AN ALGORITHM FOR INTEGER LEAST-SQUARES WITH EQUALITY, SPARSITY AND RANK CONSTRAINTS

ARUN AYYAR∗ AND NIRAV P. BHATT†

Abstract. In this work, we deal with rank-constrained integer least-squares optimization problems arising in low-rank matrix factorization related applications. We propose a solution for constrained integer least-squares problem subject to equality, sparsity, and rank constraints. The algorithm combines the Fincke-Pohst enumeration (or sphere decoding algorithm) with rank constraints and sparse solutions of Diophantine equations to arrive at an optimal solution. The proposed approach consists of two steps as follows: (i) find the solution set for Diophantine equations arising from the linear and sparsity constraints, (ii) find the matrix which minimizes the integer least-squares objective and satisfying the rank constraints using the solution set obtained in the step 1. The proposed algorithm is illustrated using a simple example. Then, we perform experiments to study the computational aspects of different steps of the proposed algorithm.

Key words. Integer Least-squares, Integer Programming, Nonconvex Programming, Diophantine equations, Rank Constraints, Fincke-Pohst enumeration, Sparsity, Equality constraints

1. Introduction. We consider constrained integer least-squares problems of the following form:

\[
\begin{align*}
\text{minimize} & \quad \|Y - GX\|_2^2 \\
\text{subject to} & \quad X(i, j) \in S \subset \mathbb{Z}, \forall i = 1, \ldots, N, j = 1, \ldots, L \\
& \quad AX^T = 0 \\
& \quad \text{rank}(X) = N \\
& \quad \|X^T(:, i)\|_0 \leq K, i = 1, \ldots, N
\end{align*}
\]

where \(Y \in \mathbb{R}^{M \times L}, G \in \mathbb{R}^{M \times N}, X \in \mathbb{S}^{N \times L} \) with \(S \subset \mathbb{Z}\), and \(A \in \mathbb{Z}^{P \times L}\). The equality constraints \(AX^T = 0\) are imposed on the rows of \(X\). \(\| \cdot \|_2^2\) denotes the square of Frobenius norm. \(\| \cdot \|_0\) is the zeroth norm of a given vector. The zeroth norm of a vector indicates the number of non-zero elements in a vector. The main motivation for investigating Problem 1 is to solve subproblem arising in low-rank matrix factorization problems with integer constraints [2, 21, 22, 25]. For example, let us consider a matrix factorization problem in which a data matrix \(Y\) is decomposed into an integer matrix \((X)\) and a non-negative matrix \((G)\) with the linear equality, rank and sparsity constraints as follows [20, 31]:

\[
Y \approx GX + E
\]
where \( E \in \mathbb{R}^{M \times L} \) is an error matrix. The factorization can be achieved by minimizing a square of Frobenius norm of the error matrix as follows:

\[
\begin{align*}
\text{minimize} & \quad \|Y - GX\|^2 \\
\text{subject to} & \quad G \succeq 0 \\
& \quad X(i, j) \in S \subset \mathbb{Z}, \forall i = 1, \ldots, N, j = 1, \ldots, L \\
& \quad AX^T = 0 \\
& \quad \text{rank}(X) = N \\
& \quad \|X^T(:, i)\|_0 \leq K, i = 1, \ldots, N
\end{align*}
\]

where \( \succeq 0 \) indicates each element of the matrix \( G \) is greater or equal to zero. The objective function in Eq. (3) is a non-convex function, and it can be solved using alternating least-squares (ALS) framework. Two subproblems can be written in the ALS framework as follows.

- For given \( X \)

\[
\begin{align*}
\text{minimize} & \quad \|Y - GX\|^2 \\
\text{subject to} & \quad G \succeq 0
\end{align*}
\]

- For given \( G \)

\[
\begin{align*}
\text{minimize} & \quad \|Y - GX\|^2 \\
\text{subject to} & \quad X(i, j) \in \mathbb{Z}, \forall i = 1, \ldots, N, j = 1, \ldots, L \\
& \quad AX^T = 0 \\
& \quad \text{rank}(X) = N \\
& \quad \|X^T(:, i)\|_0 \leq K, i = 1, \ldots, N
\end{align*}
\]

The optimization problem 5 is a constrained integer least-squares (CILS) problem which is an NP-hard problem [29]. An efficient algorithm needs to be investigated for solving this problem. In this work, we will propose an exact algorithm to solve Problem 5.

### 1.1. Related Work in Literature

The unconstrained integer least-squares (ILS) problem is known as closest point problem in the lattice theory literature [1]. The ILS problem also arises in several fields such as communications [18], global navigation systems[30], systems biology [3] etc. Also, the ILS problem is a subset of nonlinear integer programming (NIP), or more specific, quadratic integer programming (QIP). Then, methods to solve NIP or QIP can also be applied to solve the ILS problem. Thus, algorithms in the literature are based on approximation, heuristics, and combinations of NIP and QIP for solving the ILS problem [5]. These algorithms consist of two steps: (i) Reduction, and (ii) search. In the first step, the columns of \( G \) are orthogonalized as much as possible using lattice reduction methods such as Lenstra-Lenstra-Lovász (LLL) or Korkine-Zolotareff (KZ) reduction [5, 23]. In the search step, the solution of the transformed ILS problem is obtained using Pohst enumeration (or also known as sphere decoder in communication literature) [14, 17] or Schnorr-Euchner enumeration methods [28]. Further, the constrained ILS problem
with boxed or ellipsoid constraints have been solved in the literature of communication field [11, 12].

On the other hand, there have been efforts for developing efficient methods for mixed integer nonlinear programming (MINLP), quadratic integer programming (QIP) or nonlinear integer programming (NIP) [15, 19]. These methods can also be applied to ILS problems. Several branch-and-bound based algorithms have been developed to solve MINLP, QIP and NIP problems in an efficient manner [4, 6, 8, 16]. A fast branch-and-bound algorithm for minimizing a convex QIP with convex constraints has been proposed [6]. The proposed algorithm by the authors allows to compute tighter lower bounds of the objective function by considering ellipsoidal under-estimators having the same continuous minimizer as the original objective function. The idea of ellipsoidal under-estimators has been extended to non-convex QIP with the box constraints [8]. This approach has been generalized in [9] using the under-estimators with strong rounding properties. The proposed algorithm in [6, 8, 9] can also be applied to ILS problems with boxed constraints in [11, 12].

