Numerical Methods for Underdetermined Box-constrained Integer Least Squares Problems

Jing Zhu

Master of Science

Computer Science

McGill University

Montreal,Quebec

2016-03-15

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Science in Computer Science

DEDICATION

This thesis is dedicated to my family.

ACKNOWLEDGEMENTS

First, I would like to thank my supervisor, Prof. Xiao-Wen Chang for his guidance, encouragement and support in the last two years. I can never forget his consecutive comments, patience and discussions between us about the thesis until midnight.

I would like to thank Jiequn for her the interesting discussions about ILS problems and her help in both the research and life.

I would also like to thank Nancy, who knows nothing about ILS problems even computer science but gives her best support to me when I was frustrated in the past two years.

There are also some lab mates and friends I need to give my thanks at McGill: Tianze, Xiangyu, Shuo, Can. I am very happy to know all of you and treasured the time spending with you.

Finally, I am so grateful to my family: my parents and grandparents, who give me the endless guidance, care, and my husband Yuanlu, who makes me feel much stronger and more positive. Without yours support, I can not keep smiles when I faced challenges in both the research and job hunting.

ABSTRACT

Integer least squares (ILS) is an important class of optimization problems, which can arise in many applications, such as communications, cryptography and cryptanalysis and global navigation satellite systems. This thesis is concerned with solving the underdetermined box constrained ILS (UBILS) problems. For the two existing algorithms, the direct tree search (DTS) algorithm and the partial regularization (PR) algorithm, we propose to incorporate some lower bounds to speed up the search process. Simulation results show that the proposed lower bounds can make the search process of the DTS algorithm perform more efficiently than the original one. Then we propose a modified DTS algorithm by partially using a best-first search strategy in the search process. Numerical tests results indicate that the new search algorithm is very effective in improving the efficiency of the DTS algorithm with or without incorporating the proposed lower bounds.

ABRÉGÉ

Les moindres carrs en nombres entiers (ILS) est une mthode d'optimisation des problmes, qui peut tre applique dans beaucoup de domaines, comme la communication, la cryptographie et la cryptanalyse les systmes de navigation mondiaux par satellite. Dans cette thse, on tudiera trois types de problmes concernant les moindres carrs en nombres entiers: les problmes de moindres carrs en nombre entiers classiques(OILS), les problmes hyperdtermins(OBILS) et hypodtermins(UBILS) avec les contraintes de boites. Dans un premier temps, on va faire un rappel de diffrents algorithmes pour rsoudre ces trois types de problmes, suivi par une prsentation de diffrents minorations afin d'augmenter l'efficacit du processus de recherche. Pour les deux algorithmes existants l'algorithme de recherche en arbre directe(DTS) et l'algorithme de rgularisation partielle (RP), on va proposer les minorations correspondantes afin d'acclrer le processus de recherche. On montrera les rsultats de simulation qui justifient que les minorations qu'on propose peut faire le processus de recherche par l'algorithme de recherche en arbre directe plus performante que la mthode classique. Puis, on va proposer un nouveau algorithme de recherche pour rsoudre les problmes UBILS, qui est en fait une modification directe sur l'algorithme de recherche en arbre directe en utilsant la stratgie de recherche le meilleur en premier(best-first) dans le processus de recherche. Les rsultats de test numrique montrera que le nouvel algorithme de recherche est plus efficace que la mthode de recherche en arbre directe existante.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

1.1 Integer Least Squares Problems

Suppose we have the following linear model

$$\boldsymbol{y} = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{v}, \ \ \boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I}), \tag{1.1}$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ is a given matrix, $\boldsymbol{y} \in \mathbb{R}^m$ is an observation vector, $\hat{\boldsymbol{x}} \in \mathbb{Z}^n$ is a parameter vector, and $\boldsymbol{v} \in \mathbb{R}^m$ is a noise vector with Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$. Then, we estimate $\hat{\boldsymbol{x}}$ in (1.1) by solving the following minimization problem:

$$\min_{\boldsymbol{x}\in\mathbb{Z}^n} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2. \tag{1.2}$$

This problem is referred to as an *integer least squares* (ILS) *problem* and the solution of it is the maximum likelihood estimate of the parameter vector $\hat{\boldsymbol{x}}$ in (1.1). To distinguish (1.2) with other types of ILS problems, we also refer to it as an *ordinary integer least squares* (OILS) *problem*. As in the literature, we assume that $\boldsymbol{A}$ in the OILS problem (1.2) has full column rank, and then $\boldsymbol{A}$ can be referred to as a lattice generator matrix, which can generate the lattice $\mathcal{L}(\boldsymbol{A}) = \{\boldsymbol{A}\boldsymbol{x} : \boldsymbol{x} \in \mathbb{Z}^n\}$, see, e.g., [1]. The ILS problem (1.2) is equivalent to finding the closest point to $\boldsymbol{y}$ in the lattice $\mathcal{L}(\boldsymbol{A})$, so it can also be referred to as the *closest point problem*. The ILS problem has applications in various fields, such as wireless communications (see, e.g., [43, 1, 24]), cryptography and cryptanalysis (see, e.g., [25, 33, 6, 34]), GPS positioning (see, e.g., [35], [53]), and number theory (see, e.g., [32]), etc.

Different from a real least squares problem, the ILS problem has been proved to be NP-hard (see [41] and [54]). Various algorithms have been developed to solve the ILS problem, such as, the Monte Carlo probabilistic algorithms, see [2, 3, 9, 34], the Voronoi cell based approach, see [49] and [42], and the discrete enumeration algorithm, which enumerates all candidate lattice vectors in some given region to find the optimal solution, see, e.g., [45, 29, 36, 37, 47], and the real relaxation based branch and bound approach [4]. The discrete enumeration approach is usually the most efficient one, and in this thesis, all the algorithms we will discuss later use this approach.

The discrete enumeration approach usually consists of two phases: reduction and search. The reduction process is to reduce the problem (1.2) to an equivalent new problem, which can make the search process easier and more efficient. The search process aims to find the optimal solution of the reduced problem. For the reduction process, there are two well-known reduction strategies: the Korkine-Zolotareff (KZ) reduction [38] and the Lenstra-Lenstra-Lovász (LLL) reduction [39]. Due to the expensive computated cost of the KZ reduction, the LLL reduction is usually used in practice for solving OILS problems. Since the KZ reduction can make the search process more efficient than the LLL reduction, it is often used in applications when one solves a sequence of ILS problems in which the given matrices are the same. The details of the computed cost analysis of these two reduction algorithms can be found in [46] and we will introduce the LLL reduction in Section 2.1.1. The search process is to enumerate integer points in a region to find the optimal solution. There are also two well known search strategies: the Phost strategy (see [45] and [29]), which enumerates the integer vectors in a hyper-ellipsoid and the Kannan strategy (see [36] and [37]), which enumerates the integer vectors in a rectangular parallelotope. Later, Schnorr and Euchner [47] proposed a subtle but important improvement on the Phost strategy, which also enumerates the integer vectors in a hyper-ellipsoid but in a different order. The hyper-ellipsoid in which the Phost strategy or the Schorr-Euchner strategy enumerates integer points, is also a hyper-sphere in terms of the lattice points. For this reason, such kind of search algorithms are called shpere decoding (SD) algorithms in communications. The detailed comparison between these search strategies can be found in [1]. It shows that the Schnorr-Euchner (SE) strategy outperforms the other two and thus in this thesis we focus on the Schnorr-Euchner based search strategies. We will introduce this strategy in Section 2.1.2 in detail.

1.2 Box-constrained Integer Least Squares Problems

In some applications, such as wireless communications [43, 24], the parameter vector $\hat{\boldsymbol{x}}$ is usually constrained to a box $\mathcal{B}$ in $\mathbb{Z}^n$, i.e.,

$$\hat{\boldsymbol{x}} \in \mathcal{B} = \{\boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}, \boldsymbol{l}, \boldsymbol{u} \in \mathbb{Z}^n\}. \tag{1.3}$$

Then the ILS problem in (1.2) becomes

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

We refer to (1.4) as a *box-constrained integer least squares* (BILS) *problem.* If $m \geq n$ and the matrix $\boldsymbol{A}$ has full column rank, we refer to the BILS problem (1.4) as an *overdetermined box-constrained integer least squares* (OBILS) *problem.* Otherwise, when $m < n$ and $\boldsymbol{A}$ has full row rank, (1.4) is referred to as an *underdetermined box-constrained integer least squares* (UBILS) *problem.* In this thesis we mainly consider the UBILS problem.

When (1.4) is an OBILS problem, the LLL reduction can not be applied since it would transform the box $\mathcal{B}$ to a much complex geometry, making the search process difficult. However, the columns of $\boldsymbol{A}$ can be reordered to make the search process faster. Damen et al [24], and Chang and Han [14] suggest to use VBLAST and SQRD respectively, to make the search process faster. Both V-BLAST and SQRD use only the information of $\boldsymbol{A}$ to determine the column reordering. In [52], Su and Wassell use all information of the problem to propose a new column reordering algorithm in a geometric point of view. Independently, Chang and Han proposed another column reordering algorithm in [14], which is derived from an algebraic point of view. These two algorithms can bring more reduction in the cost of the search process than the previous three. In [11], Breen and Chang showed that the reduction algorithm proposed in [52] is essentially the same as that reduction proposed in [14], in the sense that they give the same column reordering result in theory. They combined the best part of these two algorithms and proposed a new mathematically equivalent but computationally faster reduction algorithm. This algorithm is called AIP (all-information based permutation) algorithm in [58]. For the search process, three Schnorr-Euchner based strategies were proposed to find the optimal solutions for the reduced OBILS problems (see, [10, 14, 24]). The one proposed by Boutros et al [10] is a direction extension of the original Schnorr-Euchner search algorithm by taking the box-constraint into account. The one proposed by Damen et al [24] is more natural and easier to understand. The one proposed by Chang and Han [14] is an improvement of the former two. In Section 2.2.2 and Section 2.2.1, we will mainly introduce the AIP reduction proposed by Breen and Chang in [11] and the search algorithm proposed by Chang and Han in [14].

When (1.4) is an UBILS problem, in the search process, regular sphere decoding based search algorithms for the OBILS problem can not be applied directly. Damen et al [23] proposed the first so-called generalized sphere decoding (GSD) algorithm. Its main idea is to partition the vector $\boldsymbol{x}$ in (1.4) into two subvectors $\boldsymbol{x}_{(1)} \in \mathbb{Z}^m$ and $\boldsymbol{x}_{(2)} \in \mathbb{Z}^{n-m}$. For each candidate of $\boldsymbol{x}_{(2)}$, it solves a corresponding OBILS problem to find the corresponding $\boldsymbol{x}_{(1)}$. After all the possible $\boldsymbol{x}_{(2)}$ are enumerated, the combination of $\boldsymbol{x}_{(2)}$ and $\boldsymbol{x}_{(1)}$ which gives the minimal residual, is the optimal solution for (1.4). Later, Dayal and Varanasi [27] proposed another generalized sphere decoding algorithm, which can significantly reduce the computational cost by partitioning the candidate set for $\boldsymbol{x}_{(2)}$ into disjoint ordered subsets. Then, Yang et al [60] proposed a so-called double-layer sphere decoder (DLSD) to solve the UBILS problem. Its basic idea is to apply an outer layer sphere decoder to get possible candidates for $\boldsymbol{x}_{(2)}$. Once $\boldsymbol{x}_{(2)}$ is determined, an inner layer sphere decoder is employed to get the relative $\boldsymbol{x}_{(1)}$. Then, Chang and Yang [20] proposed a direct tree search (DTS) algorithm, which modified the DV algorithm and incorporated a column reordering strategy in the reduction process, and is much faster than the one given in [27] and the one given in [5]. The DTS algorithm will be introducd in Section 2.3.1. All the above algorithms mainly consider how to generate a sequence of determined sub-ILS probelms and solve them to find the optimal solution. In [22], Cui and Tellambura solved the UBILS problem arising from communications, which has a special box constraint using a different approach. Their approach transforms the UBILS problem (1.4) to an equivalent OBILS problem by adding a term in the objective function, like the regularization method for ill-posed real least squares problem. Then the problem (1.4) becomes a corresponding OBILS problem so that a regular sphere decoding based algorithm can be applied. In [19], Chang et al proposed a partial regularization (PR) approach, which is a modification of the algorithm given in [22] and is faster than it. We will introduce it in Section 2.3.2. In [48], Shahnaz and Ali proposed another regularization approach to solve the UBILS problem on orthogonal frequency division multiplexing and space division multiple access (OFDM/SDMA) uplink system, which also transforms the UBILS problem to an OBILS problem, but using a different method to enlarge the original matrix $\boldsymbol{A}$.

A number of suboptimal algorithms also have been developed to reduce the complexity of solving the problem (1.4), see, e.g., [55, 26, 44]. In this thesis, we focus on optimal algorithms to solve the UBILS problem in (1.4).

The search methods for the OBILS problem and the UBILS problem we mentioned before are tree search approaches. The search cost is proportional to the number of tree nodes. If we can find some ways to prune the nodes of the search tree, the search process can then be speeded up. In [14, 51, 31, 12], various lower bounds have been proposed to prune the search tree for the OBILS problem. We will introduce the typical lower bounds in Chapter 3.

1.3 Organization and Contributions

The rest part of the thesis is organized as follows. In Chapter 2, we will review different algorithms for OILS, OBILS and UBILS problems respectively.

In Chapter 3, we first review three different lower bounds for the OBILS problems: the norm-wise lower bound, the component-wise lower bound, and the basis reduced lower bound respectively. Then we propose a lower bound for the UBILS problem, which is inspired by the partial regularization approach. To our knowledge, there are no lower bounds for the UBILS problem in the literature.

In Chapter 4, a new search algorithm is proposed for solving the UBILS problem, which is a modification of the DTS algorithm by partially using the best-first search approach.

In Chapter 5, numerical test results for different solvers for UBILS problems are given to show our lower bounds are effective in reducing the search cost and the new search algorithm is faster than the existing ones.

Finally, we summarize our results and discuss future research directions in Chapter 6.

1.4 Notation

In this section, we introduce the notations to be used in this thesis.

We denote scalars by normal type lower or upper case letters, column vectors by boldface lower case letters and matrices by boldface upper case letters. Usually, we use Greek letters (e.g., α, β) or Roman letters with subscripts (e.g., a_i, a_{ij}) to denote scalars, and sometimes we also use capital Roman letters (but not boldface).

Define $\mathbb{R}$, $\mathbb{Z}$ and $\mathbb{C}$ as the spaces of real scalars, integer scalars and complex numbers respectively, $\mathbb{R}^n$, $\mathbb{Z}^n$ and $\mathbb{C}^n$ as the spaces of $n-$dimensional real vectors, integer vectors and complex vectors respectively, and $\mathbb{R}^{m\times n}$, $\mathbb{Z}^{m\times n}$ and $\mathbb{C}^{m\times n}$ as the spaces of $m \times n$ real matrices, integer matrices and complex matrices respectively.

Given a column vector $\boldsymbol{x} \in \mathbb{R}^n$, let $\boldsymbol{x}_{i:j}$ or $\boldsymbol{x}(i{:}j)$ be the subvector composed of elements of $\boldsymbol{x}$ with indexes from i to j. Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, let $\boldsymbol{A}_{i:j,k:l}$ or $\boldsymbol{A}(i{:}j,k{:}l)$ be the submatrix containing all the elements of $\boldsymbol{A}$ whose row indexes are from i to j and column indexes are from k to l. Given a set $\mathcal{B} \subseteq \mathbb{Z}^n$, let $\mathcal{B}_{i:j} = \{\boldsymbol{x}_{i:j} : \boldsymbol{x} \in \mathcal{B}\}$. For a scalar $\alpha \in \mathbb{R}$, $\lfloor \alpha \rceil$ denotes the nearest integer to α (if there is a tie, $\lfloor \alpha \rceil$ is the one with smaller magnitude). Let $\lfloor \alpha \rfloor$ and $\lceil \alpha \rceil$ denote the integers we get after the floor and ceiling operations of α respectively. Analogously, for a vector $\boldsymbol{x} \in \mathbb{R}^n$, $\lfloor \boldsymbol{x} \rceil$ denotes the closest integer vector to $\boldsymbol{x}$, i.e., its i-th entry is $\lfloor x_i \rceil$, for $i = 1, \dots, n$. Let a denote a real number, $\mathcal{S}$ denote a set of integers, we use $\lfloor a \rceil|_{\mathcal{S}}$ to denote the nearest integer to a in the set $\mathcal{S}$.

We also define some special vectors and matrices here. We use $\mathbf{1}^{(n)}$ to represent the $n-$dimensional vector of ones. Let $\boldsymbol{I}$ represent an identity matrix. We use $\boldsymbol{e}_k$ to represent the kth column of the identity matrix $\boldsymbol{I}$. We use $\mathcal{N}(\mathbf{0}, \sigma^2\boldsymbol{I})$ to denote the normal distribution with zero mean and covariance matrix $\sigma^2\boldsymbol{I}$ and use $\mathcal{CN}(\mathbf{0}, \sigma^2\boldsymbol{I})$ to denote complex normal distribution with zero mean and covariance matrix $\sigma^2\boldsymbol{I}$.

CHAPTER 2
Algorithms for Solving Various ILS Problems

This chapter briefly reviews algorithms for solving various ILS problems. In Sections 2.1, 2.2 and 2.3, we will review different algorithms for solving OILS, OBILS and UBILS problems, respectively.

2.1 Ordinary ILS Problems

Given an OILS problem

$$\min_{\boldsymbol{x} \in \mathbb{Z}^n} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \tag{2.1}$$

where $\boldsymbol{y} \in \mathbb{R}^m$ and $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is a full column rank matrix, a typical approach to solving it is the Schnorr-Euchner sphere decoding method, which consists of two phases: reduction and search. We first introduce the well-known LLL reduction in Section 2.1.1 and then introduce the Schnorr-Euchner search strategy in Section 2.1.2.

2.1.1 Reduction

A. QRZ Decomposition

The QRZ decomposition of $\boldsymbol{A}$ has the following form (see [13]):

$$\boldsymbol{Q}^T \boldsymbol{A} \boldsymbol{Z} = \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{bmatrix}, \tag{2.2}$$

where $\boldsymbol{Q} = [\underbrace{\boldsymbol{Q}_1}_{m}, \underbrace{\boldsymbol{Q}_2}_{n-m}] \in \mathbb{R}^{m\times m}$ is an orthogonal matrix, $\boldsymbol{Z} \in \mathbb{Z}^{n\times n}$ is a unimodular matrix, i.e., $\det(\boldsymbol{Z}) = \pm 1$ and $\boldsymbol{R}$ is upper triangular. With (2.2) we have

$$\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 = \|\boldsymbol{Q}_1^T\boldsymbol{y} - \boldsymbol{R}\boldsymbol{Z}^{-1}\boldsymbol{x}\|_2^2 + \|\boldsymbol{Q}_2^T\boldsymbol{y}\|_2^2. \tag{2.3}$$

Define

$$\bar{\boldsymbol{y}} = \boldsymbol{Q}_1^T\boldsymbol{y}, \quad \boldsymbol{z} = \boldsymbol{Z}^{-1}\boldsymbol{x}, \tag{2.4}$$

then the OILS problem (2.1) can be transformed to an equivalent problem:

$$\min_{\boldsymbol{x}\in\mathbb{Z}^n} \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2. \tag{2.5}$$

Note that if the optimal solution to (2.5) is $\hat{\boldsymbol{z}}$, then the optimal solution to (2.1) is

$$\hat{\boldsymbol{x}} = \boldsymbol{Z}\hat{\boldsymbol{z}}. \tag{2.6}$$

B. LLL Reduction

The LLL reduction was proposed by Lenstra, Lenstra and Lovász in [39] and can be interpreted as a special QRZ decomposition, which requires the upper triangular matrix $\boldsymbol{R}$ in (2.2) to satisfy the following specific conditions:

$$|r_{k-1,j}| \le \frac{1}{2}|r_{k-1,k-1}|, \quad j = k, \dots, n \tag{2.7}$$

$$\delta r_{k-1,k-1}^2 \le r_{k-1,k}^2 + r_{kk}^2, \quad k = 2, \dots, n \tag{2.8}$$

where $\frac{1}{4} < \delta \le 1$. The inequality (2.7) is called the size reduction condition and the inequality (2.8) is called the Lovász condition. The LLL reduction will help to pursue the inequality

$$|r_{11}| \ll |r_{22}| \ll \dots \ll |r_{nn}|. \tag{2.9}$$

For the benefits of the matrix $\boldsymbol{R}$ satisfying inequality (2.9), see [13]. For the effects of the LLL reduction on the sphere decoding process, see [16]. Before introducing details of the LLL reduction algorithm, we first introduce three operations which will be used in the LLL reduction process: integer Gauss transformations, Givens rotations and column permutations.

Integer Gauss Transformations

We call a matrix integer Gauss transformation (IGT) matirx if it has the following form (see, e.g., [20])

$$\boldsymbol{Z}_{ij} = \boldsymbol{I} - \zeta_{ij}\boldsymbol{e}_i\boldsymbol{e}_j^T, \quad i \neq j, \quad \zeta_{ij} \in \mathbb{Z}. \tag{2.10}$$

It is easy to prove that $\boldsymbol{Z}_{ij}$ is a unimodular matrix and

$$\boldsymbol{Z}_{ij}^{-1} = \boldsymbol{I} + \zeta_{ij}\boldsymbol{e}_i\boldsymbol{e}_j^T.$$

Assume we have an upper triangular matrix $\boldsymbol{R}$, which does not satisfy (2.7) and (2.8). Applying $\boldsymbol{Z}_{ij}(i < j)$ to $\boldsymbol{R}$ from the right, we have

$$\bar{\boldsymbol{R}} = \boldsymbol{R}\boldsymbol{Z}_{ij} := \boldsymbol{R} - \zeta_{ij}\boldsymbol{R}\boldsymbol{e}_i\boldsymbol{e}_j^T. \tag{2.11}$$

Obviously, $\bar{\boldsymbol{R}}$ is the same as $\boldsymbol{R}$ except that

$$\bar{r}_{kj} = r_{kj} - \zeta_{ij}r_{ki}, \quad k = 1, \ldots, i.$$

Taking $\zeta_{ij} = \lfloor r_{ij}/r_{ii} \rceil$ makes $\bar{r}_{kj}$ smallest and

$$|\bar{r}_{ij}| \leq \frac{1}{2}|r_{ii}|.$$

Givens Rotations and Column Permutations

Column permutations can help make the matrix $\boldsymbol{R}$ satisfy the inequality (2.9). However, after switching two columns of $\boldsymbol{R}$, $\boldsymbol{R}$ is not an upper triangular matrix, which violates the requirement in (2.5). Therefore, some rotations are needed to make the matrix $\boldsymbol{R}$ upper triangular again and Givens rotations can achieve it. In the following, we first show how to use Givens rotations to make a permutated $\boldsymbol{R}$ back to upper triangular and then introduce when to interchange two columns of $\boldsymbol{R}$.

Suppose we interchange column $k-1$ and column k of $\boldsymbol{R}$ and then have

$$\boldsymbol{R}\boldsymbol{P}_{k-1,k} = \begin{array}{c c} & \begin{array}{ccc} {\scriptstyle k-2} & {\scriptstyle 2} & {\scriptstyle n-k} \end{array} \\ \begin{array}{c} {\scriptstyle k-2} \\ {\scriptstyle 2} \\ {\scriptstyle n-k} \end{array} & \begin{bmatrix} \boldsymbol{R}_{11} & \bar{\boldsymbol{R}}_{12} & \boldsymbol{R}_{13} \\ & \tilde{\boldsymbol{R}}_{22} & \boldsymbol{R}_{23} \\ & & \boldsymbol{R}_{33} \end{bmatrix} \end{array}, \tag{2.12}$$

where

$$\boldsymbol{P}_{k-1,k} = \begin{bmatrix} \boldsymbol{I}_{k-2} & & \\ & \boldsymbol{P} & \\ & & \boldsymbol{I}_{n-k} \end{bmatrix}, \quad \boldsymbol{P} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

$$\bar{\boldsymbol{R}}_{12} = \begin{bmatrix} \boldsymbol{R}_{1:k-2,k-1} & \boldsymbol{R}_{1:k-2,k} \end{bmatrix}, \quad \tilde{\boldsymbol{R}}_{22} = \begin{bmatrix} r_{k-1,k} & r_{k-1,k-1} \\ r_{kk} & 0 \end{bmatrix}.$$

The block $\tilde{\boldsymbol{R}}_{22}$ violates the upper triangular property and a Givens rotation can easily be applied to make $\tilde{\boldsymbol{R}}_{22}$ upper triangular. Assume $\boldsymbol{G} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$, then we have

$$\boldsymbol{G}\tilde{\boldsymbol{R}}_{22} = \begin{bmatrix} \bar{r}_{k-1,k-1} & \bar{r}_{k-1,k} \\ 0 & \bar{r}_{kk} \end{bmatrix},$$

where

$$c = \frac{r_{k-1,k}}{\sqrt{r_{k-1,k}^2 + r_{kk}^2}}, \quad s = \frac{r_{kk}}{\sqrt{r_{k-1,k}^2 + r_{kk}^2}},$$

$$\bar{r}_{k-1,k-1} = \sqrt{r_{k-1,k}^2 + r_{kk}^2}, \quad \bar{r}_{k-1,k} = \frac{r_{k-1,k-1}r_{k-1,k}}{\sqrt{r_{k-1,k}^2 + r_{kk}^2}}, \quad \bar{r}_{kk} = -\frac{r_{k-1,k-1}r_{kk}}{\sqrt{r_{k-1,k}^2 + r_{kk}^2}}. \tag{2.13}$$

Define $\bar{\boldsymbol{R}}_{22} = \begin{bmatrix} \bar{r}_{k-1,k-1} & \bar{r}_{k-1,k} \\ 0 & \bar{r}_{kk} \end{bmatrix}$, we have

$$\boldsymbol{Q}_{k-1,k}\boldsymbol{R}\boldsymbol{P}_{k-1,k} = \bar{\boldsymbol{R}} = \begin{bmatrix} \boldsymbol{R}_{11} & \bar{\boldsymbol{R}}_{12} & \boldsymbol{R}_{13} \\ & \bar{\boldsymbol{R}}_{22} & \bar{\boldsymbol{R}}_{23} \\ & & \boldsymbol{R}_{33} \end{bmatrix},$$

$$\boldsymbol{Q}_{k-1,k} = \begin{bmatrix} \boldsymbol{I}_{k-2} & & \\ & \boldsymbol{G} & \\ & & \boldsymbol{I}_{n-k} \end{bmatrix}, \quad \bar{\boldsymbol{R}}_{23} = \boldsymbol{G}\boldsymbol{R}_{23}.$$

Then the problem comes to when to interchange two columns of $\boldsymbol{R}$. Here we take column $k-1$ and column k as an example. The purpose of column permutations is to make the diagonal entries of $\boldsymbol{R}$ tend to (2.9) as much as possible. It means that we want to decrease $|r_{k-1,k-1}|$ and increase $|r_{kk}|$ by the reduction process. Since the product of $|r_{k-1,k-1}r_{kk}|$ is invariant under an IGT and the column permutation, decreasing $|r_{k-1,k-1}|$ means increasing $|r_{kk}|$ and vice versa. We can first apply an IGT $\boldsymbol{Z}_{k-1,k}$ to $\boldsymbol{R}$ from the right to guarantee $|r_{k-1,k}| \leq |r_{k-1,k-1}|/2$. Then, from (2.13), it is easy to see that a column interchange should be applied on columns k and $k-1$, if $\delta r^2_{k-1,k-1} > r^2_{k-1,k} + r^2_{kk}$. Note that even though the above column permutation can not guarantee $|\bar{r}_{k-1,k-1}| \leq |\bar{r}_{kk}|$, it can ensure that

$$|\bar{r}_{k-1,k-1}| < |r_{k-1,k-1}|, \quad |\bar{r}_{kk}| > |r_{kk}|,$$

which indicates that the permutations can make the diagonal components of $\boldsymbol{R}$ tend to (2.9).

