while the other chooses $\beta = \infty$. To compare the effects of these two choices on the search phase, we use them with Algorithm SEARCH-EILS (recall that $\beta = \infty$ cannot be used with Algorithm DEC-EILS). In the simulations, Algorithm LLL was used for reduction. The results of the average search time are given in Figure 6.3 for $\alpha = n/2$ and $\sigma = 0.1, 1$. Since the computation was too time consuming for $\sigma = 1$, we just give the results for $n = 20 : 30$.

From the results, we observe that when the noise is small, there is almost no difference between the first and the second choices. But when the noise is large, Algorithm SEARCH-EILS might be restarted many times during the search process because no solution was found within the initial search ellipsoid determined by Roger's upper bound. In this situation, the infinity is a better choice for the initial value of β. From Figure 6.3, we also observe that when the noise is large, the search time can be significantly long.

6.4. Comparison of the search algorithms. To compare the efficiency of Algorithms DEC-EILS and SEARCH-EILS, we give the average search time of the two algorithms for $\alpha = 0.5n$ and $\sigma = 0.1, 0.5$ in Figure 6.4. In the simulations, for both the search algorithms, Algorithm LLL was applied for reduction and the initial β was set to be Roger's upper bound and updated by $\beta := \sqrt{\beta^2+1}$ if no valid point was found (note that in Algorithm DEC-EILS the initial β has to be finite). Figure 6.4 indicates for $\sigma = 0.1$ that Algorithm DEC-EILS took more CPU time than Algorithm SEARCH-EILS, while, for $\sigma = 0.5$, the two algorithms cost more or less the same CPU time.

We found that when the noise is large, the cost of Algorithm DEC-EILS is more sensitive to the initial value of β than that of Algorithm SEARCH-EILS. When the

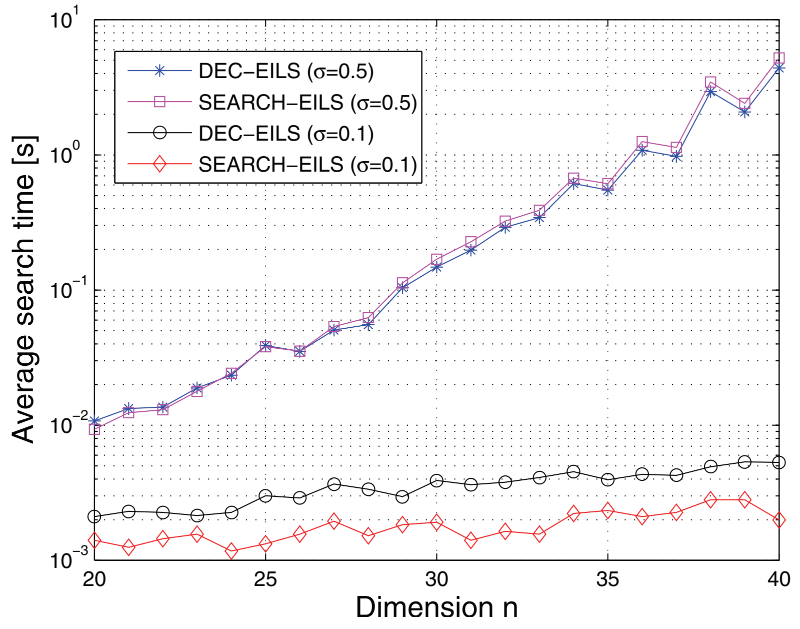

FIG. 6.4. *Average search time versus dimension, $\alpha = 0.5n$, $\beta =$ Roger's upper bound.*

