Lattice preconditioning for the real relaxation branch-and-bound approach for integer least squares problems

Journal of Global Optimization manuscript No.
(will be inserted by the editor)

Lattice Preconditioning for the Real Relaxation Branch-and-Bound Approach for Integer Least Squares Problems

Miguel F. Anjos · Xiao-Wen Chang · Wen-Yang Ku

Received: date / Accepted: date

Abstract The integer least squares problem is an important problem that arises in numerous applications. We propose a real relaxation-based branch-and-bound (RRBB) method for this problem. First, we define a quantity called the distance to integrality, propose it as a measure of the number of nodes in the RRBB enumeration tree, and provide computational evidence that the size of the RRBB tree is proportional to this distance. Since we cannot know the distance to integrality a priori, we prove that the norm of the Moore-Penrose generalized inverse of the matrix of coefficients is a key factor for bounding this distance, and then we propose a preconditioning method to reduce this norm using lattice reduction techniques. We also propose a set of valid box constraints that help accelerate the RRBB method. Our computational results show that the proposed preconditioning significantly reduces the size of the RRBB enumeration tree, that the preconditioning combined with the proposed set of box constraints can significantly reduce the computational time of RRBB, and that the resulting RRBB method can outperform the Schnorr and Eucher method, a widely used method for solving integer least squares problems, on some types of problem data.

Keywords Integer least squares · Branch-and-bound methods · Lattice reduction · Preconditioning · Box constraints

M.F. Anjos
Canada Research Chair in Discrete Nonlinear Optimization in Engineering,
GERAD & École Polytechnique de Montréal, Montreal, QC, Canada H3C 3A7
E-mail: anjos@stanfordalumni.org

X.-W. Chang
School of Computer Science, McGill University
E-mail: chang@cs.mcgill.ca

W.-Y. Ku
Department of Mechanical and Industrial Engineering, University of Toronto
E-mail: wku@mie.utoronto.ca

1 Introduction

Given a real matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with full column rank and a real vector $\boldsymbol{y} \in \mathbb{R}^m$, the integer least squares (ILS) problem has the following form:

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

In lattice theory, the *lattice* generated by $\boldsymbol{A}$ is defined as $\mathcal{L}(\boldsymbol{A}) = \{\boldsymbol{A}\boldsymbol{x} : \boldsymbol{x} \in \mathbb{Z}^n\}$ and $\boldsymbol{A}$ is referred to as a lattice basis matrix. The ILS problem is to find the closest vector in $\mathcal{L}(\boldsymbol{A})$ to $\boldsymbol{y}$. This is why the ILS problem is often referred to as the *closest vector problem*, see e.g. [5]. ILS problems arise from many applications including GPS, communications, cryptography, lattice design, bioinformatics, and finance (see e.g. [5,23,38]). Unlike the real least squares (RLS) problem, the ILS problem is known to be NP-hard [11].

There are four main families of approaches to solving ILS problems: the Voronoi approach [33,32], the approximation approach [6,7], the semidefinite programming approach [25,37] and the discrete search approach [21,24]. However, the Voronoi approach is known not to be computationally efficient, and the approximation and the semidefinite programming approaches only compute suboptimal solutions.

The most common method for solving the ILS problem to optimality in practice is the discrete search approach. This approach finds the solution by enumerating the integer points in a bounded region, see e.g. [21,24,35]. Different discrete search methods are compared in [5]. When applying the discrete search approach, one usually performs *lattice basis reduction*, such as the well-known LLL lattice reduction [30], to transform the given ILS problem (1) into a new one for which the search process in more efficient, see e.g. [5,16]. Given a lattice, the lattice basis matrix is not unique. The goal of lattice basis reduction is to find a new lattice basis matrix whose columns are shorter according to some criteria. The recent tutorial [20] and book [12] provide an introduction to the theory and algorithms of lattice basis reduction.

Lattice basis reduction has been extensively applied to integer linear programming. For the integer feasibility problem that consists of determining whether there is an integer point inside a bounded set defined by linear inequalities, Lenstra [30] applied lattice basis reduction techniques to the set of inequalities to reduce the problem to a small number of subproblems with smaller dimensions. Since then a number of lattice-based approaches have been proposed for integer linear programming problems, see e.g. the surveys [3,1] and the recent papers [26,4,34,31,2] among many others. Recent research has also considered convex quadratic objectives (as is the case for the ILS problem), see e.g. [13], and extensions to quadratic indefinite forms, see e.g. the survey chapter [36]. For the ILS problem, the Schnorr and Euchner search strategy [35] is still considered the most efficient one.

We consider an alternative approach to compute an optimal solution to (1). This approach is based on the branch-and-bound paradigm commonly used in mixed-integer optimization; we call it the *real relaxation-based branch-and-bound (RRBB)* method. The commercial mixed-integer programming solver CPLEX [18] provides a function that implements this approach for general integer convex quadratic programming problems, and it also provides a function for solving an ILS problem by converting it to an integer convex quadratic programming problem.

The RRBB method follows the standard branch-and-bound paradigm. It starts by solving subproblem P_0, the real relaxation of (1):

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

to obtain the RLS solution $\boldsymbol{x}^{\mathrm{R}}$. Then the method chooses $i \in \{1,\ldots,n\}$ such that $x_i^{\mathrm{R}} \notin \mathbb{Z}$, creates two subproblems P_1 with added constraint $x_i \le \lfloor x_i^{\mathrm{R}} \rfloor$, and P_2 with $x_i \ge \lfloor x_i^{\mathrm{R}} \rfloor + 1$, and iterates on the subproblems. Thus the RRBB process forms a binary tree. To help reduce the size of the tree, i.e., the number of subproblems, subproblem P_j is not branched on if the norm of the residual vector of the real solution to P_j is larger than the norm of the residual vector of the best known integer solution. By exploring the entire tree, the RRBB method is guaranteed to find the optimal solution of (1). In practice, for instance as implemented in CPLEX, this approach is typically uncompetitive with the discrete search approach.

This paper is concerned with improving the peformance of the RRBB method for solving the ILS problem. Specifically, we investigate how to incorporate the lattice reduction technique into RRBB to make it faster, and we show that the resulting approach can be competitive with the discrete search approach in terms of efficiency. The results in this paper are based on the work in [27].

This paper is structured as follows. First, we define in Section 2 a quantity called the *distance to integrality* and propose it as a measure of the number of nodes in the RRBB enumeration tree. To support this proposal, we provide computational evidence that the size of the RRBB tree is proportional to this distance. Since we cannot know the distance to integrality before actually solving (1), we next show in Section 3 that $\|\boldsymbol{A}^\dagger\|_2$, where $\boldsymbol{A}^\dagger$ denotes the Moore-Penrose generalized inverse of the matrix $\boldsymbol{A}$ in (1), is a key factor for bounding this distance; we refer to it as the *conditioning factor* of the ILS problem. In Section 4 we give a preconditioning method to reduce $\|\boldsymbol{A}^\dagger\|_2$ using lattice reduction techniques. In Section 5 we propose a set of valid box constraints that can help accelerate the RRBB method. Finally, in Section 6 we show computational results supporting the claims that by preconditioning to reduce $\|\boldsymbol{A}^\dagger\|_2$ before applying RRBB, the size of the RRBB tree can be reduced significantly; that the preconditioning combined with the proposed set of box constraints can significantly reduce the computational time of RRBB; and that the resulting RRBB method can outperform the Schnorr and Eucher method [35], a widely used discrete search method, on some types of problem data.