LLL Reduction Algorithm

After the LLL reduction, the matrix $\boldsymbol{R}$ satisfies two requirements we mentioned above: ineuquality (2.7) and inequality (2.8). The first requirement is referred to as the size-reduced condition and the second one is referred to as the Lovász condition. Combining these two inequalities, we can get

$$|r_{k-1,k-1}| \leq \frac{2}{\sqrt{4\delta - 1}}|r_{kk}|. \tag{2.14}$$

We first briefly introduce the process of the LLL reduction and then give the psuedocode of it in Algorithm 2.1. At the beginning, we compute the QR decomposition to obtain an upper triangular matrix $\boldsymbol{R}$. Then we work on $\boldsymbol{R}$ firom the left to the right. In the kth column of $\boldsymbol{R}$, we apply an IGT to reduce $r_{k-1,k}$ and then we check if the inequality (2.8) holds. If the inequality is satisfied, we reduce $r_{k-2,k}, r_{k-3,k}, \ldots, r_{1k}$ and move to column $k+1$. Otherwise, we interchange column $k-1$ and column k

of $\boldsymbol{R}$, and then apply a Givens rotation to bring $\boldsymbol{R}$ back to upper triangular, and move back to column $k-1$. Finally we finish this process in the last column of $\boldsymbol{R}$. Details of the LLL reduction process is given in Algorithm 2.1, which is similar to the original algorithm given in [39].

There have been some improvements of the LLL reduction algorithm. In [40] Ling and Howgrave-Graham found that when the LLL reduction is used to improve the performance of the Babai point (see [5]), the size reduction for the off-diagonal entries of $\boldsymbol{R}$ above the super-diagonal entries in the LLL algorithm is not necessary mathematically and can be removed. They proposed the so-called effective LLL (ELLL) reduction algorithm, which has less cost than the LLL reduction but it may have some numerical stability problems, see [40, 59]. In [59], Xie et al proposed a partial LLL (PLLL) reduction algorithm, which does size reductions for part of super-diagonal entries of $\boldsymbol{R}$ and the corresponding off-diagonal entries in the same columns such that it can avoid the numerical stability problem in the ELLL reduction. Furthermore, the PLLL reduction is faster than the ELLL reduction since it uses the minimum pivoting strategy in computing the QR decomposition, which can reduce the number of column permutations involved in the LLL and ELLL reductions. The PLLL reduction eliminates some unnecessary size reductions on some super-diagonal entries in the ELLL. In [59], they also proved that the ELLL and PLLL reductions are equivalent to the LLL reduction in improving the efficiency of the search process for solving (2.1).

Algorithm 2.1 LLL Reduction

Input: The matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with full column rank, and the input 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 $\bar{\boldsymbol{y}} \in \mathbb{R}^n$.
function: $[\bar{\boldsymbol{y}}, \boldsymbol{R}, \boldsymbol{Z}] = \text{LLL}(\boldsymbol{A}, \boldsymbol{y})$

```
 1: Set $\boldsymbol{Z} = \boldsymbol{I}_n$, $k = 2$
 2: Compute $[\boldsymbol{Q}_1, \boldsymbol{Q}_2]^T \boldsymbol{A} = \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{bmatrix}$ by the Householder transformations, compute
    $[\boldsymbol{Q}_1, \boldsymbol{Q}_2]^T \boldsymbol{y}$ and set $\bar{\boldsymbol{y}} = \boldsymbol{Q}_1^T \boldsymbol{y}$
 3: while $k \le n$ do
 4:     Apply size reduction $\boldsymbol{Z}_{k-1,k}$: $\boldsymbol{R} = \boldsymbol{R}\boldsymbol{Z}_{k-1,k}$
 5:     Update $\boldsymbol{Z}$, i.e., $\boldsymbol{Z} = \boldsymbol{Z}\boldsymbol{Z}_{k-1,k}$
 6:     if $\delta r_{k-1,k-1}^2 > r_{k-1,k}^2 + r_{k,k}^2$ then
 7:         Interchange columns $k-1$ and $k$ of $\boldsymbol{R}$
 8:         Transform permutated $\boldsymbol{R}$ back to upper triangular by Givens rotations
 9:         Interchange columns $k-1$ and $k$ of $\boldsymbol{Z}$
10:         Apply the same Givens rotation to $\bar{\boldsymbol{y}}$
11:         if $k > 2$ then
12:             $k = k - 1$
13:         end if
14:     else
15:         for $i = k-2 : -1 : 1$ do
16:             Apply the integer Gauss tranformation $\boldsymbol{Z}_{ik}$ to $\boldsymbol{R}$, i.e., $\boldsymbol{R} = \boldsymbol{R}\boldsymbol{Z}_{ik}$
17:             Update $\boldsymbol{Z}$, i.e., $\boldsymbol{Z} = \boldsymbol{Z}\boldsymbol{Z}_{ik}$
18:         end for
19:         $k = k + 1$
20:     end if
21: end while
```

2.1.2 Search

Suppose that we have an upper bound β for the objective funciton in (2.5) such that

$$\|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2 < \beta^2. \tag{2.15}$$

Note that the inequality (2.15) is a hyper-ellipsoid with center $\boldsymbol{R}^{-1}\bar{\boldsymbol{y}}$ in terms of $\boldsymbol{z}$. So to solve the problem (2.5) we can search this ellipsoid to find the optimal solution. The quantity β is referred to as the sphere decoding radius. We first give the basic idea of the Schnorr-Euchner strategy and then show how to choose the initial β at the end of this subsection.

Expanding the term $\|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2$ in (2.15), we have

$$\|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2 = \sum_{k=1}^{n}(\bar{y}_k - \sum_{j=k}^{n} r_{kj}z_j)^2,$$

and then we can rewrite (2.15) as

$$\sum_{k=1}^{n}(\bar{y}_k - \sum_{j=k}^{n} r_{kj}z_j)^2 < \beta^2. \tag{2.16}$$

Define

$$c_n = \bar{y}_n/r_{nn}, \quad c_k = (\bar{y}_k - \sum_{j=k+1}^{n} r_{kj}z_j)/r_{kk}, \quad k = n-1, n-2, \ldots, 1. \tag{2.17}$$

Then it follows from (2.16) that

$$\sum_{k=1}^{n} r_{kk}^2(z_k - c_k)^2 < \beta^2. \tag{2.18}$$

The above inequaltiy is equivalent to the following set of inequalities:

$$\begin{aligned}
&\text{level } n : r_{nn}^2(z_n - c_n)^2 < \beta^2, && (2.19)\\
&\quad\vdots\\
&\text{level } k : r_{kk}^2(z_k - c_k)^2 < \beta^2 - \sum_{i=k+1}^{n} r_{ii}^2(z_i - c_i)^2, && (2.20)\\
&\quad\vdots\\
&\text{level } 1 : r_{11}^2(z_1 - c_1)^2 < \beta^2 - \sum_{i=2}^{n} r_{ii}^2(z_i - c_i)^2 && (2.21)
\end{aligned}$$

With the above set of inequalities, a search process can start. We first fix the value of z_n at level n, and then z_{n-1} at level $n-1$, and so on. The value of z_k can be determined only after $z_n, \ldots, z_{k+1}$ have been fixed. In the following, we show how to determine z_k.

Suppose that $z_n, \ldots, z_{k+1}$ have been fixed. From (2.20), it is easy to see that z_k should be in the following interval

$$\left[\left\lceil c_k - [\beta^2 - \sum_{i=k+1}^{n} r_{ii}^2(z_i - c_i)^2]^{1/2}/|r_{kk}| \right\rceil, \left\lfloor c_k + [\beta^2 - \sum_{i=k+1}^{n} r_{ii}^2(z_i - c_i)^2]^{1/2}/|r_{kk}| \right\rfloor\right]. \quad (2.22)$$

The Schnorr-Euchner strategy enumerates the integers in the interval (2.22) in a zigzag sequence as

$$\begin{aligned}
&\lfloor c_k \rceil, \lfloor c_k \rceil - 1, \lfloor c_k \rceil + 1, \lfloor c_k \rceil - 2, \ldots, \quad \text{if} \quad c_k \le \lfloor c_k \rceil;\\
&\lfloor c_k \rceil, \lfloor c_k \rceil + 1, \lfloor c_k \rceil - 1, \lfloor c_k \rceil + 2, \ldots, \quad \text{if} \quad c_k > \lfloor c_k \rceil.
\end{aligned} \qquad (2.23)$$

Note that $z_k = \lfloor c_k \rceil$ makes the residual on the left hand side of (2.20) smallest and the residual increases with respect to the zigzag sequence (2.23). If (2.20) does not hold for some candidates of z_k, it does not hold for those candidates after it in (2.23) either and thus we can skip them in the search process.

Now we describe the Schnorr-Euchner search strategy starting from level n. First we take $z_n = \lfloor c_n \rceil$. If this candidate does not satisfy the inequality (2.19), no other integers can satisfy this inequality and we have to stop. It means that the ellipsoid (2.18) does not include any integer point and we have to search with a larger β, and restart the search process. Otherwise we move to level $n-1$. Based on the fixed z_n, we first compute c_{n-1} by (2.17) and then choose $z_{n-1} = \lfloor c_{n-1} \rceil$. If this $\boldsymbol{z}_{n-1}$ fails to satisfy the inequality (2.20) with $k = n-1$, then we have to go back to level n and choose z_n as the second nearest integer to c_n according to the zigzag sequence (2.23), and so on. If we can find a valid z_{n-1}, we proceed to level $n-2$. Continue this process until we reach level 1 and find a valid z_1. Then we obtain an integer point $\boldsymbol{z}^*$ satisfying (2.15). We shrink the ellipsoid by taking $\beta = \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}^*\|_2$ and search for a new integer point in the new ellipsoid. We go back to level 2 and update the value of z_2 by choosing z_2 to be the next nearest integer to c_2. If the inequality for level 2 holds for this new z_2, we move down to level 1 and update the value of z_1. Otherwise we move up to level 3 to update the value of z_3, and so on. Finally we will end up the search process at level n, where we can not choose a new integer for z_n to satisfy the inequality (2.19). Then $\hat{\boldsymbol{z}}$, the most recently found integer point, is the optimal solution of the problem (2.5).

From inequalities (2.19)-(2.21), we can see that the initial value of β is crucial to the efficiency of the search process. If the initial value of β is too large, there will be lots of candidate points inside the hyper-ellipoid and the enumeration process may be time-consuming. However, if the initial value of β is too small, the hyper-ellipsoid may not include any integer point and we need to restart with a larger β, which is also not efficient. For the OILS problem (2.5), we usually set the initial β to be ∞ and the first integer point we get by the search approach is referred to as the Babai integer point (see [13], [5]). Another simple method to initialize β is to solve a real least squares problem. Suppose $\boldsymbol{z}_{\text{rls}}$ is the real least squares solution of problem (2.5), i.e., $\boldsymbol{z}_{\text{rls}} = \boldsymbol{R}^{-1}\bar{\boldsymbol{y}}$, then we can set $\beta = \|\bar{\boldsymbol{y}} - \boldsymbol{R}\lfloor \boldsymbol{z}_{\text{rls}} \rceil\|_2$. If there is no integer point inside the hyper-ellipsoid when $\beta = \|\bar{\boldsymbol{y}} - \boldsymbol{R}\lfloor \boldsymbol{z}_{\text{rls}} \rceil\|_2$, it means that $\lfloor \boldsymbol{z}_{\text{rls}} \rceil$ is the optimal solution of (2.5).

2.2 Overdetermined Box-constrained ILS Problems

In Section 2.1, we have introduced the LLL reduction and the Schnorr-Enchner search strategy for solving the OILS problem (2.1). In this section, we will introduce the reduction algorithm in [11] and the search algorithm in [14] to solve the following OBILS problem

$$\min_{\boldsymbol{x} \in \mathcal{B}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \tag{2.24}$$

where $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is a full column rank matrix $(m \geq n)$, $\mathcal{B} = \{\boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u},\ \boldsymbol{l}, \boldsymbol{u} \in \mathbb{Z}^n\}$.

2.2.1 Reduction

The LLL reduction can be used for solving OILS problems. However, it can not be applied to OBILS problems since the geometry of the constraint box would become complicated after $\boldsymbol{x}$ is changed to $\boldsymbol{z}$ by the unimodular transformation and then the search process would be difficult. Therefore, we only apply a column permutation matrix $\boldsymbol{P}$ to $\boldsymbol{A}$ in the OBILS problem (2.24) such that

$$\boldsymbol{AP} = \boldsymbol{Q}\begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{bmatrix} = \boldsymbol{Q}_1\boldsymbol{R}, \tag{2.25}$$

where $\boldsymbol{Q} = [\underbrace{\boldsymbol{Q}_1}_{m}, \underbrace{\boldsymbol{Q}_2}_{n-m}] \in \mathbb{R}^{m\times m}$ is orthogonal and $\boldsymbol{R}$ is upper triangular. Then with (2.25), we have

$$\|\boldsymbol{y} - \boldsymbol{Ax}\|_2^2 = \|\boldsymbol{Q}_1^T\boldsymbol{y} - \boldsymbol{RP}^T\boldsymbol{x}\|_2^2 + \|\boldsymbol{Q}_2^T\boldsymbol{y}\|_2^2. \tag{2.26}$$

If we define

$$\bar{\boldsymbol{y}} = \boldsymbol{Q}_1^T\boldsymbol{y}, \quad \boldsymbol{z} = \boldsymbol{P}^T\boldsymbol{x}, \quad \bar{\boldsymbol{l}} = \boldsymbol{P}^T\boldsymbol{l}, \quad \bar{\boldsymbol{u}} = \boldsymbol{P}^T\boldsymbol{u}, \tag{2.27}$$

then the original problem (2.24) can be transformed to

$$\min_{\boldsymbol{z}\in\bar{\mathcal{B}}} \|\bar{\boldsymbol{y}} - \boldsymbol{Rz}\|_2^2, \tag{2.28}$$

where

$$\bar{\mathcal{B}} = \{\boldsymbol{z} \in \mathbb{Z}^n : \bar{\boldsymbol{l}} \le \boldsymbol{z} \le \bar{\boldsymbol{u}},\ \bar{\boldsymbol{l}}, \bar{\boldsymbol{u}} \in \mathbb{Z}^n\}. \tag{2.29}$$

In the search process, we find the optimal solution $\tilde{\boldsymbol{z}}$ for the problem (2.28) and the corresponding solution for the problem (2.24) $\tilde{\boldsymbol{x}} = \boldsymbol{P}\hat{\boldsymbol{z}}$ ($\boldsymbol{R}$ is upper triangular).

Different column reordering strategies (different permutation matrices $\boldsymbol{P}$) for OBILS problems can be found in [29, 24, 57, 52, 14, 11]. In [29], $\boldsymbol{P}$ was chosen following the criterion that the columns of the permutated $\boldsymbol{A}$ are arranged in an increasing order with respect to the 2-norm. In [30], the well-known Vertical Bell Laboratories Layered Space-Time (V-BLAST) strategy was proposed. It determines the order of columns of the permutaed $\boldsymbol{A}$ from right to left. When determining column i, it seeks from the remaining i columns of $\boldsymbol{A}$ to find a column which can maximizes the corresponding $|r_{ii}|$. In [57], the sorted QR decomposition (SQRD) algorithm chooses the order of columns of the permutated $\boldsymbol{A}$ from left to right. When determining column i, it seeks from the remaining $n-i+1$ columns of $\boldsymbol{A}$ to find a column which can minimize the corresponding $|r_{ii}|$. Details about V-BLAST and SQRD reduction algorithms can be found in [30] and [57] respectively. Unlike the above permutation strategies, the SW reduction algorithm [52] and the CH reduction algorithm [14] used all information of the OBILS problem and it turns out that they give the same column reordering, which was proved by Breen and Chang in [11], who also combined the best parts of these two algorithms to propose a reduction algorithm, to be referred to as AIP (all information permutation). Here, we give details of the AIP reduction.

Like the V-BLAST reduction, the AIP reduction algorithm determines the ordering from right to left, which means it first determines the nth column of $\boldsymbol{R}$, and then the $(n-1)$th column and so on. We show how to choose the last column of $\boldsymbol{R}$ in the AIP reduction as an example. Then the subsequent steps of choosing ordering are very similar.

First, we give some definitions which will be used later. Let

$$\boldsymbol{T} = \boldsymbol{R}^{-T}, \quad \breve{\boldsymbol{z}} = \boldsymbol{T}^T \bar{\boldsymbol{y}}, \tag{2.30}$$

where $\boldsymbol{T}$ is a lower triangular matrix and let $\boldsymbol{z}^{\mathrm{r}}$ be the closet integer point to $\breve{\boldsymbol{z}}$ in the box (2.29) (here the super script r stands for rounding), i.e.,

$$z_i^{\mathrm{r}} = \mathrm{median}(\lfloor \breve{z}_i \rceil, \bar{l}_i, \bar{u}_i), \text{ for } i = 1, 2, \ldots, n.$$

Define $\boldsymbol{z}^{\mathrm{s}}$ to be the second closest integer point to $\breve{\boldsymbol{z}}$ in the box (2.29), i.e.,

$$z_i^{\mathrm{s}} = \begin{cases} z_i^{\mathrm{r}} + 1 & \text{if } z_i^{\mathrm{r}} = \bar{l}_i \text{ or } z_i^{\mathrm{r}} < \breve{z}_i \\ z_i^{\mathrm{r}} - 1 & \text{if } z_i^{\mathrm{r}} = \bar{u}_i \text{ or } z_i^{\mathrm{r}} \geq \breve{z}_i \end{cases} \qquad i = 1, 2, \dots, n$$

Now we show how to determine the last column of the final $\boldsymbol{R}$. Suppose we interchange columns i and n of $\boldsymbol{R}$, then, the matrix $\boldsymbol{R}$ is not upper triangular and we can use Givens rotations (see Section 2.1.1) to bring $\boldsymbol{R}$ back to upper triangular. This process can be described as:

$$\hat{\boldsymbol{R}} = \boldsymbol{G}^T \boldsymbol{R} \hat{\boldsymbol{P}}, \quad \hat{\boldsymbol{y}} = \boldsymbol{G}^T \bar{\boldsymbol{y}}, \quad \hat{\boldsymbol{z}} = \hat{\boldsymbol{P}} \boldsymbol{z}. \quad \hat{\boldsymbol{z}}^{\mathrm{s}} = \hat{\boldsymbol{P}} \boldsymbol{z}^{\mathrm{s}} \tag{2.31}$$

where $\hat{\boldsymbol{P}}$ denotes the permutation matrix that interchanges columns j and n of $\boldsymbol{R}$. $\boldsymbol{G}$ is the product of Givens rotations which are used to bring the permutated $\boldsymbol{R}$ back to upper triangular and $\hat{\boldsymbol{R}}$ denotes the new upper triangular matrix. Without loss of generality, we can assume that the diagonal entries of $\hat{\boldsymbol{R}}$ are positive. Define

$$d_i^{(n)} = \hat{r}_{nn}^2 (\hat{z}_n^{\mathrm{s}} - \hat{y}_n / \hat{r}_{nn})^2, \quad i = 1, 2, \dots, n, \tag{2.32}$$

then the AIP reduction chooses column $j = \arg\max_i d_i^{(n)}$ as the new nth column. The motivation of this choice is to make the sphere decoding radius for the $(n-1)$-dimensional subproblem (see (2.33) below) small and at the same time to make r_{nn} large (for details, see [14] for explanations).

After the index j is found, the AIP reduction algorithm interchanges columns j and n of $\boldsymbol{R}$, entries j and n of $\bar{\boldsymbol{l}}$ and $\bar{\boldsymbol{u}}$, and columns j and n of $\boldsymbol{T}$ respectively. Then it applies the same Givens rotations to the permutated $\boldsymbol{T}$ and $\boldsymbol{R}$ and make these two permutated matrices back to lower and upper triangular matrices such that $\boldsymbol{T} = \boldsymbol{R}^{-T}$

still holds. Then the AIP reduction algorithm sets $z_n = \text{median}(\lfloor \bar{y}/r_{nn} \rceil, \bar{l}_n, \bar{u}_n)$ and updates $\tilde{\boldsymbol{y}} = \bar{\boldsymbol{y}} - \boldsymbol{R}_{1:n,n} z_n$. Next, the AIP reduction algorithm applies the above process recursively to the following $(n-1)$-dimensional subproblem

$$\min_{\boldsymbol{z}_{1:n-1} \in \bar{\mathcal{B}}_{n-1}} \|\tilde{\boldsymbol{y}}_{1:n-1} - \boldsymbol{R}_{1:n-1,1:n-1} \boldsymbol{z}_{1:n-1}\|_2^2 \tag{2.33}$$

where $\bar{\mathcal{B}}_{n-1} = \{\boldsymbol{z}_{1:n-1} | \boldsymbol{z}_{1:n-1} \in \mathbb{Z}^{n-1} : \bar{\boldsymbol{l}}_{1:n-1} \leq \boldsymbol{z}_{1:n-1} \leq \bar{\boldsymbol{u}}_{1:n-1}\}$. We briefly summarize the AIP reduction algorithm for n-dimensional problem first and then give the pseudocode of it in Algorithm 2.2.

Algorithm 2.2 includes the computation of the box-constrained Babai point. Since we will use it later, we give the explicit definition of it here. The box-constrained Babai point $\boldsymbol{z}_B$ is defined as

$$\begin{aligned} c_n &= \tilde{y}_n / r_{nn}, \quad z_B(n) = \text{median}(\lfloor c_n \rceil, \bar{l}_n, \bar{u}_n) \\ c_i &= (\tilde{y}_i - \sum_{j=i+1}^{n} r_{ij} z_B(j)) / r_{ii}, \quad z_B(i) = \text{median}(\lfloor c_i \rceil, \bar{l}_i, \bar{u}_i) \end{aligned} \tag{2.34}$$

for $i = n-1, \ldots, 1$.

2.2.2 Search

In this section, we will introduce the search algorithm for the OBILS problem, see [14]. Suppose for the OBILS problem (2.28) we can find β such that the optimal solution to (2.28) satisfies

$$\|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2 < \beta^2, \tag{2.35}$$

which can be rewritten with (2.17) as

$$\sum_{k=1}^{n} r_{kk}^2 (z_k - c_k)^2 \leq \beta^2. \tag{2.36}$$

Algorithm 2.2 AIP Reduction

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

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

function: $[\boldsymbol{R}, \boldsymbol{P}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{l}}, \bar{\boldsymbol{u}}, \boldsymbol{z}^*] = \text{AIP}(\boldsymbol{A}, \boldsymbol{y}, \boldsymbol{l}, \boldsymbol{u})$

```
 1: Compute $[\boldsymbol{Q}_1 \;\; \boldsymbol{Q}_2]^T\boldsymbol{A} = \begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{bmatrix}$ by the Householder transformations, compute
    $[\boldsymbol{Q}_1, \boldsymbol{Q}_2]^T$, and compute $\bar{\boldsymbol{y}} := \boldsymbol{Q}_1^T\boldsymbol{y}$.
 2: Initialize $\bar{\boldsymbol{l}} = \boldsymbol{l}$, $\bar{\boldsymbol{u}} = \boldsymbol{u}$, $\boldsymbol{z}^* = []$, and $\tilde{\boldsymbol{y}} := \bar{\boldsymbol{y}}$.
 3: Compute $\boldsymbol{L} = \boldsymbol{R}^{-T}$ and set $\boldsymbol{P} := \boldsymbol{I}_n$
 4: for $k = n : -1 : 2$ do
 5:     Solve $\boldsymbol{L}^T\breve{\boldsymbol{z}} = \tilde{\boldsymbol{y}}$
 6:     Let $z_i^{\mathrm{s}}$ be the second closest integer in $[\bar{l}_i, \bar{u}_i]$ to $\breve{z}_i$, $for$ $i = 1, \ldots, k$
 7:     Compute $d_i^{(k)} = (z_i^{\mathrm{s}} - \breve{z}_i)^2/\breve{l}_i$ for $i = 1, 2, \ldots, k$
 8:     Let $j = \arg\max_{i=1,2,\ldots,k} d_i^{(k)}$
 9:     $z_k^* = \text{median}(\lfloor\breve{z}_i\rceil, \bar{l}_i, \bar{u}_i)$
10:     $\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{y}} - \boldsymbol{R}_{1:j,j}z_k$
11:     Remove column $j$ of $\boldsymbol{L}$ and entry $j$ of $\breve{\boldsymbol{l}}$
12:     if $i \neq k$ then
13:         Interchange column $j$ and $k$ of $\boldsymbol{R}$
14:         Interchange entry $j$ and $k$ of $\bar{\boldsymbol{l}}$ and $\bar{\boldsymbol{u}}$
15:         Interchange columns $j$ and $k$ of $\boldsymbol{P}$
16:         Use Givens rotations to bring $\boldsymbol{R}$ back to upper triangular
17:         Apply the same Givens rotations to update $\boldsymbol{L}$, $\bar{\boldsymbol{y}}$ and $\tilde{\boldsymbol{y}}$
18:     end if
19:     Update $\breve{l}_i = \breve{l}_i - l_{ki}^2$ for $i = 1, 2, \ldots, k$
20:     Remove row $k$ in $\boldsymbol{L}$ and remove entry $k$ in $\tilde{\boldsymbol{y}}$
21: end for
```

The method for finding the initial β will be discussed at the end of this section. We can rewrite the inequality (2.36) as the set of inequalities (2.19)-(2.21). For the sake of readability, we repeat them here. the following set of inequalities

$$\text{level } n : r_{nn}^2(z_n - c_n)^2 < \beta^2 \tag{2.37}$$

$$\vdots$$

$$\text{level } k : r_{kk}^2(z_k - c_k)^2 < \beta^2 - \sum_{i=k+1}^{n} r_{ii}^2(z_i - c_i)^2 \tag{2.38}$$

$$\vdots$$

$$\text{level } 1 : r_{11}^2(z_1 - c_1)^2 < \beta^2 - \sum_{i=2}^{n} r_{ii}^2(z_i - c_i)^2 \tag{2.39}$$

Based on the above set of inequalities, a modified Schnorr-Euchner based search process can be developed to find the solution of (2.28). Different with the method for solving OILS problems (see Section 2.1), the box constraint (2.29) needs to be considered here. In the OILS problem, at level k, integer candidates for z_k are enumerated by an increasing order of $|z_k - c_k|$. The same startegy can be used here except that z_k should be in the constrained interval $[\bar{l}_k, \bar{u}_k]$. We choose the first nearest integer to c_k on the interval $[\bar{l}_k, \bar{u}_k]$ as the first candidate of z_k, and choose the second nearest integer to c_k on the interval $[\bar{l}_k, \bar{u}_k]$ as the second candidate of z_k, and so on.

We rewrite the inequality (2.36) as

$$r_{kk}^2(z_k - c_k)^2 < \beta^2 - \sum_{i=k+1}^{n} r_{ii}^2(z_i - c_i)^2 - \sum_{j=1}^{k-1} r_{jj}^2(z_j - c_j)^2. \tag{2.40}$$

If we can find a non-negative lower bound T_k for the third term on the right hand side of (2.40), we will have

$$r_{kk}^2(z_k - c_k)^2 < \beta^2 - \sum_{i=k+1}^{n} r_{ii}^2(z_i - c_i)^2 - T_k. \tag{2.41}$$

It is easy to see that the upper bound in (2.41) is at least as tight as the one in (2.38). If we can find a good lower bound T_k, we can shrink the search region and improve the efficiency of the search process. In [14], a simple lower bound T_k was proposed, which we will introduce below.

If we can find a lower bound $h_i \le r_{ii}^2(z_i - c_i)^2$ for each i, $1 \le i \le k-1$, then we have

$$\sum_{i=1}^{k-1} h_i \le \sum_{i=1}^{k-1} r_{ii}^2(z_i - c_i)^2.$$

Then $T_k = \sum_{i=1}^{k-1} h_i$ is the lower bound for $\sum_{i=1}^{k-1} r_{ii}^2(z_i - c_i)^2$ at level k.