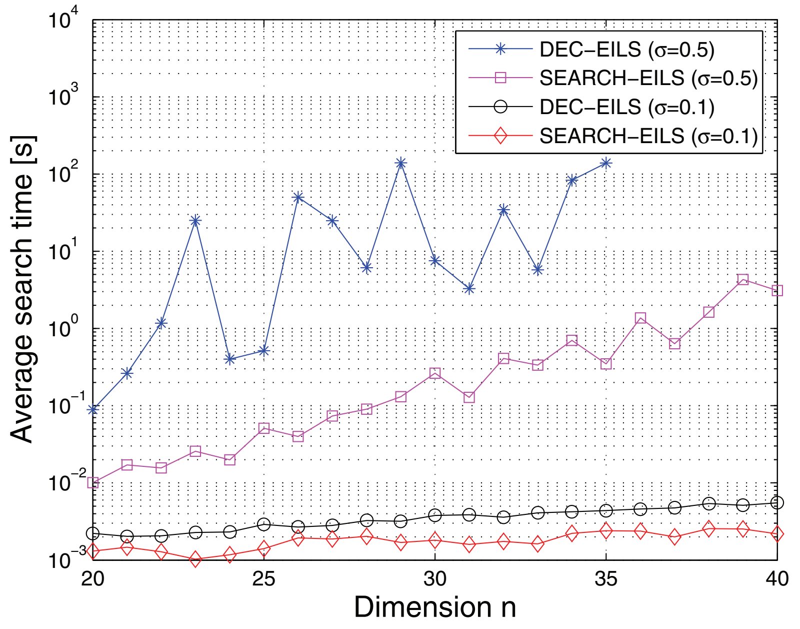

FIG. 6.5. *Average search cost versus dimension, $\alpha = 0.5n$, $\beta = 3 *$ Roger's upper bound.*

initial value of β becomes larger, Algorithm DEC-EILS can become much less efficient, while the cost of Algorithm SEARCH-EILS changes little. To illustrate this, we display the search time of the two algorithms in Figure 6.5 where the initial β is three times as large as Roger's upper bound. Since the computation was time consuming for

Algorithm DEC-EILS when $\sigma = 0.5$, we show only its result for $n = 20:35$. Comparing Figures 6.5 and 6.4, we observe that when $\sigma = 0.1$, both algorithms are as efficient as before; when $\sigma = 0.5$, Algorithm SEARCH-EILS is still as efficient as before, but Algorithm DEC-EILS becomes much less efficient and also the average search time varies dramatically for different dimensions. Here we give some explanations. We observed that when $\sigma = 0.5$, the integer points found by Algorithm DEC-EILS during the search process were often invalid; i.e., they were outside of the constraint ellipsoid in (3.4). Thus the search process became (much) less efficient. This is the main drawback with Algorithm DEC-EILS which we pointed out in section 3. For different instances with the same dimension, sometimes the number of invalid points was large and sometimes the number of invalid points was small. Thus the search time may vary dramatically for different instances with the same dimension, leading to the zig-zag behavior of the average search time over 1000 samples (see the top curve in Figure 6.5). The reason that the above phenomenon occurred with a large initial β and a large noise can be explained in geometry. In this situation, it is likely that the constraint box $\bar{\mathcal{B}}$ in (3.3) (notice that it includes those invalid integer points) lies inside the initial search ellipsoid (2.9). Since the noise is large, the search ellipsoid may not shrink much in the search process. Thus the above invalid integer point problem may last until the optimal integer point is found. This phenomenon is less likely to happen when the initial value of β is small or when the noise is small, since the intersection set of the constraint box $\bar{\mathcal{B}}$ and the search ellipsoid during the search process becomes small; i.e., the number of invalid integer points found in the search process becomes small.

6.5. Comparison of the two methods. Lastly, we compare the overall performance of our new method with the DEC method. In our simulations, $\alpha = 0.5n, n$ and $\sigma = 0.1, 0.5$. The results for the average total time (including both the reduction time and the search time) are given in Figure 6.6 for $\sigma = 0.1$ and in Figure 6.7 for $\sigma = 0.5$.