For non-convex QIP and MINLP problems, different approaches based on semi-definite relaxations have been proposed to obtain tighter lower bounds of the objective function [10, 26]. The authors in [10] proposed an algorithm in branch-and-bound approach by incorporating semi-definite relaxation for unconstrained non-convex MINLP problems. On the other hand, an approach to obtain lower and upper bounds on the objective function convex QIP using semi-definite relaxation is proposed in [26]. The algorithm proposed in [26] leads to a sub-optimal solution of the underlying problem. However, the computational experiments has shown that it provides near-optimal solutions for the larger size problem (n = 1000). Moreover, Li et al. [24] and Saxena et al. [27] have proposed algorithms based on contours and planes cuts for solving MINLP problems with inequality constraints. Recently, a fast branch-and-bound algorithms based on ellipsoidal relaxation for solving non-convex QIP with linear equality constraints has been proposed [7].

Most of the approaches in the literature can handle convex constrains or non-convex constraints like box, ellipsoid, equality, or inequality constraints. However, best of our knowledge, there is no algorithm which can solve Problem 5.

1.2. Contribution. The main contribution of this paper is development of an algorithm for matrix integer-least squares problem in (5) with equality, sparsity and rank constraints by combining techniques from solution of a system of linear Diophantine equations and modified Pohst enumeration (or modified sphere decoder). The proposed algorithm consists of two steps:

1. Find sparse solutions of a set of Diophantine equations
2. Find a solution matrix satisfying the rank constraints by combining solutions from the modified Pohst enumeration(or sphere decoding) algorithm with the solution set obtained in the step 1.

2. Proposed Algorithms to Solve Problem 5. This section proposes a novel algorithm to solve the CILS problem 5. There are three types of constraints in Problem 5: (i) linear equality constraints, (ii) sparsity constraints, and (iii) rank constraint. The entries of $X$ should be such that $X^T$ is in the null space of $A$ and each row of $X$ can have atmost $K$ non-zero elements and the rank of $X$ must be $N$. Further, the rank constraint can be checked for only after all the entries of $X$ have been found. In order to find a solution to Problem 5, we split the problem into two sub-problems based on the constraints.

1. The linear and sparsity constraints in Problem 5 dictate that the rows of $X$
must lie in the null space of $A$ and the number of non-zero elements in the rows of $X$ cannot be more than $K$. Then, the first problem is to find all the vectors that form this search space. Formally, Find the solution set $F$ for

$$\mathcal{F} = \{x \mid x \in S^L, S \subset \mathbb{Z} \& \ Ax = 0_{P \times 1} \& \ |x|_0 \leq K\}$$

A set of $N$ vectors from $\mathcal{F}$ forms the rows of $X$.

2. Consider $X = [x_1, x_2, \cdots, x_L]_{N \times L}$ and $Y = [y_1, y_2, \cdots, y_L]_{M \times L}$. Then, the objective function in (5) can be written as follows:

$$\min_{X \in S^{N \times L}} \|Y - GX\|^2 = \min_{j=1}^{L} \min_{x_j \in S_j} \|y_j - Gx_j\|^2$$

The solution to each term in the summation in (7) is obtained by modified sphere decoding algorithm. Then, the next step is to pick $N$ vectors from $\mathcal{F}$ with help of the solutions for the columns of $X$ using the modified sphere decoding algorithm such that:

$$\text{Find } X \text{ such that } \sum_{j=1}^{L} \min_{x_j \in S_j} \|y_j - Gx_j\|^2 \text{ and rank}(X) = N$$

In summary, Eq. (6) provides a set candidates for the rows of $X$ and Eq. (7) provides the solution set for the columns of $X$, while Eq. (8) finds an optimal solution of Problem 5 by combining the solutions of Eqs. (6)–(7). The following example is used to illustrate each stage of the proposed algorithm in this section.

**Example 1.** Consider the following matrices, and parameters

$$X_a = \begin{bmatrix} 1 & 1 & -1 & -1 & 0 & 0 \\ 0 & -1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 \end{bmatrix} \quad A = \begin{bmatrix} 8 & 2 & 10 & 0 & 12 & 2 \\ 0 & 6 & 9 & 1 & 14 & 5 \\ 2 & 0 & 1 & 1 & 0 & 1 \\ 2 & 1 & 3 & 0 & 4 & 0 \\ 1 \\ 0 \end{bmatrix}$$

$S = \{-1, 0, 1\}$, $K = 4$, and $N = 3$.

2.1. Solution for a System of Linear Diophantine Equations. In this section, we discuss the method to solve the problem described in Eq. (6). To solve $AX^T = 0$ with sparsity constraint, we first transform $A$ in to a upper triangular matrix $H$ using the Hermite Normal form as follows [29].

**Definition 1, Hermite Normal Form.** For every $P \times L$ matrix $A$ with integer entries, there exists a $P \times L$ matrix $H$ with integer entries such that

$$H = UA \quad \text{with} \quad U \in GL_n(\mathbb{Z})$$

where $U$ is a $P \times P$-dimensional unimodular matrix and $H$ is an upper triangular matrix. $H$ is the matrix in Hermite normal form which can be obtained from $A$ by elementary row operations.