Since $\bar{l}_j \le z_j \le \bar{u}_j$, for $j = 1, \ldots, n$, we have

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

Then it follows that

$$\bar{y}_k - \max(r_{kj}\bar{l}_j, r_{kj}\bar{u}_j) \le \bar{y}_k - r_{kj}z_j \le \bar{y}_k - \min(r_{kj}\bar{l}_j, r_{kj}\bar{u}_j). \tag{2.42}$$

If the lower bound and upper bound in (2.42) have the same sign, i.e.,

$$\text{sign}[\bar{y}_k - \sum_{j=k}^{n} \min(r_{kj}\bar{l}_j, r_{kj}\bar{u}_j)] = \text{sign}[\bar{y}_k - \sum_{j=k}^{n} \max(r_{kj}\bar{l}_j, r_{kj}\bar{u}_j)], \tag{2.43}$$

then we have

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

where $h_k = \min\{[\bar{y}_k - \sum_{j=k}^{n}\min(r_{kj}\bar{l}_j, r_{kj}\bar{u}_j)]^2, [\bar{y}_k - \sum_{j=k}^{n}\max(r_{kj}\bar{l}_j, r_{kj}\bar{u}_j)]^2\}$. Otherwise, we take $h_k = 0$. Then $T_k = \sum_{i=1}^{k-1} h_i$ is a lower bound on $\sum_{j=1}^{k-1} r_{jj}^2(z_j - c_j)^2$. We will introduce other more complicated lower bounds in Section 3.1. The pseudocode of the CH search strategy can be found in Algorithm 2.3.

For the OBILS problem (2.28), we can also set the initial β to be ∞ like what we did for the OILS problem (2.5). But since the AIP reduction (see Algorithm 2.2) has already computed the Babai point $\boldsymbol{z}_B$, we can set the initial β to be $\|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}_B\|_2$ instead. This initial β will make the search process a little faster than the one with $\beta = \infty$. Note that if there is no valid integer point inside the hyper-ellipsoid when $\beta = \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}_B\|_2$, $\boldsymbol{z}_B$ is the optimal solution of (2.28). There is another strategy which solves a box-constrained real least squares problem

$$\min_{\boldsymbol{z}\in\mathbb{R}^n, \bar{\boldsymbol{l}}\leq\boldsymbol{z}\leq\bar{\boldsymbol{u}}} \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2. \tag{2.44}$$

Suppose the optimal solution of (2.44) is $\boldsymbol{z}_{\text{brls}}$, then we can take $\beta = \|\bar{\boldsymbol{y}} - \boldsymbol{R}\lfloor\boldsymbol{z}_{\text{brls}}\rceil\|_2$. The methods for solving the box-constrained real least squares problem can be found in Sec 5.2.4 of [8] and Sec 16.6 of [56]. Note that if we can not find any integer points with this β, it means that $\lfloor\boldsymbol{z}_{\text{brls}}\rceil$ is the optimal solution of (2.28).

2.3 Underdetermined Box-constrained ILS Problems

In this section, we briefly review the direct tree search (DTS) algorithm proposed in [18] and the partial regularization algorithm proposed in [19] for solving the UBILS problem

$$\min_{\boldsymbol{x}\in\mathcal{B}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2, \tag{2.45}$$

Algorithm 2.3 CH Search

Input: The nonsingular upper triangular matrix $\boldsymbol{R} \in \mathbb{R}^{n\times n}$, the vector $\bar{\boldsymbol{y}} \in \mathbb{R}^n$, the lower bound vector $\bar{\boldsymbol{l}} \in \mathbb{Z}^n$, the upper bound vector $\bar{\boldsymbol{u}} \in \mathbb{Z}^n$, and the initial hyper-ellipsoid bound β.

Output: The solution $\hat{\boldsymbol{z}} \in \mathbb{Z}^n$ to the OBILS problem (2.28).

function: $\hat{\boldsymbol{z}}$ = CH-SEARCH($\boldsymbol{R}, \bar{\boldsymbol{y}}, \bar{\boldsymbol{l}}, \bar{\boldsymbol{u}}, \beta$)

```
Step 1 : (Initialization)
  Set $k := n$ and $Q_k := 0$, compute $T_i$ for $i = 2, \dots, n$
Step 2 :
  Compute $c_k := (\bar{y}_k - \sum_{j=k+1}^{n} r_{kj} z_j)/r_{kk}$, $z_k := \lfloor c_k \rceil$
  Set $lbound_k := 0$ and $ubound_k := 0$
  if $z_k \le \bar{l}_k$ then
      Set $z_k := \bar{l}_k$, $lbound_k := 1$ and $\Delta_k := 1$
  else if $z_k \ge \bar{u}_k$ then
      Set $z_k := \bar{u}_k$, $ubound_k := 1$ and $\Delta_k := -1$
  else
      Set $\Delta_k := \text{sign}(c_k - z_k)$
  end if
Step 3 :
  if $T_k + Q_k + r_{kk}^2 (z_k - c_k)^2 \ge \beta^2$ then
      Go to step 4
  else if $k > 1$ then
      Compute $Q_{k-1} := Q_k + r_{kk}^2 (z_k - c_k)^2$,
      Set $k := k - 1$, go to Step 2
  else
      Compute $\beta := \sqrt{Q_1 + r_{11}^2 (z_1 - c_1)^2}$,
      Set $\hat{\boldsymbol{z}} := \boldsymbol{z}$ and $k := k + 1$, go to Step 5
  end if
Step 4 :
  if $k = n$ then
      Terminate
  else
      Set $k := k + 1$
  end if
Step 5 :(Enumeration of level $k$)
  if $ubound_k = 1$ and $lbound_k = 1$ then
      Go to Step 4
  end if
```

```
Set $z_k := z_k + \Delta_k$
if $z_k = \bar{l}_k$ then
    Set $lbound_k := 1$
    Compute $\Delta_k := -\Delta_k - \text{sign}(\Delta_k)$
else if $z_k = \bar{u}_k$ then
    Set $ubound_k := 1$
    Compute $\Delta_k := -\Delta_k - \text{sign}(\Delta_k)$
else if $lbound_k = 1$ then
    Set $\Delta_k := 1$
else if $ubound_k = 1$ then
    Set $\Delta_k := -1$
else
    Compute $\Delta_k := -\Delta_k - \text{sign}(\Delta_k)$
end if
Go to Step 3
```

where $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{m \times n}(m < n)$ with $\text{rank}(\boldsymbol{A}) = m$, and $\mathcal{B} = \{\boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}, \boldsymbol{l}, \boldsymbol{u} \in \mathbb{Z}^n\}$.

2.3.1 Direct Tree Search (DTS) Algorithm

In this section, we introduce the DTS algorithm which includes two phases: reduction and search. We describe the search process first since the reduction process needs some definitions given in the search part.

Search Process

Suppose that the problem in (2.45) has been reduced to

$$\min_{\boldsymbol{z} \in \mathcal{B}} \|\tilde{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2 \tag{2.46}$$

where $\tilde{\boldsymbol{y}} \in \mathbb{R}^m$, $\boldsymbol{R} \in \mathbb{R}^{m \times n}$ is an upper trapezoidal matrix. To solve (2.46), we find a constant $\beta > 0$ such that the optimal solution of (2.46) satisfies

$$\|\tilde{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2 < \beta^2. \tag{2.47}$$

A method of finding β can be found in [17], which first solves a box constrained real least squares problem and then rounds the solution to an integer vector in $\mathcal{B}$, say $\boldsymbol{z}^*$, and then takes $\beta = \|\tilde{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}^*\|_2$.

Expanding the objective function $\|\tilde{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2$ in (2.47), we can rewrite the inequality (2.47) as

$$\sum_{i=1}^{m}(\tilde{y}_i - \sum_{j=i}^{n} r_{ij}z_j)^2 < \beta^2. \tag{2.48}$$

Thus like the OILS problem and the OBILS problem (see Section 2.1 and Section 2.2), we can derive the following set of inequalities from (2.48):

$$\text{level } m : (\tilde{y}_m - \sum_{j=m}^{n} r_{mj}z_j)^2 < \beta^2, \tag{2.49}$$

$$\vdots$$

$$\text{level } k : (\tilde{y}_k - \sum_{j=k}^{n} r_{kj}z_j)^2 < \beta^2 - \sum_{i=k+1}^{m}(\bar{y}_i - \sum_{j=i}^{n} r_{ij}z_j)^2, \tag{2.50}$$

$$\vdots$$

$$\text{level } 1 : (\tilde{y}_1 - \sum_{j=1}^{n} r_{1j}z_j)^2 < \beta^2 - \sum_{i=2}^{m}(\bar{y}_i - \sum_{j=i}^{n} r_{ij}z_j)^2. \tag{2.51}$$

For simplicity, define

$$c_i = (\tilde{y}_i - \sum_{j=i+1}^{n} r_{ij}z_j)/r_{ii}, \quad i = m, \ldots, 1. \tag{2.52}$$

Thus for $k = m-1, \ldots, 1$, we can define $\mathcal{B}_k = \{l_k, l_k+1, \ldots, u_k\}$ and have

$$\begin{aligned} z_k \in \mathcal{Z}_k &= \mathcal{B}_k \cap (\lambda_k, \varphi_k) \\ \lambda_k &= c_k - \sqrt{\beta^2 - \sum_{i=k+1}^{m} r_{ii}^2(z_i - c_i)^2}/|r_{kk}| \\ \varphi_k &= c_k + \sqrt{\beta^2 - \sum_{i=k+1}^{m} r_{ii}^2(z_i - c_i)^2}/|r_{kk}|. \end{aligned} \tag{2.53}$$

However, unlike the OBILS problem, at level m, the inequality (2.49) has $n-m+1$ variables which needs to be determined. And thus the search approach given in Section 2.2.2 can not work here since it fixes only one variable at each level.

In [60], Yang et al proposed a strategy, which first determines $z_n, \ldots, z_m$ based on (2.49) and then uses the search algorithm to determine $z_{m-1}, \ldots, z_1$ based on (2.50) with $k = m-2, \ldots, 1$. In [18], the DTS approach was proposed to integrate the above two processes into one seamlessly.

Let $\mathcal{J}^+ = \{j | r_{mj} \geq 0, m \leq j \leq n\}$, and $\mathcal{J}^- = \{j | r_{mj} < 0, m \leq j \leq n\}$. Then (2.49) is equvialent to

$$\tilde{y}_m - \beta < \sum_{j \in \mathcal{J}^+} r_{mj} z_j + \sum_{j \in \mathcal{J}^-} r_{mj} z_j < \tilde{y}_m + \beta. \tag{2.54}$$

Define the following transformation for z_j, $j = m, m+1, \ldots, n$

$$\bar{z}_j = \begin{cases} -l_j + z_j & \text{if} \quad j \in \mathcal{J}^+ \\ u_j - z_j & \text{if} \quad j \in \mathcal{J}^- \end{cases}, \tag{2.55}$$

so that

$$\bar{z}_j \in \bar{\mathcal{B}}_j \triangleq \{0, 1, \ldots, u_j - l_j\}. \tag{2.56}$$

Define

$$\alpha = \tilde{y}_m - \sum_{j \in \mathcal{J}^+} |r_{mj}| l_j + \sum_{j \in \mathcal{J}^-} |r_{mj}| u_j, \tag{2.57}$$

then (2.54) becomes

$$\alpha - \beta < \sum_{j=m}^{n} |r_{mj}| \bar{z}_j < \alpha + \beta. \tag{2.58}$$

Assume $r_{mj} \neq 0$ for $j = m, m+1, \ldots, n$, then from (2.56) and (2.58) it is easy to verify that for $j = n, n-1, \ldots, m$

$$\bar{z}_j \in \bar{\mathcal{Z}}_j = \bar{\mathcal{B}}_j \cap (\bar{\lambda}_j, \bar{\varphi}_j) \tag{2.59}$$

$$\bar{\lambda}_j = (\alpha - \beta - \textstyle\sum_{i=j+1}^{n} |r_{mi}|\bar{z}_i - \sum_{i=m}^{j-1} |r_{mi}|(u_i - l_i))/|r_{mj}| \tag{2.60}$$

$$\bar{\mu}_j = (\alpha + \beta - \textstyle\sum_{i=j+1}^{n} |r_{mi}|\bar{z}_i)/|r_{mj}|, \tag{2.61}$$

With inequalities (2.59) and (2.53), a depth-first tree search algorithm can be developed to find the optimal solution of (2.46). Now we describe the DTS strategy starting from level n. First we take $\bar{z}_n = \left\lfloor \frac{\alpha}{|r_{mn}|} \right\rceil |_{\bar{\mathcal{Z}}_n}$, which denotes the nearest integer to $\frac{\alpha}{|r_{mn}|}$ in the set $\bar{\mathcal{Z}}_n$. Then we move to level $n-1$ and compute the set $\bar{\mathcal{Z}}_{n-1}$. If $\bar{\mathcal{Z}}_{n-1}$ is empty, it means that the $\bar{z}_n$ we chose is invalid. Then we move back to level n and choose $\bar{z}_n$ to be the next nearest integer to $\frac{\alpha}{|r_{mn}|}$ in the set $\bar{\mathcal{Z}}_n$ and go to level $n-1$ again. If $\bar{\mathcal{Z}}_{n-1}$ is not empty, we choose $\bar{z}_{n-1} = \left\lfloor \frac{\alpha - |r_{mn}\bar{z}_n|}{|r_{mn-1}|} \right\rceil |_{\bar{\mathcal{Z}}_{n-1}}$. Continue this process until we reach level m and find a valid integer for $\bar{z}_m$. At this point, we have fixed $\bar{z}_n, \ldots, \bar{z}_m$ and can transform $\bar{\boldsymbol{z}}_{m:n}$ to the original integer vector $\boldsymbol{z}_{m:n}$ by (2.55). Then we move to level $m-1$ and compute $\mathcal{Z}_{m-1}$ with (2.53). If $\mathcal{Z}_{m-1}$ is empty, we move up to level m. Otherwise, choose $z_{m-1} = \lfloor c_{m-1} \rceil |_{\mathcal{Z}_{m-1}}$ and move down to level $m-2$. Continue this procedure until we reach level 1 and choose a valid integer for z_1. At this time, we have fixed all the variables in the vector $\boldsymbol{z}$ and a full integer point $\boldsymbol{z}^*$ is found. Then we update β by taking $\beta = \|\tilde{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}^*\|_2$. Next, we start the search process again within the new hyper-ellipsoid by updating $\boldsymbol{z}$. First we go back to level 2 to update the value of z_2 to be the next nearest integer to c_2 in the set $\mathcal{Z}_2$. Note that the candidate set $\mathcal{Z}_2$ has also been updated by using the new β. If it satisfies the inequality (2.50) with $k = 2$, we move down to level 1 to update the value of z_1. Otherwise, we move up to level 3 to update the value of z_3, and so on. In general, in this tree search process, if we find a valid integer at level $j+1$ we move down to level j. If we fail to find any integer value at level j, we move up to level $j+1$. Finally, if we fail to find a new valid integer for $\bar{z}_n$ at level n, the search process terminates and the latest found integer point $\hat{\boldsymbol{z}}$ is the optimal solution we seek.

It is possible that in (2.58) $r_{mj} = 0$ for some j. Under this condition all the integers in the set $\bar{\mathcal{Z}}_j$ can be candidates for $\bar{z}_j$. When we go to level j from level $j+1$, we choose $\bar{z}_j$ to be the smallest integer in the set $\bar{\mathcal{Z}}_j$, i.e., $\bar{z}_j = 0$. And if we go to level j from level $j-1$, we set $\bar{z}_j$ to be the next smallest integer in the set $\bar{\mathcal{Z}}_j$. Fortunately, this rarely occurs in practice.

Reduction Process

In this subsection, we introduce a reduction process to transform the original problem (2.45) to the reduced problem (2.46). Like the AIP reduction introduced in Section 2.2.1, we can apply a permutation matrix $\boldsymbol{P}$ from the right to $\boldsymbol{A}$ such that we can have the QR decomposition

$$\boldsymbol{AP} = \boldsymbol{QR}, \tag{2.62}$$

where $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ is an orthogonal matrix and $\boldsymbol{R} \in \mathbb{R}^{m\times n}$ is an upper trapezoidal matrix. Then we can derive

$$\|\boldsymbol{y} - \boldsymbol{Ax}\|_2^2 = \|\boldsymbol{Q}_1^T\boldsymbol{y} - \boldsymbol{RP}^T\boldsymbol{x}\|_2^2 + \|\boldsymbol{Q}_2^T\boldsymbol{y}\|_2^2. \tag{2.63}$$

Algorithm 2.4 DTS Search

Input: The nonsingular upper trapezoidal matrix $\boldsymbol{R} \in \mathbb{R}^{n\times n}$, the vector $\tilde{\boldsymbol{y}} \in \mathbb{R}^n$, the lower bound vector $\boldsymbol{l} \in \mathbb{Z}^n$, the upper bound vector $\boldsymbol{u} \in \mathbb{Z}^n$, and the initial hyper-ellipsoid bound β.

Output: The solution $\hat{\boldsymbol{z}} \in \mathbb{Z}^n$ to the UBILS problem (2.46).

function: $\hat{\boldsymbol{z}} = \text{DTS-SEARCH}(\boldsymbol{R}, \tilde{\boldsymbol{y}}, \boldsymbol{l}, \boldsymbol{u}, \beta)$

```
Step 1 : (Initialization)
  Set $k = n$
Step 2 :
  Compute $\bar{\lambda}_k$ and $\bar{\mu}_k$ by following (2.60) and (2.61)
  Set $\bar{\mathcal{Z}}_k = \{0, 1, \dots, u_k - l_k\} \cap (\bar{\lambda}_k, \bar{\mu}_k)$
  **if** $\bar{\mathcal{Z}}_k$ is empty **then** Go to step 4
  **else**
      Compute $c_k = \frac{\alpha - \sum_{j=k+1}^{n} |r_{mj}\bar{z}_j|}{|r_{mk}|}$
      Compute $\bar{z}_k = \lfloor c_k \rceil |_{\bar{\mathcal{Z}}_k}$
  **end if**
Step 3 :
  **if** $k > m-1$ **then** Set $k = k-1$, go to step 2
  **else**
      Use (2.55) to transform $\bar{\boldsymbol{z}}_{m:n}$ back to $\boldsymbol{z}_{m:n}$
      Compute $\bar{\boldsymbol{y}} = \boldsymbol{y}_{1:m-1} - \boldsymbol{R}_{1:m-1,m:n}\boldsymbol{z}_{m:n}$, and $T_m = (\tilde{y}_m - \boldsymbol{R}_{m,m:n}\boldsymbol{z}_{m:n})^2$
      Compute $\boldsymbol{z}_{1:m-1}$ = CH-SEARCH($\boldsymbol{R}_{1:m-1,1:m-1}, \bar{\boldsymbol{y}}, \boldsymbol{l}_{1:m-1}, \boldsymbol{u}_{1:m-1}, \sqrt{\beta^2 - T_m}$), see
        Algorithm 2.3
      Set $\hat{\boldsymbol{z}} = \boldsymbol{z}$, $\beta = \sqrt{(\bar{\boldsymbol{y}} - \boldsymbol{R}_{1:m-1,1:m-1}\boldsymbol{z}_{1:m-1})^2 + T_m}$
      Set $k = k+1$ and go to step 5
  **end if**
Step 4 :
  **if** $k = n$ **then**
      Terminate
  **else**
      Set $k = k+1$
  **end if**
Step 5 :
  Choose $\bar{z}_k \in \bar{\mathcal{Z}}_k$ to be the next nearest integer to $c_k$
  **if** $\bar{z}_k$ does not exist **then**
      Go to step 4
  **else**
      Go to step 3
  **end if**
```

With $\tilde{\boldsymbol{y}} = \boldsymbol{Q}_1^T \boldsymbol{y}, \quad \boldsymbol{z} = \boldsymbol{P}^T \boldsymbol{x}, \quad \bar{\boldsymbol{l}} = \boldsymbol{P}^T \boldsymbol{l}, \quad \bar{\boldsymbol{u}} = \boldsymbol{P}^T \boldsymbol{u}$, we transform (2.45) to (2.46). Note that if $\hat{\boldsymbol{z}}$ is the optimal solution to the reduced problem (2.46), $\tilde{\boldsymbol{x}} = \boldsymbol{P}\hat{\boldsymbol{z}}$ is the solution to the problem (2.45).

In the reduction process, we first determine how to choose $n-m+1$ columns as the right part of the permutated $\boldsymbol{A}$, and the remaining $m-1$ columns are the left part of the permutated $\boldsymbol{A}$. Then we determine how to order the columns for each part.

Assuming that the two parts of $\boldsymbol{A}$ have been determined (we will show how to do it later) and we have a permutated $\boldsymbol{A}$ and a corresponding upper trapezoidal matrix $\boldsymbol{R}$. In the following, we show how to reorder the last $n-m+1$ columns of $\boldsymbol{R}$. At level j $(m \le j \le n)$, the integer set $\bar{\mathcal{Z}}_j$ can be computed by (2.59). The smaller the number of integers in $\bar{\mathcal{Z}}_j$ is, the more efficient the search process would be. This motivates the following reordering strategy. Define

$$L_j = \min\{u_j - l_j, \lfloor \bar{\mu}_j \rfloor + \text{sign}(\bar{\mu}_j - \lfloor \bar{\mu}_j \rfloor) - 1\} - \max\{0, \lceil \bar{\lambda}_j \rceil - \text{sign}(\lceil \bar{\lambda}_j \rceil - \bar{\lambda}_j) + 1\}. \quad (2.64)$$

Here if $L_j > 0$, L_j is the number of integers in $\bar{\mathcal{Z}}_j$. If $L_j \le 0$, it means that the set $\bar{\mathcal{Z}}_j$ is empty. Suppose that we have determined the last $n-j$ columns of $\boldsymbol{R}$, now we need to choose the jth column $(j \ge m)$ of $\boldsymbol{R}$. First we compute L_j with the current jth column of $\boldsymbol{R}$, and then we interchange column j and column i of $\boldsymbol{R}$ for $i = m, m+1, \ldots, j-1$. Then we compute the corresponding L_j after each interchange and find the smallest L_j. Then the corresponding colmun with smallest L_j is chosen to be the jth column of $\boldsymbol{R}$ and we go to level $j-1$ to determine column $j-1$. We will repeat this procedure until we order all the last $n-m+1$ columns or we encounter a column for which the smallest L_j is nonpositive. The above procedure is described in Algorithm 2.5.

In Algorithm 2.5, there are two outputs which we need to explain. One is *index*, which is the largest column number of those columns which have not been reordered in Algorithm 2.5. The other is *Prod*, which is a product of numbers of candidates at levels higher than *index*. The empty set $\bar{\mathbb{Z}}_{index}$ means the search process has to move to level $index + 1$. In other words, we cut off a branch at level *index* in the search tree. When the value of *index* is larger, the branch we can cut off is larger, and the search process is likely to be more efficient. Therefore, we would like to get a larger *index* by reordering some columns of $\boldsymbol{R}$.

Now we are ready to describe the entire reduction process. (1) Compute the Householder QR decomposition (2.64) and in the jth step of this process for $j = 1 : m$, a column permutation will be applied such that $|r_{jj}|$ is the smallest positive number we can achieve. This permutation will tend to make $|r_{mj}|$ larger for $j = m + 1 : n$, as in terms of 2-norm, m smaller columns of $\boldsymbol{A}$ have been moved to the left. Larger $|r_{mj}|$ tends to make $\bar{u}_j$ smaller (see (2.61)). We mentioned before that λ_j is usually nonpositive. Hence, from (2.64), we see that larger $|r_{mj}|$ is likely to lead smaller L_j. In this step, the last $n - m$ columns are as a group determined. (2) For $j = 1 : m$, we first interchange columns j and m of $\boldsymbol{R}$ and bring $\boldsymbol{R}$ back to upper trapezoid by using Givens rotations. Note that when we apply a Given rotation to $\boldsymbol{R}$, we always simultaneously update $\tilde{\boldsymbol{y}}$ by the same rotation. Then we apply Algorithm 2.5 to the new $\boldsymbol{R}$ to reorder (part of) the last $n - m + 1$ columns and get corresponding *index* and *Prod*. Then after those m steps we find the largest one among the m values of

Algorithm 2.5 Reorder part of columns of $\boldsymbol{R}_{:,m:n}$

Input: Upper trapezoidal matrix $\boldsymbol{R} \in \mathbb{R}^{m\times n}$, $\tilde{y}_m, \beta$

Output: Permuted $\boldsymbol{R}$, $n \times n$ permutation matrix $\boldsymbol{P}$, variable $index$ for which $\bar{\mathcal{Z}}_{index}$ is empty, variable $Prod = \Pi_{j=index+1}^{n} L_j$

function: $[\boldsymbol{R}, \boldsymbol{P}, index, Prod] = \text{REORDER}(\boldsymbol{R}, \tilde{y}_m, \beta)$

```
 1: Set $\boldsymbol{P} = \boldsymbol{I}_n$, $index = m-1$, $Prod = 1$, $j = n$
 2: while $j \geq m$ do
 3:     Compute $L_j$ according to (2.64), set $p = j$
 4:     for $i = m : j-1$ do
 5:         Set $\boldsymbol{R}' = \boldsymbol{R}$, and interchange columns $i$ and $j$ of $\boldsymbol{R}'$, then compute the
                corresponding $L_i'$
 6:         if $L_i' < L_j$ then
 7:             Set $p = i$, $L_j = L_i'$
 8:         end if
 9:     end for
10:     if $p \neq j$ then
11:         Interchange columns $p$ and $j$ of $\boldsymbol{P}$ and $\boldsymbol{R}$
12:     end if
13:     if $L_j \leq 0$ then
14:         Set $index = j$, and break the while loop
15:     end if
16:     Compute $\bar{z}_j = \left\lfloor \frac{\alpha - \sum_{l=i+1}^{n} |r_{ml}\bar{z}_l|}{|r_{mj}|} \right\rceil|_{\bar{\mathcal{Z}}_j}$
17:     Compute $Prod = Prod * L_j$, $j = j-1$
18: end while
```

index and the corresponding order. Note that if there are more than one orderings giving the same largest *index*, we choose the one which gives the smallest *Prod*. For the chosen ordering, if $index > m+1$, we order columns $m, m-1, \ldots, index$ as follows. From (2.64), (2.60) and (2.61) we can observe that a larger $|r_{mj}|$ leads to a smaller L_j if $\bar{\lambda}_j \leq 0$. The simulations in [18] indicated that usually $\bar{\lambda}_j \leq 0$ if $\tilde{y}_m \leq 0$. And $\tilde{y}_m \leq 0$ can always be guaranteed because that if $\tilde{y}_m > 0$, we can simply multiply both $\tilde{y}_m$ and the last row of $\boldsymbol{R}$ by -1. Thus we reorder columns $m, m+1, \ldots, index-1$ of $\boldsymbol{R}$ such that $|r_{mm}| \leq |r_{m,m+1}| \leq \ldots \leq |r_{m,index-1}|$. (3) In this step we reorder the first $m-1$ columns. Note that the last $n-m+1$ columns have been found and ordered, we can get the first valid $\boldsymbol{z}_{(2)}$ by the general search process introduced in Section 2.3.1. Then problem (2.45) becomes an OBILS problem and we can employ the AIP algorithm given in [58], see Section 2.2.1, to order the first $m-1$ columns of $\boldsymbol{R}$. The details about the algorithm of reordering strategy of the DTS algorithm can be found in Algorithm 2.6.

2.3.2 Partial Regularization (PR) Algorithm

Different from the direct tree search algorithm introduced in Section 2.3.1, in this subsection, the partial regularization approach proposed in [19] will be first presented to solve the UBILS problem (2.45), where $\mathcal{B}$ is replaced by the following constraint:

$$\mathcal{X}_p^n = \mathcal{X}_p \times \mathcal{X}_p \cdots \times \mathcal{X}_p, \;\; \mathcal{X}_p = \{\pm 1, \pm 3, \ldots, \pm(2^p-3), \pm(2^p-1)\},$$

which arises from MIMO applications. Later we slightly modify the approach to deal with a box constraint problem.