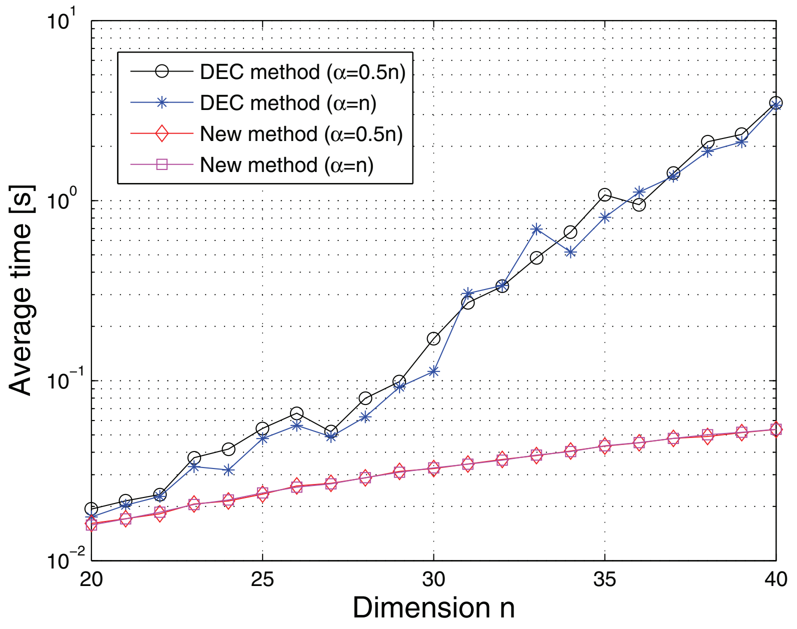

FIG. 6.6. *Average search cost versus dimension, $\sigma = 0.1$.*

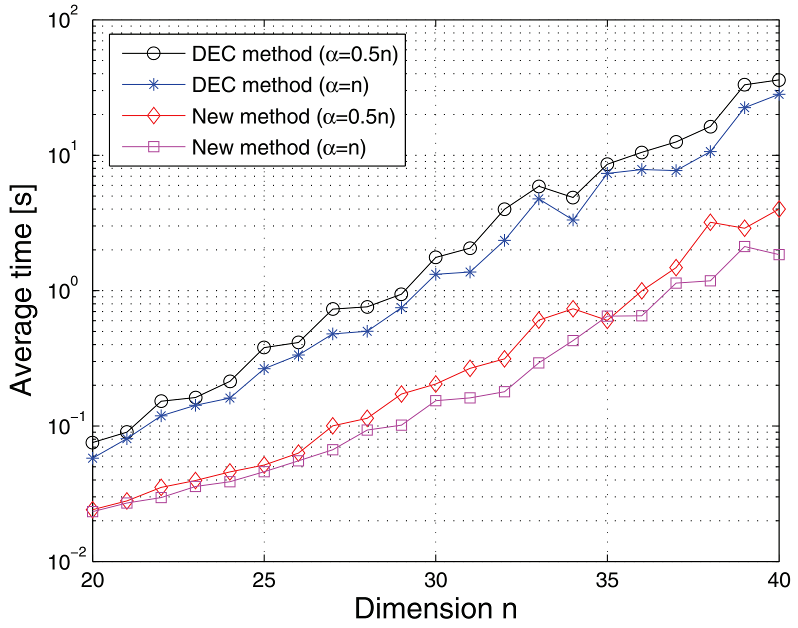

FIG. 6.7. *Average search time versus dimension, $\sigma = 0.5$.*

From the results we observe that our new method is (much) faster than the DEC method. Usually the improvement becomes more significant as the dimension n increases, in particular, when the noise is small. For the case $\sigma = 0.1$, when α changes from $0.5n$ to n, the cost of the DEC method changes only slightly, while the cost of the new method changes little (note that the two bottom curves in Figure 6.6 are almost identical). For the case $\sigma = 0.5$, when α changes from $0.5n$ to n, the cost of each method decreases.

