An ADMM Based Method for Underdetermined Box-constrained Integer Least Squares Problems

Xiao-Wen Chang and Tianchi Ma

School of Computer Science, McGill University, Montreal, QC H3A 0E9, Canada

ARTICLE HISTORY
Compiled November 11, 2023

ABSTRACT
To solve underdetermined box-constrained integer least squares (UBILS) problems, we propose an integer-constrained alternating direction method of multipliers (IADMM), which can be much more accurate than the ADMM method. To guarantee to find the optimal solution, then we incorporate IADMM to DTS, a tree search method, to make the latter more efficient. Numerical tests show that the combined method IADMM-DTS can be much faster than the original DTS method. Finally we apply the combined method to a practical communication problem. Numerical results indicate that IADMM-DTS typically performs better than the commercial solvers CPLEX and MOSEK in terms of both efficiency and accuracy, and it can be used as an alternative to the commercial solver Gurobi for UBILS problems.

KEYWORDS
Integer parameter estimation; underdetermined integer least squares; box constraints; alternating direction method of multipliers; enumeration

AMS CLASSIFICATION
90C10, 90C59, 90C90, 62F10, 62H12, 62P30, 94A13

1. Introduction

Given a real matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, a real vector $\mathbf{y} \in \mathbb{R}^m$, a box $\mathcal{B} = \{\mathbf{x} \in \mathbb{Z}^n : \boldsymbol{\ell} \le \mathbf{x} \le \mathbf{u}, \boldsymbol{\ell}, \mathbf{u} \in \mathbb{Z}^n\}$, a *box-constrained integer least squares* (BILS) problem has the following form:

$$\min_{\mathbf{x}\in\mathcal{B}} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2. \tag{1}$$

When $\mathbf{A}$ has full column rank, we refer to (1) as the *overdetermined box-constrained integer least squares* (OBILS) problem, and when $\mathbf{A}$ has full row rank with $m < n$, we refer to it as the *underdetermined box-constrained integer least squares* (UBILS) problem. When $\mathbf{A}$ is rank deficient, we can use the QR factorization with column pivoting (see, e.g., [20, Sec. 5.5.5]) to transform it to a UBILS problem. This paper is concerned with solving the UBILS problem.

The BILS (either OBILS or UBILS) problems arise in some applications, such as multiple-input and multiple-output (MIMO) communications (see, e.g., [12, 15, 17,

CONTACT Xiao-Wen Chang. Email: chang@cs.mcgill.ca

29, 36, 44]) and control (see, e.g., [26, 28]). Typically the motivation of solving the BILS problems in applications is to estimate an integer parameter vector $\mathbf{x}^*$ in a linear model:

$$\mathbf{y} = \mathbf{A}\mathbf{x}^* + \mathbf{v}, \quad \mathbf{x}^* \in \mathcal{B}, \quad \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \sigma_{\mathbf{v}}^2 \mathbf{I}), \tag{2}$$

where $\mathbf{x}^*$ is a random integer-valued parameter vector uniformly distributed over the box $\mathcal{B}$, and $\mathbf{v}$ is a noise vector following the normal distribution $\mathcal{N}(\mathbf{0}, \sigma_{\mathbf{v}}^2 \mathbf{I})$. Given $\mathbf{A}$ and $\mathbf{y}$, one would like to recover $\mathbf{x}^*$. This is an estimation or detection problem. The solution to the BILS problem (1) is both the maximum likelihood estimator and the maximum a posterior estimator of $\mathbf{x}^*$, see, e.g., [27, Sec. 1.3.1].

If we do not consider the box constraint, then (1) becomes

$$\min_{\mathbf{x} \in \mathbb{Z}^n} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2. \tag{3}$$

In this case, we assume that $\mathbf{A}$ has full column rank and refer to (3) as an *overdetermined ordinary integer least squares* (OOILS) problem. The OOILS problem arises in communications (see, e.g., [1, 29, 36]), GPS positioning (see, e.g., [13, 23, 41]), and lattice cryptography (see, e.g., [22, 33]).

The OOILS problem (3) is NP-hard [42]. Although there are various approaches to solving it (see, e.g., [2, 22]), the enumeration approach is the one most widely used in applications [1, 7, 34, 41]. Typically it searches for the optimal integer points within an ellipsoid to find the optimal one. The enumeration approach can easily be extended to the OBILS problem, see, e.g., [5, 9, 17].

The enumeration approach has also been extended to the UBILS problem, see, e.g., [11, 16]. For the communications applications, the box constraint in the UBILS problem is special and there is another approach, which transforms the UBILS problem to an equivalent enlarged OBILS problem and then solve it by the enumeration approach, see [12, 15]. But both approaches may be too time consuming when $n - m$ is a little large and the box is a little large. Essentially this is because the number of equations is smaller than the number of unknowns in the model (2), leading to large degrees of freedom for the unknowns, i.e., large search space.

Recently Takapoui et al. [40] have applied the alternating direction method of multipliers (ADMM) to mixed integer quadratic programming (MIQP) problem. This is a new approach to solving the BILS problem, a special type of MIQP. ADMM is a simple but powerful algorithm that is well suited to distributed convex optimization, see Boyd et al. [6]. In [40] it showed that for the MIQP problems it considered ADMM has favourable computational costs and in many cases the global solution can be found although it is not guaranteed. However, for the UBILS problem we found in our numerical experiments that the ADMM algorithm is not a good heuristic in that it hardly converges to the optimal solution.

The main issue with the ADMM algorithm applied to the UBILS problem is its overrelaxation - in each iteration it solves an *overdetermined ordinary real least squares* (OORLS) problem and both the integer constraint and box constraint are relaxed. We will propose to modify it by imposing the integer constraint, i.e., we solve an OOILS problem instead in each iteration. For convenience, the resulting algorithm is referred to as IADMM, where "I" is for integer. One important component of ADMM is the penalty parameter. We discuss how to choose an initial value of the penalty parameter and update it during iterations in IADMM. A suggestion for choosing initial points

for IADMM iterations is also proposed. Numerical test results will be given to show superiority of IADMM over ADMM.

IADMM, as a heuristic algorithm, may not give the optimal solution. We propose to incorporate it into the enumeration approach, which finds the optimal solution, to make the enumeration process faster. The specific enumeration method for the UBILS problem to be considered is the the *direct tree search* (DTS) algorithm proposed by Chang and Yang [11]. We use IADMM to aid DTS in two aspects. One is to use the IADMM algorithm to find a solution and then use it to define the initial search radius for the DTS algorithm, which may reduce the search region significantly, compared with the default initial search radius. The other is to use the IADMM algorithm to find lower bounds for some sub-UBILS problems encountered in DTS's search process to prune the search tree. Specifically we derive a lower bound on the optimal value of a general UBILS problem based on the IADMM algorithm. We manage to do this because in each step of the IADMM algorithm we solve a relaxed ILS problem exactly. We then apply the technique to find lower bounds for some sub-UBILS problems encountered in DTS's search process, which are used to prune the search tree. The combined algorithm is to be referred to as IADMM-DTS.

There are various solvers which can be used to solve the UBILS problem. We compare IADMM-DTS with the well known commercial software packages CPLEX, Gurobi and MOSEK for solving UBILS problems arising in MIMO communications. Numerical results will show the advantages of IADMM-DTS.

The rest of the paper is organized as follows. In Section 2 we briefly describe the enumeration approach for various ILS problems and introduce ADMM for MIQP. In Section 3 we propose the IADMM algorithm. In Section 4 we present the main ideas of IADMM-DTS. In Section 5 we give numerical results. Finally Section 6 concludes the paper with a summary and future work.

Notation. For a vector $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x}_{i:j}$ denotes the subvector composed of elements of $\mathbf{x}$ with indices from i to j. For a matrix $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{m\times n}$, $\mathbf{A}_{i:j,k:\ell}$ denotes the submatrix of $\mathbf{A}$ formed by rows from i to j and columns from k to ℓ, and $\mathbf{A}_{i:j,k}$ denotes column k of $\mathbf{A}$ with row indices from i to j. For a scalar $x \in \mathbb{R}$, $\lfloor x \rceil$ denotes the nearest integer to x (if there is a tie, the one with smaller magnitude is chosen). A vector whose entries are ones is denoted by $\mathbf{1}$. Given a vector $\mathbf{x} \in \mathbb{R}^n$ and a set of n-vectors $\mathcal{X}$, $\mathbf{\Pi}_{\mathcal{X}}(\mathbf{x})$ denotes the projection of $\mathbf{x}$ onto $\mathcal{X}$, i.e., the vector in $\mathcal{X}$ nearest to $\mathbf{x}$ in the 2-norm. For a random vector $\mathbf{x}$, we denote $\mathrm{E}\{\mathbf{x}\}$ as its mean and $\mathrm{cov}\{\mathbf{x}\}$ as its covariance matrix. If a real-valued vector $\mathbf{x}$ is normally distributed with zero mean and covariance matrix $\sigma^2\mathbf{I}$, we write $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$.

2. Background

In this section, we first briefly review the enumeration approach to solving the OOILS, OBILS and UBILS problems and then introduce the general ADMM method and its application to the MIQP problem.

2.1. The enumeration approach for various ILS problems

The enumeration approach for the OOILS problem (3) has two stages. The first stage is the reduction, which makes the second stage easier and more efficient, and the second stage is the search, see, e.g., Agrell et al. [1].

In the first stage we compute the QRZ factorization of $\mathbf{A}$:

$$\mathbf{AZ} = \mathbf{QR}, \tag{4}$$

where $\mathbf{Z} \in \mathbb{Z}^{n\times n}$ is unimodular (i.e., $\det(\mathbf{Z}) = \pm 1$), $\mathbf{Q} \in \mathbb{R}^{m\times n}$ is column-orthonormal (i.e., $\mathbf{Q}^{\mathsf{T}}\mathbf{Q} = \mathbf{I}_n$), $\mathbf{R} \in \mathbb{R}^{n\times n}$ is upper triangular. Different choices for $\mathbf{Z}$ lead to different reductions. Typically one employs LLL lattice reduction proposed by Lenstra, Lenstra and Lovász [30]. For theoretical justifications of using LLL for solving OOILS, see Chang, Wen and Xie [10]. With $\bar{\mathbf{y}} := \mathbf{Q}^{\mathsf{T}}\mathbf{y}$ and $\bar{\mathbf{x}} := \mathbf{Z}^{-1}\mathbf{x}$, the original OOILS problem (3) is transformed to

$$\min_{\bar{\mathbf{x}}\in\mathbb{Z}^n} \|\bar{\mathbf{y}} - \mathbf{R}\bar{\mathbf{x}}\|_2^2. \tag{5}$$

In the second stage, we search the solution to the reduced problem (5). Suppose that the solution to (5) satisfies the following bound:

$$\|\bar{\mathbf{y}} - \mathbf{R}\bar{\mathbf{x}}\|_2^2 < \beta^2, \tag{6}$$

where β is called the search radius, and later we will say how to choose it. Note that (6) is a hyper-ellipsoid and our goal is to search this ellipsoid to find the optimal integer point. Here we briefly introduce the most widely used search method given by Schnorr and Euchner [37], which is an improvement of the original method proposed by Fincke and Pohst [19]. Write

$$\bar{\mathbf{y}} = \begin{bmatrix} \bar{\mathbf{y}}_{1:n-1} \\ \bar{y}_n \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} \mathbf{R}_{1:n-1,1:n-1} & \mathbf{R}_{1:n-1,n} \\ & r_{nn} \end{bmatrix}, \quad \bar{\mathbf{x}} = \begin{bmatrix} \bar{\mathbf{x}}_{1:n-1} \\ \bar{x}_n \end{bmatrix}.$$