Since $U$ is a non-singular matrix in Eq. (10), $Ax = 0$ in Eq. (6) can be written as

$$UAx = 0 \quad \text{and} \quad Hx = 0$$

4
Since $H$ is an upper triangular matrix, Eq. (12) can be written as

$$
\begin{bmatrix}
  h_{1,1} & h_{1,2} & h_{1,3} & \cdots & h_{1,k} & \cdots & h_{1,L} \\
  0 & h_{2,2} & h_{2,3} & \cdots & h_{2,k} & \cdots & h_{2,L} \\
  0 & 0 & h_{3,4} & \cdots & h_{3,k} & \cdots & h_{3,L} \\
  \vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots \\
  0 & 0 & 0 & \cdots & h_{P,k} & \cdots & h_{P,L}
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
\vdots \\
x_L
\end{bmatrix}
\begin{bmatrix}
0 \\
0 \\
0 \\
\vdots \\
0
\end{bmatrix} =
$$

Eq. (13) leads to the following set of equations

$$
h_{1,1}x_1 + h_{1,2}x_2 + h_{1,3}x_3 + \cdots + h_{1,L}x_L = 0
$$

$$
h_{2,2}x_2 + h_{2,3}x_3 + \cdots + h_{2,L}x_L = 0
$$

$$
\vdots
$$

$$
h_{P,k}x_k + \cdots + h_{P,L}x_L = 0
$$

The upper triangular structure of $H$ can be exploited in finding solutions for Eq. (13). Before explaining the methodology of finding the solution for a system of linear equations with constraints, we illustrate a method to find all solutions of a single linear Diophantine equation with $L_0$ norm constraint. The set of equations in (14) can be written as follows:

$$
h_i^Tx = 0, \quad i = 1, \ldots, P
$$

where $x \in S^{N_g}$ with $N_g \leq M$. Then, the objective is to

$$
\text{Find all } x \in S^{N_g} \text{ such that } h_i^Tx = 0 \text{ and } ||x||_0 \leq K
$$

Now, let us divide the variables $x$ in to: (i) pivot variable ($x_p$) and (ii) free variables ($x_f$). The free variables can take all the possible values from $S$. The left most variable can be taken as the pivot variable and the rest variables can be taken as the free variables. Then, the values of $x_p$ in the $i$th equation can be computed as follows

$$
x_{p,i} = -\frac{\sum_{l=1}^{N_g-1} h_{i,l}x_{f,l}}{h_{p,i}}
$$

Then, we need to find all possible solutions for $x_f$. Each element of $x_f$ takes all the values in $S$. For example, there are three elements in $x_f$, i.e., $x_f = [x_f(1), x_f(2), x_f(3)]^T$. Then, the possible values for each variable are as follows: $S_{x_f(1)} = S_{x_f(2)} = S_{x_f(3)} = \{-1, 0, 1\} = S$. Let us define the solution set till the $i$th variable as $S_{x_{f,i}}$. Then, the set containing all the solutions for the first variable, $S_{x_{f,1}} = S$. Then, the set containing all the solutions till the two variables variable $x_{f,2} = [x_f(1), x_f(2)]^T$ can be obtained by a Cartesian product (denoted as $\times$) between $S_{x_{f,1}}$ and the all solution $S_{x_{f,2}}$ as follows

$$
S_{x_{f,2}} = S_{x_{f,1}} \times S_{x_{f,2}} = S \times S = \{(x_1, x_2) \mid x_1 \in S_{x_{f,1}} \text{ and } x_2 \in S_{x_{f,2}}\}
$$

Similarly, the solution set containing all three variables can be obtained by a Cartesian product between the previous stage solution set, $S_{x_{f,2}}$ and the all the possible solution for the third variable $x_f(3)$, $S$ as follows:

$$
S_{x_{f,3}} = S_{x_{f,2}} \times S_{x_f(3)} = S_{x_{f,2}} \times S = \{(x_i) \mid x_i \in S_{x_{f,i}} \forall i = 1, 2, 3\}
$$
Eq. (19) can be interpreted as follows. The solution set for the three variables $S_{x_f,3}$ is a Cartesian product of the solution set for the two variables $S_{x_f,2}$ and the set consisting all the possible values for $x_f(3)$, i.e. $S$. Then, the solution set for the first $l$ variables, $x_{f,l}$, can be expressed as a Cartesian product of the solution set for the $l-1$ variables, $S_{x_f,l-1}$ and the set consisting all possible values for the $l$ element of $x_f$, $x_f(l)$, $S$ as follows:

$$S_{x_f,l} = S_{x_f,l-1} \times S$$

Since we are interested in the solutions that satisfy the sparsity constraint ($L_0$-norm) in Problem (16), this constraint needs to be imposed on the set $S_{x_f,l}$ in Eq. (20) at each stage. To do so, let us define a function $f_s : D \rightarrow B$

$$f_s = \{\forall d \in D \mid \#(i|d_i \neq 0) \leq K\}$$

The function in (21) imposes $L_0$-norm constraints on the set $D$ and find solutions satisfying the constraints. Then, the solution set till the $l$ variables satisfying sparsity constraint ($L_0$-norm), $S_{x_f,l}$, can be obtained as follows:

$$\hat{S}_{x_f,l} = f_s(S_{x_f,l}) = f_s(S_{x_f,l-1} \times S)$$

Then, the complete solution set for Problem (16) can be obtained by applying Eq. (22) till $l = N_g - 1$.

This solution set at any stage for Eq. (16) can be stored using the concept of a rooted tree. Then, this rooted tree can be expanded to add more variables by solving a set of equations in Eq. (13). Next, a methodology to obtain the solution set for a system of linear Diophantine equations with $L_0$-norm constraint is described briefly.

- Reduce the matrix $A$ to its Hermite Normal form.
- Start from the last equation.
- Initialize the solution tree to the root node.
- For the free variables in the present equation, expand the existing solution tree. Check for $L_0$ norm constraint at each node.
- For the pivot variable in the present equation, compute the pivot variable values using Eq. (17) and also if the $L_0$ norm constraint is satisfied.
- Retain only the final leaf nodes that satisfy both - the current equation and the $L_0$ norm constraint.
- Start from the solution tree for the previous equation and repeat the procedure for the next equation.
- The solution tree that remains after applying the method to all equations in Eq. (13) is the final solution set $F$ ans stored as tree $\mathcal{T}$.

The detailed method is given in Algorithm 1. Next, the elements of the algorithms are demonstrated using Example 1.