Algorithm 2.6 DTS Reduction

Input: Given matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{y} \in \mathbb{R}^n$, β

Output: Permutation matrix $\boldsymbol{P} \in \mathbb{Z}^{n \times n}$, upper trapezoidal matrix $\boldsymbol{R} \in \mathbb{R}^{m \times n}$ and $\tilde{\boldsymbol{y}} \in \mathbb{R}^m$.

function: $[\boldsymbol{P}, \boldsymbol{R}, \tilde{\boldsymbol{y}}] = \text{DTS-RED}(\boldsymbol{A}, \boldsymbol{y}, \beta)$

```
1: Compute $\boldsymbol{AP} = \boldsymbol{QR}$ with minimum pivoting
2: Set $\tilde{\boldsymbol{y}} = \boldsymbol{Q}^T\boldsymbol{y}$, $index = 0$, $Prod = +\infty$
3: for $j = 1 : m$ do
4:     Set $\boldsymbol{R}' = \boldsymbol{R}$, $\tilde{\boldsymbol{y}}' = \tilde{\boldsymbol{y}}$
5:     if $j \neq m$ then
6:         Swap columns $j$ and $m$ of $\boldsymbol{R}'$ and transform $\boldsymbol{R}'$ to an upper trapezoidal
               matrix by Givens rotations. Apply the same Givens rotations to $\tilde{\boldsymbol{y}}'$
7:     end if
8:     if $\tilde{y}'_m > 0$ then
9:         Set $\tilde{y}'_m = -\tilde{y}'_m$, $\boldsymbol{R}'_{m,:} = -\boldsymbol{R}'_{m,:}$
10:    end if
11:    $[\boldsymbol{R}', \boldsymbol{P}', ind_{tmp}, Prod_{tmp}] = \text{REORDER}(\boldsymbol{R}', \tilde{y}'_m, \beta)$, see Algorithm 2.5
12:    if $ind_{tmp} > index$ then
13:        Set $index = ind_{tmp}$, $p = j$, $\boldsymbol{R}_{tmp} = \boldsymbol{R}'$, $\tilde{\boldsymbol{y}}_{tmp} = \tilde{\boldsymbol{y}}'$, $\boldsymbol{P}_{tmp} = \boldsymbol{P}'$
14:    else if $ind_{tmp} = index$ then
15:        if $Prod_{tmp} < Prod$ then
16:            $Prod = Prod_{tmp}$, $p = j$, $\boldsymbol{R}_{tmp} = \boldsymbol{R}'$, $\tilde{\boldsymbol{y}}_{tmp} = \tilde{\boldsymbol{y}}'$, $\boldsymbol{P}_{tmp} = \boldsymbol{P}'$
17:        end if
18:    end if
19: end for
20: if $p \neq m$ then
21:    Swap columns $p$ and $m$ of $\boldsymbol{P}$
22:    Set $\boldsymbol{P} = \boldsymbol{PP}_{tmp}$, $\boldsymbol{R} = \boldsymbol{R}_{tmp}$, $\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{y}}_{tmp}$
23: end if
24: if $index > m + 1$ then
25:    Reorder the columns of $\boldsymbol{R}_{:,m:index-1}$ such that $|r_{mm}| \leq |r_{m,m+1}| \leq \ldots \leq$
           $|r_{m,index-1}|$,and reorder the columns of $\boldsymbol{P}$ correspondingly
26: end if
27: Use the tree search algorithm to find the first $\boldsymbol{z}_{(2)}$ then use Algorithm 2.2 to
    reorder the first $m-1$ columns of $\boldsymbol{R}$, leading to new $\boldsymbol{R}$, $\bar{\boldsymbol{y}}$ and $\boldsymbol{P}$ correspondingly.
```

Partition $\boldsymbol{A}$ and $\boldsymbol{x}$ as follows:

$$\boldsymbol{A} = [\underset{m}{\boldsymbol{A}_1}, \ \underset{l}{\boldsymbol{A}_2}], \quad \boldsymbol{x} = \begin{array}{c} m \\ l \end{array} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} \tag{2.65}$$

where $l = n - m$. Following [21], we can write $\boldsymbol{x}_2 \in \mathcal{X}_p^l$ as a linear combination of $\boldsymbol{x}_2^{(i)} \in \mathcal{X}_1^l$ for $0 \le i \le p - 1$:

$$\boldsymbol{x}_2 = \sum_{i=0}^{p-1} 2^i \boldsymbol{x}_2^{(i)}. \tag{2.66}$$

Define

$$\bar{\boldsymbol{A}}_2 = \left[\boldsymbol{A}_2, \quad 2\boldsymbol{A}_2, \quad \dots, \quad 2^{p-1}\boldsymbol{A}_2\right] \in \mathbb{R}^{m\times pl}, \quad \bar{\boldsymbol{x}}_2 = \begin{bmatrix} \boldsymbol{x}_2^{(0)} \\ \boldsymbol{x}_2^{(1)} \\ \vdots \\ \boldsymbol{x}_2^{(p-1)} \end{bmatrix} \in \mathbb{R}^{pl}. \tag{2.67}$$

Notice that $\|\bar{\boldsymbol{x}}_2\|_2^2 = pl$, so $\|\bar{\boldsymbol{x}}_2\|_2^2$ is constant. Then from (2.66) and (2.67), we see that the UBILS problem (2.45) with $\mathcal{B}$ replaced by $\mathcal{X}_p^n$ is equivalent to

$$\min_{\boldsymbol{x}_1 \in \mathcal{X}_p^m, \bar{\boldsymbol{x}}_2 \in \mathcal{X}_1^{pl}} \left\| \boldsymbol{y} - \left[\boldsymbol{A}_1, \quad \bar{\boldsymbol{A}}_2\right] \begin{bmatrix} \boldsymbol{x}_1 \\ \bar{\boldsymbol{x}}_2 \end{bmatrix} \right\|_2^2 + \alpha^2 \|\bar{\boldsymbol{x}}_2\|_2^2, \tag{2.68}$$

where α is a regularization parameter. Therefore with

$$\begin{aligned}
&\bar{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A}_1 & \bar{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \alpha \boldsymbol{I} \end{bmatrix} \in \mathbb{R}^{(m+pl)\times(m+pl)}, \quad \bar{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x}_1 \\ \bar{\boldsymbol{x}}_2 \end{bmatrix} \in \mathbb{R}^{m+pl}, \quad \bar{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{m+pl} \\
&\bar{\mathcal{X}} = \left\{ \begin{bmatrix} \boldsymbol{x}_1 \\ \bar{\boldsymbol{x}}_2 \end{bmatrix} : \boldsymbol{x}_1 \in \mathcal{X}_p^m, \bar{\boldsymbol{x}}_2 \in \mathcal{X}_1^{pl} \right\},
\end{aligned} \tag{2.69}$$

the problem (2.68) can be rewritten as

$$\min_{\bar{\boldsymbol{x}} \in \bar{\mathcal{X}}} \|\bar{\boldsymbol{y}} - \bar{\boldsymbol{A}}\bar{\boldsymbol{x}}\|_2^2. \tag{2.70}$$

Obviuosly, (2.70) is an overdetermined ILS problem and we slightly modify the method introduced in Section 2.2 to solve it (the modification is needed to deal with the special constraint).

It is difficult to extend the PR approach to a general box constraint, however, we can extend it to the following box constraint:

$$\mathcal{B} = \{\boldsymbol{x} \in \mathbb{Z}^n, \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}, \boldsymbol{l}, \boldsymbol{u} \in \mathbb{Z}^n, u_j - l_j = 2^{p_j} - 1, p_j \in \mathbb{Z}^+, j = 1, 2, \ldots, n\}.$$

First, for each x_j, $j = m+1, m+2, \ldots, n$, we do the following shift:

$$\bar{x}_j = x_j - l_j \in \{0, 1, \ldots, u_j - l_j\}. \tag{2.71}$$

Since $\bar{x}_j$ is non-negative we can write it as a linear combination of $\bar{x}_j^{(i)} \in \{0, 1\}$ for $i = 0, 1, \ldots, p_j - 1$:

$$\bar{x}_j = \sum_{i=0}^{p_j - 1} 2^i \bar{x}_j^{(i)}. \tag{2.72}$$

Define the following transformation for each $\bar{x}_j^{(i)}$, $j = m+1, m+2, \dots, n$, and $i = 0, 1, \dots, p_j - 1$:

$$z_j^{(i)} = 2\bar{x}_j^{(i)} - 1, \tag{2.73}$$

then combining (2.71), (2.72) and (2.73) we have

$$x_j = \sum_{i=0}^{p_j-1} 2^i (z_j^{(i)} + 1)/2 + l_j, \quad j = m+1, m+2, \dots, n \tag{2.74}$$

where $z_j^{(i)} \in \{-1, 1\}$. Then we can define

$$\bar{\boldsymbol{A}}_2 = \left[\boldsymbol{a}_{m+1}^{(0)}, \quad \dots, \quad \boldsymbol{a}_{m+1}^{(p_{m+1}-1)}, \quad \boldsymbol{a}_{m+2}^{(0)}, \quad \dots, \quad \boldsymbol{a}_{m+2}^{(p_{m+2}-1)}, \quad \dots, \quad \boldsymbol{a}_{n}^{(0)}, \quad \dots, \quad \boldsymbol{a}_{n}^{(p_n-1)} \right],$$

$$\bar{\boldsymbol{x}}_2 = \begin{bmatrix} z_{m+1}^{(0)} \\ \vdots \\ z_{m+1}^{(p_{m+1}-1)} \\ z_{m+2}^{(0)} \\ \vdots \\ z_{m+2}^{(p_{m+2}-1)} \\ \vdots \\ z_n^{(0)} \\ \vdots \\ z_n^{(p_n-1)} \end{bmatrix} \in \{-1, 1\}^q, \tag{2.75}$$

where $\boldsymbol{a}_j^{(i)} = 2^{i-1}\boldsymbol{A}_{:,j}$ and $q = \sum_{j=m+1}^{n} p_j$. It is easy to see that the original UBILS problem (2.46) can be transformed to an equivalent problem

$$\min_{\boldsymbol{x}_1 \in \mathcal{B}_{1:m}, \bar{\boldsymbol{x}}_2 \in \{-1,1\}^q} \left\| (\boldsymbol{y} - \boldsymbol{A}_2\boldsymbol{c}) - \begin{bmatrix} \boldsymbol{A}_1, & \bar{\boldsymbol{A}}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \bar{\boldsymbol{x}}_2 \end{bmatrix} \right\|_2^2, \tag{2.76}$$

where $c_j = l_{m+j} + \frac{1}{2}(2^{p_{m+j}} - 1)$ for $j = 1, 2, \ldots, n-m$. This problem is also equivalent to the OBILS problem:

$$\min_{\bar{\boldsymbol{x}} \in \bar{\mathcal{X}}} \|\bar{\boldsymbol{y}} - \bar{\boldsymbol{A}}\bar{\boldsymbol{x}}\|_2^2,$$

where

$$\bar{\boldsymbol{A}} = \begin{bmatrix} \boldsymbol{A}_1 & \bar{\boldsymbol{A}}_2 \\ \boldsymbol{0} & \alpha \boldsymbol{I} \end{bmatrix} \in \mathbb{R}^{(m+q)\times(m+q)}, \quad \bar{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x}_1 \\ \bar{\boldsymbol{x}}_2 \end{bmatrix} \in \mathbb{R}^{m+q}, \quad \bar{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{y} - \boldsymbol{A}_2\boldsymbol{c} \\ \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{m+q}$$

$$\bar{\mathcal{X}} = \left\{ \begin{bmatrix} \boldsymbol{x}_1 \\ \bar{\boldsymbol{x}}_2 \end{bmatrix} : \boldsymbol{x}_1 \in \mathcal{B}_{1:m}, \bar{\boldsymbol{x}}_2 \in \{-1,1\}^q \right\}, \ \alpha \text{ is a constant parameter.}$$

This problem can be solved by algorithms we introduced for the OBILS problem in Section 2.2.

CHAPTER 3
Lower Bounds for UBILS Problems

To speed up the search process for OBILS problems, lower bounds have been used in the literature. In this chapter we first review three existing lower bounds for OBILS problems and then we propose a new lower bound for UBILS problems.

3.1 Lower Bounds for OBILS Problems

In Section 2.2.1, we introduced the AIP reduction algorithm for the OBILS problem

$$\min_{\boldsymbol{x}\in\mathcal{B}} \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2, \tag{3.1}$$

where $\boldsymbol{y}\in\mathbb{R}^m$, $\boldsymbol{A}\in\mathbb{R}^{m\times n}$ is a full column rank matrix $(m\geq n)$, $\mathcal{B}=\{\boldsymbol{x}\in\mathbb{Z}^n : \boldsymbol{l}\leq\boldsymbol{x}\leq\boldsymbol{u},\ \boldsymbol{l},\boldsymbol{u}\in\mathbb{Z}^n\}$. For convenience, we will first repeat some part given in Section 2.2.1. First a QRP decomposition is used to reduce the problem (3.1):

$$\boldsymbol{A}\boldsymbol{P}=\boldsymbol{Q}\begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{bmatrix}=\boldsymbol{Q}_1\boldsymbol{R}, \tag{3.2}$$

where $\boldsymbol{P}$ is a permutation matrix, $\boldsymbol{Q}=[\underset{m}{\boldsymbol{Q}_1},\underset{n-m}{\boldsymbol{Q}_2}]\in\mathbb{R}^{m\times m}$ is orthogonal, $\boldsymbol{R}\in\mathbb{R}^{n\times n}$ is an upper triangular matrix. With (3.2), we have

$$\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2=\|\boldsymbol{Q}_1^T\boldsymbol{y}-\boldsymbol{R}\boldsymbol{P}^T\boldsymbol{x}\|_2^2+\|\boldsymbol{Q}_2^T\boldsymbol{y}\|_2^2. \tag{3.3}$$

Define

$$\bar{\boldsymbol{y}}=\boldsymbol{Q}_1^T\boldsymbol{y},\quad \boldsymbol{z}=\boldsymbol{P}^T\boldsymbol{x},\quad \bar{\boldsymbol{l}}=\boldsymbol{P}^T\boldsymbol{l},\quad \bar{\boldsymbol{u}}=\boldsymbol{P}^T\boldsymbol{u}, \tag{3.4}$$

then the original problem (3.1) can be transformed to

$$\min_{\boldsymbol{z}\in\bar{\mathcal{B}}} \|\bar{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2, \tag{3.5}$$

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

After the reduction, a search process can start to search for the optimal solution to the reduced OBILS problem (3.5), see Section 2.2.2. The search process at level k uses the following inequality (see (2.40)):

$$r_{kk}^2(z_k - c_k)^2 < \beta^2 - \sum_{i=k+1}^{n} r_{ii}^2(z_i - c_i)^2 - T_k, \tag{3.6}$$

where $T_k \le \sum_{i=1}^{k-1} r_{ii}^2(z_i - c_i)^2$. Obviously, the tighter T_k we get, the more nodes we can prune on the search tree. However, if the cost of computing lower bounds is significant, we can not get a good performance even if the lower bounds are tight. Thus a trade-off between the tightness and the computational cost of lower bounds needs to be considered. In the following part, we first review some existing lower bounds and then introduce three lower bounds in detail.

To make the search process for the OBILS problems more efficient, various lower bounds have been proposed. In [51], Stojnic et al proposed one lower bound based on solving quadratic optimization problems (QOP) and another lower bound which only works for binary variables by solving semidefinite programming (SDP). They claim that the second lower bound can be extended to the general problem but the extension is complex. In [12], Buchheim et al proposed a lower bound by solving SDP but it is expensive to compute. In [31], Garcia et al proposed a lower bound involving the smallest singular value of the matrix. Since it does not need to solve

QOP or SDP problems, the cost of this lower bound is smaller than those proposed in [51] and [12] but it is less tighter. In the following, three of those lower bounds for OBILS problems will be introduced in details.

At level k we find a lower bound T_k, i.e.,

$$T_k \leq \sum_{j=1}^{k-1} r_{jj}^2 (z_j - c_j)^2 = \|(\bar{\boldsymbol{y}}_{1:k-1} - \boldsymbol{R}_{1:k-1,k:n}\boldsymbol{z}_{k:n}) - \boldsymbol{R}_{1:k-1,1:k-1}\boldsymbol{z}_{1:k-1}\|_2^2, \tag{3.7}$$

where $\boldsymbol{z}_{k:n}$ has been chosen and $\boldsymbol{z}_{1:k-1}$ is unknown. To simplify notation, we consider lower bounds for the following model

$$\min_{\boldsymbol{z}\in\mathcal{B}} \|\boldsymbol{y} - \boldsymbol{R}\boldsymbol{z}\|_2^2, \tag{3.8}$$

where each quantity is defined as before. Then we can obtain T_k for $k = n, n-1, \ldots, 2$. Define $\boldsymbol{z}_{\text{rls}} = \boldsymbol{R}^{-1}\boldsymbol{y}$, the real least squares solution to (3.8) if $\mathcal{B}$ is replaced by $\mathbb{R}^n$. Now we want to find a lower bound T, such that

$$0 \leq T \leq \|\boldsymbol{y} - \boldsymbol{R}\boldsymbol{z}\|_2^2. \tag{3.9}$$

Let $\boldsymbol{C} \in \mathbb{R}^{k\times n}$ be any arbitrary matrix. Then

$$\|\boldsymbol{C}(\boldsymbol{z} - \boldsymbol{z}_{\text{rls}})\|_2 = \|\boldsymbol{C}\boldsymbol{R}^{-1}(\boldsymbol{R}\boldsymbol{z} - \boldsymbol{y})\|_2 \leq \|\boldsymbol{C}\boldsymbol{R}^{-1}\|_2 \|\boldsymbol{y} - \boldsymbol{R}\boldsymbol{z}\|_2$$

leading to a lower bound on the objective function of the OBILS problem (3.8):

$$\min_{\boldsymbol{z}\in\mathcal{B}} \|\boldsymbol{y} - \boldsymbol{R}\boldsymbol{z}\|_2 \geq \frac{\min_{\boldsymbol{z}\in\mathcal{B}} \|\boldsymbol{C}(\boldsymbol{z} - \boldsymbol{z}_{\text{rls}})\|_2}{\|\boldsymbol{C}\boldsymbol{R}^{-1}\|_2}. \tag{3.10}$$

The inequality (3.10) gives a framework for a class of lower bounds. We introduce three different lower bounds based on different choices of $\boldsymbol{C}$.

3.1.1 A Norm-wise Based Lower Bound

Take $\boldsymbol{C} = \boldsymbol{I}$. From (3.10), we obtain

$$\min_{\boldsymbol{z}\in\mathcal{B}} \|\boldsymbol{y} - \boldsymbol{R}\boldsymbol{z}\|_2 \geq \frac{\|\boldsymbol{z}_{\text{rls}}|_{\mathcal{B}} - \boldsymbol{z}_{\text{rls}}\|_2}{\|\boldsymbol{R}^{-1}\|_2}, \tag{3.11}$$

where $(\boldsymbol{z}_{\text{rls}}|_{\mathcal{B}})(i) = z_{\text{rls}}(i)|_{\mathcal{B}_i}$ with

$$z_{\text{rls}}(i)|_{\mathcal{B}_i} = \begin{cases} \lfloor z_{\text{rls}}(i) \rceil & \text{if} \quad l_i \leq z_{\text{rls}}(i) \leq u_i \\ l_i & \text{if} \quad z_{\text{rls}}(i) < l_i \\ u_i & \text{if} \quad z_{\text{rls}}(i) > u_i \end{cases}, \quad \text{for } i = 1, \ldots, n.$$

This lower bound was proposed in [50] for OBILS problems.

3.1.2 A Component-wise Based Lower Bound

Take $\boldsymbol{C} = \boldsymbol{e}_i^T$ for $i = 1, \ldots, n$. From (3.10), we obtain

$$\min_{\boldsymbol{z}\in\mathcal{B}} \|\boldsymbol{y} - \boldsymbol{R}\boldsymbol{z}\|_2 \geq \max_i \frac{|z_{\text{rls}}(i)|_{\mathcal{B}_i} - z_{\text{rls}}(i)|}{\|\boldsymbol{e}_i^T \boldsymbol{R}^{-1}\|_2}, \tag{3.12}$$

where $\boldsymbol{z}_{\text{rls}}(i)|_{\mathcal{B}_i}$ is defined as before. This lower bound was given in [31].

3.1.3 A Basis Reduction Based Lower Bound

Take $\boldsymbol{C} = \boldsymbol{t}^T \in \mathbb{Z}^{1\times n}$. From (3.10), we obtain

$$\min_{\boldsymbol{z}\in\mathcal{B}} \|\boldsymbol{y} - \boldsymbol{R}\boldsymbol{z}\|_2 \geq \min_{\boldsymbol{z}\in\mathcal{B}} \frac{|\boldsymbol{t}^T(\boldsymbol{z} - \boldsymbol{z}_{\text{rls}})|}{\|\boldsymbol{t}^T \boldsymbol{R}^{-1}\|_2}. \tag{3.13}$$

Suppose $\boldsymbol{T} = [\boldsymbol{t}_1, \ldots, \boldsymbol{t}_n] \in \mathbb{Z}^{n\times n}$ is a unimodular matrix which reduces the lattice basis matrix $\boldsymbol{R}^{-T}$, i.e., the columns of $\boldsymbol{R}^{-T}\boldsymbol{T}$ becomes shorter. In order to make the lower bound in (3.13) large, [12] suggests to take $\boldsymbol{t} = \boldsymbol{t}_i$ for $i = 1, \ldots, n$. Then from

(3.13) we have

$$\begin{aligned}\min_{\boldsymbol{z}\in\mathcal{B}} \|\boldsymbol{y}-\boldsymbol{R}\boldsymbol{z}\|_2 &\geq \max_i \min_{\boldsymbol{z}\in\mathcal{B}} \frac{|\boldsymbol{t}_i^T(\boldsymbol{z}-\boldsymbol{z}_{\text{rls}})|}{\|\boldsymbol{t}_i^T\boldsymbol{R}^{-1}\|_2} \\ &\geq \max_i \frac{\max\{|\lfloor \boldsymbol{t}_i^T\boldsymbol{z}_{\text{rls}}\rceil - \boldsymbol{t}_i^T\boldsymbol{z}_{\text{rls}}|, \boldsymbol{t}_i^T\boldsymbol{z}_{\text{rls}} - \tau_i^{\max}, \tau_i^{\min} - \boldsymbol{t}_i^T\boldsymbol{z}_{\text{rls}}\}}{\|\boldsymbol{t}_i^T\boldsymbol{R}^{-1}\|_2},\end{aligned} \tag{3.14}$$

where $\tau_i^{\max} = \max_{\boldsymbol{z}\in\mathcal{B}} \boldsymbol{t}_i^T\boldsymbol{z}$ and $\tau_i^{\min} = \min_{\boldsymbol{z}\in\mathcal{B}} \boldsymbol{t}_i^T\boldsymbol{z}$.

It is easy to verify that

$$\tau_i^{\max} = \boldsymbol{t}_i^T\boldsymbol{z}^{\max}, \quad \tau_i^{\min} = \boldsymbol{t}_i^T\boldsymbol{z}^{\min}, \tag{3.15}$$

where $\boldsymbol{z}^{\max}, \boldsymbol{z}^{\min} \in \mathbb{Z}^n$ and

$$z_j^{\max} = \begin{cases} u_j & \text{if} \quad \boldsymbol{t}_i(j) > 0 \\ l_j & \text{if} \quad \boldsymbol{t}_i(j) \leq 0 \end{cases}, \quad z_j^{\min} = \begin{cases} l_j & \text{if} \quad \boldsymbol{t}_i(j) > 0 \\ u_j & \text{if} \quad \boldsymbol{t}_i(j) \leq 0 \end{cases}, \quad \text{for } j = 1, \ldots, n. \tag{3.16}$$

Now we make a comment about how to get the unimodular matrix $\boldsymbol{T}$. When we solve the OILS problem, we use the LLL reduction to reduce the basis matrix $\boldsymbol{R}$, see Section 2.1.1 and we can use the same reduction algorithm here to reduce $\boldsymbol{R}^{-T}$. Thus $\boldsymbol{R}^{-T}$ can be reduced if we apply the LLL reduction on it. Then, we can set $\boldsymbol{T}$ to be the unimodular matrix. This idea was given in [12] for convex quadratic integer programming and here we use it for OBILS problems.

3.2 Lower Bounds for UBILS Problems

In Section 3.1, we introduced three different lower bounds for solving OBILS problems. However, to our best knowledge, no lower bounds have been proposed for the UBILS problem (see (2.45)). In this section, we propose lower bounds for this problem, which is repeated here for convenience:

$$\min_{\boldsymbol{x} \in \mathcal{B}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2, \tag{3.17}$$

where $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}(m < n)$ with $\text{rank}(\boldsymbol{A}) = m$, and $\mathcal{B} = \{\boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}, \boldsymbol{l}, \boldsymbol{u} \in \mathbb{Z}^n\}$.

In Section 3.2.1 we derive lower bounds which can be incorporated into the DTS algorithm. And then in Section 3.2.2 we show how to incorporate lower bounds into the partial regularization algorithm.

3.2.1 Incorporating Lower Bounds in the DTS Algorithm

Lower Bounds for the Underdetermined Part

In Section 2.3.1, we introduced the DTS algorithm proposed in [18] for solving (3.1). For convenience, we briefly repeat some parts given in Section 2.3.1. We first compute the QRP decomposition

$$\boldsymbol{A}\boldsymbol{P} = \boldsymbol{Q}\boldsymbol{R}, \tag{3.18}$$

where $\boldsymbol{P}$ is a unimodualr matrix, $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ is an orthogonal matrix and $\boldsymbol{R} \in \mathbb{R}^{m\times n}$ is an upper trapezoidal matrix. Then with $\tilde{\boldsymbol{y}} = \boldsymbol{Q}^T\boldsymbol{y}$, $\boldsymbol{z} = \boldsymbol{P}^T\boldsymbol{x}$, $\bar{\boldsymbol{l}} = \boldsymbol{P}^T\boldsymbol{l}$, $\bar{\boldsymbol{u}} = \boldsymbol{P}^T\boldsymbol{u}$, the original UBILS problem (3.1) can be transformed to

$$\min_{\boldsymbol{z} \in \bar{\mathcal{B}}} \|\tilde{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2 \tag{3.19}$$

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

To solve (3.19), we find a constant β, see Section 2.3, such that

$$\|\tilde{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2 < \beta^2. \tag{3.20}$$

Partition $\boldsymbol{z}$, $\tilde{\boldsymbol{y}}$, $\boldsymbol{R}$ in (3.19) as follows

$$\boldsymbol{z} = \begin{array}{c} m-1 \\ l \end{array} \begin{bmatrix} \boldsymbol{z}_{(1)} \\ \boldsymbol{z}_{(2)} \end{bmatrix}, \quad \tilde{\boldsymbol{y}} = \begin{array}{c} m-1 \\ 1 \end{array} \begin{bmatrix} \bar{\boldsymbol{y}} \\ \tilde{y}_m \end{bmatrix}, \quad \boldsymbol{R} = \begin{array}{c} \\ m-1 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} m-1 & l \end{array} \\ \begin{bmatrix} \boldsymbol{R}_1 & \boldsymbol{R}_2 \\ & \boldsymbol{r}_m^T \end{bmatrix} \end{array} \tag{3.21}$$

where $l = n - m + 1$, then it is easy to see that the inequality (3.20) is equivalent to a set of two inequalities:

$$(\tilde{y}_m - \sum_{j=m}^{n} r_{mj} z_j)^2 < \beta^2 \tag{3.22}$$

$$\|\bar{\boldsymbol{y}} - \boldsymbol{R}_1 \boldsymbol{z}_{(1)} - \boldsymbol{R}_2 \boldsymbol{z}_{(2)}\|_2^2 < \beta^2 - (\tilde{y}_m - \sum_{j=m}^{n} r_{mj} z_j)^2. \tag{3.23}$$

In [18], the inequality (3.22) is used to determine $z_n, z_{n-1}, \ldots, z_m$ and (3.23) is used to determine remaining elements $z_{m-1}, z_{m-2}, \ldots, z_1$. However, the upper bound β^2 in (3.22) may be very loose, since we ignore the other elements of the vector on the left hand side of (3.20) when we derived (3.22) from (3.20). If we can find a smaller upper bound for (3.22), we may make the search process more efficient in determining $z_n, z_{n-1}, \ldots, z_m$. Note that we can rewrite (3.23) as

$$(\tilde{y}_m - \sum_{j=m}^{n} r_{mj} z_j)^2 < \beta^2 - \|\bar{\boldsymbol{y}} - \boldsymbol{R}_1 \boldsymbol{z}_{(1)} - \boldsymbol{R}_2 \boldsymbol{z}_{(2)}\|_2^2. \tag{3.24}$$

Suppose we can find a lower bound T for the second term on the right hand side of (3.24), i.e.,

$$0 < T \le \min_{\boldsymbol{z} \in \bar{\mathcal{B}}} \|\bar{\boldsymbol{y}} - \boldsymbol{R}_1 \boldsymbol{z}_{(1)} - \boldsymbol{R}_2 \boldsymbol{z}_{(2)}\|_2^2, \tag{3.25}$$

then we have

$$(\tilde{y}_m - \sum_{j=m}^{n} r_{mj} z_j)^2 < \beta^2 - T. \tag{3.26}$$

Now we show how to find a lower bound T. In Section 3.1, we introduced three lower bounds for the OBILS problems. Note that the ILS problem in (3.25) is not an overdetermined one and thus we can not use those lower bounds directly here. In Section 2.3.2, We showed how to transform a UBILS problem with a special box constraint to an OBILS problem. We can do the same thing here. But we need to assume the constrained box $\bar{\mathcal{B}}$ in (3.25) has the same form as that in Section 2.3.2, i.e., $\bar{\mathcal{B}} = \{\boldsymbol{z} \in \mathbb{Z}^n, \bar{\boldsymbol{l}}, \bar{\boldsymbol{u}} \in \mathbb{Z}^n, \bar{u}_j - \bar{l}_j = 2^{p_j} - 1, p_j \in \mathbb{Z}^+, \text{ for } 1 \le j \le n\}$.