Then we can easily observe that (6) is equivalent to

$$\|(\bar{\mathbf{y}}_{1:n-1} - \mathbf{R}_{1:n-1,n}\bar{x}_n) - \mathbf{R}_{1:n-1,1:n-1}\bar{\mathbf{x}}_{1:n-1}\|_2^2 < \beta^2 - r_{nn}^2(c_n - \bar{x}_n)^2, \tag{7}$$

$$r_{nn}^2(c_n - \bar{x}_n)^2 < \beta^2, \tag{8}$$

where $c_n := \bar{y}_n/r_{nn}$. We say that (8) gives the search region for $\bar{x}_n$ at *level* n. The search method first takes $\bar{x}_n = \lfloor c_n \rceil$, the nearest integer to c_n. If (8) does not hold, then the hyper-ellipsoid (6) does not include any integer point and the search process stops. If (8) holds, we denote this value of $\bar{x}_n$ by $\bar{x}_n'$ and attempt to find an $(n-1)$-integer point within the $(n-1)$-hyper-ellipsoid (7) that is the solution to the sub-problem

$$\min_{\bar{\mathbf{x}}_{1:n-1}\in\mathbb{Z}^{n-1}} \|(\bar{\mathbf{y}}_{1:n-1} - \mathbf{R}_{1:n-1,n}\bar{x}_n') - \mathbf{R}_{1:n-1,1:n-1}\bar{\mathbf{x}}_{1:n-1}\|_2^2, \tag{9}$$

which is an OOILS problem with dimension $n-1$. There are two cases. In case 1, the solution $\bar{\mathbf{x}}_{1:n-1}'$ to (9) is within the (n–1)-hyper-ellipsoid (7). In this case, $\bar{\mathbf{x}}' := \begin{bmatrix} \bar{\mathbf{x}}_{1:n-1}' \\ \bar{x}_n' \end{bmatrix}$ is an integer point in the n-hyper-ellipsoid (6), and we first update β^2 by defining $\beta^2 := \|\bar{\mathbf{y}} - \mathbf{R}\bar{\mathbf{x}}'\|_2^2$, and then go back to level n. In case 2, the solution $\bar{\mathbf{x}}_{1:n-1}'$ to (9) is outside the $(n-1)$-hyper-ellipsoid (7). In this case, $\bar{\mathbf{x}}'$ does not satisfy (6), and we also go back to level n. In both cases, when we go back to level n (see (8)), we choose $\bar{x}_n$ to be the next nearest integer to c_n and then continue the search process as before. Finally, when the search process stops at level n, i.e., no new integer can be chosen

to satisfy (8), the latest found n-dimensional integer point is the solution to (5). Here we want to point out when we solve (9), we use exactly the same search method. The whole process is a depth-first tree search. When an integer point in the hyper-ellipsoid (6) is found, the search radius β is updated so that the search region becomes smaller. The typical choice for the initial β is ∞.

The enumeration approach for solving the OBILS problem (1) is very similar to that for solving the OOILS problem (3), see Chang and Han [9] and the references therein. There are two main differences. One is that in the QRZ factorization (4), $\mathbf{Z}$ is chosen to be a permutation matrix rather than a more general unimodular matrix to keep the constraint in the optimization problem as a box after the reduction. The other difference is that in the search process, the box constraint has to be taking into account.

Now we briefly introduce the main idea of the enumeration approach for the UBILS problem (1), where $\mathbf{A} \in \mathbb{R}^{m\times n}$ has full row rank and $m < n$. In general UBILS is much more challenging than OOILS and OBILS. We first compute the QRZ factorization (4), where $\mathbf{Z} \in \mathbb{Z}^{n\times n}$ is a permutation matrix, $\mathbf{Q} \in \mathbb{R}^{m\times m}$ is orthogonal, $\mathbf{R} \in \mathbb{R}^{m\times n}$ is upper trapezoidal with nonzero diagonal entries (i.e., $r_{ij} = 0$ for all $i > j$ and $r_{ii} \neq 0$ for all i), Then, with

$$\bar{\mathbf{y}} := \mathbf{Q}^{\mathsf{T}}\mathbf{y}, \quad \bar{\mathbf{x}} := \mathbf{Z}^{\mathsf{T}}\mathbf{x}, \quad \bar{\boldsymbol{\ell}} := \mathbf{Z}^{\mathsf{T}}\boldsymbol{\ell}, \quad \bar{\mathbf{u}} := \mathbf{Z}^{\mathsf{T}}\mathbf{u}, \quad \bar{\mathcal{B}} := \{\bar{\mathbf{x}} \in \mathbb{Z}^n, \bar{\boldsymbol{\ell}} \leq \bar{\mathbf{x}} \leq \bar{\mathbf{u}}\}, \tag{10}$$

the original UBILS problem (1) can be transformed to

$$\min_{\bar{\mathbf{x}}\in\bar{\mathcal{B}}} \|\bar{\mathbf{y}} - \mathbf{R}\bar{\mathbf{x}}\|_2^2. \tag{11}$$

Suppose that the optimal solution of (11) satisfies

$$\|\bar{\mathbf{y}} - \mathbf{R}\bar{\mathbf{x}}\|_2^2 < \beta^2. \tag{12}$$

We can choose β as follows. We first solve the corresponding underdetermined box-constrained *real* least squares problem and then round the solution to the nearest integer vector, say $\bar{\mathbf{x}}^{\text{UBRLS}}$, and then take $\beta = \|\bar{\mathbf{y}} - \mathbf{R}\bar{\mathbf{x}}^{\text{UBRLS}}\|_2$. With appropriate partitioning of $\bar{\mathbf{y}}$, $\mathbf{R}$ and $\bar{\mathbf{x}}$ in (12), we see (12) is equivalent to

$$\begin{aligned} &\|(\bar{\mathbf{y}}_{1:m-1} - \mathbf{R}_{1:m-1,m:n}\bar{\mathbf{x}}_{m:n}) - \mathbf{R}_{1:m-1,1:m-1}\bar{\mathbf{x}}_{1:m-1}\|_2^2 \\ &\quad \leq \beta^2 - \Big(\bar{y}_m - \sum_{j=m}^{n} r_{mj}\bar{x}_j\Big)^2, \end{aligned} \tag{13}$$

$$\Big(\bar{y}_m - \sum_{j=m}^{n} r_{mj}\bar{x}_j\Big)^2 < \beta^2. \tag{14}$$

The search process uses (14) and the corresponding interval constraints to enumerate $\bar{x}_n, \bar{x}_{n-1}, \ldots, \bar{x}_m$. There may be many candidates for the sub-vector $\bar{\mathbf{x}}_{m:n}$ satisfying (14) and the corresponding box constraint. Once a value of $\bar{\mathbf{x}}_{m:n}$ is chosen, by (13) we solve the OBILS problem:

$$\min_{\bar{\mathbf{x}}_{1:m-1}\in\bar{\mathcal{B}}_{1:m-1}} \|(\bar{\mathbf{y}}_{1:m-1} - \mathbf{R}_{1:m-1,m:n}\bar{\mathbf{x}}_{m:n}) - \mathbf{R}_{1:m-1,1:m-1}\bar{\mathbf{x}}_{1:m-1}\|_2^2,$$

where $\bar{\mathcal{B}}_{1:m-1} := \{\bar{\mathbf{x}}_{1:m-1} \in \mathbb{Z}^{m-1} : \bar{\boldsymbol{\ell}}_{1:m-1} \leq \bar{\mathbf{x}}_{1:m-1} \leq \bar{\mathbf{u}}_{1:m-1}\}$. Note that here we use (14) to search for the sub-vector $\bar{\mathbf{x}}_{m:n}$, while in solving (5) we use (8) to search for the element $\bar{x}_n$. The former can be much more expensive due to a lot of choices for $\bar{\mathbf{x}}_{m:n}$ when $n-m$ is a little large, say $n-m=6$ when $m=20$. The specific enumeration method to be used in this paper for solving the UBILS problem is the DST algorithm proposed by Chang and Yang [11], which has been implemented in the MATLAB MILES package [8].

2.2. ADMM as a heuristic for mixed-integer quadratic programming problems

In [40], Takapoui et al. proposed to apply the ADMM approach to solve the MIQP problem:

$$\begin{aligned} &\min_{\mathbf{x}\in\mathbb{R}^n} (1/2)\mathbf{x}^\mathsf{T}\mathbf{M}\mathbf{x} + \mathbf{c}^\mathsf{T}\mathbf{x} + r \\ &\text{s.t. } \mathbf{B}\mathbf{x} = \mathbf{d}, \quad \mathbf{x} \in \mathcal{X}, \end{aligned} \tag{15}$$

where $\mathbf{M} \in \mathbb{R}^{n\times n}$ is a symmetric positive semidefinite matrix, $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{B} \in \mathbb{R}^{m\times n}$, $\mathbf{d} \in \mathbb{R}^m$, $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, with each $\mathcal{X}_i$ being an integer set or a convex subset of $\mathbb{R}$. Note that the MIQP problem (15) can be rewritten as

$$\begin{aligned} &\min_{\mathbf{x},\mathbf{z}} f(\mathbf{x}) + g(\mathbf{z}) \\ &\text{s.t. } \begin{bmatrix}\mathbf{B}\\ \mathbf{I}\end{bmatrix}\mathbf{x} - \begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\mathbf{z} = \begin{bmatrix}\mathbf{d}\\ \mathbf{0}\end{bmatrix}, \end{aligned}$$

where

$$f(\mathbf{x}) = (1/2)\mathbf{x}^\mathsf{T}\mathbf{M}\mathbf{x} + \mathbf{c}^\mathsf{T}\mathbf{x} + r, \quad g(\mathbf{z}) = \begin{cases} 0, & \text{if } \mathbf{z} \in \mathcal{X} \\ \infty, & \text{otherwise} \end{cases}.$$

Applying the ADMM approach leads to the following iteration scheme (in the scaled form) given in [40]:

$$\begin{aligned} \mathbf{x}^{(k+1)} = \underset{\mathbf{x}\in\mathbb{R}^n}{\operatorname{argmin}} \Bigg(&(1/2)\mathbf{x}^\mathsf{T}\mathbf{M}\mathbf{x} + \mathbf{c}^\mathsf{T}\mathbf{x} \\ &+ (\rho/2)\left\| \begin{bmatrix}\mathbf{B}\\ \mathbf{I}\end{bmatrix}\mathbf{x} - \begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\mathbf{z}^{(k)} - \begin{bmatrix}\mathbf{d}\\ \mathbf{0}\end{bmatrix} + \mathbf{w}^{(k)} \right\|_2^2 \Bigg), \end{aligned} \tag{16a}$$

$$\mathbf{z}^{(k+1)} = \mathbf{\Pi}_{\mathcal{X}}\left(\mathbf{x}^{(k+1)} + [\underset{m}{\mathbf{0}}, \underset{n}{\mathbf{I}}]\mathbf{w}^{(k)}\right), \tag{16b}$$

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \begin{bmatrix}\mathbf{B}\\ \mathbf{I}\end{bmatrix}\mathbf{x}^{(k+1)} - \begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\mathbf{z}^{(k+1)} - \begin{bmatrix}\mathbf{d}\\ \mathbf{0}\end{bmatrix}, \tag{16c}$$

where ρ is a constant penalty parameter, $\mathbf{\Pi}_{\mathcal{X}}$ is the projector onto $\mathcal{X}$. The $\mathbf{x}$-update is to solve a strongly convex quadratic programming problem in real space, the $\mathbf{z}$-update involves projection onto the set $\mathcal{X}$, and the $\mathbf{w}$-update is for updating the dual variable.