**Example 1. (continued) A** can be factorized into its Hermite normal form as follows:

$$A = \begin{bmatrix} 2 & 0 & 0 & 2 & -2 & -8 & 10 \\ 0 & 1 & 0 & 1 & 0 & -19 & 21 \\ 0 & 0 & 1 & -1 & 2 & 9 & -10 \\ 0 & 0 & 0 & 0 & 0 & -18 & 18 \end{bmatrix} = \begin{bmatrix} -3 & -1 & 3 & 12 \\ -6 & -2 & 3 & 25 \\ 3 & 1 & -2 & -12 \\ -5 & -2 & 2 & 22 \end{bmatrix} \begin{bmatrix} 8 & 2 & 10 & 0 & 12 & 2 & 0 \\ 4 & 6 & 9 & 1 & 14 & 5 & 2 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 2 & 1 & 3 & 0 & 4 & 0 & 1 \end{bmatrix}$$
Algorithm 1 Algorithm to find solution to set of Diophantine equations with sparsity constraints (Problem 6)

Input: $K, S, A \in \mathbb{Z}^{P \times L}$
Output: The set $F$ in terms of tree $T_{sol}$ containing all the solution vectors $x$ such that $x \in S^L$ and $||x||_0 \leq K$

1: procedure SOLVESYSOLINDIQEON($A, S, K$) 
2: $H = UA$ \hspace{1cm} \triangleright \text{Harmit Normal Form of } A$
3: $T = \{\}$
4: $N_{prev} = 0$
5: $N = \text{number of rows in } H$
6: for $i = N$ to 1 do 
7:  $I_d = \text{Index of First non-zero element in } i^{th} \text{ row of } H \text{ starting from left}$
8:  if $|T| == 0$ then 
9:     Add all elements of $S$ as individual nodes to $T$
10:  else 
11:     $N_{prev} = \text{length of each node in } T$
12:     for $k = 1$ to $I_d - N_{prev} - 1$ do 
13:         $Z = \text{All the leaves in the Tree } T$
14:         $N_z = \text{Number of elements in } Z$
15:         $T = \{\}$ \hspace{1cm} \triangleright \text{Empty the tree}$
16:         for $j = 1$ to $N_z$ do 
17:             $z = Z(j)$ \hspace{1cm} \triangleright \text{Assign } j^{th} \text{ vector in } Z \text{ to } z$
18:             for $l = 1$ to $|S|$ do 
19:                 $z = [z \ S(l)]$ \hspace{1cm} \triangleright \text{Append } l^{th} \text{ element in } S \text{ to } z$
20:                 if $||z||_0 \leq K$ then 
21:                     Add $z$ as node to $T$ \hspace{1cm} \triangleright \text{Add the vector if it satisfies } L_0 \text{ norm constraint}$
22:     $z = Z = \text{All the leaves in the Tree } T$
23:     $N_z = \text{Number of elements in } Z$
24:     $T = \{\}$ \hspace{1cm} \triangleright \text{Empty the tree}$
25:     $h_i = H(i)$ \hspace{1cm} \triangleright \text{$i^{th} \text{ row of } H}$
26:     for $j = 1$ to $N_z$ do 
27:         $z_j = Z(j) \cdot h_i(I_d + 1 : end)/h_i(I_d)$
28:         $z = [z \ z_j]$
29:         if $z_j \in S$ and $||z||_0 \leq K$ then 
30:             Add $z$ as node to $T$
31: return $T$

The system of equations $Ax = 0$ can thus be written as
\[
\begin{align*}
2x_1 + 2x_4 - 2x_5 - 8x_6 + 10x_7 &= 0 \\
x_2 + x_4 - 19x_6 + 21x_7 &= 0 \\
x_3 - x_4 + 2x_5 + 9x_6 - 10x_7 &= 0 \\
-18x_6 + 18x_7 &= 0
\end{align*}
\]

The last equation in (24) is considered to demonstrate building of the solution tree.
The problem to be solved is as follows:

\[(25) \quad -18x_6 + 18x_7 = 0\]

such that \(\|x\|_0 \leq 4\) and \(x \in \{-1, 0, 1\}^2\), where \(x = [x_6 \ x_7]\). We need to find all possible solutions of Eq. (25). In (25), \(x_6\) is the pivot variable and \(x_7\) is the free variable. Hence, \(x_7\) can take all the values from \(S\) while the values of \(x_6\) depend on \(x_7\). We start with an empty tree. At each stage, we consider the values that a particular variable can take. For example, in first stage, \(x_7\) can take all possible values from \(S\). Hence, to the root node, we add 3 nodes one each for the values \(-1, 0,\) and \(1\).

We start with an empty tree. At each stage, we consider the values that a particular variable can take. For example, in first stage, \(x_7\) can take all possible values from \(S\). Hence, to the root node, we add 3 nodes one each for the values \(-1, 0,\) and \(1\). At each subsequent stage, we concatenate the value taken by a variable to the value present at its parent node. In the next stage, we consider the variable \(x_6\). \(x_6\) can be computed using Eq. (17). Then, the solution set for Eq. (25) is shown in Figure. 1 in terms of a rooted tree.

![Fig. 1: Solution tree to Eq. (25)](image)

Then, by applying Algorithm 1, the solution tree for Eq. (24) is given in Fig. 2. The intermediate solutions are also depicted.

![Fig. 2: Solution set \(F\) for Example 1.](image)

There are seven solutions that satisfy \(AX^T = 0\) with \(K = 4\) which are shown in the box and labelled as the final solution in Figure 2. The solution set \(F\) contains all possible rows of \(X\).
2.2. Solution to Constrained Integer least-squares. In the previous section, we outlined the method to find the solution to the sub-problem (6). This section proposes an approach to solve the posed CILS problem 5 using the solution set $\mathcal{F}$. To solve the proposed problem, we first revisit the algorithm to solve of Problem 7 using the modified sphere decoding method. The individual terms within the summation in Eq. (7) can be formulated as the following problem. Given $S \subseteq \mathbb{Z}^N$, $G \in \mathbb{R}^{M \times N}$ and $y \in \mathbb{R}^M$,

$$\min_{x_j \in S} ||y_j - Gx_j||^2.$$

The problem presented in (26) is the classical integer least–squares problem. Since the entries of $x_j$ are restricted to $S \subseteq \mathbb{Z}^N$, the search space is a finite subset of the infinite “rectangular” lattice $\mathbb{Z}^N$. When multiplied by a matrix $G$, $Gx_j$ spans a “skewed lattice”. Hence, given a “skewed lattice” $Gx_j$ and a vector $y_j$, the integer least–squares problem is to find the lattice point closest to $y_j$ in the Euclidean sense. However, in contrast to classical ILS problem (26), it is possible that each element of $x_j$ can belong to a different set in this work, i.e., $x_j(i) \in S_{i,j} \subseteq S$, $\forall i = 1, \ldots, N$. By defining a set $S_j = \{S_{1,j}, S_{2,j}, \ldots, S_{N,j}\}$, Probolen 26 can be written as follows

$$\min_{x_j \in S_j} ||y_j - Gx_j||^2.$$

Then, Problem (27) can be solved by the existing algorithms for solving the classical ILS problem in Eq. (26). We can exploit the nature of the problem by modifying the existing algorithm to include the information of the set $S_j$. This helps us to reduce computational complexity. Hence, we will modify the existing sphere decoding algorithm in the current work [14,18]. Then, the existing sphere decoding algorithm can be modified by changing the bounds on each element $i$ of $x_j(i)$ as presented in Algorithm 2.