Following Section 2.3.2, we define

$$\bar{\boldsymbol{R}}_2 = \left[\boldsymbol{r}_m^{(0)}, \quad \ldots, \quad \boldsymbol{r}_m^{(p_m-1)}, \quad \boldsymbol{r}_{m+1}^{(0)}, \quad \ldots, \quad \boldsymbol{r}_{m+1}^{(p_{m+1}-1)}, \quad \ldots, \quad \boldsymbol{r}_n^{(0)}, \quad \ldots, \quad \boldsymbol{r}_n^{(p_n-1)}\right],$$

$$\bar{\boldsymbol{z}}_{(2)} = \begin{bmatrix} \bar{z}_m^{(0)} \\ \vdots \\ \bar{z}_m^{(p_m-1)} \\ \bar{z}_{m+1}^{(0)} \\ \vdots \\ \bar{z}_{m+1}^{(p_{m+1}-1)} \\ \vdots \\ \bar{z}_n^{(0)} \\ \vdots \\ \bar{z}_n^{(p_n-1)} \end{bmatrix} \in \{-1, 1\}^q, \tag{3.27}$$

where $\boldsymbol{r}_j^{(i)} = 2^{i-1}\boldsymbol{R}_{1:m-1,j}$ and $q = \sum_{j=m}^{n} p_j$. Then the UBILS problem in (3.25) can be transformed to an equivalent problem

$$\min_{\boldsymbol{z}_{(1)}\in\bar{\mathcal{B}}_{1:m-1}, \bar{\boldsymbol{z}}_{(2)}\in\{-1,1\}^q} \left\| (\bar{\boldsymbol{y}} - \boldsymbol{R}_2\boldsymbol{c}) - \begin{bmatrix} \boldsymbol{R}_1, & \bar{\boldsymbol{R}}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_{(1)} \\ \bar{\boldsymbol{z}}_{(2)} \end{bmatrix} \right\|_2^2 \tag{3.28}$$

where $c_j = \bar{l}_{j+m-1} + \frac{1}{2}(2^{p_{j+m-1}} - 1)$ for $j = 1, 2, \dots, n-m+1$. Then by adding a constant term $\alpha^2 \|\bar{\boldsymbol{z}}_{(2)}\|_2^2 = \alpha^2 q$ to (3.28) we can transform it to the equivalent overdetermined ILS problem:

$$\min_{\bar{\boldsymbol{z}}\in\mathcal{Z}} \|\breve{\boldsymbol{y}} - \bar{\boldsymbol{R}}\bar{\boldsymbol{z}}\|_2^2, \tag{3.29}$$

where

$$\begin{aligned}
\bar{\boldsymbol{R}} &= \begin{bmatrix} \boldsymbol{R}_1 & \bar{\boldsymbol{R}}_2 \\ \boldsymbol{0} & \alpha\boldsymbol{I} \end{bmatrix} \in \mathbb{R}^{(m+q-1)\times(m+q-1)} \\
\bar{\boldsymbol{z}} &= \begin{bmatrix} \boldsymbol{z}_{(1)} \\ \bar{\boldsymbol{z}}_{(2)} \end{bmatrix} \in \mathbb{R}^{(m+q-1)} \\
\breve{\boldsymbol{y}} &= \begin{bmatrix} \bar{\boldsymbol{y}} - \boldsymbol{R}_2\boldsymbol{c} \\ \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{(m+q-1)} \\
\mathcal{Z} &= \left\{ \begin{bmatrix} \boldsymbol{z}_{(1)} \\ \bar{\boldsymbol{z}}_{(2)} \end{bmatrix} : \boldsymbol{z}_{(1)} \in \mathcal{B}_{1:m-1}, \bar{\boldsymbol{z}}_{(2)} \in \{-1,1\}^q \right\}.
\end{aligned} \tag{3.30}$$

Next, we can find a lower bound $\bar{T}$ for (3.29) by a method introduced in Section 3.1. A lower bound for the UBILS problem in (3.25) can then be defined as $T = \bar{T} - \alpha^2 q$.

Lower Bound for the Overdetermined Part

After $z_n, z_{n-1} \dots, z_m$ are fixed, the problem (3.19) becomes an OBILS problem

$$\min_{\boldsymbol{z} \in \mathcal{B}} \|\hat{\boldsymbol{y}} - \boldsymbol{R}_{1:m-1,1:m-1}\boldsymbol{z}_{1:m-1}\|_2^2, \tag{3.31}$$

where $\hat{\boldsymbol{y}} = \tilde{\boldsymbol{y}}_{1:m-1} - \boldsymbol{R}_{1:m-1,m:n}\boldsymbol{z}_{m:n}$. We can use lower bounds introduced in Section 3.1 to get lower bounds for the remaining levels $m-1, m-2, \dots, 1$.

3.2.2 Incorporating Lower Bounds in the PR Algorithm

In Section 2.3.2, we introduced the PR algorithm proposed in [19] and we can also incorporate lower bounds in it. In the PR algorithm, we first transform the original UBILS problem to an OBILS problem and then use CH algorithm to solve the OBILS problem, see Section 2.2. Therefore, the way to incorporate lower bounds in the PR algorithm is the same as the way to incorporate lower bounds in CH for OBILS problems, which can be found in Section 3.1.

CHAPTER 4
A New Search Algorithm for UBILS Problems

In Chapter 2, we presented the DTS algorithm and the PR algorithm for solving the UBILS problem

$$\min_{\boldsymbol{x}\in\mathcal{B}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2, \tag{4.1}$$

where $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ $(m < n)$ with $\operatorname{rank}(\boldsymbol{A}) = m$, and $\mathcal{B} = \{\boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}, \boldsymbol{l}, \boldsymbol{u} \in \mathbb{Z}^n\}$. (For the PR algorithm, it is assumed that $u_i - l_i = 2^{p_i} - 1, p_i \in \mathbb{Z}^+$ for $i = 1 : n$.) In this chapter, we propose a new search algorithm, which is a modification of the DTS search algorithm. Numerical test results will be given in Chapter 5.

4.1 Reordering Strategy for the Search Process

Recall in the DTS algorithm, we first compute the QRP decomposition:

$$\boldsymbol{A}\boldsymbol{P} = \boldsymbol{Q}\boldsymbol{R}, \tag{4.2}$$

where $\boldsymbol{P}$ is a permutation matrix, $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ is an orthogonal matrix and $\boldsymbol{R} \in \mathbb{R}^{m\times n}$ is an upper trapezoidal matrix. Then we have

$$\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 = \|\boldsymbol{Q}_1^T\boldsymbol{y} - \boldsymbol{R}\boldsymbol{P}^T\boldsymbol{x}\|_2^2 + \|\boldsymbol{Q}_2^T\boldsymbol{y}\|_2^2. \tag{4.3}$$

With $\tilde{\boldsymbol{y}} = \boldsymbol{Q}_1^T\boldsymbol{y}$, $\boldsymbol{z} = \boldsymbol{P}^T\boldsymbol{x}$, $\bar{\boldsymbol{l}} = \boldsymbol{P}^T\boldsymbol{l}$, and $\bar{\boldsymbol{u}} = \boldsymbol{P}^T\boldsymbol{u}$, the original UBILS problem (4.1) can be transformed to

$$\min_{\boldsymbol{x}\in\bar{\mathcal{B}}} \|\tilde{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2 \tag{4.4}$$

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

An example of the search tree for the DTS algorithm is depicted in Figure 4–1 (where there are three candidate values $1, 2, 3$ for z_n). Note that the root node (the white one) does not correspond to any search operation and is an artificial node. In this example, the search tree has 3 different branches at level n and we use $z_n = 1$ (red branch), $z_n = 2$ (yellow branch), $z_n = 3$ (blue branch) to denote them respectively.

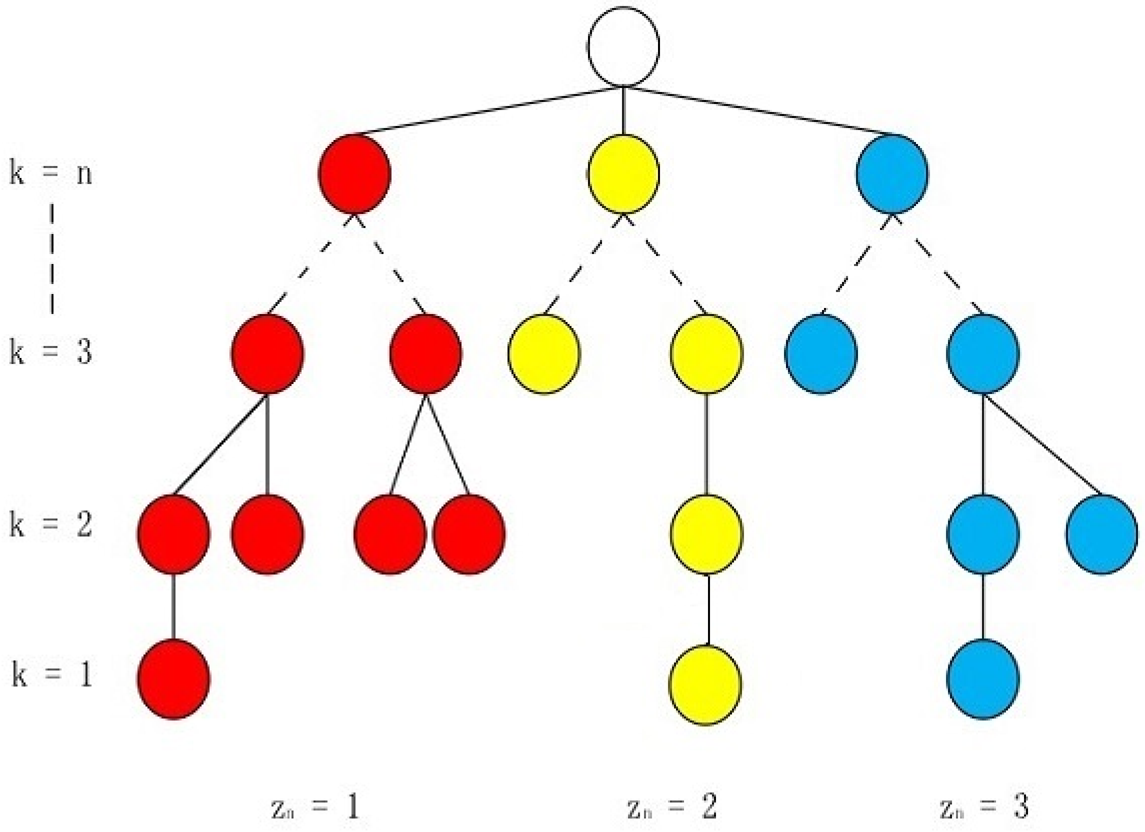

Figure 4–1: An example of search tree

Suppose we can find an initial upper bound β^2 for (4.4):

$$\|\tilde{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}\|_2^2 < \beta^2. \tag{4.5}$$

Partition $\boldsymbol{R}$ and $\boldsymbol{z}$ as

$$\boldsymbol{z} = \begin{array}{c} m-1 \\ l \end{array} \begin{bmatrix} \boldsymbol{z}_{(1)} \\ \boldsymbol{z}_{(2)} \end{bmatrix}, \quad \boldsymbol{R} = \begin{array}{c|cc} & m-1 & l \\ m-1 & \boldsymbol{R}_1 & \boldsymbol{R}_2 \\ 1 & & \boldsymbol{r}_m^T \end{array} \tag{4.6}$$

where $l = n - m + 1$, then we can derive the following set of inequalities:

$$(\tilde{y}_m - \sum_{j=m}^{n} r_{mj} z_j)^2 < \beta^2 \tag{4.7}$$

$$\|\tilde{\boldsymbol{y}}_{1:m-1} - \boldsymbol{R}_1\boldsymbol{z}_{(1)} - \boldsymbol{R}_2\boldsymbol{z}_{(2)}\|_2^2 < \beta^2 - (\tilde{y}_m - \sum_{j=m}^{n} r_{mj} z_j)^2. \tag{4.8}$$

In the search process of the DTS algorithm (see Section 2.3.1), we first use (4.7) to determine $\boldsymbol{z}_{m:n}$ and then use (4.8) to determine $\boldsymbol{z}_{1:m-1}$. Note that the transformation (2.55) was used to transform each z_i, $m \leq i \leq n$ to $\bar{z}_i$ and it can also be used to transform $\bar{\boldsymbol{z}}_{m:n}$ back to $\boldsymbol{z}_{m:n}$ at level m. The search process starts with the branch $\bar{z}_n = z_n^{(1)} \triangleq \left\lfloor \frac{\alpha}{|r_{mn}|} \right\rceil|_{\bar{\mathcal{Z}}_n}$. After all the tree nodes at this branch have been enumerated, the search process moves to another branch. In the DTS algorithm, it chooses $\bar{z}_n = z_n^{(2)}$ as the next branch to search, where $z_n^{(2)} \in \bar{\mathcal{Z}}_n$ is the next nearest integer to $\frac{\alpha}{|r_{mn}|}$ and so on. However, our simulations indicated that $\frac{\alpha}{|r_{mn}|}$ is usually larger than $\bar{u}_n$(the upper box bound of z_n), which means that $z_n^{(1)}$ is often equal to $\bar{u}_n$. And thus the search process often starts with branch $\bar{z}_n = \bar{u}_n$ and enumerates all the branches by the order of $\bar{z}_n = \bar{u}_n, \bar{z}_n = \bar{u}_n - 1, \ldots, \bar{z}_n = \bar{l}_n$. We do not have a rigorous explanation for this observation yet and it will be investigated in the future.

This search order does not seem to be an efficient one. Here we give an explanation. If the optimal solution or nearly optimal solution of (4.4) are in some branches with small values for $\bar{z}_n$, then the search process will come to these branches at a late stage, making the search process inefficient. We would like to enter these branches earlier. In the following we first propose a new search ordering strategy at level n and then give a new search algorithm based on it.

4.2 A New Search Ordering Strategy at Level n

Suppose that there are q elements in the set $\bar{\mathcal{Z}}_n$ at level n, and we use $a_1, a_2, \ldots, a_q$ to denote these integers from the smallest to the largest, i.e., $\bar{\mathcal{Z}}_n = \{a_1, \ldots, a_q\}$ and $a_1 < a_2 < \ldots < a_q$. Then there are q branches at level n in the search tree and we use $\bar{z}_n = a_1, \ldots, \bar{z}_n = a_q$ to represent these branches respectively. We also call the branch with $\bar{z}_n = a_i$ the i-th branch. Now the question comes to what criterion we should use to choose the visiting order for the q branches. We would like to find a local upper bound for the minimal value of the corresponding objective function in each branch and then use these upper bounds to help us choose the visiting order. If the upper bounds are tight enough, we can assume that the smaller the upper bound a branch has, the higher the possibility that the optimal solution exists in this branch. In the following, we will make use of (4.8) and (4.7) to derive an upper bound for the minimal value of the objective function in (4.4) in each branch.

First, suppose we can find the first valid subvector $\boldsymbol{z}_{m:n}^{(i)}$ for $\boldsymbol{z}_{m:n}$ in branch i by the DTS search process (see Section 2.3.1) for each i. Then we have q candidates for

$\boldsymbol{z}_{m:n}$: $\boldsymbol{z}_{m:n}^{(1)}, \boldsymbol{z}_{m:n}^{(2)}, \ldots, \boldsymbol{z}_{m:n}^{(q)}$. After $\boldsymbol{z}_{m:n}$ is fixed in branch i, the UBILS problem (4.4) becomes

$$\min_{\boldsymbol{z} \in \bar{\mathcal{B}}} \|\bar{\boldsymbol{y}}^{(i)} - \boldsymbol{R}_{1:m-1,1:m-1} \boldsymbol{z}_{1:m-1}\|_2^2, \tag{4.9}$$

where $\bar{\boldsymbol{y}}^{(i)} = \tilde{\boldsymbol{y}}_{1:m-1} - \boldsymbol{R}_{1:m-1,m:n} \boldsymbol{z}_{m:n}^{(i)}$. Now we try to find an upper bound on the minimal value of the objective function in (4.9).

We use $\boldsymbol{z}_B^{(i)} \in \mathbb{Z}^{m-1}$ to denote the box-constrained Babai integer point of (4.9) in branch i, the definition of which can be found in Section 2.2.1. Then $\|\bar{\boldsymbol{y}}^{(i)} - \boldsymbol{R}_{1:m-1,1:m-1} \boldsymbol{z}_B^{(i)}\|_2^2$ is an upper bound on the minimal value of the objective function in (4.9) for branch i. Then we can define an upper bound on the minimal value of the objective function in (4.4) as follows:

$$\eta^{(i)} = (\tilde{y}_m - \sum_{j=m}^{n} r_{mj} z_j^{(i)})^2 + \|\bar{\boldsymbol{y}}^{(i)} - \boldsymbol{R}_{1:m-1,1:m-1} \boldsymbol{z}_B^{(i)}\|_2^2. \tag{4.10}$$

Now we sort the q upper bounds $\eta^{(1)}, \ldots, \eta^{(q)}$ in nondecreasing order and then visit these branches one by one, from the branch with the smallest upper bound to the one with the largest upper bound. We will give details about this in the next subsection.

Note that it is possible that for some branch i we may not find a valid vector $\boldsymbol{z}_{m:n}^{(i)}$ for $\boldsymbol{z}_{m:n}$. It means that the solution for (4.4) must not be in this branch ($z_n = a_{ni}$) and we can set the upper bound $\eta^{(i)}$ of this branch to be ∞. Later, in the new search process, we can directly skip those branches with ∞ as upper bounds.

The new search ordering strategy at level n is inspired by the idea of best first search (BFS) algorithm. We do not apply the same idea to other levels, as this may make the search process less efficient because that there would be too many branches.

Algorithm 4.1 FIND SEARCH ORDER at LEVEL n

Input: The upper trapezoidal matrix $\boldsymbol{R} \in \mathbb{R}^{m\times n}$, the vector $\tilde{\boldsymbol{y}} \in \mathbb{R}^m$, the lower bound vector $\bar{\boldsymbol{l}} \in \mathbb{Z}^n$, the upper bound vector $\bar{\boldsymbol{u}} \in \mathbb{Z}^n$, and the initial hyper-ellipsoid bound β.

Output: Z,$\boldsymbol{s}$ and $\mathcal{U}$ are three ordered sets which store the information of $\bar{\boldsymbol{z}}_{m:n}^{(i)}$, $\eta^{(i)}$ and $\{\bar{\mathcal{Z}}_m, \bar{\mathcal{Z}}_{m+1}, \dots, \bar{\mathcal{Z}}_n\}$ in each branch, respectively.

function: $[\boldsymbol{s},\ Z,\ \mathcal{U}]$ = NEW-ORDER($\boldsymbol{R}, \tilde{\boldsymbol{y}}, \bar{\boldsymbol{l}}, \bar{\boldsymbol{u}}, \beta$)

```
Step 1 :(Initialization)
  Set $i = 0$, $\boldsymbol{s} = \{\}$, $Z = \{\}$, $\mathcal{U} = \{\}$, $k = n$
Step 2 :
  **if** $k = n$ **then**
      **if** $i > \bar{u}_n - \bar{l}_n$ **then** Go to step 6
      **end if**
      Set $\bar{z}_k = i$, $\eta^{(i)} = \infty$, $k = k - 1$ //$\bar{\boldsymbol{z}}_{m:n}$ is transformed from $\boldsymbol{z}_{m:n}$ by (2.55)
  **end if**
  Compute $\bar{\lambda}_k$ and $\bar{\mu}_k$ by following (2.60) and (2.61)
  Set $\bar{\mathcal{Z}}_k = \{\bar{l}_k, \bar{l}_k + 1, \dots, \bar{u}_k\} \cap (\bar{\lambda}_k, \bar{\mu}_k)$
  **if** $\bar{\mathcal{Z}}_k$ is not empty **then**
      Compute $c_k = \frac{\alpha - \sum_{j=k+1}^{n} |r_{mj}\bar{z}_j|}{|r_{mk}|}$ and $\bar{z}_k = \lfloor c_k \rceil|_{\bar{\mathcal{Z}}_k}$, $\alpha$ is defined in (2.57)
  **else**
      Go to step 4
  **end if**
Step 3 :
  **if** $k = m - 1$ **then**
      Use the transformation (2.55) to transform $\bar{\boldsymbol{z}}_{m:n}$ back to $\boldsymbol{z}_{m:n}$
      Compute $\bar{\boldsymbol{y}} = \tilde{\boldsymbol{y}}_{1:m-1} - \boldsymbol{R}_{1:m-1,m:n}\boldsymbol{z}_{m:n}$
      Compute $\eta^{(i)}$ by following (4.10)
      Go to step 4
  **else**
      $k = k - 1$ and go to step 2
  **end if**
```

```
Step 4 :
  if $k = n - 1$ or $k = m - 1$ then
      if $k = n - 1$ then
          Update $\boldsymbol{s} = \{\boldsymbol{s}, \eta^{(i)}\}$, $Z = \{Z, []\}$, $\mathcal{U} = \{\mathcal{U}, []\}$ //No valid $z_{m:n}$ in this branch
      else
          Set $\bar{\mathcal{U}} = \{\}$
          for $k = m : n$ do Update $\bar{\mathcal{U}} = \{\bar{\mathcal{U}}, \bar{\mathcal{Z}}_k\}$
          end for      //$\bar{\mathcal{U}} = \{\bar{\mathcal{Z}}_m, \bar{\mathcal{Z}}_{m+1}, \dots, \bar{\mathcal{Z}}_n\}$
          Update $\boldsymbol{s} = \{\boldsymbol{s}, \eta^{(i)}\}$, $Z = \{Z, \boldsymbol{z}_{m:n}\}$, $\mathcal{U} = \{\mathcal{U}, \bar{\mathcal{U}}\}$ //Store information
            in the current branch, which will be used in Algorithm 4.2 later
      end if
      Set $i = i + 1$, $k = n$ and go to step 2
  else
      Set $k = k + 1$
  end if
Step 5 :
  Choose $\bar{z}_k \in \bar{\mathcal{Z}}_k$ to be next valid integer to $c_k$
  if $\bar{z}_k$ does not exist then
      Go to step 4
  else
      Go to step 2
  end if
Step 6 :
  Sort $\boldsymbol{s}$ such that $s(1) \leq s(2) \leq \ldots \leq s(\bar{u}_n - \bar{l}_n + 1)$. Then reorder $Z$ and $\mathcal{U}$ corre-
  spondingly
```

We have shown how to find the new search order at level n. The corresponding algorithm is given in Algorithm 4.1.

4.3 The New Search Algorithm

Let $\eta^{(i_1)} \leq \eta^{(i_2)} \leq \ldots \leq \eta^{(i_q)}$. Suppose we found an initial upper bound β^2 for (4.4) by the method we introduced in Section 2.3.1. Since $\eta^{(i_1)}$ is an upper bound of branch i_1, it must be an upper bound of the global optimal solution of (4.4). Thus, if the current β^2 is larger than $\eta^{(i_1)}$, we update β^2 to be $\eta^{(i_1)}$ when we start the search process.

In the following, we give a description of the new search process. After the search order at level n is determined, we start the search process in branch i_1. Note that the first valid value for $\boldsymbol{z}_{m:n}$ in branch i_1 (i.e., $\boldsymbol{z}_{m:n}^{(i_1)}$) has been found thus our search algorithm can immediately start at level $m-1$. The search process between level $m-1$ and level 1 is the same as the DTS algorithm. Each time we find a valid full integer vector $\boldsymbol{z}^*$ at level 1, we update β to be $\|\tilde{\boldsymbol{y}} - \boldsymbol{R}\boldsymbol{z}^*\|_2$. After the search process finishes enumerations in branch i_i, it chooses branch i_2 to continue search in the same way as it does in branch i_1. We repeat such procedure until all the branches whose upper bounds have finite values at level n are enumerated. The latest found integer point $\tilde{\boldsymbol{z}}$ is the optimal solution of (4.4).

The details of the search process is given in Algorithm 4.2.

4.4 Incorporating Lower Bounds in the New Search Algorithm

To make the new search process more efficient, lower bounds can also be incorporated.

Algorithm 4.2 NEW SEARCH

Input: The nonsingular upper trapezoidal matrix matrix $\boldsymbol{R} \in \mathbb{R}^{n\times n}$, the vector $\tilde{\boldsymbol{y}} \in \mathbb{R}^n$, the lower bound vector $\bar{\boldsymbol{l}} \in \mathbb{Z}^n$, the upper bound vector $\bar{\boldsymbol{u}} \in \mathbb{Z}^n$, and the initial hyper-ellipsoid bound β.

Output: The solution $\hat{\boldsymbol{z}} \in \mathbb{Z}^n$ to the UBILS problem (2.46).

function: $\hat{\boldsymbol{z}}$ = NEW-SEARCH($\boldsymbol{R}, \tilde{\boldsymbol{y}}, \bar{\boldsymbol{l}}, \bar{\boldsymbol{u}}, \beta$)

```
Step 1 :
  [$\boldsymbol{s}, Z, \mathcal{U}$] = NEW-ORDER($\boldsymbol{R}, \tilde{\boldsymbol{y}}, \bar{\boldsymbol{l}}, \bar{\boldsymbol{u}}, \beta$), see Algorithm 4.1
  Set $i = 0$
Step 2 :
  if $i > \bar{u}_n - \bar{l}_n$ or $s(i) = \infty$ then
      Terminate
  end if
  Set $k = m - 1$, $\boldsymbol{z}_{m:n} = Z(i+1)$, $\bar{\mathcal{U}} = \mathcal{U}(i+1)$
  for $k = m : n$ do $\bar{\mathcal{Z}}_k = \bar{\mathcal{U}}(k)$
  end for
Step 3 :
  if $k = m - 1$ then
      Use (2.55) to transform $\bar{\boldsymbol{z}}_{m:n}$ back to $\boldsymbol{z}_{m:n}$
      Compute $\bar{\boldsymbol{y}} = \boldsymbol{y}_{1:m-1} - \boldsymbol{R}_{1:m-1,m:n}\boldsymbol{z}_{m:n}$, and $T_m = (\tilde{y}_m - \boldsymbol{R}_{m,m:n}\boldsymbol{z}_{m:n})^2$
      Compute $\boldsymbol{z}_{1:m-1}$ = CH-SEARCH($\boldsymbol{R}_{1:m-1,1:m-1}, \bar{\boldsymbol{y}}, \boldsymbol{l}_{1:m-1}, \boldsymbol{u}_{1:m-1}, \sqrt{\beta^2 - T_m}$), see
        Algorithm 2.3
      Set $\hat{\boldsymbol{z}} = \boldsymbol{z}$, $\beta = \sqrt{(\bar{\boldsymbol{y}} - \boldsymbol{R}_{1:m-1,1:m-1}\boldsymbol{z}_{1:m-1})^2 + T_m}$
      Set $k = k + 1$ and go to step 6
  else
      Set $k = k - 1$
  end if
Step 4 :
  if $k > m - 1$ then
      Compute $\bar{\lambda}_k$ and $\bar{\mu}_k$ by following (2.60) and (2.61)
      Set $\bar{\mathcal{Z}}_k = \{\bar{l}_k, \bar{l}_k + 1, \ldots, \bar{u}_k\} \cap (\bar{\lambda}_k, \bar{\mu}_k)$
      if $\bar{\mathcal{Z}}_k$ is not empty then
          Compute $c_k = \frac{\alpha - \sum_{j=k+1}^{n} |r_{mj}\bar{z}_j|}{|r_{mk}|}$
          Compute $\bar{z}_k = \lfloor c_k \rceil|_{\bar{\mathcal{Z}}_k}$ and go to step 3 //$\bar{\boldsymbol{z}}_{m:n}$ is transformed from $\boldsymbol{z}_{m:n}$ by
          (2.55)
      end if
  end if
```

```
Step 5 :
  if $k = n - 1$ then
      Set $i = i + 1$ and go to step 2
  else
      Set $k = k + 1$
  end if
Step 6 :
  Choose $\bar{z}_k \in \bar{\mathcal{Z}}_k$ to be the next nearest integer to $c_k$
  if $\bar{z}_k$ does not exist then
      Go to step 5
  else
      Go to step 3
  end if
```

4.4.1 Lower Bounds in the Order Choosing Stage

To speed up the new search process above, two lower bounds can be incorporated.