Through numerical experiments [40] showed that the ADMM method is an effective tool for MIQP problems arising from many embedded optimization applications, since ADMM usually finds a feasible point with reasonable objective value and is substantially faster than global optimization methods. The authors of [40] also applied this approach to a particular OBILS problem (see (1)) arising in communications and shown that satisfactory bit error rates can be obtained with substantially less computing time than the relax-and-round method. However, our experimental results indicated that for the UBILS problem the method hardly converges to the optimal solution and is not a good heuristic. Thus, some modifications need to be made in applying the ADMM approach.

Here we would like to mention a related work [39] by Soute and Lopes, which applied the ADMM approach to recover integer valued sparse signals. In the ADMM iterations, it provided a strategy to deals with the sparsity requirement.

3. IADMM: A modified ADMM method for the UBILS problem

In this section, first we propose an integer-constrained ADMM iteration scheme for solving the UBILS problem (1), then we discuss the choice of the penalty parameter and initial points, and finally we give a detailed description of the algorithm.

3.1. Integer-constrained ADMM iteration scheme

The UBILS problem (1) is a special case of the MIQP problem (15). When we apply the ADMM iteration scheme (16) to (1), it becomes

$$\mathbf{x}^{(k+1)} = \underset{\mathbf{x}\in\mathbb{R}^n}{\operatorname{argmin}} \left(\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + (\rho/2)\|\mathbf{x} - \mathbf{z}^{(k)} + \mathbf{w}^{(k)}\|_2^2 \right), \tag{17a}$$

$$\mathbf{z}^{(k+1)} = \mathbf{\Pi}_{\mathcal{B}}(\mathbf{x}^{(k+1)} + \mathbf{w}^{(k)}), \tag{17b}$$

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \mathbf{x}^{(k+1)} - \mathbf{z}^{(k+1)}. \tag{17c}$$

Here $\mathbf{x}^{(k+1)}$ is the *real* solution to a regularized real least squares problem, $\mathbf{z}^{(k+1)}$ is an integer vector obtained by projecting $\mathbf{x}^{(k+1)} + \mathbf{w}^{(k)}$ onto the box $\mathcal{B}$, and then the scaled dual variable $\mathbf{w}^{(k)}$ is updated, leading to $\mathbf{w}^{(k+1)}$.

Unfortunately the above ADMM iteration scheme does not work well. Often it does not converge, and even if it does, it does not converge to the optimal solution to the UBILS problem. Later we will give numerical examples to illustrate this. Here we propose a small change to the iteration scheme. Specifically, we impose the integrity constraint on $\mathbf{x}$ in (17a). For convenience, we replace the penalty parameter ρ by $2\lambda^2$ ($\lambda > 0$). Then (17) becomes

$$\mathbf{x}^{(k+1)} = \underset{\mathbf{x}\in\mathbb{Z}^n}{\operatorname{argmin}} \left(\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda^2\|\mathbf{x} - \mathbf{z}^{(k)} + \mathbf{w}^{(k)}\|_2^2 \right), \tag{18a}$$

$$\mathbf{z}^{(k+1)} = \mathbf{\Pi}_{\mathcal{B}}(\mathbf{x}^{(k+1)} + \mathbf{w}^{(k)}), \tag{18b}$$

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \mathbf{x}^{(k+1)} - \mathbf{z}^{(k+1)}. \tag{18c}$$

We refer to the above as the integer-constrained ADMM (IADMM) iteration scheme. Although the change is small, it brings a huge improvement.

Now we show how to solve the regularized ILS problem in (18a). The objective function in (17a) or (18a) can be rewritten as

$$\|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2+\lambda^2\|\mathbf{x}-\mathbf{z}^{(k)}+\mathbf{w}^{(k)}\|_2^2=\left\|\begin{bmatrix}\mathbf{y}\\ \lambda(\mathbf{z}^{(k)}-\mathbf{w}^{(k)})\end{bmatrix}-\begin{bmatrix}\mathbf{A}\\ \lambda\mathbf{I}\end{bmatrix}\mathbf{x}\right\|_2^2. \tag{19}$$

Note that the augmented matrix $\left[\begin{smallmatrix}\mathbf{A}\\ \lambda\mathbf{I}\end{smallmatrix}\right]$ has full column rank, thus (17a) is an OORLS problem, while (18a) is an OOILS problem. We can use the enumeration approach introduced in Section 2.1 to solve the OOILS problem. Although solving this OOILS problem may cost much more than solving the corresponding OORLS problem, this overcomes the overrelaxed issue with the latter, where both integer and box constraints are relaxed, and can make the iteration converge much faster. An OOILS problem is usually much easier to solve than a UBILS problem with the same dimension n.

The optimality conditions of the standard ADMM on convex problems are that the primal residual and dual residual both reach zero. For the IADMM iteration scheme (18), when the primal residual $\mathbf{r}^{(k+1)}=\mathbf{x}^{(k+1)}-\mathbf{z}^{(k+1)}$ and the dual residual $\mathbf{s}^{(k+1)}=2\lambda^2(\mathbf{z}^{(k+1)}-\mathbf{z}^{(k)})$ both reach zero, the algorithm stops ($\mathbf{x}$ and $\mathbf{z}$ updates will not change their values over further iterations). We found in our numerical experiments that if a constant λ is used over all iterations, sometimes the IADMM method may not converge, i.e., the above conditions cannot be reached. We will propose a varying strategy for choosing the penalty parameter in the following subsection.

3.2. Initial points and penalty parameter

In this subsection, we discuss how to set the initial points $\mathbf{z}^{(0)}$, $\mathbf{w}^{(0)}$ and how to choose the penalty parameter λ during iteration in IADMM.

It has been shown that in the convex case the choice of parameters in ADMM only affects the speed of the convergence, while in the non-convex case the choice can have a critical role in the quality of approximate solution, as well as the speed at which this solution is found (see, e.g., Diamond, Takapoui and Boyd [18] and Xu, Figueiredo, and Goldstein [43]). Note that the integer constraint is a special type of non-convex constraints.

For $\mathbf{w}^{(0)}$, it is reasonable to set it as $\mathbf{0}$ because the scaled dual variable $\mathbf{w}$ represents the running sum of residuals.

To choose $\mathbf{z}^{(0)}$ and the initial λ, we consider two situations in the following.

Suppose the UBILS problem comes from the linear model based estimation, i.e., the model (2) is known. For simplicity, we assume that $u_i-l_i=d$ for $i=1,2,...,n$ (in communication applications typically $\mathcal{B}$ is assumed to be a cube). Then, from the distribution of the true parameter vector $\mathbf{x}^*$ in (2), we have

$$\mathrm{E}\{\mathbf{x}^*\}=\frac{1}{2}(\boldsymbol{\ell}+\mathbf{u}),\quad \mathrm{cov}\{\mathbf{x}^*\}=\sigma_{\mathbf{x}^*}^2\mathbf{I},\quad \sigma_{\mathbf{x}^*}^2=\frac{(d+1)^2-1}{12}. \tag{20}$$

The real minimum mean square error (MMSE) estimator of $\mathbf{x}^*$ is the solution of a regularized real least squares problem (see, e.g., [32, Lesson 13]):

$$\mathbf{x}^{\mathrm{MMSE}}=\underset{\mathbf{x}\in\mathbb{R}^n}{\operatorname{argmin}}\left(\|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2+\lambda^{*2}\|\mathbf{x}-\mathrm{E}(\mathbf{x}^*)\|_2^2\right),$$

where

$$\lambda^* := \sigma_{\mathbf{v}}/\sigma_{\mathbf{x}^*}. \tag{21}$$

The MMSE guides us to set $\mathbf{z}^{(0)}$ as $\mathrm{E}(\mathbf{x}^*)$ and λ as λ^*. Then $\mathbf{x}^{(1)}$ given by IADMM in the first iteration has the same form as the MMSE estimator of $\mathbf{x}^*$ if we do not impose the integer restriction on $\mathbf{x}$ in (18a). In our numerical tests, we find that if λ is chosen to be larger than λ^* and it is fixed during the iteration, the accuracy of the final solution may drop dramatically, while setting λ to a smaller value usually does not change the accuracy of the final solution. Thus, we set the initial $\lambda^{(0)} = \alpha\lambda^*$ for a positive parameter $\alpha \leq 1$.

Suppose we have no knowledge about the linear model (2). In this case, it is still reasonable to set $\mathbf{z}^{(0)} = \frac{1}{2}(\boldsymbol{\ell} + \mathbf{u})$. For the initial $\lambda^{(0)}$, we suggest to choose a small value, say 10^{-2}.

Now we discuss how to update λ during the iterations. It has been shown that using appropriate varying penalty parameter over iterations can make ADMM more efficient and effective for both convex and non-convex optimisation problems (see, e.g., Boyd et al. [6], Diamond, Takapoui and Boyd [18], and Xu, Figueiredo and Goldstein [43]). For the IADMM method for the UBILS problem, we propose to increase the value of penalty parameter λ for every certain number of iterations until the method stops. The scheme is as follows:

$$\lambda^{(k+1)} = \begin{cases} \tau\lambda^{(k)} & \text{if } (k \bmod q) = 0, \\ \lambda^{(k)} & \text{otherwise,} \end{cases} \tag{22}$$

where $\tau \geq 1$ is a factor of inflating λ and $(k \bmod q)$ is the remainder of the division of the current iteration k by an integer parameter q. When q is set as 1, λ is inflated at each iteration. In the following, we give some explanations about this strategy.

First, it is inspired by the quadratic penalty function method (see e.g., Bertsekas [4]), whose idea is to eliminate some or all of the constraints by adding to the objective function a penalty term which prescribes a high cost to infeasible points. Associated with this method is a penalty parameter λ, which determines the severity of the penalty and as a consequence the extent to which the resulting unconstrained problem approximates the original constrained problem. Thus the quadratic penalty method consists of solving a sequence of problems with $\{\lambda^{(k)}\}$ as a sequence of penalty parameters satisfying:

$$\forall k, \quad 0 < \lambda^{(k)} < \lambda^{(k+1)}, \quad \lambda^{(k+1)} \to \infty.$$

Similarly, for the IADMM method, if such a sequence of penalty parameters $\{\lambda^{(k)}\}$ is adopted, it is not hard to see that the algorithm is guaranteed to stop. In fact, from (18a), we see that when $\lambda^{(k)}$ is large enough we must have $\mathbf{x}^{(k+1)} - \mathbf{z}^{(k)} + \mathbf{w}^{(k)} = \mathbf{0}$, because all of these vectors here are integer vectors. Then from (18b) we obtain $\mathbf{z}^{(k+1)} = \mathbf{z}^{(k)}$ and from (18b) we obtain $\mathbf{w}^{(k+1)} = \mathbf{0}$. In the next step we have $\mathbf{x}^{(k+2)} = \mathbf{z}^{(k+1)}$ from (18a) and $\mathbf{z}^{(k+2)} = \mathbf{z}^{(k+1)}$ from (18b). Therefore the primal residual $\mathbf{r}^{(k+2)} = \mathbf{0}$ and the dual residual $\mathbf{s}^{(k+2)} = \mathbf{0}$, leading to termination of the algorithm. In (22) we increase λ every certain number of iterations instead of every iteration because we would like the algorithm to stop with a relatively small λ. Another reason for keeping λ unchanged over some iterations is that we need to perform

the partial LLL lattice reduction only once in those iterations when we solve the associated OOILS problems for $\mathbf{x}-$update in (18a) since the augmented matrix $\left[\begin{smallmatrix}\mathbf{A}\\ \lambda\mathbf{I}\end{smallmatrix}\right]$ in its objective function (see (19)) is unchanged during those iterations.