They key concept that sphere decoding uses to solve the problem is a parameter $d$ which is the radius of the sphere within which the algorithm will search for $x_j$. Assuming the centre of the sphere to be the given vector $y_j$, the algorithm searches for the lattice points of $Gx_j$ which lies within a sphere of radius $d$. It can be seen that, the closest lattice point inside the sphere will also be the closest lattice point for the whole lattice. If a point is found within the sphere, then the vector is returned, else the radius of the sphere is increased and the search is performed all over again. The details of the sphere decoding algorithm and its complexity can be found in [13].

There is a rank constraint in the sub-problem 8. Since the closest lattice point inside the sphere for each column does not guarantee that $X$ is of rank $N$. The output of algorithm 2 provides all lattice points inside the radius of sphere $d$ which will be used to combine with the set $\mathcal{F}$.

Remark 1. The Sphere decoding returns candidate solutions for each of the columns of $X$. As can be seen from (7), we need to run the sphere decoder algorithm for all the $L$ columns of $X$. The inputs to the sphere decoding for the solution of the jth column of $X$ will be as seen from Eq. (27) are $S_j^N$, $G$ and jth column of $Y$, i.e. $y_j$. The rows of $X$ are chosen from the solution set $\mathcal{F}$ of Eq. (6). Hence, the search space for the sphere decoding need not be chosen as $S^N$ for all the $L$ runs. The search space $S_j$ for the jth run will be decided by the $i$th entry of the solutions to Problem 6. This will help in reducing the overall search space while solving Problem 27 using the modified sphere decoding algorithm.
Algorithm 2 Modified sphere decoding algorithm to solve Problem (27)

Input: $y$, $G$. Search radius $d$, Set $S$
Output: Solution vectors $x$

1: procedure SphereDecoder($S$, $y$, $G$, $d$)
2: \[ G = [Q_1, Q_2] \begin{bmatrix} R \\ 0 \end{bmatrix} \] \quad \triangleright \quad QR \text{ decomposition of } G
3: Compute Hermitian transpose $Q_1^*$ and $Q_2^*$ of the sub-matrices $Q_1$ and $Q_2$, and $z = Q_1^* y$.
4: Set $i = N$, $d_i^2 = d^2 - \|Q_2^* y\|_2^2$, $z_{i,t+1} = z_N$.
5: Set $u_b(x_i) = \min(\max(S_i), \lceil \frac{d_i^{t+2} + z_{i,t+1}}{r_{i,t}} \rceil)$.
6: \[ x_i = \max(\min(S_i), \lceil -\frac{d_i^{t+2} + z_{i,t+1}}{r_{i,t}} \rceil) - 1 \] \quad \triangleright \quad \text{Bounds for } x_i
7: \[ x_i = x_i + 1, \]
8: if $x_i \leq u_b(x_i)$ then, go to 12
9: else go to 9
10: $i = i + 1$ \quad \triangleright \quad \text{Increase } i
11: if $i = N + 1$ then, terminate algorithm
12: else go to 7
13: if $i = 1$ then, go to 14 \quad \triangleright \quad \text{Decrease } i
14: else $i = i - 1$, $z_{i,t+1} = z_i - \sum_{j=1}^N r_{i,j} x_j$, $d_i^2 = d_i^{t+1} - (z_{i+1,t+2} - r_{i+1,t+1} x_{i+1})^2$
15: Solutions found. Save $x$, $d_n^2 - d_1^2 + (z_1 - r_1 x_1)^2$, and go to 7.
16: return $x$

As explained earlier, the candidates for the rows of $X$, $x_R$, are generated from the solutions to Problem 6, i.e. $F$, and the column sets of $X$, $x_C$ are generated by Algorithm 2. Then, the objective is to find the actual entries of $X$ using the solutions to Eqs. (6) and (7) such that the rank constraint is also satisfied. To achieve this objective, $X$ has to be constructed from the solutions of rows and columns candidates, and then check for the rank condition. Let us define the solution set for $x_C$ is $S_C$ with $|S_C| = L$ where $|\cdot|$ indicates the cardinality of a given set. Further, the solution set for $i$th row of $X$ at the computation of the $j$th column, $F_{i,j}$, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, L$, is given by

\begin{equation}
F_{i,j} = \{x_R \mid x_R(j) = x_{C,j}(i), \forall x_R \in F, x_C \in S_C\}
\end{equation}

where $x_R(j)$ is the $j$th element of the row vector $x_R$ and $x_{C,j}(i)$ indicates the $i$th element of the $j$th column in the ordered set $S_C$. Then, the sets $F_{i,j}$ can be used to create $X$ matrices, and the rank of $X$ matrices can be checked. One optimal way to implement this solution strategy is to incrementally solve problem for each column $x_{C,j}$ and then refine the elements of $F_{i,j}$ at $j$th iteration. Next, we describe briefly this approach. Note that $F_{i,0} = F$, $\forall i = 1, \ldots, N$.