From (4.5), the optimal solution satisfies

$$\|\tilde{\boldsymbol{y}}_{1:m-1} - \boldsymbol{R}_1 \boldsymbol{z}_{1:m-1} - \boldsymbol{R}_2 \boldsymbol{z}_{m:n}\|_2^2 + (\tilde{y}_m - \sum_{j=m}^{n} r_{mj} z_j)^2 < \beta^2. \tag{4.11}$$

First, we use the method introduced in Section 3.2.1 to find a lower bound T_1 for the first term of the left hand side of (4.11). As before, we need to assume the box constraint in (4.4) satisfies: $\bar{\mathcal{B}} = \{\boldsymbol{z} \in \mathbb{Z}^n, \bar{\boldsymbol{l}}, \bar{\boldsymbol{u}} \in \mathbb{Z}^n, \bar{u}_j - \bar{l}_j = 2^{p_j - 1}, p_j \in \mathbb{Z}^+ \text{ for } j = m:n\}$. Then we have

$$(\tilde{y}_m - \sum_{j=m}^{n} r_{mj} z_j)^2 < \beta^2 - T_1, \tag{4.12}$$

which can be used to replace (4.7) as a tighter upper bound to find $\boldsymbol{z}_{m:n}$. Note that the lower bound T_1 is independent of any value of $\boldsymbol{z}_{m:n}$ we choose in the search process and it needs to be computed only once. The search method of finding $\boldsymbol{z}_{m:n}^{(i)}$ is the same as before, see Section 2.3.1.

Second, suppose $z_{m:n}$ has been fixed and we denote its value by $z_{m:n}^{(i)}$. From (4.11), which holds for the optimal solution of the UBILS problem, we have

$$\min_{\boldsymbol{z}_1 \in \bar{\mathcal{B}}_{1:m-1}} \|(\tilde{\boldsymbol{y}}_{1:m-1} - \boldsymbol{R}_2 \boldsymbol{z}_{m:n}) - \boldsymbol{R}_1 \boldsymbol{z}_1\|_2^2 + (\tilde{y}_m - \sum_{j=m}^{n} r_{mj} z_j)^2 < \beta^2. \tag{4.13}$$

Note that the first term of the left hand side of (4.13) is an OBILS problem and a lower bound $T0_2$ on its objective function can be computed by using the method introduced in Section 3.1. If the current $\boldsymbol{z}_{m:n}^{(i)}$ is valid, it must satisfy

$$T_2 + (\tilde{y}_m - \sum_{j=m}^{n} r_{mj} z_j^{(i)})^2 < \beta^2. \tag{4.14}$$

Otherwise, $\boldsymbol{z}_{m:n}^{(i)}$ is invalid, which means it can not be part of an integer point $\boldsymbol{z}$ satisfying (4.11). In this case, we need to continue the search process in branch i to update $\boldsymbol{z}_{m:n}^{(i)}$.

4.4.2 Lower Bounds for the Overdetermined Part

After fixing $z_n, z_{n-1} \ldots, z_m$, the problem (4.4) can be transformed to the OBILS problem (4.9). We can use lower bounds introduced in Section 3.1 to get lower bounds for the remaining levels $m-1, m-2, \ldots, 1$.

CHAPTER 5
Numerical Experiments

In this chapter we use numerical experiments to show the effectiveness of the lower bounds and the new search algorithm for solving the UBILS problem (2.45). Algorithms to be tested in this chapter are implemented in MATLAB 2013a. All tests were run on a PC with Intel Core 2 Quad CPU and 3.8 GiB memory running on Ubuntu 14.04 LTS, 64 bits OS.

5.1 System Model and Algorithm Notations

In the numerical experiments, we construct the data based on the following linear model

$$\boldsymbol{y} = \boldsymbol{A}\hat{\boldsymbol{x}} + \boldsymbol{v}, \tag{5.1}$$

where $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}(m < n)$ with $\text{rank}(\boldsymbol{A}) = m$, $\hat{\boldsymbol{x}} \in \mathcal{B} = \{\boldsymbol{x} \in \mathbb{Z}^n : \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u}, \boldsymbol{l}, \boldsymbol{u} \in \mathbb{Z}^n\}$, and $\boldsymbol{v} \in \mathbb{R}^m$ is a Guassian noise vector following $\mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$.

In the simulations, we generate three different types of cases for the model (5.1). Here we introduce some MATLAB built-in functions which will be used later: $\text{randn}(p,q)$ generates a p-by-q matrix of normally distributed random numbers; $\text{randi}([i_{min}, i_{max}], p, q)$ generates a p-by-q matrix of pseudo random integers drawn from the discrete uniform distribution on the interval $[i_{min}, i_{max}]$; $\text{ones}(p,q)$ generates a p-by-q matrix of ones. For both Cases 1 and 2, we use $\sigma * \text{randn}(m,1)$ to generate an m dimensional noise vector $\boldsymbol{v}$, use $\text{ones}(n,1) * l$ and $\text{ones}(n,1) * u$ to generate constrained box $[\boldsymbol{l},\ \boldsymbol{u}]$, where $l, u \in \mathbb{Z}$ and use randi$([l\ \ u], n, 1)$ to generate the true parametric vector $\hat{\boldsymbol{x}}$. Note that each element of $\hat{\boldsymbol{x}}$ has the same range (this is the case in communications). Then, $\boldsymbol{y}$ can be generated according to (5.1) after $\boldsymbol{A}$ is generated. We generate three different types of the matrix $\boldsymbol{A}$:

Case 1 (random case): We use randn(m, n) to generate the matrix $\boldsymbol{A}$. For the m and n we chose in our tests, we noticed that the condition numbers of $\boldsymbol{A}$ are usually below 100.

Case 2 (ill-conditioned case): We use randn(m, n) to generate an $m \times n$ matrix $\bar{\boldsymbol{A}}$ and then use the built-in function svd$(\bar{\boldsymbol{A}})$ to get the singular value decomposition $\bar{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$. Then we replace $\boldsymbol{D}$ by a new diagonal matrix $\bar{\boldsymbol{D}} \in \mathbb{R}^{m\times n}$ with $\bar{d}_{11} = 1$, $\bar{d}_{ii} = d_{ii}$ for $i = 2, 3, \ldots, n-1$, $\bar{d}_{mm} = 0.001$. Then we can form a new matrix $\boldsymbol{A} = \boldsymbol{U}\tilde{\boldsymbol{D}}\boldsymbol{V}^T$. The condition numbers of $\boldsymbol{A}$ are at least 1000.

Case 3 (flat-fading in communications): This case is similar to the case tested in [15] and [19]. In the Gaussian MIMO linear flat-fading channel system, the relation between received signal vectors and transmit signal vectors of this system can be written as a complex linear system

$$\boldsymbol{y}_c = \boldsymbol{A}_c\boldsymbol{x}_c + \boldsymbol{v}_c \tag{5.2}$$

where $\boldsymbol{A}_c \in \mathbb{C}^{N_r\times N_t}$ represents the channel matrix with N_t transmitter antennas and N_r receiver antennas. The elements of $\boldsymbol{A}_c$ are complex i.i.d Gaussian variables with distribution $\mathcal{CN}(\boldsymbol{0}, \boldsymbol{I})$ and $\boldsymbol{v}_c \in \mathbb{C}^{N_r}$ is the white Gaussian noise vector with distribution $\mathcal{CN}(\boldsymbol{0}, 2\sigma^2\boldsymbol{I})$. The elements of the unknown vector $\boldsymbol{x}_c$ are odd numbers in set $\mathcal{X}_c(p) = \{p_1 + p_2 j : p_1, p_2 = \pm1, \pm3, \ldots, \pm(2^p - 3), \pm(2^p - 1)\}$ where $j^2 = -1$, $p = 1, 2, 3$

correspond to the 4QAM, 16QAM, 64QAM constellations, respectively. To deal with the complex case, we first transform (5.2) to the following real linear model

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{v} \tag{5.3}$$

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_c^R \\ \boldsymbol{y}_c^I \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_c^R & -\boldsymbol{A}_c^I \\ \boldsymbol{A}_c^I & \boldsymbol{A}_c^R \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_c^R \\ \boldsymbol{x}_c^I \end{bmatrix}, \quad \boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}_c^R \\ \boldsymbol{v}_c^I \end{bmatrix} \tag{5.4}$$

where $\boldsymbol{A}_c = \boldsymbol{A}_c^R + j\boldsymbol{A}_c^I$, $\boldsymbol{y}_c = \boldsymbol{y}_c^R + j\boldsymbol{y}_c^I$, $\boldsymbol{x}_c = \boldsymbol{x}_c^R + j\boldsymbol{x}_c^I$ and $\boldsymbol{v}_c = \boldsymbol{v}_c^R + j\boldsymbol{v}_c^I$. Thus $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with $m = 2N_r$, $n = 2N_t$ and $a_{ij} \sim \mathcal{N}(\boldsymbol{0}, 1/2)$, $\boldsymbol{v} \sim \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I}_m)$, and $\boldsymbol{x} \in \mathcal{X}(p)^n = \mathcal{X}(p) \times \mathcal{X}(p) \dots \mathcal{X}(p)$ with

$$\mathcal{X}(p) = \{\pm 1, \pm 3, \dots, \pm(2^p - 3), \pm(2^p - 1)\}. \tag{5.5}$$

And thus estimating $\boldsymbol{x}_c \in \mathbb{C}^{N_t}$ in (5.2) is equivalent to estimating $\boldsymbol{x} \in \mathbb{R}^n$ in (5.3).

In this case, we can construct the matrix $\boldsymbol{A}$ by setting $\boldsymbol{A}_c^R = \frac{1}{\sqrt{2}}\text{randn}(N_r, N_t)$, $\boldsymbol{A}_c^I = \frac{1}{\sqrt{2}}\text{randn}(N_r, N_t)$. Then we show how to construct $\boldsymbol{x}$. First we generate each element of vector $\bar{\boldsymbol{x}}$ as $\bar{x}_i = 2 * \text{randi}([1 \;\; 2^{p-1}]) - 1$. Then x_i can be generated as $x_i = (3 - 2^{\text{randi}([1,2])}) * \bar{x}_i$ for $i = 1, 2, \dots, n$. The method to generate $\boldsymbol{y}$, $\boldsymbol{v}$ is the same as what we introduced in Cases 1 and 2. To estimate $\boldsymbol{x}$ in (5.3), we solve the following constrained ILS problem:

$$\min_{\boldsymbol{x} \in \mathcal{X}(p)^n} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \tag{5.6}$$

Note that the above problem is not a standard UBILS problem since the constraint on $\boldsymbol{x}$ is not a box. However, we can transform it to a standard problem. For each x_i

$(i = 1 : n)$, we define the following transformation:

$$\bar{x}_i = \frac{2^p - 1 + x_i}{2}, \tag{5.7}$$

so that $\bar{x}_i \in \bar{\mathcal{X}}(p) = \{0, 1, \ldots, 2^p - 1\}$. Then the problem (5.6) becomes a standard UBILS problem:

$$\min_{\bar{\boldsymbol{x}} \in \bar{\mathcal{X}}(p)^n} \|\bar{\boldsymbol{y}} - \bar{\boldsymbol{A}}\bar{\boldsymbol{x}}\|_2, \tag{5.8}$$

where $\bar{\boldsymbol{y}} = \boldsymbol{y} - (2^p - 1)\boldsymbol{A} * \boldsymbol{1}$, $\bar{\boldsymbol{A}} = 2\boldsymbol{A}$.

Here we first give the formulation to compute the signal-to-noise-rations (SNR),

$$\text{SNR} = 10\log_{10}(((M - 1)/3)/2\sigma^2), \tag{5.9}$$

which will be used later for M-QAM.

In the simulations, we compare the existing DTS algorithm and the PR algorithm with four new solvers for the above three cases. For convenience, we use the following abbreviations for different solvers:

DTS: The direct tree search algorithm without lower bounds (Algorithm 2.5)

DTSLB: The direct tree search algorithm with lower bounds (see Section 2.3.2)

PR: The partial regularization algorithm without lower bounds (see Section 3.2.1)

PRLB: The partial regularization algorithm with lower bounds (see Section 3.2.2)

NS: The new search algorithm without lower bounds (Algorithm 4.2)

NSLB: The new search algorithm with lower bounds (see Section 4.4)

According to [19], the regularization parameter α in the PR and the PRLB was set to $2^{7/4}\sigma$.

In Section 3.1, we reviewed three different lower bounds for the OBILS problems, each of which can be used as a lower bound for the over-determined part of the UBILS problem, see Section 3.2.1. The lower bounds we used in our tests for the over-determined parts in the DTSLB and the NSLB are the component-wise lower bounds, see Section 3.1. It is because that the cost of the component-wise lower bound is smaller than the basis-reduced lower bound and it is usually tighter than the norm-wise lower bound.

We compare the above six different solvers in terms of average and median running time and performance profiles for different scenarios.

5.2 Simulation Results

5.2.1 Simulation Result for Case 1

We first consider Case 1 in various scenarios.

Figures 5–1 and 5–2 display the average and median running time of six algorithms versus $n-m$ of 200 random instances in Case 1. Here $\sigma = 0.1$, $m = 15$ and $n = 16:20$. The time limit we set for each instance is 1000 seconds. If a solver can not solve the corresponding problem in the limited time, it will be terminated. For this case, NSLB is the most efficient one in terms of average running time and PR is the most efficient one in terms of median running time overall. In Figure 5–1, when $n-m = 3$ PR is a little faster than NSLB from the aspect of average running time. In Figure 5–2, when $n-m = 5$, the median running time of NSLB is faster than PR and NS is very close to PR.

Now we look at the effectiveness of lower bounds for DTS, PR and NS based on the above figures. NSLB outperforms NS and DTSLB outperforms DTS, while

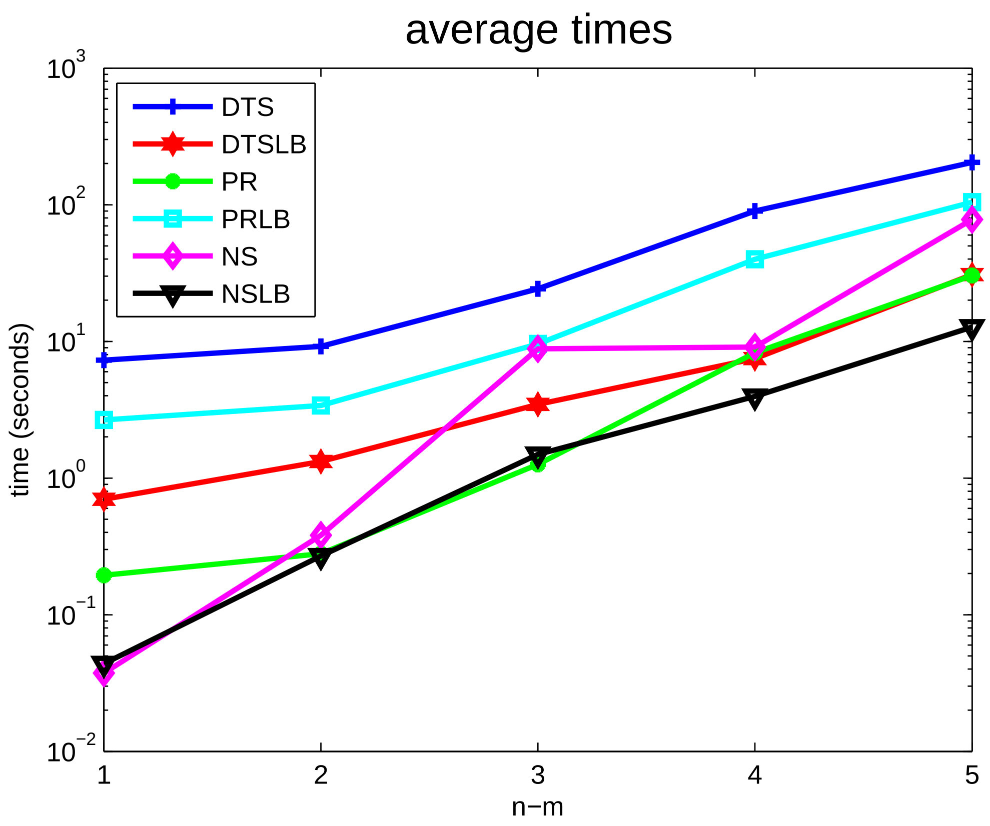

Figure 5–1: Case 1, $\sigma = 0.1$, $l = 0$, $u = 7$, $m = 15$, $n = 16, 17, 18, 19, 20$

PRLB performs worse than PR in both average and median running time. Then, we can conclude that lower bounds can improve the performances of NS and DTS but can not improve PR in Case 1.

To compare these six algorithms further, we use a different metric, performance profiles to evaluate them more fairly. Performance profiles provide an effective mean to compare solver performances for several solvers at once, eliminating some of the bias some comparisons have. It was first proposed by Dolan and More in [28], which is an extension of initial performance comparisons developed in [7] by Billups et al.

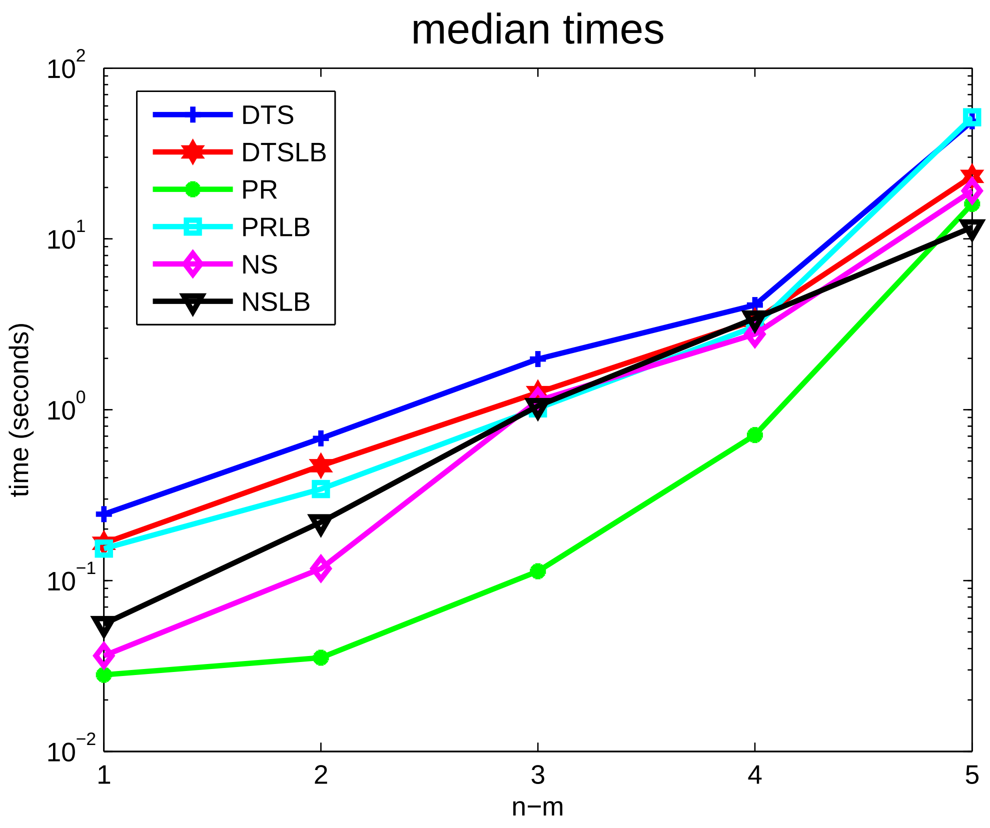

Figure 5–2: Case 1, $\sigma = 0.1$, $l = 0$, $u = 7$, $m = 15$, $n = 16, 17, 18, 19, 20$

In [28], they define the performance profile of a solver as a (cumulative) distribution function for a performance metric. Suppose we solved a set of n_p problems denoted by P with different solvers. Let p denote a particular problem and s denote a particular solver. In particular, they use the ratio of the solver resource time for a given solver versus the best time of all the solvers as the performance ratio, i.e.,

$$\rho(p, s) = \frac{t_{(p,s)}}{\min\{t_{(p,s)} : 1 \le s \le n_s\}}, \tag{5.10}$$

where $t_{(p,s)}$ refers to the time solver s spent on problem p, and n_s refers to the number of solvers in the model test set. For practical purposes, if a solver does not solve a problem, we set its performance ratio to ∞. In order to obtain an overall assessment of a solver on the given model test set, define a cumulative distribution function $P_s(\tau)$:

$$P_s(\tau) = \frac{1}{n_p}\text{size}\{p \text{ in P} : \rho(p,s) \leq \tau\}. \tag{5.11}$$

So $P_s(\tau)$ is the probability that a performance ratio $\rho(p,s)$ is within a factor of τ of the best possible ratio.

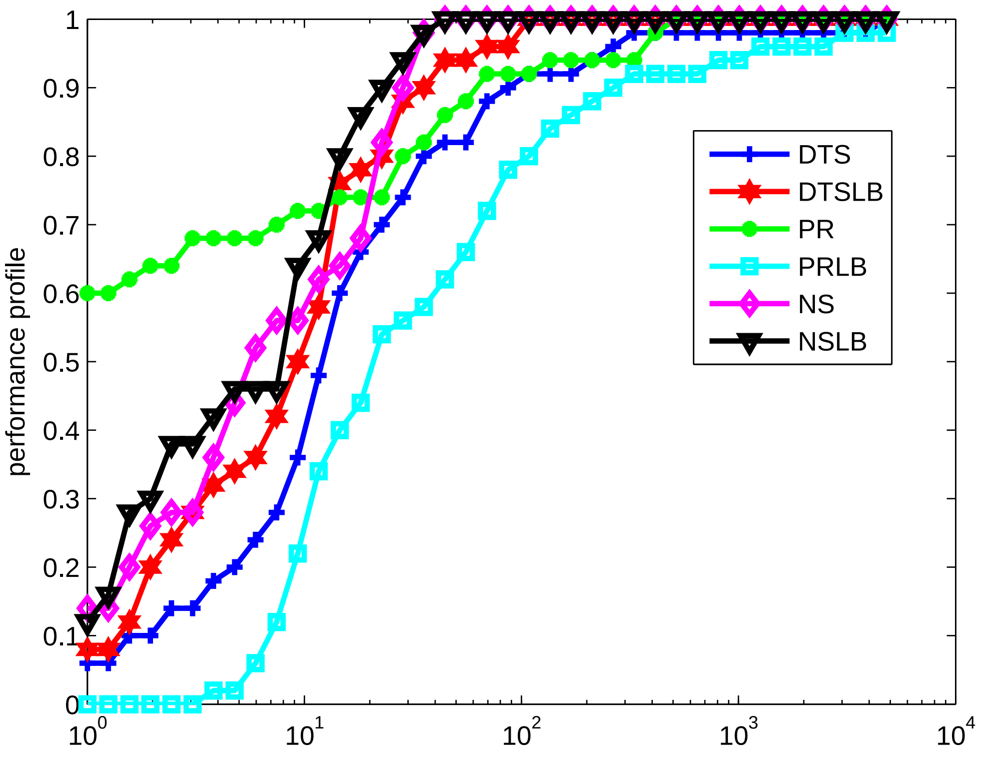

Figure 5–3: Case 1, $\sigma = 0.1$, $l = 0$, $u = 7$, $m = 15$, $n = 20$

	Min(s)	Average(s)	Median(s)	Max(s)
200 rand instances for Case 1				
DTS	0.0499	15.7095	1.4206	410.1331
DTSLB	0.0744	1.9432	1.0465	18.4673
PR	**0.0259**	17.2582	**0.1219**	500.0001
PRLB	0.1638	44.9894	1.7269	507.7474
NS	0.0390	3.0665	0.7086	44.7135
NSLB	0.0490	**0.9948**	0.8402	**6.8056**

Table 5–1: Case 1 with $m = 15$, $n = 20$

Figure 5–3 shows the performance profile of six algorithms for Case 1. In all of these 200 instances $\sigma = 0.1$, $l = 0$, $u = 7$, $m = 15$, $n = 20$ and the time limit we set is 1000 seconds. It shows that PR can first finish around 60% of all instances when τ in (5.11) around 1.25 but NSLB and NS can only finish around $12 - 13\%$ at the same time. The percentages of DTS, DTSLB and PRLB can finish when τ in (5.11) around 1.25 are all below 10%. There is a turning point for the performances of six solvers: the time to finish 72% of all instances. From this point, NSLB becomes the first solver which can finish 72% of all instances and keeps this advantage until the end (finish all instances). DTSLB, NS, DTS perform better than PR gradually after the turning point and finish all test instances earlier than PR.

From Figures 5–1 and 5–2, we find that there is a discrepancy between performances of average and median running time: NSLB and PR have best performances in terms of average and median running time, respectively. Figure 5–3 may help us to explain it. First, let us recall what the average and median running time mean. The average running time is the summation of all time divided by the number of instances and the median running time is the middle one of the sorted time set.

And thus an algorithm like PR, which can finish 60% instances when τ in (5.11) around 1.25 but need quite a long time to finish the remaining 30% instances easily has discrepancy performances in terms of average and median running time. That is because average time considers running time of all instances and median time only considers the smallest 51% running time of all instances.

Through Figures 5–1, 5–2 and Figure 5–3, we can find that PRLB can not outperform PR in terms of all three metrics. Here we give a rough explanation about why lower bounds can not improve the performance of PR. First, in Figure 5–3, we can see that PR can finish 60% instances when τ in (5.11) around 1.25 and thus adding lower bounds can easily bring overhead for these instances. Second, different from DTS and NS, PR solves UBILS problems by transforming UBILS problems into larger dimensions OBILS problems, which means matrices used for computing lower bounds in PRLB are larger than those in DTSLB and NSLB. Thus, lower bounds in PRLB more easily bring overhead than DTSLB and NSLB. Third, the information given in the above Figures is not complete. For example, we set a time limit of 1000 seconds in all three tests above. There is a possibility that PRLB can solve some problem in 1100 seconds and PR can not solve the problem in it. But they are all counted as 1000 seconds in our simulations.

Table 5–1 summarizes the minimum, average, median and maximum time of the same instances shown in Figure 5–3. From the table we see that PR has the smallest minimum and median running time, while NSLB has smallest average and maximum running time. We can also find that the difference between average, median and maximum running time of NSLB is much smaller than others.