Second, with the goal of making the performance less dependent on the initial choice of the penalty parameter and accelerating convergence as well, we would like to inflate the penalty parameter when the primal residual $\mathbf{r}^{(k+1)} = \mathbf{x}^{(k+1)} - \mathbf{z}^{(k+1)}$ appears large compared to the dual residual $\mathbf{s}^{(k+1)} = 2\lambda^2(\mathbf{z}^{(k+1)} - \mathbf{z}^{(k)})$, and deflate it when the primal residual seems too small relative to the dual residual. In our numerical tests we found that the primal residual is much larger than the dual residual over iterations. Thus we do not need to deflate the penalty parameter. Adopting (22) will ensure that IADMM terminates.

3.3. The IADMM algorithm

Now we can give a full description of the IADMM method for the UBILS problem in Algorithm 1.

Algorithm 1 IADMM for a UBILS problem

Input: The matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ $(m < n)$ with full row rank, the vector $\mathbf{y} \in \mathbb{R}^m$, the lower bound $\boldsymbol{\ell} \in \mathbb{Z}^n$ and the upper bound $\mathbf{u} \in \mathbb{Z}^n$ of the box $\mathcal{B} = \{\mathbf{x} \in \mathbb{Z}^n : \boldsymbol{\ell} \le \mathbf{x} \le \mathbf{u}\}$, the initial penalty parameter $\lambda^{(0)}$, the constants q and τ for initializing and updating the penalty parameter and the maximal number of iterations K.

Output: The feasible solution (not necessarily optimal) $\mathbf{x}^{\text{IA}}$ to the UBILS problem (1) and the corresponding residual norm $\beta = \|\mathbf{y} - \mathbf{A}\mathbf{x}^{\text{IA}}\|_2$.

Function: $(\mathbf{x}^{\text{IA}}, \beta) = \text{IADMM}(\mathbf{A}, \mathbf{y}, \boldsymbol{\ell}, \mathbf{u}, \lambda^{(0)}, q, \tau, K)$

1: Set $\mathbf{z}^{(0)} = (\boldsymbol{\ell} + \mathbf{u})/2$, $\mathbf{w}^{(0)} = \mathbf{0}$, $\lambda = \lambda^{(0)}$, $\beta = \boldsymbol{\infty}$
2: **for** $k = 1 : K$ **do**
3: $\quad \mathbf{x}^{(k)} = \underset{\mathbf{x}\in\mathbb{Z}^n}{\text{argmin}} \left\| \begin{bmatrix} \mathbf{y} \\ \lambda(\mathbf{z}^{(k-1)} - \mathbf{w}^{(k-1)}) \end{bmatrix} - \begin{bmatrix} \mathbf{A} \\ \lambda\mathbf{I} \end{bmatrix} \mathbf{x} \right\|_2^2$
4: $\quad \mathbf{z}^{(k)} = \boldsymbol{\Pi}_{\mathcal{B}}(\mathbf{x}^{(k)} + \mathbf{w}^{(k-1)})$
5: $\quad \mathbf{w}^{(k)} = \mathbf{w}^{(k-1)} + \mathbf{x}^{(k)} - \mathbf{z}^{(k)}$
6: $\quad$ **if** $\beta > \|\mathbf{y} - \mathbf{A}\mathbf{z}^{(k)}\|_2$ **then**
7: $\quad\quad$ Set $\mathbf{x}^{\text{IA}} = \mathbf{z}^{(k)}$ and $\beta = \|\mathbf{y} - \mathbf{A}\mathbf{z}^{(k)}\|_2$
8: $\quad$ **end if**
9: $\quad$ **if** $\mathbf{x}^{(k)} = \mathbf{z}^{(k)} = \mathbf{z}^{(k-1)}$ **then**
10: $\quad\quad$ Terminate
11: $\quad$ **end if**
12: $\quad$ **if** $k \pmod q = 0$ **then**
13: $\quad\quad$ $\lambda = \lambda\tau$
14: $\quad\quad$ $\mathbf{w}^{(k)} = \mathbf{w}^{(k)}/\tau^2$
15: $\quad$ **end if**
16: **end for**

Here we give some remarks on Algorithm 1:

(1) Line 3: When λ is not changed from the previous iteration, we use the partial LLL reduction (see Sec. 2.1) of the augmented matrix obtained or used in the previous iteration and start the search phase directly. If λ is changed from the previous iteration, the change is expected small. We use the reduction result of the previous iteration to make the reduction process for the new augmented

matrix faster.

(2) Lines 6–8: Since the objective function may not decrease monotonically during iterations, after each iteration, we check if the latest iterate $\mathbf{z}^{(k)}$ is better than the current best solution $\mathbf{x}^{\text{IA}}$. If so, we update the latter. This means that the final solution $\mathbf{x}^{\text{IA}}$ may not be the last iterate.

(3) Line 14: Note that $\mathbf{w}$ is the scaled form of the dual variable. Therefore, after updating λ, $\mathbf{w}$ must also be updated; see, e.g., [6, Chap. 3].

4. Incorporating IADMM into the enumeration approach

Although in general IADMM is much faster and more accurate than ADMM for solving the UBILS problem, the solution obtained by the former may still not be the optimal one. This is different from the enumeration approach, which finds the optimal solution. To find the optimal solution, in this section we propose to incorporate IADMM into an enumeration method named DTS proposed by Chang and Yang [11]. The goal is to make the latter faster. The combined method is to be referred to as IADMM-DTS.

4.1. Initial search radius

One factor which affects the efficiency of the enumeration approach is the initial search radius β in (12). If β is small, the initial search region is small. One would like to find a β as small as possible. As the solution obtained by IADMM is usually good, we propose to use it to define the initial β. Specifically, we apply IADMM to solve the reduced UBILS problem (11). Let the solution be denoted by $\bar{\mathbf{x}}^{\text{IA}}$. Then we define the initial search radius $\beta = \|\bar{\mathbf{y}} - \mathbf{R}\bar{\mathbf{x}}^{\text{IA}}\|_2$. The DTS method uses the rounded value of the solution of the corresponding real problem to define the initial β (see the lines after (12)). Numerical results will be given to show that the new initial radius can reduce the total running time significantly, although finding the new initial radius takes more time.

4.2. Lower bounds

In this subsection, we first propose an IADMM based method to find a lower bound for a general UBILS problem, and then apply the method to find lower bounds for some sub-UBILS problems encountered in the search process of the DTS method to prune the search tree.

Suppose we use IADMM to solve the UBILS problem (1). From (18a) we have

$$\begin{aligned}\forall \mathbf{x} \in \mathbb{Z}^n,\ \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 \geq & -\lambda^2 \|\mathbf{x} - \mathbf{z}^{(k)} + \mathbf{w}^{(k)}\|_2^2 \\ & + \|\mathbf{y} - \mathbf{A}\mathbf{x}^{(k+1)}\|_2^2 + \lambda^2 \|\mathbf{x}^{(k+1)} - \mathbf{z}^{(k)} + \mathbf{w}^{(k)}\|_2^2.\end{aligned} \tag{23}$$

Then we have a lower bound as follows:

$$\begin{aligned}\forall \mathbf{x} \in \mathcal{B},\ \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 \geq & -\lambda^2 \sum_{i=1}^{n} \left[\max(|u_i - z_i^{(k)} + w_i^{(k)}|, |l_i - z_i^{(k)} + w_i^{(k)}|)\right]^2 \\ & + \|\mathbf{y} - \mathbf{A}\mathbf{x}^{(k+1)}\|_2^2 + \lambda^2 \|\mathbf{x}^{(k+1)} - \mathbf{z}^{(k)} + \mathbf{w}^{(k)}\|_2^2.\end{aligned} \tag{24}$$

Then we take the maximum lower bound over all iterations to get a lower bound on $\min_{\mathbf{x}\in\mathcal{B}} \|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2$.

We modify Algorithm 1 so that the algorithm produces either the maximum lower bound or a flag which indicates the current lower bound is larger than a given value. The latter is for reducing unnecessary computation and an explanation will be given later. The modified algorithm is described in Algorithm 2.

Algorithm 2 IADMM-based Lower Bound (IADMM-LB)

Input: The matrix $\mathbf{A}\in\mathbb{R}^{m\times n}$ $(m<n)$ with full row rank, the vector $\mathbf{y}\in\mathbb{R}^m$, the lower bound $\boldsymbol{\ell}\in\mathbb{Z}^n$ and the upper bound $\mathbf{u}\in\mathbb{Z}^n$ of the box $\mathcal{B}=\{\mathbf{x}\in\mathbb{Z}^n : \boldsymbol{\ell}\le\mathbf{x}\le\mathbf{u}\}$, the initial penalty parameter $\lambda^{(0)}$, the constants q and τ for initializing and updating the penalty parameter, the maximal number of iterations K, and a constant γ.

Output: The lower bound T of $\min_{\boldsymbol{\ell}\le\mathbf{x}\le\mathbf{u},\mathbf{x}\in\mathbb{Z}^n} \|\mathbf{y}-\mathbf{A}\mathbf{x}\|_2^2$ and $Flag$, which is 1 if the lower bound is larger than γ and 0 otherwise.

Function: $(T, Flag) = \text{IADMM-LB}(\mathbf{A}, \mathbf{y}, \boldsymbol{\ell}, \mathbf{u}, \lambda^{(0)}, q, \tau, K, \gamma)$

Set $\mathbf{z}^{(0)} = (\boldsymbol{\ell}+\mathbf{u})/2$, $\mathbf{w}^{(0)} = \mathbf{0}\in\mathbb{R}^n$, $\lambda = \lambda^{(0)}$, $T=0$, $Flag=0$
for $k = 1 : K$ **do**
  $\mathbf{x}^{(k)} = \underset{\mathbf{x}\in\mathbb{Z}^n}{\operatorname{argmin}} \left\| \begin{bmatrix} \mathbf{y} \\ \lambda(\mathbf{z}^{(k-1)} - \mathbf{w}^{(k-1)}) \end{bmatrix} - \begin{bmatrix} \mathbf{A} \\ \lambda\mathbf{I} \end{bmatrix} \mathbf{x} \right\|_2^2$
  $T = \max(T, \|\mathbf{y}-\mathbf{A}\mathbf{x}^{(k)}\|_2^2 + \lambda^2\|\mathbf{x}^{(k)} - \mathbf{z}^{(k-1)} + \mathbf{w}^{(k-1)}\|_2^2$
    $-\lambda^2 \sum_i \max((l_i - z_i^{(k-1)} + w_i^{(k-1)})^2, (u_i - z_i^{(k-1)} + w_i^{(k-1)})^2))$
  if $T \ge \gamma$ **then**
    $Flag = 1$
    Terminate
  end if
  $\mathbf{z}^{(k)} = \boldsymbol{\Pi}_{\mathcal{B}}(\mathbf{x}^{(k)} + \mathbf{w}^{(k-1)})$
  $\mathbf{w}^{(k)} = \mathbf{w}^{(k-1)} + \mathbf{x}^{(k)} - \mathbf{z}^{(k)}$
  if $\mathbf{x}^{(k)} = \mathbf{z}^{(k)} = \mathbf{z}^{(k-1)}$ **then**
    Terminate
  end if
  if $k \pmod q = 0$ **then**
    $\lambda = \lambda \times \tau$
    $\mathbf{w}^{(k)} = \mathbf{w}^{(k)}/\tau^2$
  end if
end for

Our numerical tests indicate that the lower bound obtained by Algorithm 2 is usually tight for small or moderate residual of the optimal solution, see [31, Sec. 5.2.3] for details.