2 Distance to Integrality and Size of the BB Tree

Let $\boldsymbol{x}^{\mathrm{R}}$ denote the optimal solution to the RLS problem (2) and $\boldsymbol{x}^{\mathrm{I}}$ the solution to the ILS problem (1). We define the *distance to integrality* as the Euclidean distance between the real and integer optimal solutions to the ILS problem:

$$d(\boldsymbol{x}^{\mathrm{R}},\boldsymbol{x}^{\mathrm{I}}) = \left\|\boldsymbol{x}^{\mathrm{R}}-\boldsymbol{x}^{\mathrm{I}}\right\|_2.$$

We are interested in the distance to integrality because it is a useful predictor of the size of the BB tree explored by CPLEX. The following computational results support this claim by showing that for ILS problems, as the root node $\boldsymbol{x}^{\mathrm{R}}$ is closer to the leaf node $\boldsymbol{x}^{\mathrm{I}}$, then the size of the RRBB tree is generally smaller.

All numerical experiments were performed on a Macbook 2.26GHz Intel Core 2 Duo machine with 4GB memory running Mac OS X 10.6.6 with one thread. For the RRBB method, we call CPLEX 12.5 from MATLAB 7.7.0 via the MATLAB toolbox of CPLEX Optimization Studio v12.5. All CPLEX settings are set to their defaults.

We randomly generate instances using the commonly used linear model

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{v}, \tag{3}$$

where the noise vector $\boldsymbol{v} \in \mathbb{R}^n$ is randomly generated by $\sigma * \texttt{randn(m,1)}$, the coefficient matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is randomly generated in two different ways (see the next paragraph), and $\boldsymbol{x} \in \mathbb{Z}^n$ is randomly generated by $10 * \texttt{round(rand(n,1))}$. Here `randn(m,1)` is the MATLAB function that returns an m-dimensional vector containing pseudorandom values drawn from the standard normal distribution $N(0,1)$, `rand(m,1)` is the MATLAB function that returns an m-dimensional vector containing pseudorandom values drawn from the standard uniform distribution on the interval $(0,1)$, and $\texttt{round}(\boldsymbol{x})$ is the MATLAB function that rounds the elements of $\boldsymbol{x}$ to the nearest integer.

We considered two different types of randomly generated coefficient matrices:

1. Type 1: $\boldsymbol{A} = \texttt{rand(n,n)}$, $n = 2:2:38$, $\sigma = 0.05:0.05:1$ (380 instances in total);
2. Type 2: $\boldsymbol{A} = \texttt{randn(n,n)}$, $n = 2:2:40$, $\sigma = 0.05:0.05:1$ (400 instances in total),

where we use MATLAB notation for loops; for example, $n = 1:2:5$ is equivalent to $n = 1, 3, 5$, i.e., n ranges from 1 to 5 in steps of 2. The slightly different numbers of instances for each type is due to the MATLAB memory limit.

Figure 1 gives the graphs of the number of nodes of the search tree versus the distance to integrality for both types of instances (logarithmic scales are used for both axes). The experimental correlations between the two quantities in logarithmic scales are:

$$\text{Type 1: } 0.69 \quad \text{and} \quad \text{Type 2: } 0.75.$$

These results lead us to propose ways to bound, and hence to reduce, the distance to integrality for ILS problems.

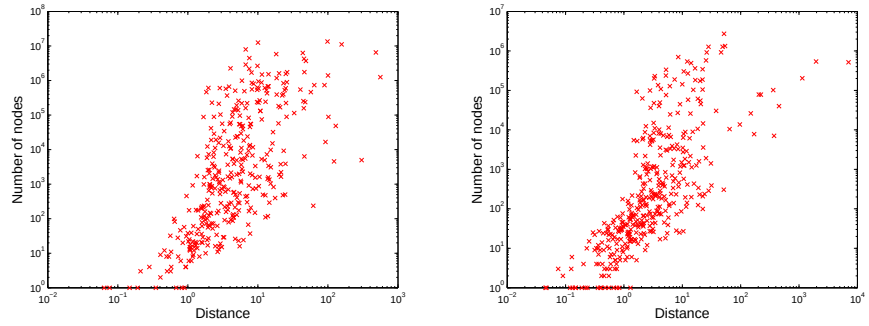


Fig. 1 Number of branch-and-bound nodes versus distance to integrality

3 An Upper Bound on the Distance to Integrality

In this section we give a theoretical upper bound on $d(\boldsymbol{x}^{\mathrm{R}}, \boldsymbol{x}^{\mathrm{I}})$ and show how it provides a means to reduce the distance to integrality.

3.1 Upper Bound

Using $\boldsymbol{A}^{\dagger}$, the Moore-Penrose generalized inverse of $\boldsymbol{A}$, we can write $\boldsymbol{x}^{\mathrm{R}} = \boldsymbol{A}^{\dagger}\boldsymbol{y}$ and $\boldsymbol{A}^{\dagger}\boldsymbol{A} = \boldsymbol{I}$. Therefore

$$d(\boldsymbol{x}^{\mathrm{R}}, \boldsymbol{x}^{\mathrm{I}}) = \|\boldsymbol{x}^{\mathrm{R}} - \boldsymbol{x}^{\mathrm{I}}\|_2 = \|\boldsymbol{A}^{\dagger}\boldsymbol{y} - \boldsymbol{x}^{\mathrm{I}}\|_2 = \|\boldsymbol{A}^{\dagger}(\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^{\mathrm{I}})\|_2,$$

leading to

$$d(\boldsymbol{x}^{\mathrm{R}}, \boldsymbol{x}^{\mathrm{I}}) \leq \|\boldsymbol{A}^{\dagger}\|_2 \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^{\mathrm{I}}\|_2, \tag{4}$$

where the equality holds if $\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^{\mathrm{I}}$ lies in the space spanned by the left singular vectors of $\boldsymbol{A}$ corresponding to the smallest singular value of $\boldsymbol{A}$.

Let us first consider the term $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^{\mathrm{I}}\|_2$. Recall that a square integer matrix is said to be unimodular if its determinant equals ± 1, and that the inverse of a unimodular matrix is an integer matrix. If $\boldsymbol{U} \in \mathbb{Z}^{n \times n}$ is unimodular and $\boldsymbol{z} = \boldsymbol{U}^{-1}\boldsymbol{x}$, then the ILS problem (1) is equivalent to

$$\min_{\boldsymbol{z} \in \mathbb{Z}^n} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{U}\boldsymbol{z}\|_2^2, \tag{5}$$

and the RLS problem (2) is equivalent to

$$\min_{\boldsymbol{z} \in \mathbb{R}^n} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{U}\boldsymbol{z}\|_2^2. \tag{6}$$

Let the solutions to (5) and (6) be denoted by $\boldsymbol{z}^{\mathrm{I}}$ and $\boldsymbol{z}^{\mathrm{R}}$, respectively. Then $\boldsymbol{z}^{\mathrm{I}} = \boldsymbol{U}^{-1}\boldsymbol{x}^{\mathrm{I}}$, $\boldsymbol{z}^{\mathrm{R}} = \boldsymbol{U}^{-1}\boldsymbol{x}^{\mathrm{R}}$, and analogously to (4):

$$d(\boldsymbol{z}^{\mathrm{R}}, \boldsymbol{z}^{\mathrm{I}}) \leq \|(\boldsymbol{A}\boldsymbol{U})^{\dagger}\|_2 \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{U}\boldsymbol{z}^{\mathrm{I}}\|_2. \tag{7}$$

Since $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^{\mathrm{I}}\|_2 = \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{U}\boldsymbol{z}^{\mathrm{I}}\|_2$, this means that the second term in the upper bound (4) is unaffected by unimodular transformations $\boldsymbol{U}$ of $\boldsymbol{x}$ to $\boldsymbol{z}$. However, $\|(\boldsymbol{A}\boldsymbol{U})^{\dagger}\|_2 \neq \|\boldsymbol{A}^{\dagger}\|_2$ in general, therefore it is in principle possible to improve the bound on the distance to integrality by using a unimodular transformation $\boldsymbol{U}$. In other words, it may be possible to decrease the distance to integrality by transforming the problem using a unimodular matrix $\boldsymbol{U}$ that minimizes $\|(\boldsymbol{A}\boldsymbol{U})^{\dagger}\|_2$. We provide a practical method to find a good $\boldsymbol{U}$ in Section 4.