Since the solution sets to Eq. (6) are contained in the solution trees and there are $N$ rows in $X$, each of the $N$ rows is chosen from the solution set $F_{i,j}$. As a first step, we repeat the solution tree $N$ times, once for each row. Once Algorithm 2 finds the solution for a given column, the solution tree for each row will be pruned in order to find the candidate vector for each of the rows of $X$ using the set definition in Eq. (28). The method of finding the entries of $X$ is briefly described as follows:

- Find the solution tree containing solutions of the $x_R$ to Problem 6 using
Algorithm 1.
- Repeat the solution tree $N$ times.
- Assume a value for $d$, the radius in Algorithm 2.
- In the $i$th solution tree (i.e. for $i$th row of $X$), denote the set of unique entries of the $j$th column of the solutions present in the tree as $S_j$. Define $S_j = [S_{1,j} \ S_{2,j} \ldots S_{N,j}]^T$. $S_j$ is the input to Algorithm 2 in the $j$th run.
- Algorithm 2 for the $j^{th}$ column will output all candidates which are within the sphere radius. Denote the output of Algorithm 2 as $Z_j$.
- Run the Algorithm 2 for all the columns.
- For each of the columns, from $Z_j$ choose the vector in the ascending order of distance which has not been considered previously. Denote this vector as $x_{C,j} = [x_{C,j}(1) \ x_{C,j}(2) \ldots x_{C,j}(N)]^T$.
- In the $i$th solution tree, remove all the vectors whose $j$th element is not same $x_{C,j}(i)$. This is equivalent to refining the solution sets $F_{i,j}$. If only one vector is left in the tree do not perform the pruning step for the $i$th tree.
- Repeat this procedure for all the columns of $X$.
- The $i$th solution tree contains the candidate solutions for the $i$th row of $X$.
- Form the matrix $X$ and check for the rank condition. If the rank condition is satisfied, declare the derived matrix as the solution, else try out other vectors from $Z_j$ which have not yet been considered.
- If all the vectors in $Z_j$ are exhausted and still the rank condition is not satisfied, then reduce $j$ by 1 and re-run the procedure.
- If $j$ becomes zeros and still no solution is found, then increase $d$ and repeat the procedure till one $X$ is found that satisfies the rank condition.

The complete algorithm for finding the solution to Problem 5 can be given in Algorithm 3.
Algorithm 3 Algorithm to solve Problem \((5)\)

**Input:** \(G \in \mathbb{R}^{M \times N}, Y \in \mathbb{R}^{M \times L}\) and \(A \in \mathbb{Z}^{P \times L}, S, K\)

**Output:** Matrix \(X \in \mathbb{S}^{N \times L}\) such that \(AX^T = 0, \|X^T(i)\|_0 \leq K, \text{rank}(X) = N\) and \(||Y - GX||_2^2\) is minimum

1: procedure SOLVESPDecWCons\((G, Y, A, S, K)\)
2: \(T = \text{SolveSysOfLin DioEqn}(A, S, K)\)
3: \(X = 0\)
4: Discard the vectors in \(Z_j\) that are already present in \(Z_{p,j}\)
5: Repeat the solution tree \(T, N\) times, once for each row.
6: Connect the root nodes of the repeated trees to a common root node.
7: Denote this tree as \(T_N\)
8: while rank\((X) < N\) do
9: \(j = 1, J = L, M = 0\)
10: while \(j \leq L\) do
11: for \(i = 1\) to \(N\) do
12: \(S_{i,j} = \text{Set of unique elements at } j^{th} \text{ position in the } i^{th} \text{ branch vectors of } T_N\)
13: \(S_j = \{S_{1,j}, S_{2,j}, \ldots, S_{N,j}\}\)
14: \(Z_j = \text{SphereDecoder}(S_j, Y(:,j), G, d_{sp})\)
15: \(x_{C,j} = \text{Vector from } Z_j \text{ which has not be considered previously in the ascending order of distance}\)
16: for \(k = 1\) to \(N\) do
17: if Number of vectors in \(k^{th}\) branch of \(T_N > 1\) then
18: In \(k^{th}\) branch of \(T_N\), remove all vectors which dont have \(x_{C,j}(k)\) in the \(j^{th}\) position
19: if \(j == L\) then
20: Create a matrix \(X\) containing one vector from each branch in \(T_N\)
21: if rank\((X) \neq N\) then
22: \(n=\text{Number of vectors in } Z_j\)
23: \(M = M + 1\)
24: if \(M == n\) then
25: \(J = J - 1, M = 0\)
26: if \(J == 0\) then
27: \(j = 1, J = L\)
28: \(d_{sp} = d_{sp} + 1\)
29: for \(k = 1\) to \(L\) do
30: \(Z_{p,k} = Z_k\)
31: else
32: \(j = J\)
33: Go To 16
34: end if
35: \(j = j + 1\)
36: return \(X\)

Remark 2. Note that if the equality constraints is of form \(AX = 0\) instead of \(AX^T = 0\), the construction of \(X\) is straightforward using the set \(F\).

Next, Algorithm 3 is explained using Example 1.

Example 1. (continued)
There are three rows in $X$. Therefore, the solution set $F$ in Fig. 2 is replicated three times as shown in Fig. 3 one solution tree for each row. It can be seen that

\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & -1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

Fig. 3: Solution tree (6) repeated 3 times for each rows, and the corresponding the solution sets are denoted as $F_{i,0}$, $\forall i = 1, 2, 3$

in the first tree set of unique entries for the first column is $S_{1,1} = \{-1, 0, 1\}$. The set of unique entries for all the 3 rows is the same. Hence for the first column $S_1 = \{-1, 0, 1\}$.

Using the radius $d = 0.5$, the solution given Algorithm 2 for the first column ($j = 1$) of $X$ is $x_{C,1} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$. This means that the candidate vectors for the rows $R1$, $R2$, and $R3$ must have 1, 0 and 0 in the first position. Then, the new sets $F_{i,1}$, $i = 1, 2, 3$ are obtained by deleting the candidates in $F_{i,0}$, $i = 1, 2, 3$ which do not contain the specified elements in $x_{C,1}$ on the first position for the corresponding rows. The resulting solution tree is shown in Fig. 4. As seen from Fig. 4, there is

\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & -1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

Fig. 4: Solution tree for (9) and pruning operation using the solution for $x_{C,1}$

only one candidate vector for the $x_{R,1}$ of $X$, while there are five candidate vectors for the rows $x_{R,2}$ and $x_{R,3}$. As it can be seen from Fig. 4, the set for each element of $x_{C,2}$ can be re-defined as: $S_2 = \{S_{1,2}, S_{2,2}, S_{3,2}\}$, $S_{1,2} = \{1\}$, $S_{2,2} = \{-1, 0, 1\}$ and
\( S_{3,2} = \{-1,0,1\} \). With these sets and the solutions from Algorithm 2 with \( d = 0.5 \), we get the solution set for the second column \((j = 2)\), \( x_{C,2} = [1 -1 1]^T \). This means that we need to eliminate all the vectors that do not have \(-1\) and \(1\), respectively in their second position from the candidate solution sets \( F_{2,1} \) and \( F_{3,1} \). The resulting tree is shown in Fig. 5.