In the following we show how to incorporate Algorithm 2 into the DTS method to find lower bounds to prune the search tree.

In the search process to determine $\bar{\mathbf{x}}_{m:n}$, DTS uses the inequality (14). To reduce the upper bound in (14) so that the search space is reduced, we apply Algorithm 2 to the UBILS problem whose objective function is $\|\bar{\mathbf{y}}_{1:m-1} - \mathbf{R}_{1:m-1,1:n}\bar{\mathbf{x}}\|_2^2$:

$$(T_{n+1}, Flag_{n+1}) = \text{IADMM-LB}(\mathbf{R}_{1:m-1,1:n}, \bar{\mathbf{y}}_{1:m-1}, \bar{\boldsymbol{\ell}}, \bar{\mathbf{u}}, \lambda^{(0)}, q, \tau, K, \beta^2). \tag{25}$$

Thus, from the inequality (13), the optimal solution to (11) satisfies

$$T_{n+1} \leq \|\bar{\mathbf{y}}_{1:m-1} - \mathbf{R}_{1:m-1,1:n}\bar{\mathbf{x}}\|_2^2 \leq \beta^2 - \Big(\bar{y}_m - \sum_{j=m}^{n} r_{mj}\bar{x}_j\Big)^2.$$

Then

$$\Big(\bar{y}_m - \sum_{j=m}^{n} r_{mj}\bar{x}_j\Big)^2 < \beta^2 - T_{n+1}.$$

Based on this instead of (14), the DTS method enumerates $\bar{\mathbf{x}}_{m:n}$.

Next we consider using Algorithm 2 to prune the search tree at level $k = m + 1, \ldots, n-1, n$. Suppose the search process is now at level k and a value for $\bar{x}_k$ is just chosen. Note that the current value of $\bar{\mathbf{x}}_{k:n}$ is known. If it is a part of the optimal solution of the UBILS problem (11), then the following inequality should hold:

$$\min_{\bar{\mathbf{x}}_{1:k-1} \in \bar{\mathcal{B}}_{1:k-1}} \|(\bar{\mathbf{y}} - \mathbf{R}_{:,k:n}\bar{\mathbf{x}}_{k:n}) - \mathbf{R}_{:,1:k-1}\bar{\mathbf{x}}_{1:k-1}\|_2^2 < \beta^2, \tag{26}$$

where $\bar{\mathcal{B}}_{1:k-1} = \{\mathbf{x} \in \mathbb{Z}^{k-1} : \bar{\boldsymbol{\ell}}_{1:k-1} \leq \mathbf{x} \leq \bar{\mathbf{u}}_{1:k-1}\}$. We employ Algorithm 2 to compute a lower bound T_k on the left hand side of (26), i.e.,

$$(T_k, Flag_k) = \text{IADMM-LB}(\mathbf{R}_{:,1:k-1}, \bar{\mathbf{y}} - \mathbf{R}_{:,k:n}\bar{\mathbf{x}}_{k:n}, \bar{\boldsymbol{\ell}}_{1:k-1}, \bar{\mathbf{u}}_{1:k-1}, \lambda^{(0)}, q, \tau, K, \beta^2). \tag{27}$$

If $Flag_k = 0$, then $T_k < \beta^2$. In this case, the current value of $\bar{x}_k$ is a valid candidate and we move down level $k-1$ to choose $\bar{x}_{k-1}$. Otherwise, i.e., $Flag_k = 1$, then $T_k \geq \beta^2$. In this case, (26) does not hold, thus, the current value of $\bar{x}_k$ is invalid and the branch $\bar{\mathbf{x}}_{k:n}$ (corresponding to the current value of $\bar{\mathbf{x}}_{k:n}$) in the search tree cannot be a part of the optimal path (corresponding to the optimal solution). Then we should choose another integer in $\bar{\mathcal{B}}_k = \{x_k \in \mathbb{Z} : \bar{l}_k \leq x_k \leq \bar{u}_k\}$ for $\bar{x}_k$ based on the DTS method.

Computing a lower bound involves an extra cost. The benefits of applying the lower-bound technique for pruning branches decrease when k decreases. In practice we should use the lower-bound technique only at a few levels near the root of the search tree, i.e., k is close to n.

The detailed description of the whole IADMM-DTS algorithm can be found in Ma [31, Sec. 4.2].

5. Numerical Experiments

In this section we demonstrate efficiency and accuracy of the IADMM algorithm proposed in Section 3 and of the IADMM-DST algorithm proposed in Section 4 for solving the UBILS problem. Our proposed algorithms are implemented in MATLAB 2019a and all tests are run on a laptop with 2.3 GHz 8-Core Intel i9 CPU, 16 GB memory and MacOS. We will use the MATLAB package MILES [8] to solve the OOILS problem and the UBILS problem in our experiments.

5.1. Experiment Setup

We use some MATLAB built-in functions to generate data in our experiments: `randn(p,q)`, which generates a $p \times q$ matrix containing pseudorandom values drawn from the standard normal distribution; and `randi([imin,imax],p,q)`, which generates a $p \times q$ matrix containing pseudorandom integer values drawn from the discrete uniform distribution on [imin, imax]. We consider two different examples:

Example 1 (Real model). Take $\mathbf{A} =$ `randn(m,n)`, $\mathbf{x}^* =$ `randi([l,u],n,1)`, and $\mathbf{v} = \sigma_{\mathbf{v}} *$ `randn(m,1)`, and then compute $\mathbf{y} = \mathbf{A}\mathbf{x}^* + \mathbf{v}$.

Example 2 (Complex model in MIMO communications). Here we consider a real-world application, see, e.g., [3]. In this application, the relation between received signal vectors and transmit signal vectors of this system can be written as a complex linear model:

$$\mathbf{y}_c = \mathbf{A}_c\mathbf{x}_c^* + \mathbf{v}_c, \tag{28}$$

where $\mathbf{A}_c \in \mathbb{C}^{N_r \times N_t}$ represents the channel matrix with N_t transmitter antennas and N_r receiver antennas, the elements of $\mathbf{x}_c^*$ are independently uniformly distributed over the set $\mathcal{X}_c = \{k_1 + k_2 i : k_1, k_2 = \pm 1, \pm 3, \ldots, \pm(2^k - 3), \pm(2^k - 1)\}$ which is referred to as 4^k-QAM (QAM stands for quadrature amplitude modulation) and $\mathbf{v}_c \in \mathbb{C}^{N_r}$ is a noise vector following the complex normal distribution $\mathcal{CN}(\mathbf{0}, \sigma_{\mathbf{v}}^2\mathbf{I})$ (i.e., its real and imaginary parts are independent and each follows $\mathcal{N}(\mathbf{0}, \frac{1}{2}\sigma_{\mathbf{v}}^2\mathbf{I})$). In communications, a measure of the quality of the data is signal-to-noise-ratio (SNR). We take SNR $= 10\log_{10}(4^k - 1)/(3k\sigma_{\mathbf{v}}^2)$, as defined on [3, p49]. We consider two types of channel matrices which are typically used for numerical tests in the literature.

- **(a)** $\mathbf{A}_c$ is an uncorrelated Rayleigh-fading matrix. Specifically, the elements of $\mathbf{A}_c$ are i.i.d normal random variables with distribution $\mathcal{CN}(0, 1)$; see [24].
- **(b)** $\mathbf{A}_c$ is a correlated Rayleigh-fading matrix: $\mathbf{A}_c = \mathbf{\Psi}^{1/2}\mathbf{H}_c\mathbf{\Phi}^{1/2}$, where $\mathbf{H}_c \in \mathbb{C}^{N_r \times N_t}$ is defined in the same way as $\mathbf{A}_c$ in (**a**), $\psi_{ij} = a^{|i-j|}$ and $\phi_{ij} = b^{|i-j|}$ with $a, b \in [0, 1]$, $\mathbf{\Psi}^{1/2}\mathbf{\Psi}^{1/2} = \mathbf{\Psi}$ and $\mathbf{\Phi}^{1/2}\mathbf{\Phi}^{1/2} = \mathbf{\Phi}$; see [14] and [38]. In our numerical tests, we choose $a = b = 0.9$ to make $\mathbf{A}_c$ significantly more ill-conditioned than $\mathbf{A}_c$ in (**a**).

To deal with the complex case, we first transform (28) into a real linear model. We split each quantity in (28) into a real part and an imaginary part:

$$\mathbf{A}_c = \mathbf{A}_{\mathrm{R}} + i\mathbf{A}_{\mathrm{I}}, \quad \mathbf{y}_c = \mathbf{y}_{\mathrm{R}} + i\mathbf{y}_{\mathrm{I}}, \quad \mathbf{x}_c^* = \mathbf{x}_{\mathrm{R}}^* + i\mathbf{x}_{\mathrm{I}}^*, \quad \mathbf{v}_c = \mathbf{v}_{\mathrm{R}} + i\mathbf{v}_{\mathrm{I}}.$$

Then (28) is equivalent to the real linear model:

$$\bar{\mathbf{y}} = \bar{\mathbf{A}}\bar{\mathbf{x}}^* + \mathbf{v}, \tag{29}$$

where

$$\bar{\mathbf{y}} = \begin{bmatrix} \mathbf{y}_{\mathrm{R}} \\ \mathbf{y}_{\mathrm{I}} \end{bmatrix}, \quad \bar{\mathbf{A}} = \begin{bmatrix} \mathbf{A}_{\mathrm{R}} & -\mathbf{A}_{\mathrm{I}} \\ \mathbf{A}_{\mathrm{I}} & \mathbf{A}_{\mathrm{R}} \end{bmatrix}, \quad \bar{\mathbf{x}}^* = \begin{bmatrix} \mathbf{x}_{\mathrm{R}}^* \\ \mathbf{x}_{\mathrm{I}}^* \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} \mathbf{v}_{\mathrm{R}} \\ \mathbf{v}_{\mathrm{I}} \end{bmatrix},$$
$$\bar{\mathbf{x}}^* \in \bar{\mathcal{X}} \times \cdots \times \bar{\mathcal{X}}, \quad \bar{\mathcal{X}} := \{\pm 1, \pm 3, ..., \pm(2^k - 1)\}.$$

To transform the linear model (29) with the constraint to the standard one, we do the

following transformations:

$$\mathbf{x}^* = \frac{1}{2}(2^k - 1)\mathbf{1} + \frac{1}{2}\bar{\mathbf{x}}^*, \quad \mathbf{y} = \bar{\mathbf{y}} + (2^k - 1)\bar{\mathbf{A}}\mathbf{1}, \quad \mathbf{A} = 2\bar{\mathbf{A}}. \tag{30}$$

Then (29) becomes the standard one:

$$\begin{aligned} &\mathbf{y} = \mathbf{A}\mathbf{x}^* + \mathbf{v}, \\ &\mathbf{x}^* \in \mathcal{B} := \mathcal{X} \times \cdots \times \mathcal{X}, \quad \mathcal{X} := \{0, 1, 2, \ldots, 2^k - 1\}. \end{aligned} \tag{31}$$

Note that here $\mathbf{A} \in \mathbb{R}^{2N_r \times 2N_t}$, the elements of $\mathbf{x}^*$ are independently uniformly distributed over $\mathcal{B}$ with $\mathrm{E}\{\mathbf{x}^*\} = \frac{1}{2}(2^k - 1)\mathbf{1}$ and $\mathrm{cov}\{\mathbf{x}^*\} = \frac{1}{12}(4^k - 1)\mathbf{I}$, each element of $\mathbf{x}^*$ can be represented by k bits, and $\mathbf{v} \sim N(\mathbf{0}, \frac{1}{2}\sigma_{\mathbf{v}}^2\mathbf{I})$.