Because the term $\|\boldsymbol{A}^{\dagger}\|$ becomes key to bounding the distance to integrality, we call it the *conditioning factor* of the ILS problem, and the process of using the unimodular transformation to reduce the conditioning factor is referred to as *preconditioning*.

3.2 Numerical Illustration

We illustrate the relationship between $d(\boldsymbol{x}^{\mathrm{R}}, \boldsymbol{x}^{\mathrm{I}})$ and the tree size using a 2-dimensional example, a basic branch-and-bound algorithm with depth first search, and the least feasible variable selection strategy for branching.

Consider the following data for the example:

$$\boldsymbol{A} = \begin{bmatrix} 8.2279 & 5.4482 \\ 8.4560 & 5.5348 \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} 16.4771 \\ 17.2832 \end{bmatrix}, \quad \boldsymbol{U} = \begin{bmatrix} 21 & 2 \\ -32 & -3 \end{bmatrix}.$$

Then $\|\boldsymbol{A}^{\dagger}\|_2 = 26.6411$ and $\|(\boldsymbol{AU})^{\dagger}\|_2 = 3.0627$, hence the unimodular transformation $\boldsymbol{U}$ reduces $\|\boldsymbol{A}^{\dagger}\|_2$ significantly. (We explain how $\boldsymbol{U}$ was computed in Section 4.)

Fig. 2 is the RRBB tree for the original ILS problem, where the root node is the real LS solution $\boldsymbol{x}^{\mathrm{R}} = [5.5, -5.4]^T$ and the leaf node with the star sign is the ILS solution $\boldsymbol{x}^{\mathrm{I}} = [4, -3]^T$. Thus $d(\boldsymbol{x}^{\mathrm{R}}, \boldsymbol{x}^{\mathrm{I}}) = 2.8302$. In this example, we first add the constraint $x_2 \geq -5$ and get the solution $[5.3, -5]^T$, whose second entry satisfies the equality $x_2 = -5$. Then we add the constraint $x_1 \leq 5$ and get the solution $[5, -4.5]^T$, whose first entry satisfies $x_1 = 5$. Subsequently we add $x_2 \geq -4$, $x_1 \leq 4$, $x_2 \geq -3$, $x_1 \geq 4$ and get solutions that have similar behavior along the branch we traverse from $\boldsymbol{x}^{\mathrm{R}}$ to $\boldsymbol{x}^{\mathrm{I}}$.

Fig. 3 is the RRBB tree for the reduced problem, where $\boldsymbol{z}^{\mathrm{R}} = [65.1, -5.9]^T$ and $\boldsymbol{z}^{\mathrm{I}} = [65, -6]^T$. In this case, $d(\boldsymbol{z}^{\mathrm{R}}, \boldsymbol{z}^{\mathrm{I}}) = 0.1414 < d(\boldsymbol{x}^{\mathrm{R}}, \boldsymbol{x}^{\mathrm{I}})$, and we see that only two constraints ($z_1 \leq 65$ and $z_2 \leq -6$) are added along the branch leading to the ILS solution.

Our computational results in Section 6 show a similar impact for larger instances of ILS.

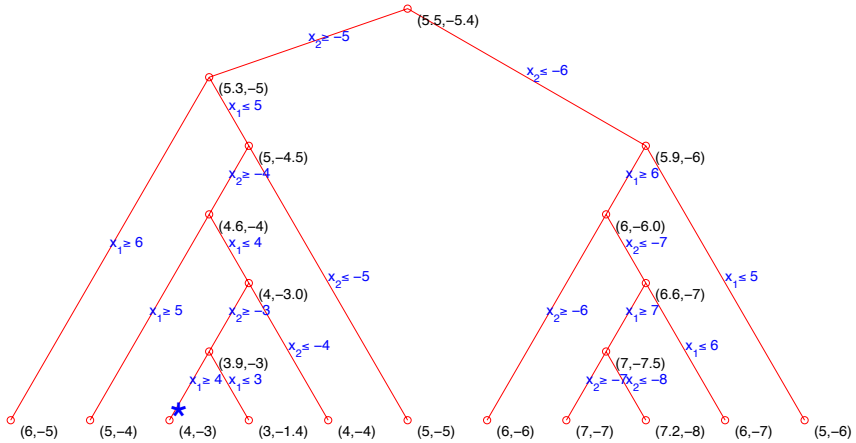


Fig. 2 The RRBB tree for the original ILS problem

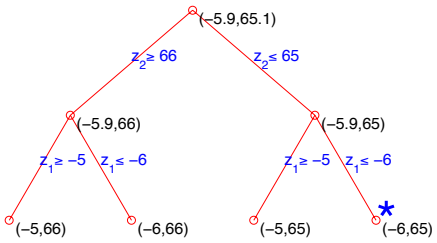


Fig. 3 The RRBB tree for the reduced ILS problem.

4 Preconditioning to Reduce the Conditioning Factor

We now explain our technique to reduce the conditioning factor $\|\boldsymbol{A}^{\dagger}\|_2$ using a unimodular transformation $\boldsymbol{U}$. Since it is much simpler to compute the Frobenius-norm of a matrix than its 2-norm, and since $\|\boldsymbol{A}^{\dagger}\|_2 \leq \|\boldsymbol{A}^{\dagger}\|_F \leq \sqrt{n}\|\boldsymbol{A}^{\dagger}\|_2$, we choose to reduce $\|\boldsymbol{A}^{\dagger}\|_F$ instead. The computational results reported in Section 6 show the impact of this preconditioning.

First we recall the well-known Lenstra-Lenstra-Lovász (LLL) lattice reduction [29]. Given a lattice basis matrix $\boldsymbol{B}$ which has long column vectors, lattice reduction looks for a unimodular matrix $\boldsymbol{Z}$ such that the columns of the new basis matrix $\boldsymbol{BZ}$ are short. The best-known lattice reduction is the LLL reduction, which can be described as a matrix factorization:

$$\boldsymbol{Q}^T\boldsymbol{BZ} = \begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{bmatrix}, \tag{8}$$

where $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ is orthogonal, $\boldsymbol{R}$ is upper triangular and its elements satisfy

$$|r_{ij}| \leq |r_{ii}|/2, \quad \delta r_{ii}^2 \leq r_{i,i+1}^2 + r_{i+1,i+1}^2, \quad j = i+1,\ldots,n, \quad i = 1,\ldots,n-1, \tag{9}$$

where δ is a parameter satisfying $1/4 < \delta \leq 1$. In (9), the first and second inequalities are referred to as the size-reduce condition and the Lovász condition respectively. If $\boldsymbol{B}$ is an integer matrix and $\delta \neq 1$, it can be shown that the LLL algorithm is a polynomial-time algorithm [29].