5.2.2 Simulation Result for Case 2

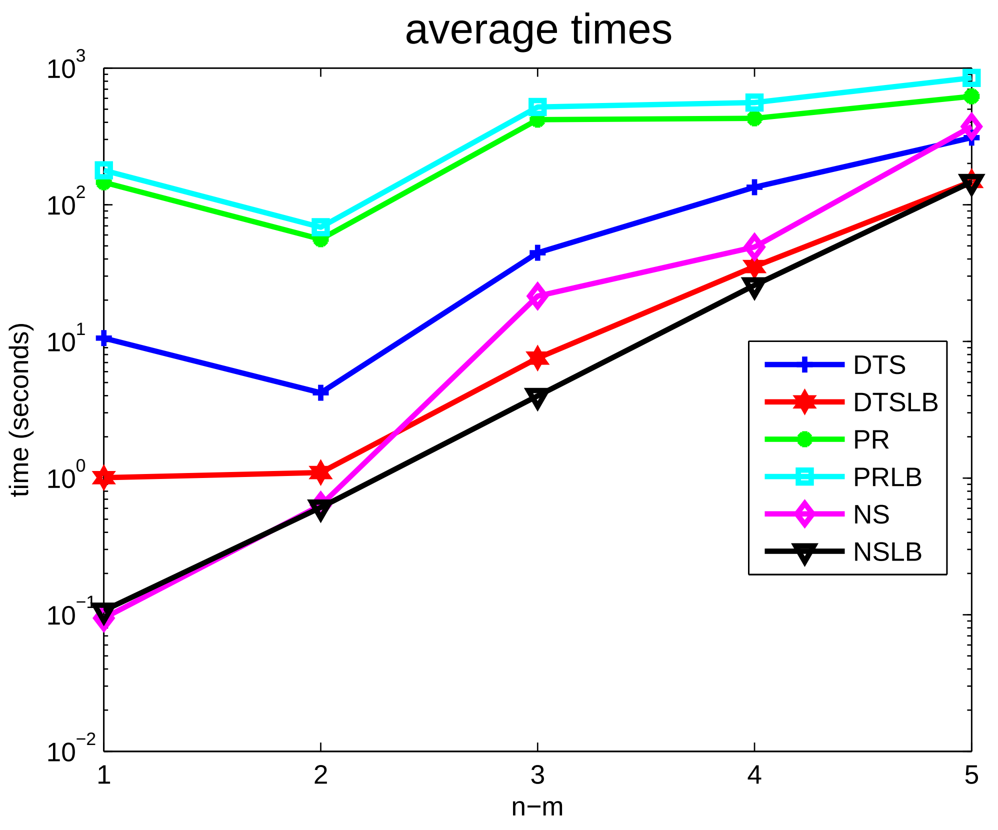

Figure 5–4: Case 2, $\sigma = 0.1$, $l = 0$, $u = 7$, $m = 15$, $n = 16, 17, 18, 19, 20$

Figures 5–4 and 5–5 display the average and median running time of six algorithms versus $n - m$ of 200 random instances in Case 2. Here $\sigma = 0.1$, $m = 15$ and $n = 16 : 20$. The time limit we set for each instance is 1000 seconds. In this case, NSLB is the most efficient one in terms of average running time. As to the median running time, NS and NSLB perform best alternatively. NSLB has the best performance when $n - m$ equals 3, 5 and NS has the best performance when $n - m$ equals

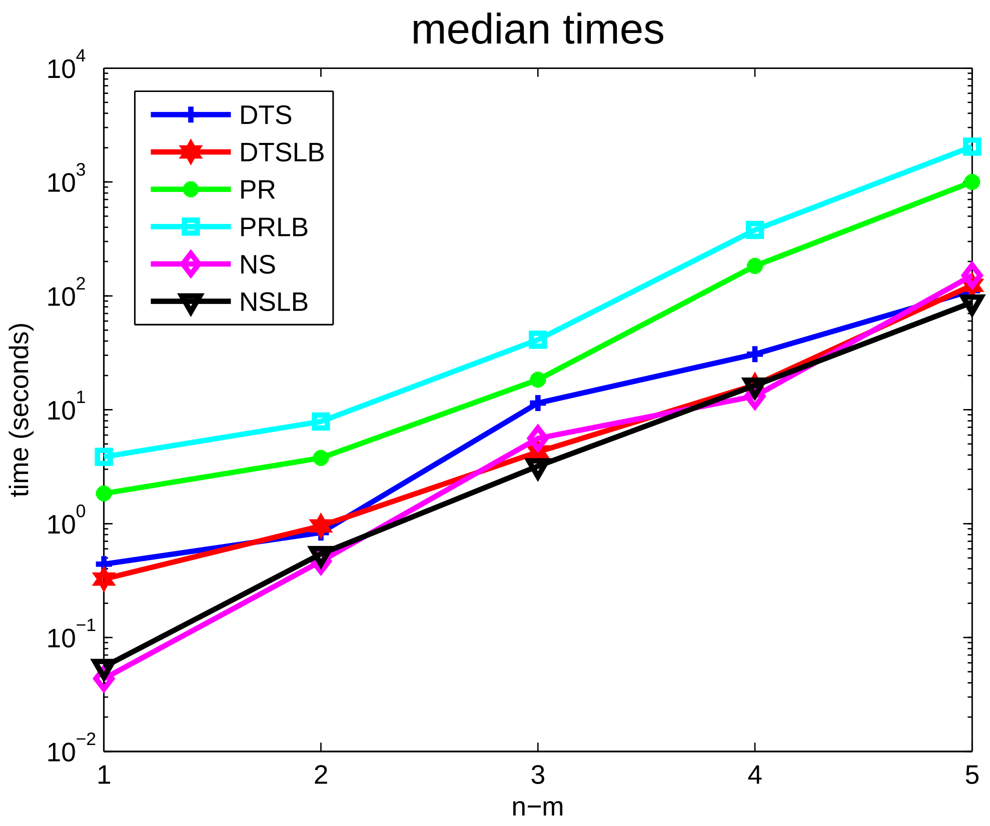

Figure 5–5: Case 2, $\sigma = 0.1$, $l = 0$, $u = 7$, $m = 15$, $n = 16, 17, 18, 19, 20$

1, 2, 4. But the performances of these two solvers are very close when $n - m$ equals 1, 2, 4.

For the effectiveness of lower bounds for DTS, PR and NS, NSLB can outperform NS and DTSLB can outperform DTS in most instances but PRLB performs worse than PR in both average and median running time. Then, we can conclude that lower bounds can improve the performances of NS and DTS algorithms but can not improve the PR algorithm for Case 2. It is consistent with what we found in Case 1.

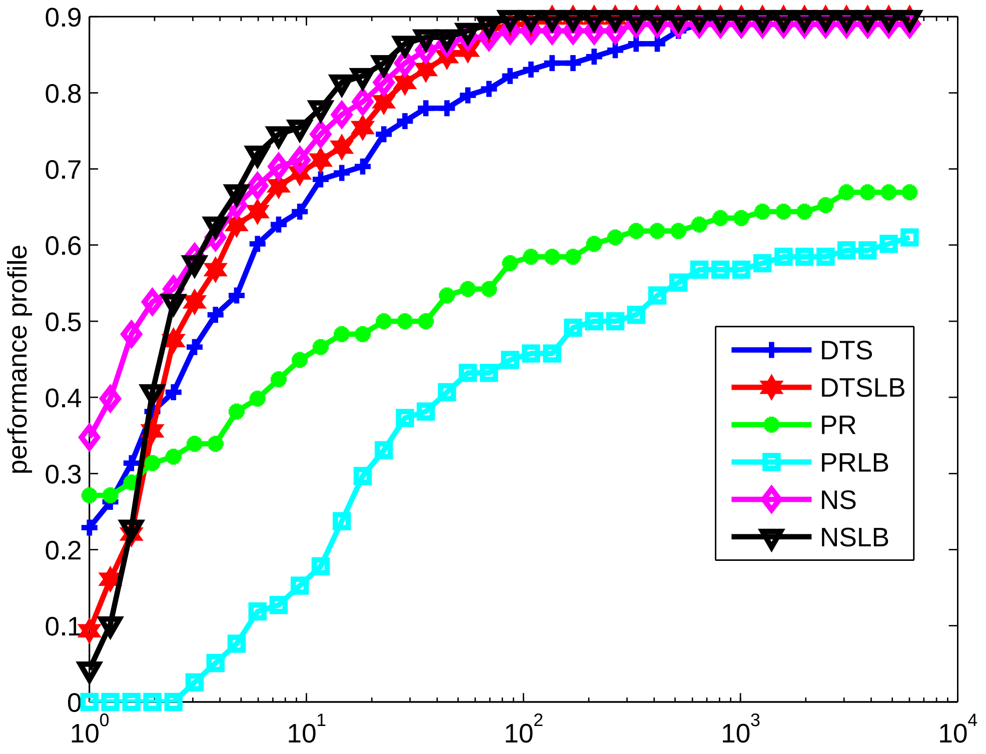

Figure 5–6: Case 2, $\sigma = 0.1$, $l = 0$, $u = 7$, $m = 15$, $n = 20$

Figure 5–6 shows the performance profile of six algorithms in Case 2. In all of these 200 instances $\sigma = 0.1$, $l = 0$, $u = 7$, $m = 15$, $n = 20$ and the time limit we set is 1000 seconds. It shows that NS, PR, DTS, DTSLB, NSLB, PRLB can finish around 40%, 27%, 22%, 10%, 5% and 1% of instances when τ in (5.11) around 1.25, respectively. NSLB and DTSLB can catch up with PR when solving around 31% of instances almost at the same time and can catch up with DTSLB when solving around 38% of instances. NSLB begins to perform better than NS when solving around 60% of instances and has the best performance over all solvers after

	Min(s)	Average(s)	Median(s)	Max(s)
200 rand instances for Case 2				
DTS	0.0386	198.3417	14.1826	1000.0000
DTSLB	0.0413	172.1622	**9.6048**	1000.0000
PR	**0.0309**	381.0334	60.6704	1000.0000
PRLB	0.0312	395.7107	69.7932	1000.2000
NS	0.0377	181.1204	10.6955	1000.0000
NSLB	0.0532	**170.6295**	10.4604	1000.0000

Table 5–2: Cases 2 with $m = 15$, $n = 20$

this. None of six solvers can finish all instances within the time limit. NSLB, NS, DTSLB, DTS can finish 90% of instances, while PR and PRLB can finish 65% and 60% of instances, respectively.

Table 5–2 summarizes the minimum, average, median and maximum time of the same instances shown in Figure 5–6.

For Case 2, PR still has the smallest minimum time and the minimum running time of all six algorithms are quite close. NSLB and DTSLB have the best average and median running time, respectively. Note that the median running times of DTSLB, NS, NSLB are quite close. As to the maximum time, all of six algorithms have some extreme instance which can not be solved within the time limit (1000 seconds) we set.

5.2.3 Simulation Result for Case 3

Figures 5–7 and 5–8 give the average time and median running time of six algorithms against $N_t - N_r$ for 4QAM ($p = 1$) with $N_r = 8$, $N_t = 9 : 12$ in all of 200 random instances. Each of instance has a time limit 2000 seconds. From the aspect of average running time: when $N_t - N_r \leq 2$, NSLB has the best performance; when

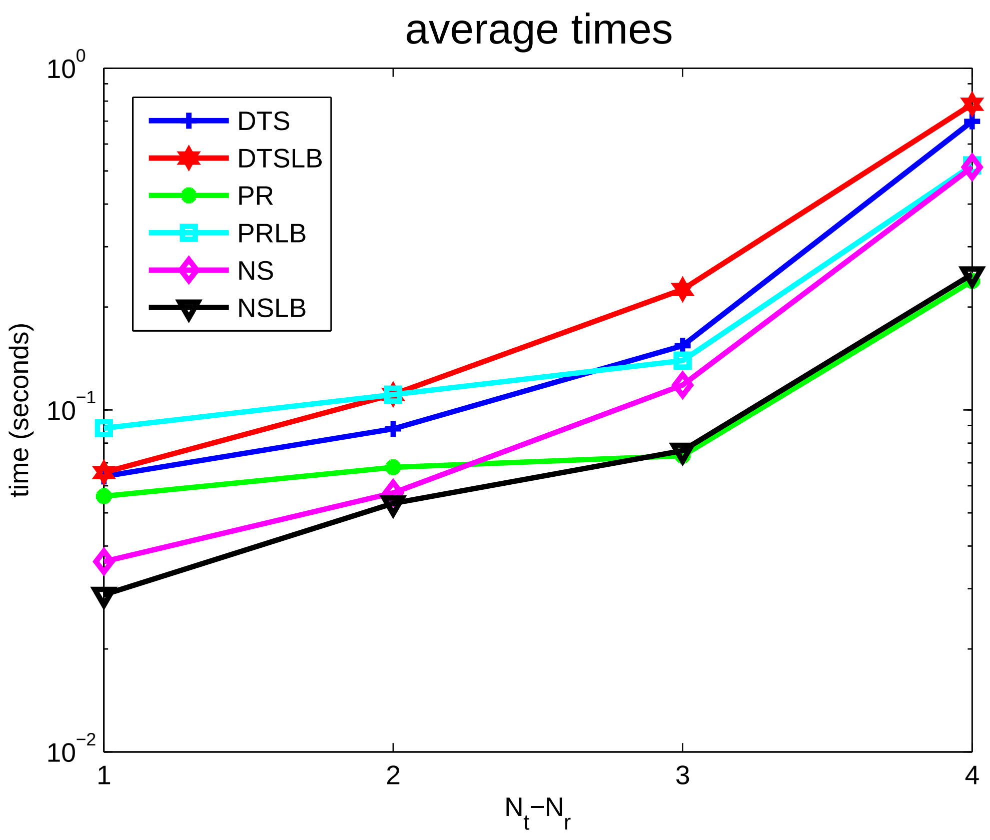

Figure 5–7: 4QAM, flat-fading system

$N_t - N_r = 3, 4$ the performances of PR and NSLB are almost the same and PR is little faster than NSLB. When it comes to median running time, PR always performs best.

Figure 5–9 shows the performance profile of six solvers for 4QAM flat-fading system. In all of these 200 instances SNR = 30, $N_r = 8$, $N_t = 12$ and the time limit we set is 2000 seconds. PR has the best performance in the whole process, which can finish around 80% instances when τ in (5.11) around 1.25. However, the other five solvers can only finish around (NS, DTS) or below (NSLB, DTSLB, PRLB) 10%

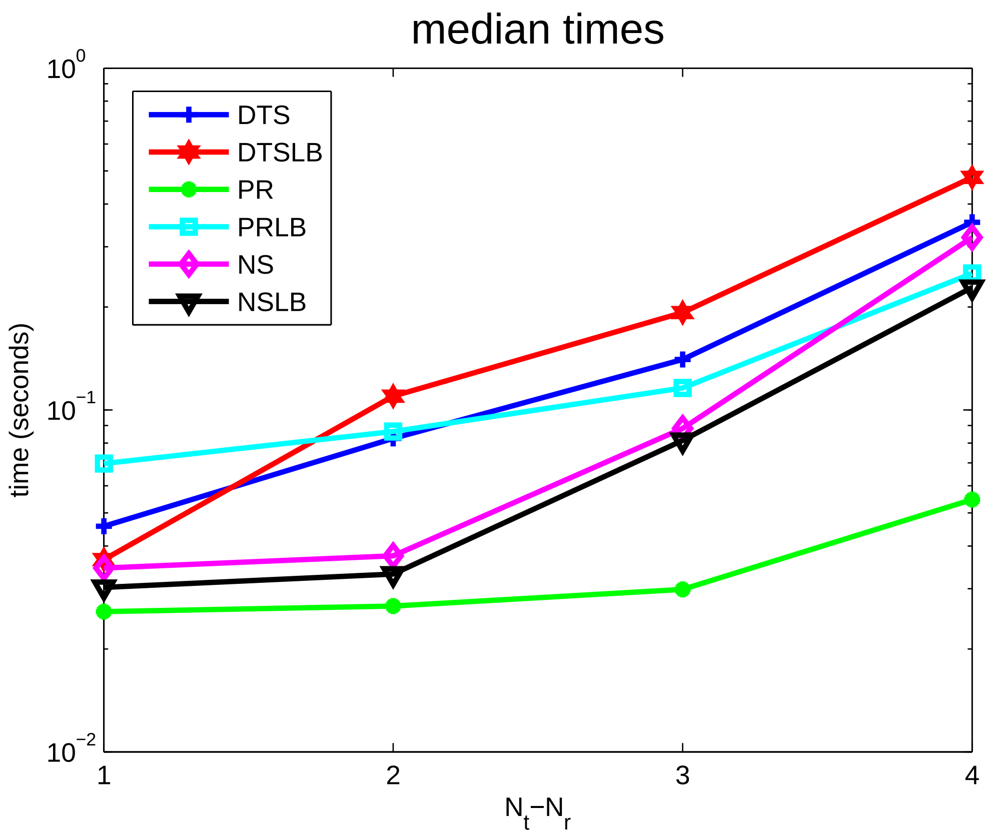

Figure 5–8: 4QAM, flat-fading system

at that time. NS has the second best performance for the first 75% instances and PRLB has the second best performance for instances between 76%-95%. For the rest part (the last 5%) instances, the performances of NS, NSLB, DTS, DTSLB, PRLB are very close.

Figures 5–10 and 5–11 give the average and median running time of six solvers against $N_t - N_r$ for 16QAM ($p = 2$) with $N_r = 8$, $N_t = 9 : 12$ in all of 200 random instances. Each of instance has a time limit of 2000 seconds. It is easy to see that

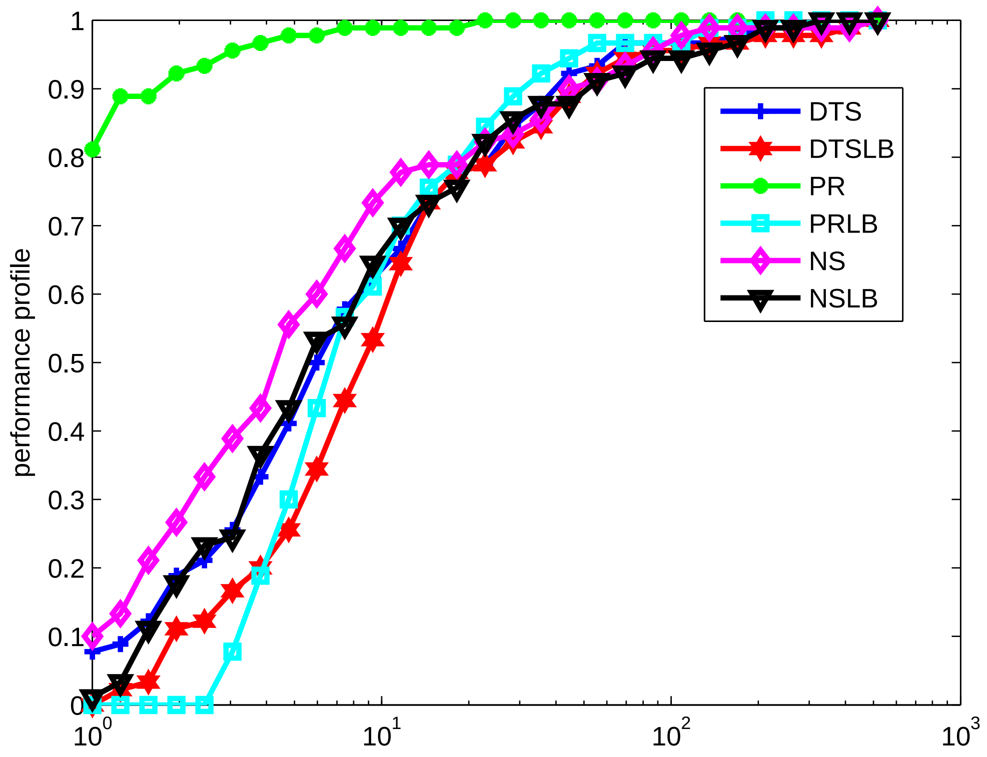

Figure 5–9: 4QAM, flat-fading system

NSLB and PR always have the best performance in terms of average and median running time, respectively.

Figure 5–12 shows the performance profile of six solvers for 16QAM flat-fading system. In all of these 200 instances SNR = 30, N_r = 8 and the time limit we set is 2000 seconds. When τ in (5.11) around 1.25, PR can finish around 65% of instances but NSLB, NS and DTSLB can only finish around 10% instances. The percentages of instances DTS and PRLB can finish when τ in (5.11) around 1.25 are both below 10%. In terms of finishing 85% of instances, the performances of NSLB and NS

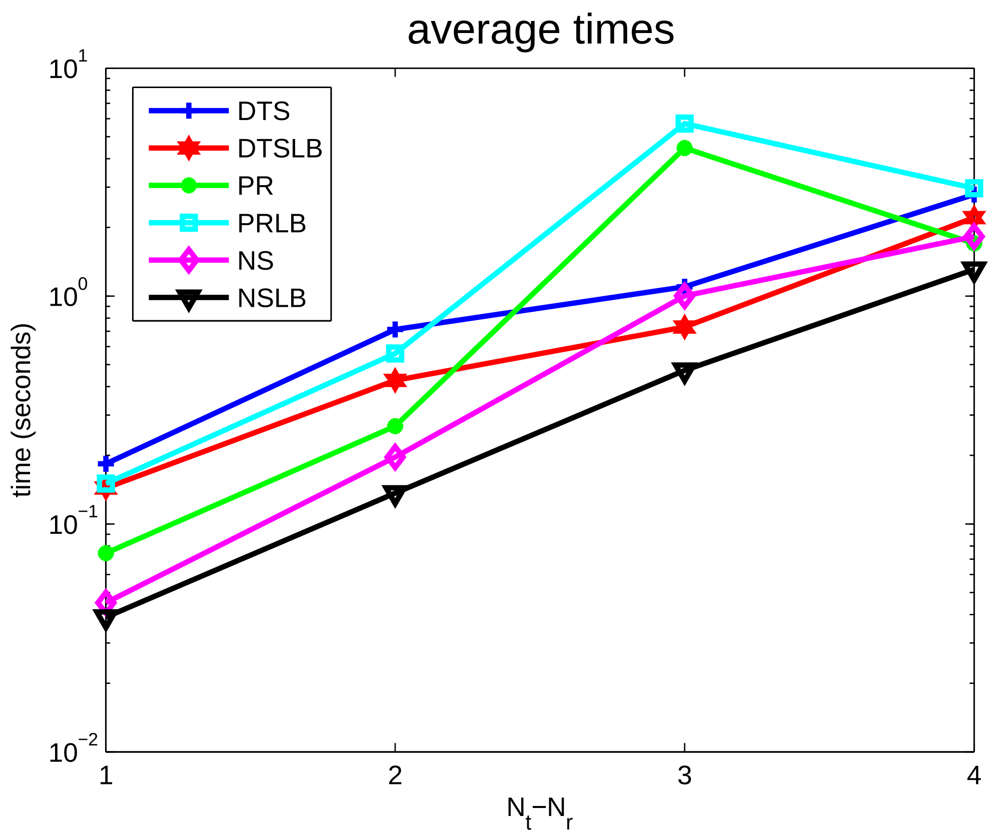

Figure 5–10: 16QAM, flat-fading system

can catch up with that of PR. NSLB and NS have best performances after that and NSLB can finish all instances first.

Figures 5–13 and 5–14 give the average time and median running time of six solvers against $N_t - N_r$ for 64QAM ($p = 3$) with $N_r = 8$, $N_t = 9:12$ in all of 200 random instances. Each of instance has a time limit of 2000 seconds. When $N_t - N_r \geq 2$, NSLB has the best performance in terms of average running time and NS has the best performance when $N_t - N_r = 1$. For the median running time, PR has the best performance when $N_t - N_r \geq 2$ and NSLB has the best performance when $N_t - N_r = 1$.

	Min(s)	Average(s)	Median(s)	Max(s)
200 rand instances with 4QAM				
DTS	0.0392	1.3946	0.3327	13.7556
DTSLB	0.0393	1.8795	0.4745	23.7022
PR	**0.0259**	**0.1422**	**0.0509**	**1.5403**
PRLB	0.0333	1.6176	0.2422	12.2163
NS	0.0360	1.5099	0.2434	22.4341
NSLB	0.0430	1.8128	0.3572	18.8811
100 rand instances with 16QAM				
DTS	0.0307	46.9653	3.4256	2000.0000
DTSLB	0.0694	28.8014	2.4976	451.9706
PR	**0.0276**	72.0403	**0.4116**	2000.0000
PRLB	0.0525	77.6904	1.4313	2000.0000
NS	0.0500	29.1598	1.8829	422.8588
NSLB	0.0690	**23.6563**	1.3815	**263.4167**
100 rand instances with 64QAM				
DTS	**0.0307**	46.9653	3.4256	2000.0000
DTSLB	0.0694	28.8014	2.4976	451.9706
PR	0.0334	101.9469	**0.8625**	2000.0000
PRLB	0.0492	120.9281	1.1281	2000.0000
NS	0.0500	29.1598	1.8829	422.8588
NSLB	0.0690	**20.7674**	1.3815	**248.5187**

Table 5–3: Case 3 with $N_t = 8$, $N_r = 12$, for $4, 16, 64$QAMs

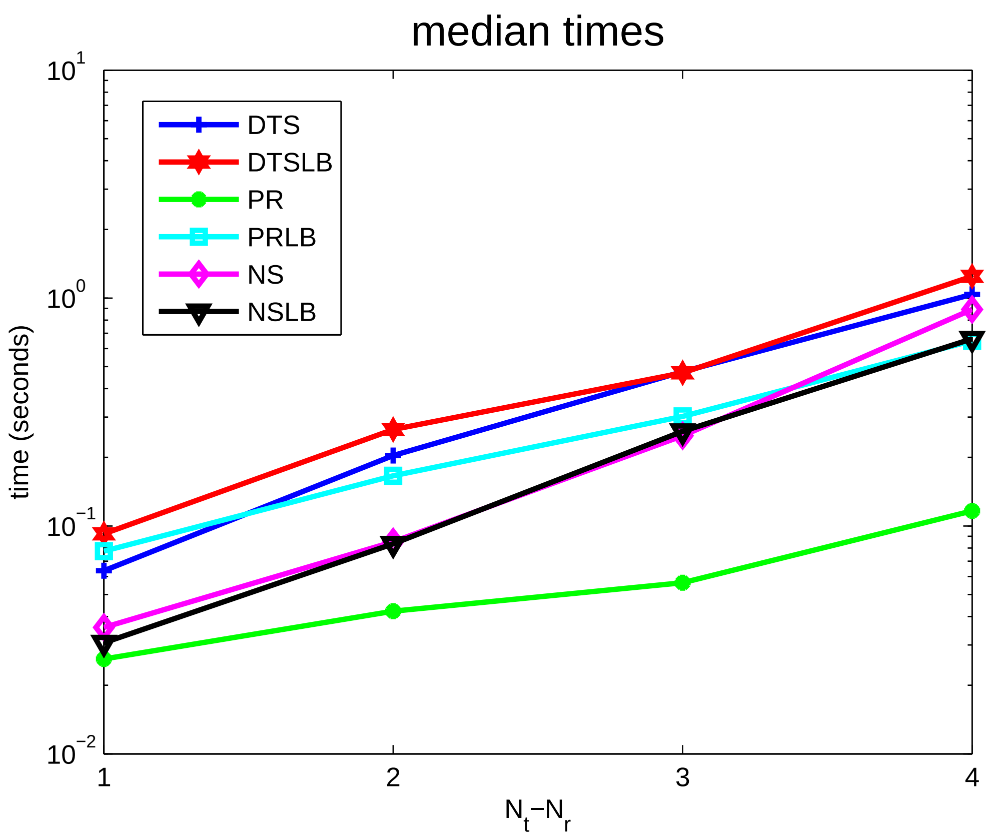

Figure 5–11: 16QAM, flat-fading system

Figure 5–15 shows the performance profile of six solvers for 64QAM flat-fading system. In all of these 200 instances SNR = 30, $N_r = 8$ and the time limit we set is 2000 seconds. When τ in (5.11) around 1.25, PR can finish around 67% of instances and NSLB, NS and DTSLB can only finish around 10% instances. The percentages of instances DTS and PRLB can finish at that time are below 10%. In terms of 90% of instances, the performances of NSLB and NS catch up with that of PR. NSLB and NS have best performances after that and NSLB can finish all instances first.