7. Summary and future work.

We have proposed a new method to solve the EILS problem (1.4). Simulations indicated that the new method can be significantly faster than the DEC method proposed in [8]. The new method first transformed (1.4) into a new problem (3.4) through what we refer to as the QRZ factorization of the lattice generator matrix, where the QRZ factorization was computed by the LLL reduction algorithm. This reduction strategy is much more effective for a search process than the V-BLAST reduction used in the DEC method. Then a search algorithm based on the Schnorr–Euchner enumerating strategy was proposed for the reduced EILS problem. The enumerated integer points by the new search algorithm always satisfy the ellipsoidal constraint (4.1). This avoids the main shortcoming with the search algorithm used in the DEC method. We also discussed the choice of β of the initial search ellipsoid (2.9).

For solving a box-constrained ILS problem, a new reduction strategy has been recently proposed in [5], which uses all available information of the problem to make the search process more efficient. Note that the LLL reduction uses only the information of the generator matrix $\boldsymbol{A}$. In the future, we would like to extend the ideas of [5] to develop a more effective reduction strategy for the EILS problem (1.4).

Acknowledgments. We are indebted to Qing Han and Xiaohua Yang for the numerical simulations they did for this paper. We are also grateful to the referees for their helpful suggestions.

REFERENCES

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

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

[3] J. Boutros, N. Gresset, L. Brunel, and M. Fossorier, *Soft-input soft-output lattice sphere decoder for linear channels*, in Proceedings of the IEEE 2003 Global Communications Conference, San Francisco, 2003.

[4] L. Brunel and J. J. Boutros, *Lattice decoding for joint detection in direct-sequence CDMA systems*, IEEE Trans. Inform. Theory, 49 (2003), pp. 1030–1037.

[5] X.-W. Chang and Q. Han, *Solving box-constrained integer least squares problems*, IEEE Trans. Wireless Comm., 7 (2008), pp. 277–287.

[6] X.-W. Chang and C. C. Paige, *Euclidean distances and least squares problems for a given set of vectors*, Appl. Numer. Math., 57 (2007), pp. 1240–1244.

[7] J. H. Conway and N. J. A. Sloane, eds., *Sphere Packings, Lattices and Groups*, Springer, New York, 1998.

[8] 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, 49 (2003), pp. 2389–2402.

[9] M. O. Damen, A. Chkeif, and J.-C. Belfiore, *Lattice code decoder for space-time codes*, IEEE Commun. Lett., 4 (2000), pp. 161–163.

[10] 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. Sel. Areas Commun., 17 (1999), pp. 1841–1852.

[11] G. H. Golub and C. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[12] A. Hassibi and S. Boyd, *Integer parameter estimation in linear models with applications to GPS*, IEEE Trans. Signal Process., 46 (1998), pp. 2938–2952.

[13] R. Kannan, *Improved algotithms for integer programming and related lattice problems*, in Proceedings of the ACM Symposium on Theory of Computing, Boston, MA, 1983.

[14] A. K. Lenstra, H. W. Lenstra, and L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann., 261 (1982), pp. 515–534.

[15] A. D. Murugan, H. El Gamal, M. O. Damen, and G. Caire, *A unified framework for tree search decoding: Rediscovering the sequential decoder*, IEEE Trans. Inform. Theory, 52 (2006), pp. 933–953.

[16] M. Pohst, *On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications*, ACM SIGSAM Bull., 15 (1981), pp. 37–44.

[17] C. P. Schnorr and M. Euchner, *Lattice basis reduction: Improved practical algorithms and solving subset sum problems*, Math. Program., 66 (1994), pp. 181–199.

[18] P. Teunissen, *GPS carrier phase ambiguity fixing concepts*, in GPS for Geodesy, 2nd ed., P. Teunissen and A. Kleusberg, eds., Springer, New York, 1998, pp. 317–388.

[19] P. van Emde Boas, *Another NP-complete Partition Problem and the Complexity of Computing Short Vectors in a Lattice*, Technical report 81-04, Mathematisch Institut, Amsterdam, The Netherlands, 1981.

[20] E. Viterbo and E. Biglieri, *A universal lattice decoder*, in Proceedings of the 14-ème Colloque GRETSI, Juan-Les-Pins, France, 1993.

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