We use the LLL reduction to reduce $\boldsymbol{A}^{\dagger}$. For a unimodular matrix $\boldsymbol{U} \in \mathbb{Z}^{n\times n}$, it is easy to verify that $((\boldsymbol{AU})^{\dagger})^T = (\boldsymbol{A}^{\dagger})^T\boldsymbol{U}^{-T}$. Thus in order to minimize $\|(\boldsymbol{AU})^{\dagger}\|_F$, we perform the LLL reduction on $(\boldsymbol{A}^{\dagger})^T$ and obtain a unimodular matrix $\boldsymbol{Z}$. Then we take $\boldsymbol{U} = \boldsymbol{Z}^{-T}$ and solve the ILS problem (5) by branch-and-bound. The unimodular matrix $\boldsymbol{U}$ used in the example in Section 3.2 was obtained in this way.

A remark about the lattice reduction is in order here. When one solves the ILS problem (1) by the discrete search approach, typically one performs a lattice reduction (usually LLL) on $\boldsymbol{A}$, while here we perform a lattice reduction on $(\boldsymbol{A}^{\dagger})^T$. In lattice theory, $\mathcal{L}((\boldsymbol{A}^{\dagger})^T)$ is referred to as the dual lattice of $\mathcal{L}(\boldsymbol{A})$. If the LLL reduction algorithm is used in the discrete approach, it was shown in [39] that the size-reduce condition (i.e., the first inequality in (9)) is not necessary for the off-diagonal entries above the supradiagonal entries. In fact only the diagonal entries of $\boldsymbol{R}$ affects the search speed. The size-reduce condition for the supradiagonal

entries of $\boldsymbol{R}$ should be met if it can result in changes in the diagonal entries. However, in our RRBB approach we perform size reductions fully in applying the LLL reduction algorithm to $(\boldsymbol{A}^\dagger)^T$ because we want to reduce all entries of $(\boldsymbol{A}^\dagger)^T$.

In the remainder of this section, we explain in detail how we compute the unimodular matrix $\boldsymbol{U}$. We first compute the QL factorization of $\boldsymbol{A}$ using the minimum column pivoting strategy (see [15, Sec. 3]):

$$\boldsymbol{Q}^T\boldsymbol{A}\boldsymbol{P} = \begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{L}\end{bmatrix}, \tag{10}$$

where $\boldsymbol{Q} = [\underset{m-n}{\boldsymbol{Q}_1}, \underset{n}{\boldsymbol{Q}_2}] \in \mathbb{R}^{m\times m}$ is orthogonal, $\boldsymbol{P}$ is a permutation matrix (and thus unimodular), and $\boldsymbol{L}$ is lower triangular. Specifically, in the first step we find a column of $\boldsymbol{A}$ with the minimum 2-norm and interchange it with the last column. Then we perform a Householder transformation to zero $\boldsymbol{A}(1:m-1,n)$. In the second step, we find the column in the remaining matrix $\boldsymbol{A}(1:m-1,1:n-1)$ with the minimum 2-norm and interchange the whole column in $\boldsymbol{A}$ with the 2nd to the last column of $\boldsymbol{A}$. Then we perform a Householder transformation to zero $\boldsymbol{A}(1:m-2,n-1)$. We repeat these steps until we obtain (10).

Now it follows from (10) that

$$\boldsymbol{Q}^T(\boldsymbol{A}^\dagger)^T\boldsymbol{P} = \begin{bmatrix}0\\ \boldsymbol{L}^{-T}\end{bmatrix}. \tag{11}$$

Since $\|\boldsymbol{Q}^T(\boldsymbol{A}^\dagger)^T\boldsymbol{P}\|_F = \|\boldsymbol{L}^{-T}\|_F$, we only need to reduce $\|\boldsymbol{L}^{-T}\|_F$. Thus we compute $\tilde{\boldsymbol{R}} = \boldsymbol{L}^{-T}$, and then apply the LLL reduction algorithm to $\boldsymbol{R}$ to obtain

$$\tilde{\boldsymbol{Q}}^T\tilde{\boldsymbol{R}}\boldsymbol{Z} = \boldsymbol{R}, \tag{12}$$

where $\tilde{\boldsymbol{Q}} \in \mathbb{R}^{n\times n}$ is orthogonal, $\boldsymbol{Z} \in \mathbb{Z}^{n\times n}$ is unimodular and $\boldsymbol{R}$ is upper triangular satisfying the LLL reduced conditions in (9).

We conclude with an explanation of the computational benefits of using the QL factorization (10) with minimum column pivoting. Note that $\tilde{\boldsymbol{R}}$ in (12) is already in upper triangular form. Thus, when we apply the LLL reduction algorithm, the QR factorization of $\tilde{\boldsymbol{R}}$, which is usually the first step of the algorithm, is not needed. The LLL algorithm tries to move large diagonal entries of $\tilde{\boldsymbol{R}}$ to the bottom right corner and small diagonal entries to the upper left corner by column permutations (cf. (9)) . The minimum column pivoting strategy used in computing (10) helps this process, i.e., it reduces the cost of the LLL algorithm by decreasing the number of permutations in the reduction process.

5 Box Constraints

In order to further reduce the size of the RRBB tree, we show how to compute box constraints that reduce the feasible set of the ILS problem while preserving the optimal solution.

Suppose that the QR factorization of $\boldsymbol{AU}$ is

$$[\bar{\boldsymbol{Q}}_1, \bar{\boldsymbol{Q}}_2]^T\boldsymbol{A}\boldsymbol{U} = \begin{bmatrix}\bar{\boldsymbol{R}}\\ \boldsymbol{0}\end{bmatrix},$$

where $[\underset{n}{\bar{\boldsymbol{Q}}_1}, \underset{m-n}{\bar{\boldsymbol{Q}}_2}] \in \mathbb{R}^{m\times m}$ is orthogonal and $\bar{\boldsymbol{R}}$ is upper triangular, then with $\bar{\boldsymbol{y}} = \bar{\boldsymbol{Q}}_1^T \boldsymbol{y}$, the ILS problem (5) is transformed to the equivalent ILS problem:

$$\min_{\boldsymbol{z}\in\mathbb{Z}^n} \left\|\bar{\boldsymbol{y}} - \bar{\boldsymbol{R}}\boldsymbol{z}\right\|_2^2. \tag{13}$$

Suppose that the solution of the new ILS problem (13) satisfies the following inequality for some β:

$$\left\|\bar{\boldsymbol{y}} - \bar{\boldsymbol{R}}\boldsymbol{z}\right\|_2^2 \le \beta^2. \tag{14}$$

It is easy to show that (14) is a hyperellipsoid with center $\bar{\boldsymbol{R}}^{-1}\bar{\boldsymbol{y}}$ by applying the singular value decomposition to $\boldsymbol{R}$. The value of β is chosen as

$$\beta = \|\bar{\boldsymbol{y}} - \bar{\boldsymbol{R}}\boldsymbol{z}^{\mathrm{B}}\|_2$$

where $\boldsymbol{z}^{\mathrm{B}} \in \mathbb{Z}^n$ is the so-called Babai point [9]:

$$z_n^{\mathrm{B}} = \lfloor \bar{y}_n / r_{nn} \rceil, \quad z_k^{\mathrm{B}} = \left\lfloor \left(\bar{y}_k - \sum_{j=k+1}^{n} r_{kj} z_j^{\mathrm{B}}\right)/r_{nn} \right\rceil, \quad k = n-1, \ldots, 1.$$

While any integer point can be used, the Babai point can easily be computed and is often used as an approximate solution to the ILS problem [9,5].