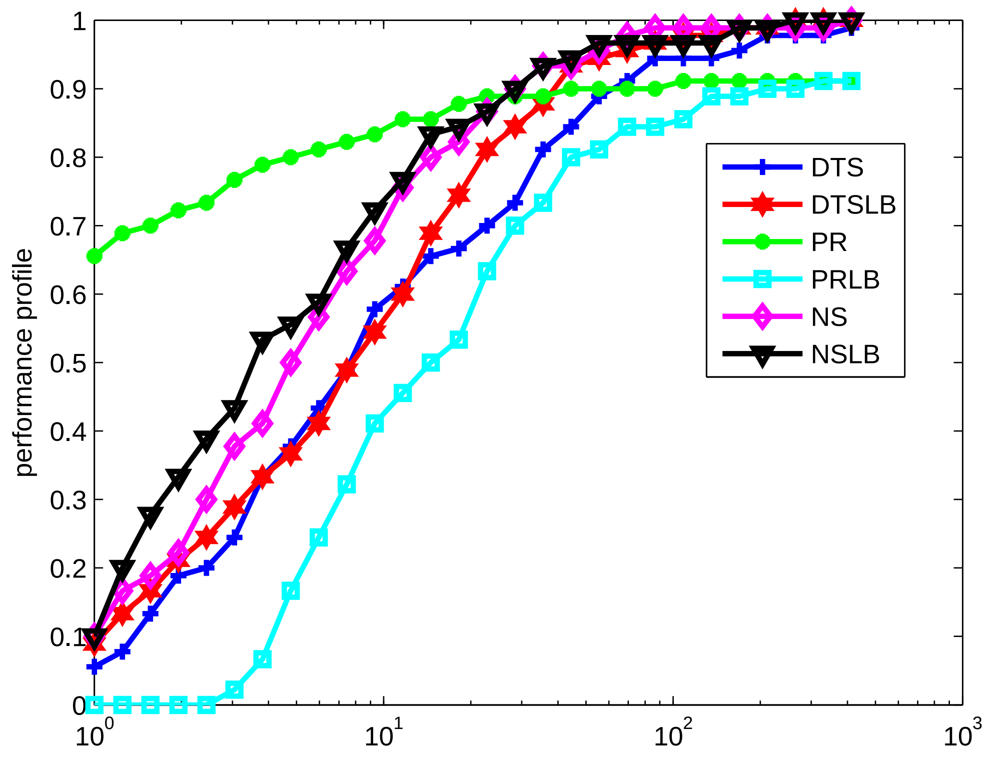

Figure 5–12: 16QAM, flat-fading system

Table 5–3 summarizes the minimum, average, median and maximum time of the same instances shown in Figures. 5–9, 5–12 and 5–15. We can see that PR has best values in terms of all four aspects for 4QAM instances. It also has best minimum and median values for 16QAM instances. NSLB has best average and maximum values for 16QAM instances. For 64QAM instances, DTS has the best minimum value, PR has the best median value and NSLB has best average and maximum values.

Figures 5–16 and 5–17 show the average and median running time of the six solvers against different SNRs for 16QAM with $N_r = 8$, $N_t = 10$ and SNR = 9 : 3 :

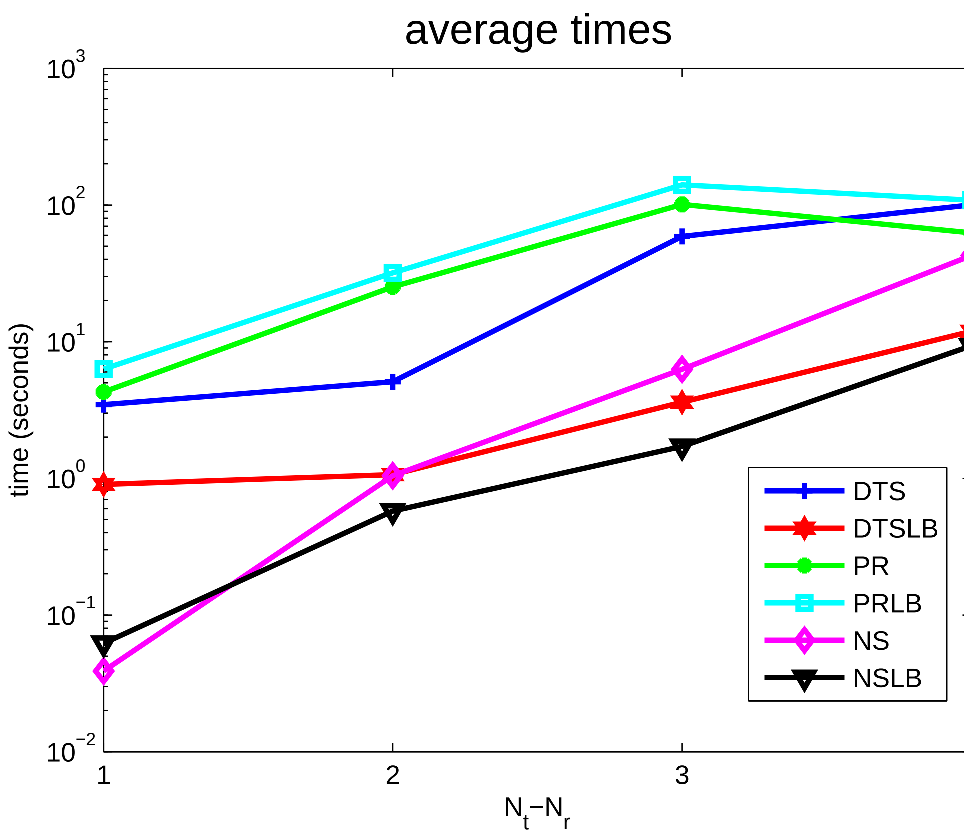

Figure 5–13: 64QAM, flat-fading system

30. The time limit for each instance is 2000 seconds. NSLB always has the best performance in terms of average running time and PR is the most efficient one from the aspect of median running time. Also we can find that with SNR decreasing, the performance of NSLB is more close to PR.

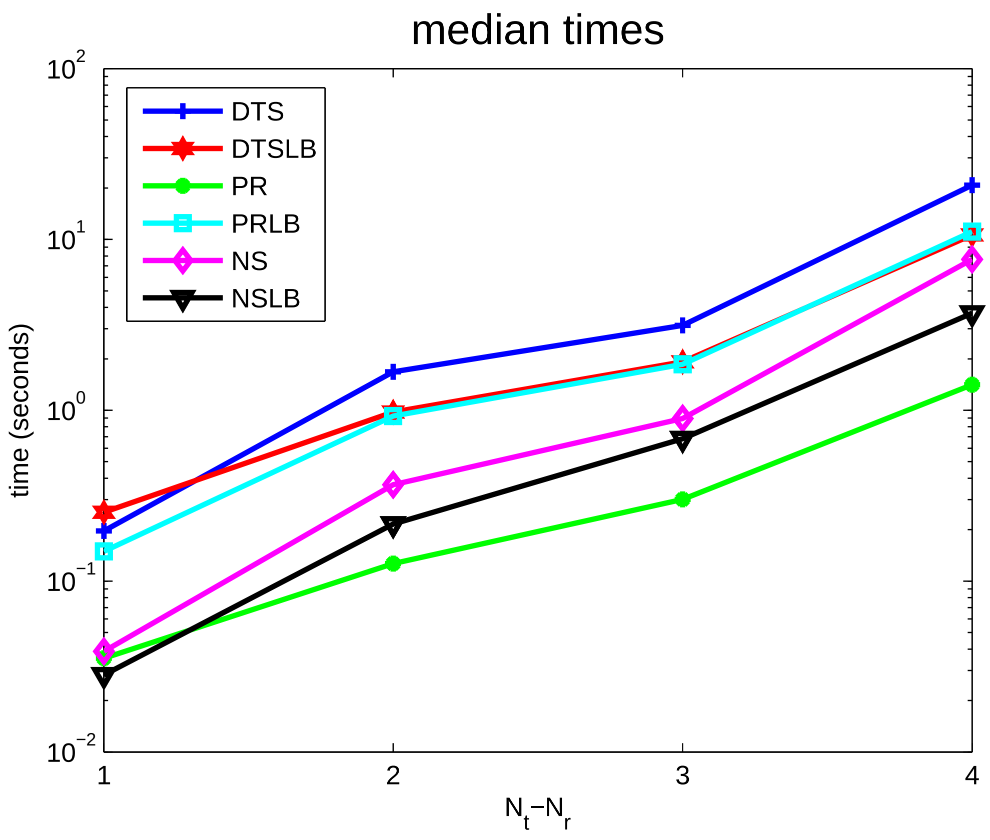

Figure 5–14: 64QAM, flat-fading system

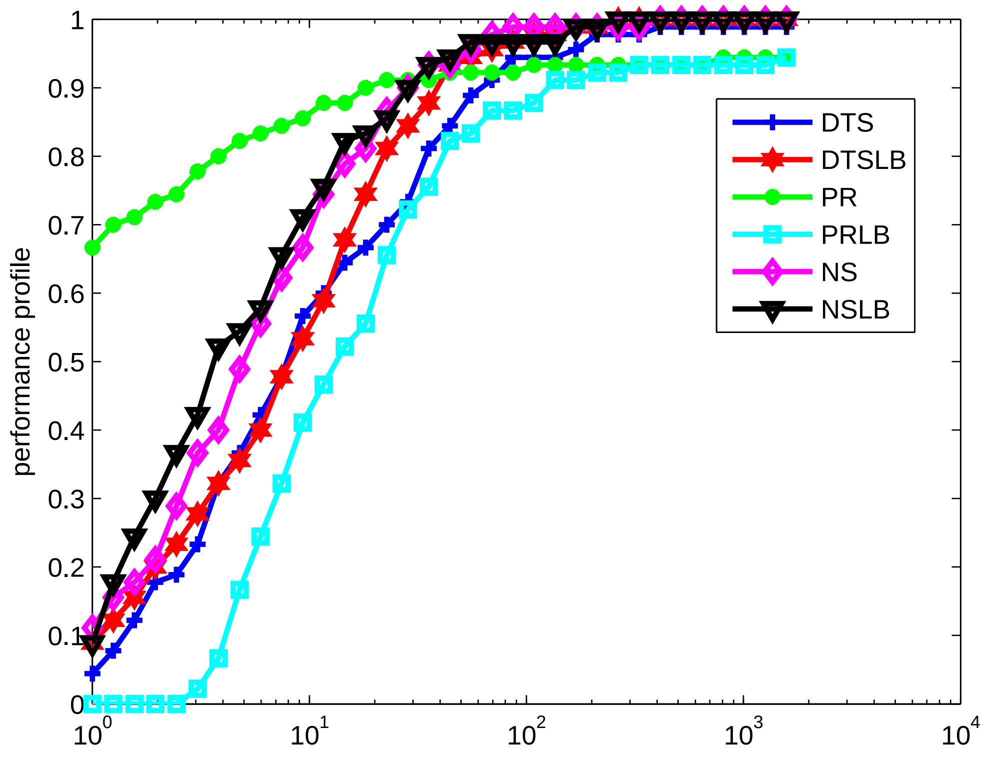

Figure 5–15: 64QAM, flat-fading system

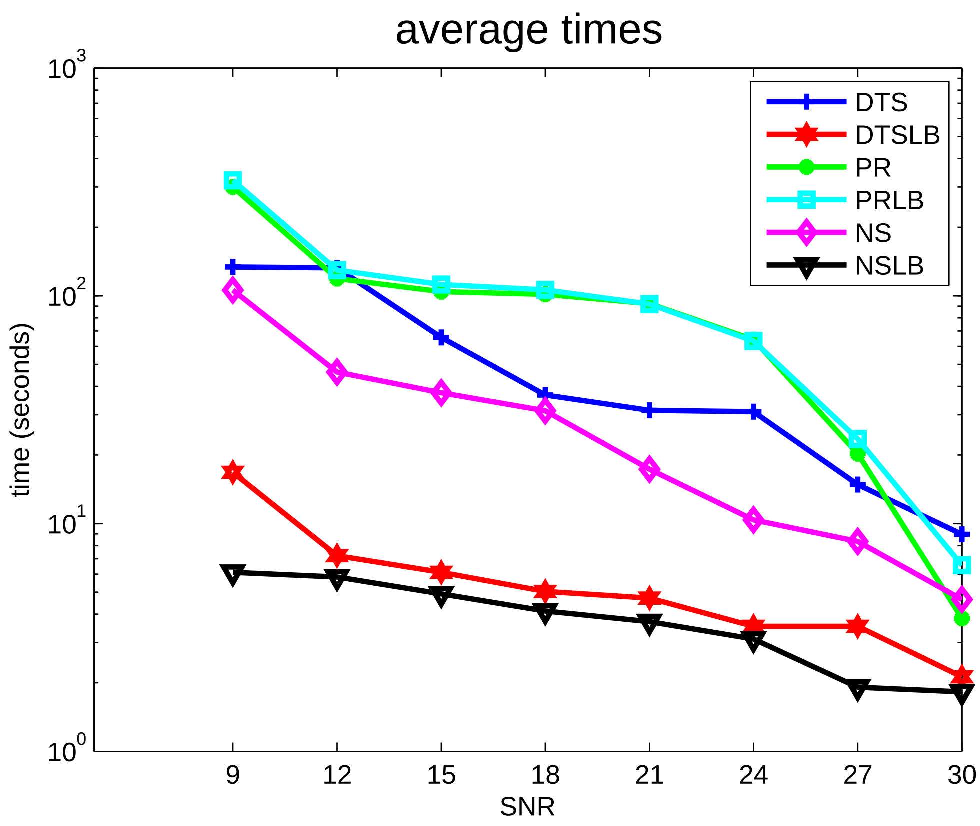

Figure 5–16: Case 3, average time against SNR (16QAM, flat-fading system)

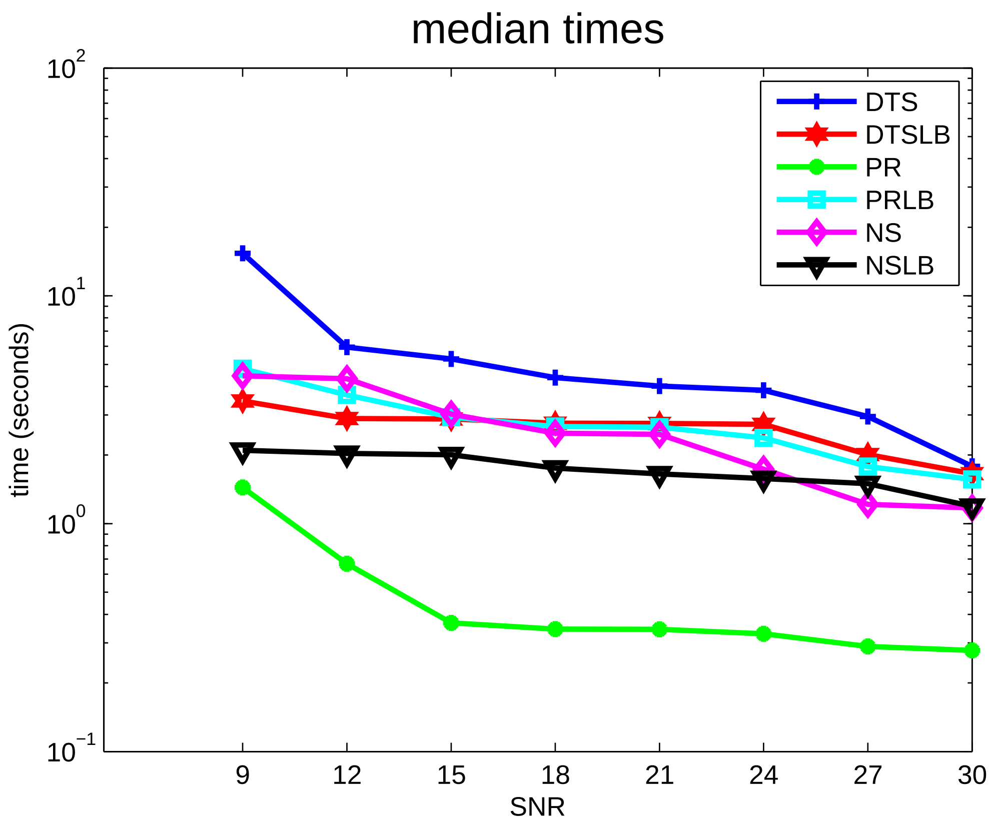

Figure 5–17: Case 3, median time against SNR (16QAM, flat-fading system)

CHAPTER 6
Summary and Future Work

In this thesis, we first reviewed several algorithms for different kinds of ILS problems, and then focused on the investigation of methods to improve the search process for UBILS problems. We proposed a lower bound and a new search algorithm for UBILS problems.

In Chapter 2, we reviewed existing sphere decoding algorithms for solving OILS, OBILS and UBILS problems respectively. For the UBILS problem, we gave two algorithms: DTS and PR.

In Chapter 3, first three lower bounds were reviewed for speeding up the search process for OBILS problems: norm-wise lower bound, component-wise lower bound, and basis reduced lower bound. Then, a new lower bound was proposed for the search process in the underdetermined part of the UBILS problem, which is inspired by the partial regularization approach. Numerical tests results indicate that lower bounds added to UBILS problems can speed up the search process of the direct tree search algorithm and the new search algorithm we proposed in Chapter 4.

In Chapter 4, we proposed a new search algorithm for solving the UBILS problem, which is a modification of the direct tree search algorithm and uses a best-first search idea. Numerical tests show that the new search algorithm outperforms than the direct tree search algorithm with or without incorporating lower bounds.

In Chapter 5, we compared six solvers: DTS, DTSLB, NS, NSLB, PR and PRLB in terms of average, median running and performance profiles. We also tested the above six solvers under three different conditions: random case, ill-conditioned case and flat-fading models in communications. The results show that it is difficult to find one solver can always have best performance in all cases. But we can give a rough recommendation for different cases here. If we pursue a good average or median running time, NSLB and PR usually give a good performance, respectively. If we have a strong preference for avoiding of extreme instances, PR will not be a good choice and NSLB, NS usually can guarantee good performances. If we only consider about cases which can be done in a very short time and do not mind a small percentage of extreme instances, PR will be a good choice. We can hardly list every specified condition and give the corresponding recommendation here and thus need to find a better algorithm which can perform well in more conditions in the future.

In the future work, we shall investigate the following problems: (1) The current column reordering strategy in the DTS algorithm is complicated and it deals with the under-determined part and the determined part quite differently. We intend to find a more unified and simpler strategy. (2) We noticed that unlike DTSLB and DTS, NSLB and NS, PRLB can not outperform PR in almost all test cases. One reason is the overhead of the lower bounds is significant. We plan to use lower bounds only in part of the search process to reduce the cost. (3) Simulation results show that the lower bound we proposed for underdetermined part seems not tight enough and we plan find a tighter bound for it.

References

[1] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger. Closest point search in lattices. *IEEE Transactions on Information Theory*, 48(8):2201–2214, 2002.

[2] M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proc. Annual ACM Symposium on Theory of Computing*, pages 601–610, 2001.

[3] M. Ajtai, R. Kumar, and D. Sivakumar. Sampling short lattice vectors and the closest lattice vector problem. In *Proc. IEEE Annual Conference on Computational Complexity*, pages 41–45, 2002.

[4] M.F. Anjos, X.-W. Chang, and W.-Y. Ku. Lattice preconditioning for the real relaxation branch-and-bound approach for integer least squares problems. *Journal of Global Optimization.*, (59):227–242, 2014.

[5] L. Babai. On lovasz lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[6] D.J. Bernstein, J. Buchmann, and E. Dahmen. *Post-quantum cryptography.* Springer Science & Business Media, 2009.

[7] S.C. Billups, S.P. Dirkse, and M.C. Ferris. A comparison of large scale mixed complementarity problem solvers. *Computational Optimization and Applications*, 7(1):3–25, 1997.

[8] A. Björck. *Numerical methods for least squares problems.* SIAM, 1996.

[9] J. Blömer and S. Naewe. Sampling methods for shortest vectors, closest vectors and successive minima. *Theoretical Computer Science*, 410(18):1648–1665, 2009.

[10] J. Boutros, N. Gresset, L. Brunel, and M. Fossorier. Soft-input soft-output lattice sphere decoder for linear channels. In *Proc. IEEE GLOBECOM*, pages 213–217, 2003.

[11] S. Breen and X.-W. Chang. Column reording for box-bonstrained integer least squares problems. In *Proc. IEEE GLOBECOM*, page 6, 2011.

[12] C. Buchheim, A. Caprara, and A. Lodi. An effective branch-and-bound algorithm for convex quadratic integer programming. *Math. Program*, 135:369–395, 2012.

[13] X.-W. Chang. Integer least squares estimation. *Lecture Notes*, 2005.

[14] X.-W. Chang and Q. Han. Solving box-constrained integer least squares problems. *IEEE Transactions on Wireless Communications*, 7(1):277–287, 2008.

[15] X.-W. Chang and C.C. Paige. Euclidean distances and least squares problems for a given set of vectors. *Applied Numerical Mathematics*, 57(1):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*, 59(8):4915–4926, 2013.

[17] X.-W. Chang and X. Yang. A new fast generalized sphere decoding algorithm for under-determined mimo systems. In *Proc. IEEE Biennial Symposium on Communications*, pages 18–21, 2006.

[18] X.-W. Chang and X. Yang. An effective tree search decoder for underdetermined mimo system. In *Proc. IEEE GLOBECOM*, pages 4375–4379, 2007.

[19] X.-W. Chang, X. Yang, T. Le-Ngoc, and P. Wang. Partial regularization approach for detection problems in underdetermined linear systems. *IET Communications*, 3:17–24, 2009.

[20] X.-W. Chang, X. Yang, and T. Zhou. MLAMBDA: A modified LAMBDA method for integer least-squares estimation. *Journal of Geodesy*, 79(9):552–565, 2005.

[21] T. Cui and C. Tellambura. An efficient generalized sphere decoder for rank-deficient mimo systems. In *Proc. IEEE Vehicular Technology Conference*, volume 5, pages 3689–3693, 2004.

[22] T. Cui and C. Tellambura. An efficient generalized sphere decoder for rank-deficient mimo systems. *IEEE Commun. Lett.*, pages 423–425, 2005.

[23] M. Damen, K. Abed-Meraim, and J. Belfiore. Generalized sphere decoder for asymmetrical space-time communication architecture. *Electron Letters*, 36:166–167, 2000.

[24] M. O. Damen, H. E. Gamal, and G. Caire. On maximum likelihood detection and the search for the closest lattice point. *IEEE Transactions on Information Theory*, 49(10):2389–2402, 2003.

[25] M. Daniele and G. Shafi. *Complexity of Lattice Problems:a cryptographic perspective*. Kluwer Academic Publishers, 2002.

[26] T. Datta, N. Srinidhi, A. Chockalingam, and B.S. Rajan. Low-complexity near-optimal signal detection in underdetermined large-mimo systems. In *Proc. IEEE National Conference on Communications*, pages 1–5, 2012.

[27] P. Dayal and M.K. Varanasi. A fast generalized sphere decoder for optimum decoding of under-determined mimo systems. In *Proc. Annual Allerton Conference on Communication Control and Computing*, volume 41, pages 1216–1225, 2003.

[28] E.D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

[29] U. Fincke and M. Pohst. Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. *Mathematics of Computation*, 44(170):463–471, 1985.

[30] 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 Journal on Selected Areas in Communications*, 17(11):1841–1852, 1999.

[31] V.M. Garcia, S. Roger, R.A. Trujillo, A.M. Vidal, and A. Gonzalez. A deterministic lower bound for the radius in sphere decoding search. In *International Conference on Advanced Technologies for Communications*, 2010.

[32] V. Guruswami, A. Sahai, and M. Sudan. Soft-decision decoding of chinese remainder codes. In *Proc. Annual Symposium on Foundations of Computer Science*, pages 159–168, 2000.

[33] G. Hanrot and D. Stehlé. Improved analysis of kannan's shortest lattice vector algorithm. In *Proc. Annual International Cryptology Conference on Advances in Cryptology*, pages 170–186, 2007.

[34] G. Hanrot, X. Xavier Pujol, and D. Stehlé. Algorithms for the shortest and closest lattice vector problems. In *Proc. International Conference on Coding and Cryptology*, pages 159–190, 2011.

[35] A. Hassibi and S. Boyd. Integer parameter estimation in linear models with applications to GPS. *IEEE Transactions on Singal Processing*, 46(11):2938–2952, 1998.

[36] R. Kannan. Improved algorithms for integer programming and related lattice problems. In *Proc. Annual ACM Symposium on Theory of Computing*, pages 193–206, 1983.

[37] R. Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of operations research*, 12(3):415–440, 1987.

[38] A. Korkine and G. Zolotareff. Sur les formes quadratiques. *Mathematische Annalen*, 6(3):366–389, 1873.

[39] A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[40] C. Ling and N. Howgrave-Graham. Effective LLL reduction for lattice decoding. In *Proc. IEEE International Symposium on Information Theory*, pages 196–200, 2007.

[41] D. Micciancio. The hardness of the closest vector problem with preprocessing. *IEEE Transactions on Information Theory*, 47(3):1212–1215, 2001.

[42] D. Micciancio and P. Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. *SIAM Journal on Computing*, 42(3):1364–1391, 2013.

[43] W.-H. Mow. Maximum likelihood sequence estimation from the lattice viewpoint. *IEEE Transactions on Information Theory*, 40(5):1591–1600, 1994.

[44] G. Papa, D. Ciuonzo, G. Romano, and P.S. Rossi. A dominance-based soft-input soft-output mimo detector with near-optimal performance. *IEEE Transactions on Communications*, 62(12):4320–4335, 2014.

[45] M. Pohst. On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. *ACM Sigsam Bulletin*, 15(1):37–44, 1981.

[46] C.-P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53(2):201–224, 1987.

[47] C.-P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66(1-3):181–199, 1994.

[48] K.V. Shahnaz and C.K. Ali. A sphere decoding algorithm for underdetermined ofdm/sdma uplink system with an effective radius selection. In *Proc. Advances in Intelligent Informatics*, pages 169–177. 2015.

[49] N. Sommer, M. Feder, and O. Shalvi. Finding the closest lattice point by iterative slicing. *SIAM J. Discrete Math*, 23(2):715–731, 2009.

[50] M. Stojnic, H. Vikalo, and B. Hassibi. An h infinity based lower bound to speed up the sphere decoder. In *Proc. IEEE Signal Processing Advances in Wireless Communications*, pages 751–755, 2005.

[51] M. Stojnic, H. Vikalo, and B. Hassibi. Speeding up the sphere decoder with h infinity and sdp inspired lower bounds. *IEEE Transactions on Signal Processing*, 56(2):712–726, 2008.

[52] K. Su and I.J. Wassell. A new ordering for efficient sphere decoding. In *Proc. IEEE International Conference on Communications*, volume 3, pages 1906–1910, 2005.

[53] P.J. Teunissen. Integer least-squares theory for the gnss compass. *Journal of Geodesy*, 84(7):433–447, 2010.

[54] P. van Emde Boas. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Technical report, Technical report 81-04,Mathematics Department, University of Amsterdam, 1981.

[55] K.-K. Wong and A. Paulraj. Efficient near maximum-likelihood detection for underdetermined mimo antenna systems using a geometrical approach. *EURASIP Journal on Wireless Communications and Networking*, 2007.

[56] S.J. Wright and J. Nocedal. *Numerical Optimization.* Springer New York, 1999.

[57] D. Wübben, R. Böhnke, J. Rinas, V. Kühn, and K.-D. Kammeyer. Efficient algorithm for decoding layered space-time codes. *Electronics letters*, 37(22):1348–1350, 2001.

[58] X. Xie. *Theory and Algorithms for Some Integer Least Squares Problems*. McGill University, 2014.

[59] X. Xie, X.W. Chang, and M. Al Borno. Partial LLL reduction. In *Proc. IEEE GLOBECOM*, page 5, 2011.

[60] Z. Yang, C. Liu, and J. He. A new approach for fast generalized sphere decoding in mimo systems. *IEEE Transactions on Signal Processing*, 12(1):41–44, 2005.

Index

KEY TO ABBREVIATIONS

CDMA: code-division multiple access 100

CTAN: comprehensive TeX archive network 100

IEEE: Institute of Electrical and Electronics Engineers, Inc. 100