In our numerical tests, the values of parameters in constructing the data for both Examples 1 and 2 will be chosen to be consistent with the literature. In real-time applications, typically the dimensions of the problems are quite small, but the estimation of the true integer parameter vector needs to be done very quickly.

5.2. Performance of IADMM with Different Parameters

In IADMM (Algorithm 1), there are a few parameters. In this subsection, we give numerical test results to show how different values of some parameters affect the performance of IADMM and suggest some choices.

We focus on Example 1 in various scenarios. We take $m = 15$, $n = 20$, $l = 0$, $u = 10$, and $\sigma_{\mathbf{v}} = 0.01$, 0.1, 0.2, 0.3, 0.4, 0.5. Note that $\sigma_{\mathbf{x}^*} = \sqrt{((u - l + 1)^2 - 1)/12} = \sqrt{10}$. For each $\sigma_{\mathbf{v}}$, we generate 100 random instances of Example 1 for each scenario.

We take the initial penalty parameter $\lambda^{(0)} = \alpha\lambda^*$ with $\alpha = 0.2, 0.5, 1, 2$, where $\lambda^* = \sigma_{\mathbf{v}}/\sigma_{\mathbf{x}^*}$ (see (21)). We consider both the fixed and varying λ strategies by taking $\tau = 1, 1.05$. We take $q = 2$ and $K = 200$.

In our tests, we record the experimental probability that the first iterate $\mathbf{z}^{(1)}$ is equal to the optimal solution $\mathbf{x}^{\text{ILS}}$ (denoted by $\Pr(\mathbf{z}^{(1)} = \mathbf{x}^{\text{ILS}})$), the experimental probability that the output $\mathbf{x}^{\text{IA}}$ is equal to $\mathbf{x}^{\text{ILS}}$ (denoted by $\Pr(\mathbf{x}^{\text{IA}} = \mathbf{x}^{\text{ILS}})$), and the average number of iterations over 100 tries for each scenario. The optimal solution $\mathbf{x}^{\text{ILS}}$ to the UBILS problem (1) is found by MILES. To see how $\mathbf{z}^{(1)}$ and $\mathbf{x}^{\text{IA}}$ are close to the $\mathbf{x}^{\text{ILS}}$, we also compute the average errors over the 100 instances:

$$\text{average error for } \mathbf{z}^{(1)} = \frac{1}{100}\sum_{i=1}^{100} \frac{\|\mathbf{z}_i^{(1)} - \mathbf{x}_i^{\text{ILS}}\|_2}{\|\mathbf{x}_i^{\text{ILS}}\|_2},$$

$$\text{average error for } \mathbf{x}_i^{\text{IA}} = \frac{1}{100}\sum_{i=1}^{100} \frac{\|\mathbf{x}_i^{\text{IA}} - \mathbf{x}_i^{\text{ILS}}\|_2}{\|\mathbf{x}_i^{\text{ILS}}\|_2}.$$

The experimental results are shown in Table 1. Note that bold values in tables of this paper represent the best results.

In the following we make some comments on Table 1.

(1) The performance of IADMM drops as the noise standard deviation $\sigma_{\mathbf{v}}$ increases, and when $\sigma_{\mathbf{v}} = 0.01, 0.1$, IADMM can usually give the optimal solution in only one iteration. In this case, the algorithm terminates in the second iteration.

Table 1. Performance of IADMM ($K = 200, q = 2$, $\lambda^{(0)} = \alpha\lambda^*$)

$\sigma_{\mathbf{v}}$		$\tau = 1.05$				$\tau = 1$		
		α=0.2	α=0.5	α=1	α = 2	α=0.5	α=1	α=2
0.01	$\Pr(\mathbf{z}^{(1)} = \mathbf{x}^{\text{ILS}})$	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	average error for $\mathbf{z}^{(1)}$	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	$\Pr(\mathbf{x}^{\text{IA}} = \mathbf{x}^{\text{ILS}})$	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	average error for $\mathbf{x}^{(\text{IA})}$	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	# of iterations	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**
0.1	$\Pr(\mathbf{z}^{(1)} = \mathbf{x}^{\text{ILS}})$	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	average error for $\mathbf{z}^{(1)}$	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	$\Pr(\mathbf{x}^{\text{IA}} = \mathbf{x}^{\text{ILS}})$	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
	average error for $\mathbf{x}^{(\text{IA})}$	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	# of iterations	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**
0.2	$\Pr(\mathbf{z}^{(1)} = \mathbf{x}^{\text{ILS}})$	0.87	**0.99**	**0.99**	0.97	**0.99**	**0.99**	0.97
	average error for $\mathbf{z}^{(1)}$	0.1021	**0.0027**	0.0044	0.0150	**0.0027**	0.0044	0.0150
	$\Pr(\mathbf{x}^{\text{IA}} = \mathbf{x}^{\text{ILS}})$	**1.00**	**1.00**	0.99	0.97	**1.00**	**1.00**	0.99
	average error for $\mathbf{x}^{(\text{IA})}$	**0**	**0**	0.0039	0.0148	**0**	**0**	0.0056
	# of iterations	4.21	**2.20**	2.70	2.67	2.24	2.34	2.52
0.3	$\Pr(\mathbf{z}^{(1)} = \mathbf{x}^{\text{ILS}})$	0.40	0.75	**0.87**	0.66	0.75	**0.87**	0.66
	average error for $\mathbf{z}^{(1)}$	0.4074	0.1292	**0.0515**	0.1240	0.1292	**0.0515**	0.1240
	$\Pr(\mathbf{x}^{\text{IA}} = \mathbf{x}^{\text{ILS}})$	0.90	0.90	**0.93**	0.76	0.83	**0.93**	0.84
	average error for $\mathbf{x}^{(\text{IA})}$	0.0434	0.0391	**0.0309**	0.0846	0.0756	0.0317	0.0618
	# of iterations	27.34	11.41	**5.51**	7.02	39.29	17.01	7.02
0.4	$\Pr(\mathbf{z}^{(1)} = \mathbf{x}^{\text{ILS}})$	0.10	0.41	**0.54**	0.26	0.41	**0.54**	0.26
	average error for $\mathbf{z}^{(1)}$	0.5390	0.2633	**0.1737**	0.2410	0.2633	**0.1737**	0.2410
	$\Pr(\mathbf{x}^{\text{IA}} = \mathbf{x}^{\text{ILS}})$	0.73	**0.74**	0.72	0.37	0.46	0.73	0.43
	average error for $\mathbf{x}^{(\text{IA})}$	**0.0888**	**0.0888**	0.0905	0.2011	0.2269	0.0975	0.1889
	# of iterations	58.67	29.92	14.49	**9.90**	111.74	46.61	23.56
0.5	$\Pr(\mathbf{z}^{(1)} = \mathbf{x}^{\text{ILS}})$	0.02	0.17	**0.29**	0.14	0.17	**0.29**	0.14
	average error for $\mathbf{z}^{(1)}$	0.5421	0.3368	**0.2423**	0.2650	0.3368	**0.2423**	0.2650
	$\Pr(\mathbf{x}^{\text{IA}} = \mathbf{x}^{\text{ILS}})$	**0.51**	0.48	0.45	0.23	0.25	0.42	0.28
	average error for $\mathbf{x}^{(\text{IA})}$	**0.1555**	0.1704	0.1722	0.2401	0.2756	0.1862	0.2249
	# of iterations	77.07	43.69	18.26	**9.51**	152.15	70.54	28.47

(2) When $\alpha = 1$, $\Pr(\mathbf{z}^{(1)} = \mathbf{x}^{\text{ILS}})$ is the highest for each $\sigma_{\mathbf{v}}$ and the average error for $\mathbf{z}^{(1)}$ is the smallest for all values of σ_v except $\sigma_v = 0.2$. This verifies our argument for the choice of $\lambda^{(0)}$ given in Section 3.2.

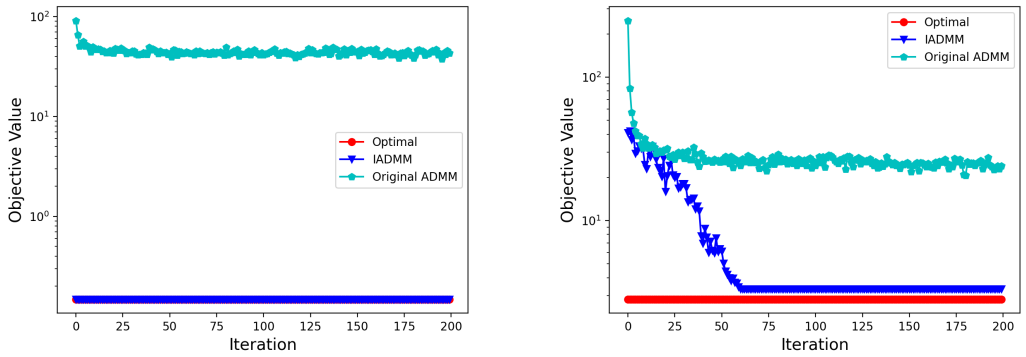
(3) Typically, when $\Pr(\mathbf{z}^{(1)} = \mathbf{x}^{\text{ILS}})$ is large/small, the average error for $\mathbf{z}^{(1)}$ is small/large. This is also true for $\mathbf{x}^{\text{IA}}$.

(4) In the varying λ strategy, larger α leads to faster convergence but worse results, especially when $\sigma_{\mathbf{v}}$ is large. The results suggest that $\alpha = 1$ is usually a good choice. For a large $\sigma_{\mathbf{v}}$, smaller α may give better results.

(5) For the fixed λ strategy, the results indicate that $\alpha = 1$ is a good choice too. When α moves away from 1, $\Pr(\mathbf{x}^{\text{IA}} = \mathbf{x}^{\text{ILS}})$ drops significantly for large $\sigma_{\mathbf{v}}$. Thus, if λ^* is unknown, using the fixed strategy may be problematic.

(6) Comparing the two strategies for λ with the same α, we see the fixed one typically needs more iterations (for all α) and gives worse results (for $\alpha = 0.5, 1$). In our tests, we noticed that for the fixed strategy, the algorithm sometimes failed to converge (e.g., when $\sigma = 0.5$, 32% instances do not stop within 200 iterations.), while for the varying strategy with different α, the algorithm always converged for all instances. Thus, these test results favor the varying strategy.