To derive an upper and a lower bound on $\boldsymbol{z}$ satisfying (14), observe that we can write the k-th entry of $\boldsymbol{z}$ as

$$z_k = \boldsymbol{e}_k^T \bar{\boldsymbol{R}}^{-1}(\bar{\boldsymbol{R}}\boldsymbol{z} - \bar{\boldsymbol{y}}) + \boldsymbol{e}_k^T \bar{\boldsymbol{R}}^{-1}\bar{\boldsymbol{y}}.$$

Then by the Cauchy-Schwarz inequality and (14), we have

$$-\beta\|\bar{\boldsymbol{R}}^{-T}\boldsymbol{e}_k\|_2 + \boldsymbol{e}_k^T \bar{\boldsymbol{R}}^{-1}\bar{\boldsymbol{y}} \le z_k \le \beta\|\bar{\boldsymbol{R}}^{-T}\boldsymbol{e}_k\|_2 + \boldsymbol{e}_k^T \bar{\boldsymbol{R}}^{-1}\bar{\boldsymbol{y}}.$$

In other words, we have

$$\boldsymbol{l} \le \boldsymbol{z} \le \boldsymbol{u}, \tag{15}$$

where

$$l_k = -\left\lceil \beta\|\bar{\boldsymbol{R}}^{-T}\boldsymbol{e}_k\|_2 + \boldsymbol{e}_k^T \bar{\boldsymbol{R}}^{-1}\bar{\boldsymbol{y}} \right\rceil, \quad u_k = \left\lfloor \beta\|\bar{\boldsymbol{R}}^{-T}\boldsymbol{e}_k\|_2 + \boldsymbol{e}_k^T \bar{\boldsymbol{R}}^{-1}\bar{\boldsymbol{y}} \right\rfloor, \quad k = 1, \ldots, n.$$

Note that here $\bar{\boldsymbol{R}}^{-T}\boldsymbol{e}_k$ can easily be obtained by solving a lower triangular linear system. Using the approach in [14], we can show that the box (15) is the smallest box that includes the hyperellipsoid (14) and whose edges are parallel to the axes of the coordinate system. Thus we may add the constraint (15) to the ILS problem (13) when we solve it with RRBB. When including these box constraints, we tighten them by rounding up the lower bounds and rounding down the upper bounds.

We conclude this section by pointing out that while [14] provides a method to compute the box, the problem solved there is different from the ILS problem considered in this paper, and the box computed there is for a complete different purpose. Furthermore, the approach proposed in [14] is an enumeration approach, which enumerates integer points within a region, and it is different from the branch and bound approach considered in this paper. Thus, using box-constraints to accelerate convergence of the branch and bound approach in the ILS context is a new idea.

6 Computational Results

We present three sets of computational results. All the test instances were generated randomly in the same way as in Section 2 using $\sigma = 0.05$ and $\sigma = 0.4$, where σ is the standard deviation of each entry of the random noise vector $\boldsymbol{v}$ in the linear model (3). This gives four types of randomly generated instances to test the algorithms.

We considered 50 instances of size 100×100 for each type of randomly generated entries. We imposed a time limit of 1000 sec for each run. All the computations were performed with the same computer and setup as described in Section 2. Computational time statistics are reported in Table 1; these are total computation times for performing all the computations needed for each algorithm. We observe that the computational costs of the preconditioning and the box constraint computation are negligible; the times required is essentially spent in the branch-and-bound process. For each of the four types of instances, the lowest average computation time is indicated in bold in Table 1. The number of instances that were not solved by a particular algorithm within the time limit are reported as "dnt" (did not terminate) in Table 1.

	min (sec)	average (sec)	max (sec)
50 `rand` instances with $\sigma = 0.05$			
No preconditioning	0.344	0.882	7.108
Preconditioning	0.198	**0.350**	0.592
No preconditioning + box constraints	0.272	0.950	7.108
Preconditioning + box constraints	0.178	0.379	0.765
50 `randn` instances with $\sigma = 0.05$			
No preconditioning	0.166	0.455	1.683
Preconditioning	0.168	0.303	0.465
No preconditioning + box constraints	0.059	0.314	1.364
Preconditioning + box constraints	0.004	**0.014**	0.077
50 `rand` instances with $\sigma = 0.4$			
No preconditioning	103.7	625.6	1000 (15 dnt)
Preconditioning	50.4	256.0	851.8
No preconditioning + box constraints	106.1	569.3	1000 (11 dnt)
Preconditioning + box constraints	50.4	**245.6**	881.9
50 `randn` instances with $\sigma = 0.4$			
No preconditioning	1.1	21.1	520.6
Preconditioning	0.7	**4.3**	29.2
No preconditioning + box constraints	0.7	14.2	191.0
Preconditioning + box constraints	0.8	4.4	30.6

Table 1 Total Computational Time Statistics (dnt stands for "did not terminate")

We first examine the impact of the preconditioning on $\|A^\dagger\|$ and on the distance to integrality. The results in Figures 4 and 5 correspond to $\sigma = 0.05$ and $\sigma = 0.4$ respectively. We see that the preconditioning is consistently successful at significantly reducing the norm of $\|A^\dagger\|$, and that this reduction in $\|A^\dagger\|$ translates into a smaller distance to integrality.

The next question is whether the preconditioning and the box constraints have a positive impact on the computational cost for solving the ILS problems. Figures 6 and 7 report, in the form of performance profiles [19], and for the same instances

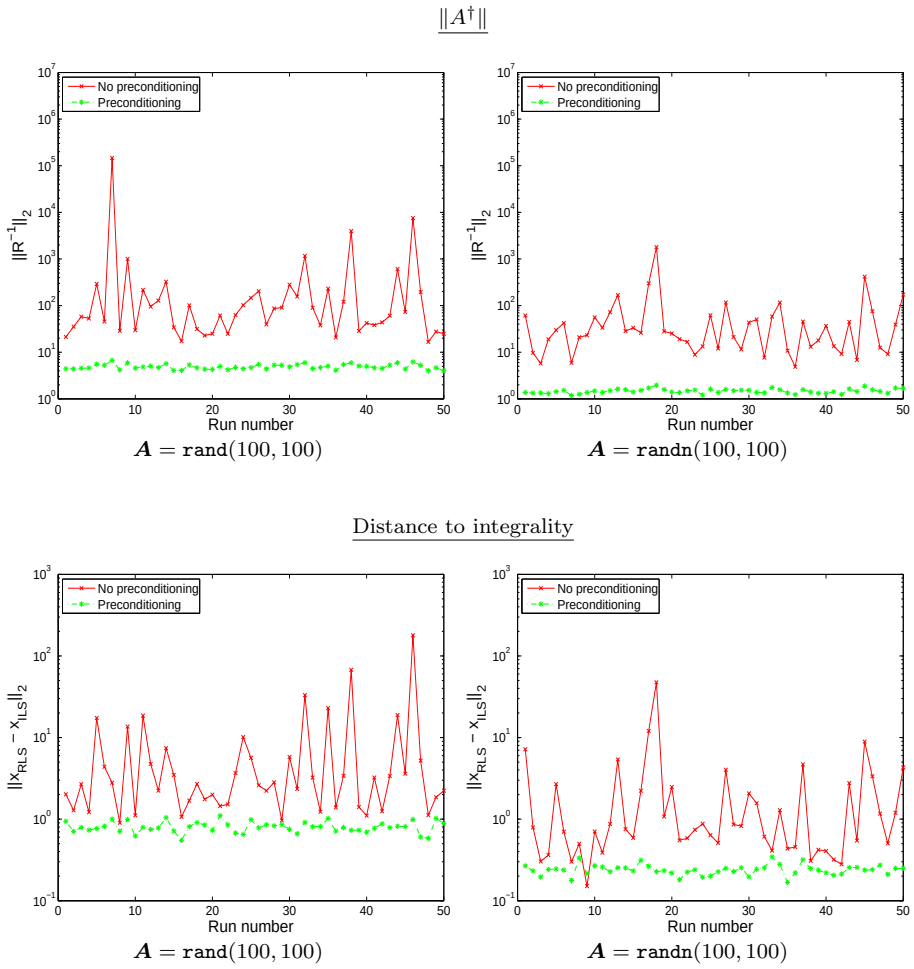


Fig. 4 Results over 100 randomly generated instances (50 rand and 50 randn) with $\sigma = 0.05$

as in Figures 4 and 5, the impact of preconditioning and/or of the box constraints on the performance of RRBB in terms of the CPU time and of the number of nodes in the enumeration tree.