\[
\begin{bmatrix}
  1 & 1 & -1 & -1 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 1 \\
  0 & 1 & 0 & 1 \\
  0 & 1 & 0 & 0 
\end{bmatrix}
\]

Fig. 5: Solution tree for (9) and pruning operation using the solution for \( x_{C,2} \)

Constructing the set of all allowable entries for \( x_{C,3} \) of \( X \), we have \( S_{1,3} = \{-1\}, S_{2,3} = \{-1,0\} \) and \( S_{3,3} = \{0,1\} \). Applying Algorithm 2 to the third column \((j = 3)\) of \( Y \) and the corresponding set \( S_{3} \), we have \( \hat{x}_{C,3} = [-1 -1 0]^T \). Eliminating candidate vectors for the rows \( R2 \) and \( R3 \) that do not contain \{-1\} and \{0\}, respectively, the solution tree is shown in Fig. 6. After pruning with \( x_{C,3} \), it can be seen from Fig. 6 that the sets \( F_{i,3}, \forall i = 1,2,3 \) contain only one element.

Hence, we cannot search any further since there are no other alternative solutions. Hence, \( X \) can be constructed as follows

\[
X = \begin{bmatrix}
  1 & 1 & -1 & -1 & 0 & 0 & 0 \\
  0 & -1 & -1 & 1 & 1 & 0 & 0 \\
  0 & 1 & 0 & 1 & 0 & -1 & -1 
\end{bmatrix}
\]

Since \( \text{rank} \left( X \right) = 3 \), the solution is optimal \( X^* \) which is same as the actual solution \( X_a \) as given in (9). Note that if \( \text{rank} \left( X \right) \neq 3 \), then \( d \) has to be increased in Algorithm 2 and repeat the procedure.
3. Special case: Integer least–squares with the linear equality and sparsity constraints. In Section 2, an algorithm to solve Problem 5 is proposed. The main advantage of the proposed algorithm is that the two parts solution to the the optimization problem (5). Using this two-parts algorithm, we can also solve integer least–squares with the linear inequality and sparsity constraints. In this case, the optimization problem is:

\[
\begin{align*}
\text{minimize} & \quad \|y - Gx\|^2 \\
\text{subject to} & \quad x_i \in S \subset \mathbb{Z}, \forall i = 1, \ldots, N \\
& \quad Ax = b \\
& \quad ||x||_0 \leq K.
\end{align*}
\]  

The algorithm for the problem (30) can be obtained by putting together the solutions of the sub-problems of the problem (1) as follows:

1. Find all the solutions of the Diophantine Equations with the sparsity constraints. Formally, we would like to find a solution set \( \mathcal{F} \) in (6).

2. Then, solve the integer least–squares problem of the following form:

\[
\text{(31) Find } x \text{ such that } \min_{x \in \mathcal{F}} ||y - Gx||^2.
\]

In contrast to the standard integer–least squares problem in (26), the solution space is constrained by the solution set \( \mathcal{F} \) in Eq. (31).

An algorithm to solve the problem (30) is described next using the solution strategies in the last section.

**Algorithm 4 Algorithm to solve (30)**

**Input:** \( G \in \mathbb{R}^{M \times N}, y \in \mathbb{R}^{M \times 1} \) and \( A \in \mathbb{Z}^{P \times L}, S, K \)

**Output:** Vector \( x \in \mathbb{S}^{N \times 1} \) such that \( AX = 0, ||x||_0 \leq K, \) and \( ||y - Gx||^2 \) is minimum

1. **procedure** SOLVEILSEQ(\( G, y, A, S, K \))
2. \( \mathcal{F} = \text{SolveSysOfLinDioEqn}(A, S, K) \)
3. \( \hat{x} = \text{SphereDecoder}(S, y, G, d_{sp}) \)
4. \( d_s = 0 \)
5. **for** \( i = 1 \) to \( |\mathcal{F}| \) **do**
6. Compute \( d_x(i) = \|\hat{x} - x_i\|^2 \) where \( x_i \) is the \( i \)th element in the set \( \mathcal{F} \)
7. Find the minimum value in the \( d_x \) and the corresponding index \( f \)
8. \( x_{\text{min}} = f \)th element of the set \( \mathcal{F} \)
9. **return** \( x_{\text{min}} \)

4. Numerical Experiments. This section reports computational results obtained by applying the proposed algorithm. We consider simulation examples of different sizes and rank of \( X \). For each simulation example, the data matrix \( Y \) were generated based on the model in Eq. (2). The matrix \( G \) is generated as follows. The random samples of desired dimension \( (G) \) are generated under the assumption of the Gaussian distribution with zero mean and the unity standard deviation. For obtaining positive matrix, the absolute of the generated matrix is taken. Then, the matrix \( X \) are generated by picking such that the constraints are satisfied. We restrict each \( X(i, j) \) element in \( S = \{-2, -1, 0, 1, 2\} \). The sparsity \( K = 4 \) is taken. Each element
Table 1: Numerical Results for Problem (5) with $S = \{-1, 0, 1\}$

| Size of $X$ | Rank of $X$ | $n$ | Avg CPU time [sec] | $N_{\mathcal{F},\text{avg}}$ |
|------------|-------------|-----|--------------------|-----------------------------|
| 10 $\times$ 2 | 2 | 20 | 0.0859 | 1767 |
| 10 $\times$ 4 | 4 | 40 | 0.1211 | 1767 |
| 10 $\times$ 5 | 5 | 50 | 0.8234 | 1767 |
| 15 $\times$ 6 | 6 | 55 | 0.9156 | 19590 |
| 20 $\times$ 5 | 5 | 100 | 8.0062 | 411639 |
| 20 $\times$ 4 | 4 | 100 | 22.406 | 1534308 |
| 20 $\times$ 7 | 7 | 140 | 9.3438 | 411639 |
| 20 $\times$ 8 | 8 | 160 | 10.5188 | 411639 |
| 20 $\times$ 9 | 9 | 180 | 10.791 | 411639 |
| 25 $\times$ 6 | 6 | 55 | 108.5531 | 7225863 |
| 25 $\times$ 7 | 7 | 140 | 94.3781 | 7225863 |
| 25 $\times$ 8 | 8 | 160 | 666.68 | 45800736 |
| 25 $\times$ 9 | 9 | 180 | 1703288769 |