In our numerical experiments, we also chose different values for the parameters τ and q to see how they affect the performance of IADMM; see [31, Sec. 5.2.1] for details. We observed that in general: (1) If λ increases more slowly over iterations (i.e., choose smaller τ and larger q), often there is a higher probability of obtaining the optimal solution $\mathbf{x}^{\text{ILS}}$ but more iterations are needed. (2) The performance of IADMM is less sensitive on τ and q than on α and changing τ and q hardly influences the results



(a) $\sigma = 0.1$ (b) $\sigma = 0.5$

Figure 1. Performance of IADMM and the original ADMM

when $\sigma_{\mathbf{v}}$ is small or moderate. (3) There is no fixed choice for the parameters that can work well for all instances. To achieve high probability of obtaining the optimal solution, one can fine-tune these parameters for different types of instances.

5.3. Comparisons of IADMM and the Original ADMM

In this subsection we do numerical tests to compare IADMM with the original ADMM algorithm in [40] for solving the UBILS problem.

The performances of the two algorithms are evaluated in terms of average objective values over iterations. We take $\sigma_{\mathbf{v}} = 0.1, 0.5$ and for each $\sigma_{\mathbf{v}}$ we generate 100 instances of Example 1 with $m = 15$, $n = 20$, $l = 0$, and $u = 10$. In the original ADMM algorithm (c.f. (16)) the penalty parameter ρ (which was replaced by $2\lambda^2$ in IADMM) is a fixed constant and in our tests we set $\rho = 2\lambda^{*2}$. For IADMM, we set $K = 200$, $\lambda^{(0)} = \lambda^*$, $\tau = 1.05$ and $q = 2$. The results of average objective value over the 100 instances versus 200 iterations are displayed in Figure 1. For an instance if an algorithm stops before iteration 200, we keep its current objective value for the remaining iterations. For comparisons, we also plot the average optimal objective value, which is obtained via computing the optimal solutions of the 100 UBILS instances by MILES. Note that this is a constant.

Figure 1 shows that IADMM is much better than the original ADMM. In fact, for $\sigma_{\mathbf{v}} = 0.1, 0.5$, IADMM gives the optimal solutions for 100% and 45% instances, respectively, while the original ADMM algorithm for mere 8% and 3% instances, respectively. Since IADMM gives the optimal solution for all instances in only one iteration for $\sigma_{\mathbf{v}} = 0.1$, the blue line and the red line coincide in Figure 1(a) making the red line invisible. From Figure 1(b) we see when the noise is large, IADMM takes more iterations before convergence, but can eventually give solutions with much smaller objective value than the original ADMM.

5.4. Comparisons of IADMM-DTS and DTS

In Section 4 we proposed to incorporate IADMM to DTS, leading to IADMM-DTS. IADMM is used to find an initial search radius for DTS and to find lower bounds to help DTS to prune the search tree. In this subsection, we do numerical tests to

compare IADMM-DTS and DTS. To see the different effects of the search radius and the lower bounds, we also compare the two algorithms with IADMM-DTS$_0$, which is IADMM-DTS without computing lower bounds. The algorithms used in this and next subsections were implemented in MATLAB, and MATLAB Coder toolbox was employed to generate C functions, because the commercial solvers to be compared in the next subsection were implemented in C/C++.

In IADMM-DTS, when IADMM is called to find an initial search radius for DTS, the parameters are taken as follows (see Algorithm 1): $\lambda^{(0)} = 0.2\lambda^*, q = 2, \tau = 1.1, K = 100$; when it is called to compute the lower bound T_{n+1} (see (25)), $\lambda^{(0)} = 0.02\lambda^*, q = 2, \tau = 1.5, K = 20$; and when it is called to compute the lower bounds T_n and T_{n-1} (see (27)), $\lambda^{(0)} = 0.05\lambda^*, q = 1, \tau = 2, K = 5$. The above choices of the parameters in IADMM and the levels at which lower bounds are computed will be kept unchanged when we compare IADMM-DTS with three commercial solvers in the next subsection.

In the test we take $u = 10, 20$ and $\sigma_{\mathbf{v}} = 0.01, 0.1, 0.5$, and for each scenario, we generate 100 instances of Example 1 with $m = 15$, $n = 20$, and $l = 0$. Table 2 summarize the minimum, maximum, median and average running time in seconds of the three solvers for the same 100 instances.

Table 2. Time comparison of DTS, IADMM-DTS$_0$ and IADMM-DTS

u	$\sigma_{\mathbf{v}}$	Solvers	Min	Max	Median	Average
10	0.01	DTS	0.0048	4.1596	0.1362	0.2433
		IADMM-DTS$_0$	**0.0026**	0.0508	**0.0160**	**0.0175**
		IADMM-DTS	0.0143	**0.0268**	0.0206	0.0203
	0.1	DTS	0.0233	3.9949	0.1647	0.2619
		IADMM-DTS$_0$	**0.0062**	0.1242	0.0410	0.0431
		IADMM-DTS	0.0137	**0.0298**	**0.0207**	**0.0204**
	0.5	DTS	**0.0114**	4.6039	0.3248	0.4652
		IADMM-DTS$_0$	0.0418	0.8619	**0.2735**	**0.2955**
		IADMM-DTS	0.0506	**0.7270**	0.2919	0.3097
20	0.01	DTS	0.2187	161.9328	2.7011	5.3550
		IADMM-DTS$_0$	0.0290	1.0347	0.3104	0.3593
		IADMM-DTS	**0.0273**	**0.1066**	**0.0375**	**0.0381**
	0.1	DTS	0.2961	174.9038	3.0593	6.2585
		IADMM-DTS$_0$	0.0932	2.6411	0.8482	0.8937
		IADMM-DTS	**0.0264**	**0.0528**	**0.0377**	**0.0374**
	0.5	DTS	1.3706	24.1324	5.3079	6.3041
		IADMM-DTS$_0$	**0.5064**	**10.2416**	**3.9285**	**4.2512**
		IADMM-DTS	0.5609	11.1708	4.2607	4.3266

In the following we make some comments on the table.

(1) DTS is slower than two other algorithms that incorporate IADMM in terms of the four running time measures, except the minimal running time for the case that $u = 10$ and $\sigma_{\mathbf{v}} = 0.5$. When $\sigma_{\mathbf{v}}$ is small (i.e., $\sigma_{\mathbf{v}} = 0.01$) or moderate (i.e., $\sigma_{\mathbf{v}} = 0.1$), the improvement brought by incorporating IADMM into DST is significant. This is not surprising, as IADMM can often find the optimal solution in one iteration when $\sigma_{\mathbf{v}}$ is small or even moderate, making IADMM-DTS faster than DTS.

(2) IADMM-DTS$_0$ takes (much) less running time than DTS, indicating that the initial search radius given by IADMM can significantly improve the search efficiency. For small and moderate $\sigma_{\mathbf{v}}$, IADMM-DTS can reduce the running time

further and for the larger box (i.e., $u = 20$) the overall improvement is more significant. Thus applying the IADMM-based lower bound technique can (significantly) improve the search efficiency too. This is understandable because a larger constraint box implies a larger search tree and using lower bounds to prune the search tree can become more effective. When $\sigma_{\mathbf{v}}$ is large (i.e., $\sigma_{\mathbf{v}} = 0.5$), IADMM-DTS often takes a little more running time than IADMM-DTS$_0$. This is because the lower bounds are not tight in this case, implying that they are not effective in pruning the search tree.

(3) When the constraint box becomes bigger, i.e., u increases from 10 to 20, the running times of all the three algorithms increase. This is because the search region becomes larger. The maximal running time of DTS increases more dramatically than that of the two other algorithms.

5.5. Comparisons of Solvers for UBILS problems

In the subsection, we compare IADMM-DTS with three well-known commercial optimization solvers CPLEX [25], Gurobi [21], and MOSEK [35] for Example 2.

The solver IAMDD-DTS finds an optimal solution to the UBILS problem (1), but the three commercial solvers may not. Thus, the solutions obtained by the four solvers may be different. In communications, the typical measure of the badness of an estimate is the bit error rate (BER). Suppose that $\hat{\mathbf{x}}$ is an estimate of $\mathbf{x}^* \in \mathbb{Z}^{2N_t}$ obtained by a solver. By comparing the binary representation of each element of $\hat{\mathbf{x}}$ with that of $\mathbf{x}^*$, whose each element can be represented by k bits, one can find the total number of wrong bits in the binary representation of $\hat{\mathbf{x}}$, which, divided by $2N_tk$, leads to the BER.

In both Examples 2(a) and 2(b), we set SNR = 20; $(N_r, N_t) = (8, 12), (12, 16)$; constellation = 4QAM (i.e., $k = 1$ in (31)), 16QAM (i.e., $k = 2$ in (31)). For each scenario, we randomly generated 100 instances. Tables 3–6 display the minimum, maximum, median and average running times in seconds, the average BER and the average residual (i.e., $\|\mathbf{y} - \mathbf{A}\hat{\mathbf{x}}\|_2$ with $\hat{\mathbf{x}}$ being a computed solution) of each solver for the 100 instances of each scenario, respectively.

From the results, we give some observations and comments as follows:

(1) For the 4QAM constellation, Tables 3-6 indicates that IADMM-DTS performs best, followed by Gurobi, MOSEK and CPLEX. Sometimes CPLEX can perform much worse than other solvers (see Table 6). For the 4QAM constellation, often the first iterate given by IADMM is the optimal solution. Thus IADMM-DTS is much more efficient than other solvers. Here we would like to point out that in communications many integer estimation problems need to be solved continuously in real-time and the computation resources are limited, thus efficiency of an algorithm is very crucial.

(2) For the 16QAM constellation, Tables 3-6 indicate that Gurobi performs best, followed by IADMM-DTS, CPLEX and MOSEK in terms of running times.

(3) When the constellation changes from 4QAM to 16QAM, CPLEX's running time can decrease significantly (see Table 6), while all other solvers takes more running time. For an enumeration method, such as DTS, when the constraint region becomes larger, typically the running time increases, as the search region becomes larger.

(4) When the dimensions (N_r, N_t) change from $(8, 12)$ to $(12, 16)$, Tables 3-4 for Example 2(a) and Tables 5-6 for Example 2(b) indicate that for 4QAM all solvers

take more running time, and this is especially true for CPLEX; for 16QAM CPLEX takes a little less running time, Gurobi's running time changes a little bit, while IADMM-DTS and MOSEK still take more running time.

(5) Solving Example 2(b) takes more running time than solving Example 2(a) for all solvers. The former's channel matrices are more ill-conditioned than the latter's channel matrices. Typically an enumeration method such as DTS takes more running time to solve the former than to solve the latter.

(6) For Example 2(a), Tables 3-4 show that all solutions are equal to the corresponding true parameter vector $\mathbf{x}^*$. Here we would like to point out that an optimal solution may not be equal to $\mathbf{x}^*$ even if it is unique and this is especially true when the noise in the model is large. For Example 2(b), Tables 5-6 show that the BER of each solution obtained by IADMM-DTS is still 0, but the BER of a solution obtained by any of three commercial solvers may not be 0. This is because a solution obtained by those solvers may not be optimal, which is confirmed by the results of the average residual in Tables 5-6.