We observe from Figure 6 that for $\sigma = 0.05$ preconditioning alone generally improves the computation time and the number of nodes for the rand instances; on the other hand, its impact on solving randn instances is limited. While the impact that the box constraints have by themselves is limited, the combination of preconditioning and box constraints is always competitive and often offers the best performance both with respect to computation time and to the number of nodes. In particular, for the randn instances, the preconditioning together with the box constraints reduce the feasible set to a single point in all 50 instances, thus clearly outperforming the other combinations. This is particularly observable in Table 1 for "preconditioning + box constraints" for randn instances with $\sigma = 0.05$: the

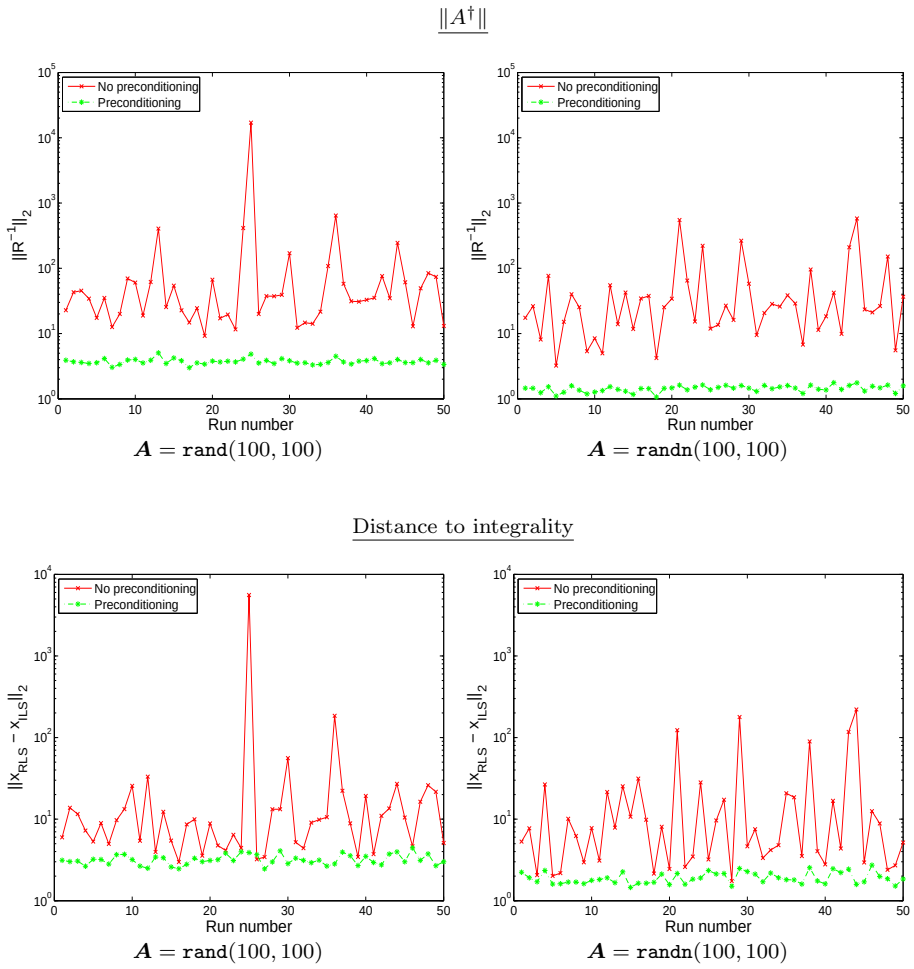


Fig. 5 Results over 100 randomly generated instances (50 rand and 50 randn) with $\sigma = 0.4$

times reported are consistently one order of magnitude lower than those of the other three combinations.

For the instances with $\sigma = 0.4$, Figure 7 shows that the box constraints have no significant effect, but that preconditioning (with or without box constraints) can reduce both the computation time and the number of nodes. The benefits from preconditioning for the instances with $\sigma = 0.4$ are also clearly seen in the bottom half of Table 1 where the average times for the approaches using preconditioning are consistently and significantly lower than those for the other two approaches. Moreover, the approaches using preconditioning were able to solve all the instances within the 1000-second time limit, while for the rand instances, the approaches without preconditioning failed to solve 26 of the 100 such instances.

The last set of results is reported in Figures 8 and 9. These results are for two additional sets of randomly generated instances of sizes 100×100 and 200×200 respectively. The purpose of these experiments is to compare RRBB with the com-

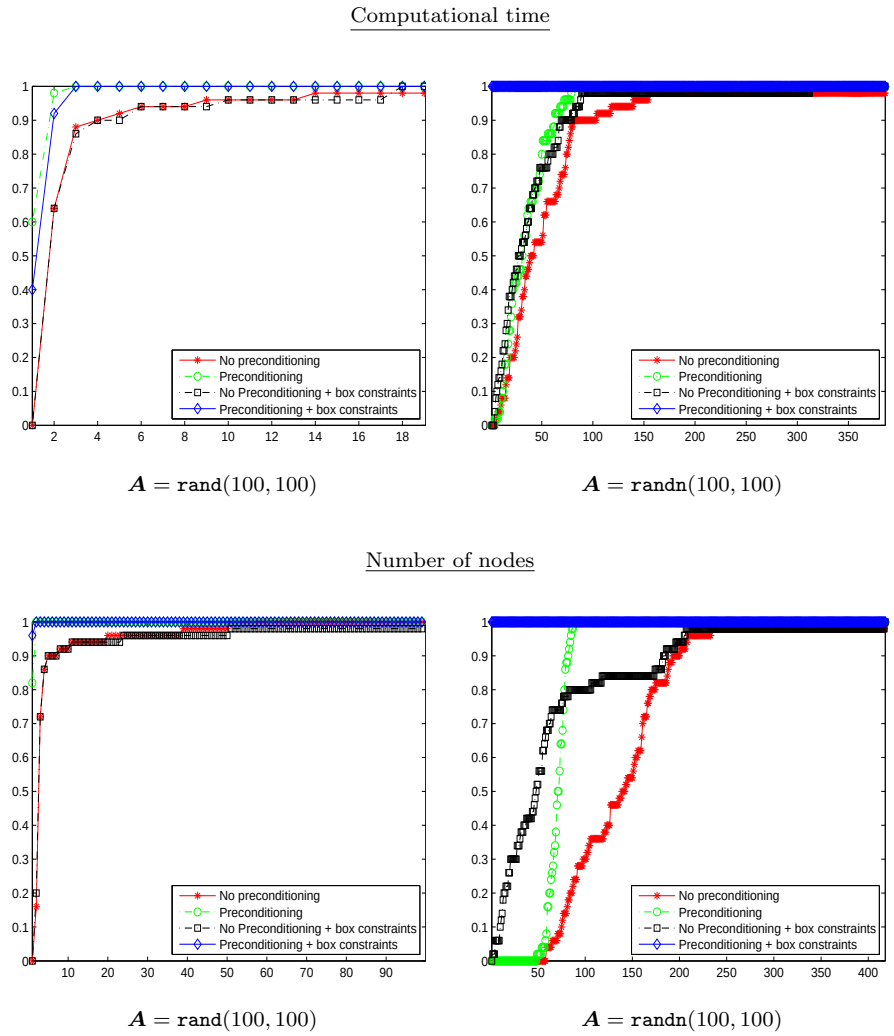


Fig. 6 Results over 100 randomly generated instances (50 rand and 50 randn) with $\sigma = 0.05$

bination of preconditioning and box constraints with the discrete search approach of Schnorr and Euchner [35] that is generally considered among the most efficient methods of this kind [5]. The LLL reduction algorithm was applied to $\boldsymbol{A}$ before applying this discrete search method. For a fair comparison, the implementation of the discrete search method (including the LLL reduction) was done in C code called from MATLAB. For a MATLAB implementation of the method, see [17].

We again use 50 instances for each type of randomly generated entries. We only report results for instances with $\sigma = 0.4$ because for $\sigma = 0.05$ both RRBB and the discrete search approach solved all the instances we tried within 1 sec. The time limit remained set to 1000 seconds. The results are summarized in the performance profiles of Figures 8 and 9.

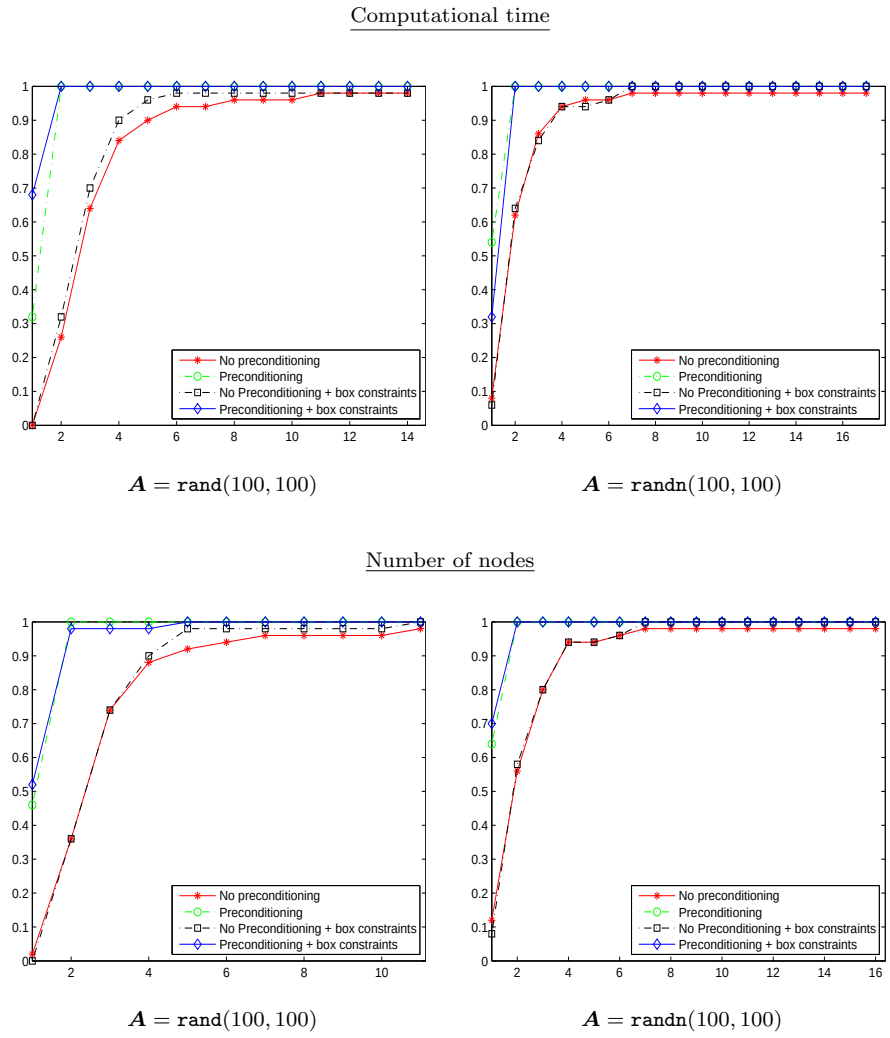
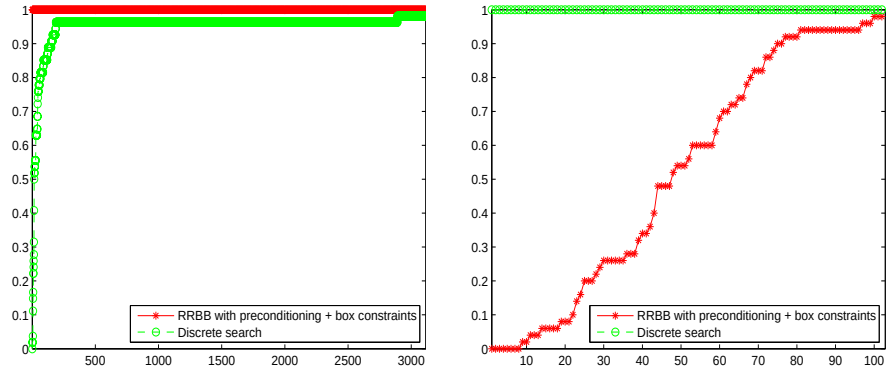


Fig. 7 Results over 100 randomly generated instances (50 rand and 50 randn) with $\sigma = 0.4$

The results differ significantly between the rand and randn instances. While the discrete search approach is clearly superior for the randn instances, the RRBB approach outperforms it consistently for the rand instances. We conclude that the proposed RRBB approach can offer a significant improvement over the well-known discrete search approach on some types of problem data.

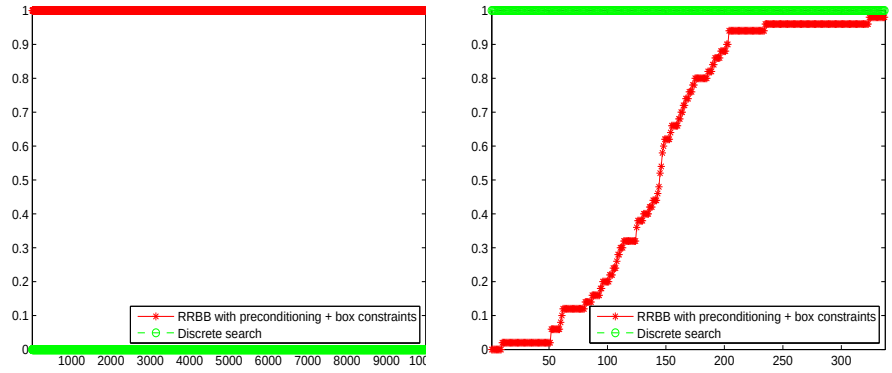
7 Conclusion

We proposed an improved RRBB method for the ILS problem and implemented it using CPLEX. After defining a quantity called the distance to integrality and providing evidence that the size of the RRBB enumeration tree is proportional



$\boldsymbol{A} = \texttt{rand}(100, 100)$ $\boldsymbol{A} = \texttt{randn}(100, 100)$

Fig. 8 Performance profiles using 100 randomly generated instances (50 rand and 50 randn) with $\sigma = 0.4$



$\boldsymbol{A} = \texttt{rand}(200, 200)$ $\boldsymbol{A} = \texttt{randn}(200, 200)$

Fig. 9 Performance profiles using 100 randomly generated instances (50 rand and 50 randn) with $\sigma = 0.4$

to it, we proved that the norm of the Moore-Penrose generalized inverse of the matrix of coefficients of the least-squares problem is a key factor for bounding this distance. We then proposed a preconditioning method to reduce this norm using lattice reduction techniques. We also derived a set of valid box constraints that help accelerate the RRBB method. Our computational results showed that the preconditioning significantly reduces the size of the RRBB enumeration tree, that the preconditioning combined with the proposed set of box constraints can significantly reduce the computational time of RRBB, and that the resulting RRBB method can offer, for some types of problem data, a significant improvement over the discrete search approach, a widely used method for solving ILS problems.