Table 3. Time, average BER and average residual for Example 2(a), $N_r = 8$, $N_t = 12$, SNR = 20

QAM	Solver	Min	Max	Median	Average	BER	Residual
4	IADMM-DTS	**0.0008**	**0.0037**	**0.0012**	**0.0013**	0	0.0806
	CPLEX	0.1702	0.4601	0.1905	0.1987	0	0.0806
	Gurobi	0.0030	0.0107	0.0064	0.0064	0	0.0806
	MOSEK	0.0264	0.0755	0.0311	0.0335	0	0.0806
16	IADMM-DTS	0.0130	0.0352	0.0238	0.0232	0	0.2016
	CPLEX	0.0721	0.2229	0.1181	0.1147	0	0.2016
	Gurobi	**0.0023**	**0.0309**	**0.0115**	**0.0128**	0	0.2016
	MOSEK	0.0373	0.3560	0.1676	0.1640	0	0.2016

Table 4. Time, average BER and average residual for Example 2(a), $N_r = 12$, $N_t = 16$, SNR = 20

QAM	Solver	Min	Max	Median	Average	BER	Residual
4	IADMM-DTS	**0.0017**	**0.0112**	**0.0025**	**0.0025**	0	0.1231
	CPLEX	0.3196	10.1165	1.1245	1.3677	0	0.1231
	Gurobi	0.0048	0.0237	0.0091	0.0092	0	0.1231
	MOSEK	0.0477	0.1697	0.0646	0.0654	0	0.1231
16	IADMM-DTS	0.0385	0.1293	0.0646	0.0669	0	0.3077
	CPLEX	0.0699	0.1789	0.0887	0.0973	0	0.3077
	Gurobi	**0.0030**	**0.0454**	**0.0107**	**0.0124**	0	0.3077
	MOSEK	0.0589	0.4346	0.1844	0.1797	0	0.3077

6. Summary and future work

A UBILS problem is a challenging problem to solve. We have proposed a heuristic integer-constrained ADMM algorithm, named IADMM. Numerical tests showed that IADMM can produce much more accurate results than the original ADMM method. To obtain the optimal solution we have proposed to incorporate IADMM into the DTS enumeration method, leading to the combined algorithm named IADMM-DTS.

Table 5. Time, average BER and average residual for Example 2(b), $N_r = 8$, $N_t = 12$, SNR = 20

QAM	Solver	Min	Max	Median	Average	BER	Residual
4	IADMM-DTS	**0.0008**	**0.0030**	**0.0012**	**0.0012**	0	0.0799
	CPLEX	0.1315	32.2897	7.6065	9.1637	0	0.0799
	Gurobi	0.0041	0.0259	0.0058	0.0085	0	0.0799
	MOSEK	0.0307	0.2116	0.0448	0.0619	2.500e-3	0.0888
16	IADMM-DTS	0.0204	0.3496	0.0728	0.0918	0	0.1998
	CPLEX	0.0779	0.4252	0.1199	0.1250	1.392e-1	0.8846
	Gurobi	**0.0040**	**0.2414**	**0.0219**	**0.0290**	1.296e-1	0.8303
	MOSEK	0.0605	18.3002	0.3654	0.9483	1.042e-2	0.2159

Table 6. Time, average BER and average residual for Example 2(b), $N_r = 12$, $N_t = 16$, SNR = 20

QAM	Solver	Min	Max	Median	Average	BER	Residual
4	IADMM-DTS	**0.0022**	**0.0165**	**0.0025**	**0.0028**	0	0.1225
	CPLEX	2.2224	412.0457	88.9543	114.0970	0	0.1225
	Gurobi	0.0048	0.0264	0.0065	0.0086	0	0.1225
	MOSEK	0.0491	0.2419	0.0663	0.0745	0	0.1225
16	IADMM-DTS	0.0570	0.9064	0.2491	0.2647	0	0.3063
	CPLEX	0.0749	0.2786	0.1027	0.1135	7.047e-2	1.0868
	Gurobi	**0.0034**	**0.2767**	**0.0206**	**0.0310**	6.000e-2	1.2426
	MOSEK	0.0806	7.3718	0.5521	1.0386	0	0.3063

Specifically, IADMM-DTS uses IADMM to find an initial search radius to make the search region smaller and some lower bounds to make the size of the search tree smaller. The advantages of IADMM-DTS over the commercial solvers CPLEX, Gurobi and MOSEK were demonstrated through a practical application.

There have been several convergence results for the ADMM methods for convex problems. Extending the convergence theory to integer problems will be interesting, although it looks very challenging. The proposed IADMM algorithm involves some parameters to choose. In applying the algorithm to solve some practical problems, one may need to tune the parameters $\lambda^{(0)}$, τ and q to get better performance. Probably deep neural network techniques can be used to determine these parameters for specific applications. The numerical comparisons of IADMM-DTS with commercial solvers in this paper are limited to MIMO communications. When new applications of UBILS problems are discovered, new comparisons will be needed. The IADMM algorithm deals with the box constraint, and we would like to extend it to deal with some other constraints. It is now common for a computer to have a multiple core processor. The three commercial solvers in our comparisons use multiple cores. We intend to modify the IADMM-DTS algorithm to fully take advantage of multiple cores to speed it up.

Funding

The research of this work was supported by NSERC of Canada grant RGPIN-2017-05138.

Disclosure statement

The authors report there are no competing interests to declare

Data Availability Statements

All data generated or analysed during this study are included in this article and [31]

Acknowledgement

We are very grateful to the referees for their detailed valuable comments and suggestions which helped us to improve the paper significantly.

References

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

[2] 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.

[3] L. Bai, J. Choi, and Q. Yu. *Low Complexity MIMO Receivers*. Springer, Cham, Germany, 2014.

[4] D.P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, Belmont, Mass., 2014.

[5] J. Boutros, N. Gresset, L. Brunel, and M. Fossorier. Soft-input soft-output lattice sphere decoder for linear channels. In *IEEE Global Telecommunications Conference*, volume 3, pages 1583–1587, 2003.

[6] S. Boyd, N. Parikh, E. Chu, B. Peleato, and E. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3:1–122, 2011.

[7] Christoph Buchheim, Alberto Caprara, and Andrea Lodi. An effective branch-and-bound algorithm for convex quadratic integer programming. *Mathematical programming*, 135(1):369–395, 2012.

[8] X.-W. Chang. MILES: MATLAB package for solving Mixed Integer LEast Squares problems, 2023, https://www.cs.mcgill.ca/∼chang/miles.php.

[9] X.-W. Chang and Q. Han. Solving box-constrained integer least squares problems. *IEEE Trans. Wirel. Commun.*, 7(1):277–287, 2008.

[10] X.-W. Chang, J. Wen, and X. Xie. Effects of the LLL reduction on the success probability of the babai point and on the complexity of sphere decoding. *IEEE Trans. Inf. Theory*, 59(8):4915–4926, 2013.

[11] X.-W. Chang and X. Yang. An efficient tree search decoder with column reordering for underdetermined MIMO systems. In *IEEE Global Telecommunications Conference*, pages 4375–4379, 2007.

[12] X.-W. Chang, X. Yang, T. Le-Ngoc, and P. Wang. Partial regularisation approach for detection problems in underdetermined linear systems. *IET Communications*, 3(1):17–24, 2009.

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

[14] Chen-Nee Chuah, D.N.C. Tse, J.M. Kahn, and R.A. Valenzuela. Capacity scaling in

MIMO wireless systems under correlated fading. *IEEE Trans. Inf. Theory*, 48(3):637–650, 2002.

[15] T. Cui and C. Tellambura. An efficient generalized sphere decoder for rank-deficient MIMO systems. In *IEEE 60th Vehicular Technology Conference*, volume 5, pages 3689–3693, 2004.

[16] M.O. Damen, K. Abed-Meraim, and J. Belfiore. Generalized sphere decoder for asymmetrical space-time communication architecture. *Electronics letters*, 36(2):166–167, 2000.

[17] M.O. Damen, H. El Gamal, and G. Caire. On maximum-likelihood detection and the search for the closest lattice point. *IEEE Trans. Inf. Theory*, 49(10):2389–2402, 2003.

[18] S. Diamond, R. Takapoui, and S. Boyd. A general system for heuristic minimization of convex functions over non-convex sets. *Optimization Methods and Software*, 33(1):165–193, 2018.

[19] 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.

[20] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins, Baltimore, 4 edition, 2013.

[21] Gurobi. Gurobi Optimizer, 9.1, https://www.gurobi.com.

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

[23] A. Hassibi and S. Boyd. Integer parameter estimation in linear models with applications to GPS. *IEEE Trans. Signal Process.*, 46(11):2938–2952, 1998.

[24] B. Hassibi and H. Vikalo. On the sphere-decoding algorithm I. Expected complexity. *IEEE Trans. Signal Process.*, 53(8):2806–2818, 2005.

[25] IBM. ILOG CPLEX Optimization Studio, v12.10.0, https://www.ibm.com/ca-en/products/ilog-cplex-optimization-studio.

[26] P. Karamanakos, T. Geyer, and R. Kennel. A computationally efficient model predictive control strategy for linear systems with integer inputs. *IEEE Trans. Control Syst. Technol.*, 24(4):1463–1471, July 2016.

[27] V. Kühn. *Wireless Communications over MIMO Channels: Applications to CDMA and Multiple Antenna Systems*. John Wiley & Sons, England, 2006.

[28] J.K. Kuusinen, J. Sorsa, and M.L. Siikonen. The elevator trip origin-destination matrix estimation problem. *Transp. Sci.*, 49(3):559–576, 2014.

[29] E.G. Larsson. MIMO detection methods: How they work. *IEEE Signal Process. Mag.*, 26(3):91–95, 2009.

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

[31] T. Ma. An admm method for underdetermined box-constrained integer least squares problems. Master's thesis, School of Computer Science, McGill University, Montreal, Quebec, Canada, May 2021.

[32] J.M. Mendel. *Lessons in Estimation Theory for Signal Processing, Communications and Control*. Prentice Hall, Englewood Cliffs, NJ, 1995.

[33] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: A Cryptographic Perspective*. Springer Science & Business Media, New York, 2012.

[34] D. Micciancio and M. Walter. Fast lattice point enumeration with minimal overhead. In *Proceedings of the 2015 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 276–294, 2015.

[35] MOSEK. MOSEK Optimization Suite Release 9.2.32, https://www.mosek.com.

[36] W.H. Mow. Maximum likelihood sequence estimation from the lattice viewpoint. *IEEE Trans. Inf. Theory*, 40(5):1594–1600, 1994.

[37] C. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. *Math. Program.*, 66:181–191, 1994.

[38] Hyundong Shin, Moe Z. Win, and Marco Chiani. Asymptotic statistics of mutual informa-

tion for doubly correlated MIMO channels. *IEEE Trans. Wirel. Commun.*, 7(2):562–573, 2008.

[39] N.M.B. Souto and H.A. Lopes. Efficient recovery algorithm for discrete valued sparse signals using an ADMM approach. *IEEE Access*, 5:19562–19569, 2017.

[40] R. Takapoui, N. Moehle, S. Boyd, and A. Bemporad. A simple effective heuristic for embedded mixed-integer quadratic programming. *Int. J. Control*, 93(1):2–12, 2020.

[41] P.J.G. Teunissen. The least-squares ambiguity decorrelation adjustment: a method for fast GPS ambiguity estitmation. *J. of Geodesy*, 70(1-2):65–82, 1995.

[42] P. van Emde Boas. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Technical Report Rep. 81-04, Mathematics Institute, Amsterdam, The Netherlands, 1981.

[43] Z. Xu, M. Figueiredo, and T. Goldstein. Adaptive admm with spectral penalty parameter selection. In *Artificial Intelligence and Statistics*, pages 718–727. PMLR, 2017.

[44] S. Yang and L. Hanzo. Fifty years of mimo detection: The road to large-scale mimos. *IEEE Commun. Surv. Tutor.*, 17(4):1941–1988, 2015.