Acknowledgements We are grateful to two anonymous referees for their detailed criticisms that helped us improve the paper. We also acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada.

References

1. K. Aardal and F. Eisenbrand. The LLL algorithm and integer programming. In P.Q. Nguyen and B. Vallée, editors, *The LLL Algorithm*, Information Security and Cryptography, pp. 293–314. Springer Berlin Heidelberg, 2010.
2. K. Aardal and F. Heymann. On the structure of reduced kernel lattice bases. In M. Goemans and J. Correa, editors, *Integer Programming and Combinatorial Optimization*, volume 7801 of *Lecture Notes in Computer Science*, pp. 1–12, Springer, 2013.
3. K. Aardal, R. Weismantel, and L. A. Wolsey, Non-standard approaches to integer programming. *Discrete Applied Mathematics*, vol. 123, pp. 5–74, 2002.
4. K. Aardal and L.A. Wolsey. Lattice based extended formulations for integer linear equality systems. *Mathematical Programming*, vol. 121, pp. 337–352, 2010.
5. E. Agrell, T. Eriksson, A. Vardy, and K. Zeger. Closest point search in lattices. *IEEE Transactions on Information Theory*, vol. 48, pp. 2201–2214, 2002.
6. M. Ajtai and C. Dwork. A public-key cryptosystem with worst-case/average-case equivalence. *Proceedings of STOC*, pp. 284–293, 1997.
7. M. Ajtai, R. Kumar, and D. Sivakumar. Sampling short lattice vectors and the closest lattice vector problem. *Proceedings of CCC*, pp. 53–57, 2002.
8. D. Axehill. *Integer Quadratic Programming for Control and Communication*. PhD Thesis, Department of Electrical Engineering, Linköping Universiity, 2008.
9. L. Babai. On Lóvasz' lattice reduction and the nearest lattice point problem. *Combinatorica*, vol. 6, pp. 1–13, 1986.
10. D. Bienstock. Computational study of a family of mix-integer quadratic programming problems. *Mathematical Programming*, vol. 74, pp. 121–140, 1995.
11. P. van Emde Boas. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Technical Report Rep.81-04, Mathematics Institute, Amsterdam, The Netherlands, 1981.
12. M.R. Bremmer. *Lattice Basis Reduction, An Introduction to the LLL Algorithm and Its Applications*. CRC Press, 2012.
13. C. Buchheim, A. Caprara, and A. Lodi. An effective branch-and-bound algorithm for convex quadratic integer programming. *Mathematical Programming*, vol. 135, pp. 369–395, 2012.
14. X.-W. Chang, G. H. Golub. Solving ellipsoid-constrained integer least squares problems. *SIAM Journal on Matrix Analysis and Applications*, vol. 31, pp. 1071–1089, 2009.
15. X.-W. Chang and C.C. Paige. Euclidean distances and least squares problems for a given set of vectors. *Applied Numerical Mathematics*, vol. 57, pp. 1240–1244, 2007.
16. X.-W. Chang, J. Wen and X. Xie. Effects of the LLL reduction on the success probability of the Babai point and on the complexity of sphere decoding. *IEEE Transactions on Information Theory*, vol. 59, pp. 4915–4926, 2013.
17. X.-W. Chang, X. Xie and T. Zhou, MILES: MATLAB package for solving Mixed Integer LEast Squares problems, Version 2.0, October 2011. http://www.cs.mcgill.ca/∼chang/software.php.
18. IBM ILOG CPLEX Optimization Studio. http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/.
19. E.D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, vol. 91, pp. 201–213, 2002.
20. F. Eisenbrand. Integer programming and algorithmic geometry of numbers. In M. Jünger, T.M. Liebling, D. Naddef, G.L.Nemhauser, W.R. Pulleyblank, G.Reinelt, G.Rinaldi, and L.A. Wolsey, editors, *50 Years of Integer Programming 1958-2008*, pages 505–559. Springer Berlin Heidelberg, 2010.
21. U. Fincke and M. Pohst. A procedure for determining algebraic integers of given norm. *Proceedings of RUROCAL*, vol. 162, pp. 194–202, 1983.
22. G. Hanrot, X. Pujol and D. Stehle Algorithms for the shortest and closest lattice vector problems. *Proceedings of IWCC*, pp. 159–190, 2011.

23. A. Hassibi and S. Boyd. Integer parameter estimation in linear models with applications to GPS. *IEEE Transactions on Signal Processing*, vol. 46, pp. 2938–2952, 1998.
24. R. Kannan. Improved algorithms for integer programming and related lattice problems. *Proceedings of STOC*, pp. 99–108, 1983.
25. M. Kisialiou and Z.-Q. Luo. Performance analysis of quasi-maximumlikelihood detector based on semi-definite programming. *Proceedings of IEEE ICASSP*, pp. 433–436, 2005.
26. B. Krishnamoorthy and G. Pataki. Column basis reduction and decomposable knapsack problems. *Discrete Optimization*, vol. 6, pp. 242–270, 2009.
27. W.-Y. Ku. Lattice Preconditioning for the Real Relaxation Based Branch and Bound Method for Integer Least Squares Problems. MSc Thesis, School of Computer Science, McGill University, 2011.
28. A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, vol. 28, pp. 497–520, 1960.
29. A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, vol. 261, pp. 515–534, 1982.
30. H. W. Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, vol. 8, pp. 538–548, 1983.
31. S. Mehrotra and Z. Li. Branching on hyperplane methods for mixed integer linear and convex programming using adjoint lattices. *Journal of Global Optimization*, vol. 49, pp. 623–649, 2011.
32. D. Micciancio and P. Voulgaris. Faster exponential time algorithms for the shortest vector problem. *Proceedings of SODA*, pp.1468–1480, 2010.
33. D. Micciancio and P. Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. *SIAM Journal on Computing*, vol. 42, pp.1364–1391, 2013.
34. G. Pataki, M. Tural, and E.B. Wong. Basis reduction and the complexity of branch-and-bound. In *Proceedings of SODA*, pp. 1254–1261, 2010.
35. 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.
36. C.P. Schnorr. Progress on lll and lattice reduction. In P.Q. Nguyen and B.Vallée, editors, *The LLL Algorithm*, Information Security and Cryptography, pp. 145–178. Springer Berlin Heidelberg, 2010.
37. 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, 2001.
38. P. J. G. Teunissen and A. Kleusberg. *GPS for Geodesy.* Springer, 1998.
39. X. Xie, X.-W. Chang, and M. Al Borno. Partial LLL reduction. *Proceedings of IEEE GLOBECOM 2011*, 5 pages.