of matrix $E$, $E(i,j)$ is considered to be normally distributed with zero mean and standard deviation $\sigma = 0.2$. The data matrix, $Y$ is obtained by the multiplying the generated $G$ and $X$ and adding $E$ of the appropriate size. The equality constraints are generated using the matrix $A$ of $7 \times L$ where $L$ is the number of columns of $X$. For comparison purpose, the number of rows of $A$ is not changed. All the simulations are conducted in using MATLAB R2016 on Intel(R) Core(TM) i7-6700 CPU@ 3.40 GHz processor. Each optimization problem is repeated five times and the average values are reported.

Tables 1 and 2 reports the results of numerical experiments with the two domains $S = \{-1, 0, 1\}$ and $S = \{-2, -1, 0, 1, 2\}$. In Tables 1 and 2, $n$ indicates the number of variables to be optimized, and $N_{\mathcal{F},\text{avg}}$ the number of average nodes the algorithm visited during the computing set $\mathcal{F}$. For each simulation, the exact solution for $X$ is obtained.

The results in Tables 1-2 show that the $N_{\mathcal{F},\text{avg}}$ does not depend on the rank of $X$ but the number of rows of $X$. For example, for the number of row equals to ten and the different ranks of $X$, $N_{\mathcal{F},\text{avg}}$ is same. However, the the number of row increases, the $N_{\mathcal{F},\text{avg}}$ also increases. It can be seen that the average CPU time depends on the dimension of $X$ for the same number of decision variables. For example, $n = 100$, 300 the average CPU times increases with the dimension of $X$. This is due to the number of nodes need to be visited to compute set $\mathcal{F}$. Further, the domains have an effect on the average CPU time dramatically. It can be seen that the average CPU time increased fifty folds for $n = 300$ when the domain has extended two more integers.

5. Conclusions. In this work, we have proposed an exact algorithm to solve a rank-constrained integer least-squares (ILS) problem arising low-rank matrix factorization related applications. The algorithm can also handle the equality constraints
Table 2: Numerical Results for Problem (5) with \( S = \{-2, -1, 0, 1, 2\} \)

| Size of \( X \) | Rank of \( X \) | \( n \) | CPU time (s) | \( N_{F, \text{avg}} \) |
|-----------------|-----------------|-----|-------------|-----------------|
| \( 10 \times 2 \) | 2               | 20  | 0.5828      | 31815           |
| \( 10 \times 3 \) | 3               | 30  | 0.6937      | 31815           |
| \( 10 \times 4 \) | 4               | 40  | 0.8234      | 31815           |
| \( 11 \times 5 \) | 5               | 55  | 0.9859      | 31940           |
| \( 14 \times 5 \) | 5               | 70  | 4.8609      | 240815          |
| \( 15 \times 6 \) | 6               | 90  | 11.484      | 540400          |
| \( 15 \times 7 \) | 7               | 105 | 12.083      | 540400          |
| \( 20 \times 7 \) | 7               | 140 | 306.79      | 26713445        |
| \( 20 \times 8 \) | 8               | 160 | 318.25      | 26713445        |
| \( 20 \times 10 \)| 10              | 200 | 366.36      | 26713445        |
| \( 22 \times 10 \)| 10              | 220 | 423.53      | 33838935        |
| \( 22 \times 11 \)| 11              | 242 | 488.14      | 33838935        |
| \( 25 \times 12 \)| 12              | 300 | 1244.7      | 97556730        |
| \( 30 \times 10 \)| 10              | 300 | 5096.8      | 466803555       |

and the sparsity constraints. The algorithm is based on the modified Fincke-Pohst enumeration and sparse solutions of Diophantine equations. The proposed algorithm has been explained using an example. Further, numerical experiments have been carried out to test the applicability of the algorithm. The results of numerical experiments show that the proposed algorithm has solved moderate scale rank-constrained integer least-squares problems with the sparsity and equality constraints which arise in intended practical applications. It is shown that the proposed solution can be useful to solve ILS problems with linear equality and sparsity constraints. It has been also observed that the computing sparse solution of Diophantine equations to find row solution sets takes majority time in the algorithm. In future, the proposed algorithm can be improved by reducing time taken in this step.

REFERENCES

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

[2] M. Berry, M. Browne, A. M. Langville, V. P. Pauca, and R. J. Plemmons, Algorithms and applications for approximate nonnegative matrix factorization, Computational Statistics & Data Analysis, 52 (2007), pp. 155–173.

[3] N. Bhattacharyya and A. Ayvaz, Monotonous (semi-) nonnegative matrix factorization, in Proceedings of the Second ACM IKDD Conference on Data Sciences, ACM, 2015, pp. 92–97.

[4] A. Billionnet, S. Eloumi, and A. Lambert, A branch and bound algorithm for general mixed-integer quadratic programs based on quadratic convex relaxation, Journal of Combinatorial Optimization, 28 (2014), pp. 376–399.

[5] M. A. Borno, Reduction in solving some integer least squares problems, arXiv preprint arXiv:1101.0382, (2011).

[6] C. Buchheim, A. Cappara, and A. Lodi, An effective branch-and-bound algorithm for convex quadratic integer programming, Mathematical programming, 135 (2012), pp. 369–395.

[7] C. Buchheim, M. De Santis, and L. Palagi, A fast branch-and-bound algorithm for nonconvex quadratic integer optimization subject to linear constraints using ellipsoidal relaxations, Operations Research Letters, 43 (2015), pp. 384–388.

[8] C. Buchheim, M. De Santis, L. Palagi, and M. Piacentini, An exact algorithm for nonconvex quadratic integer minimization using ellipsoidal relaxations, SIAM Journal on Op-
[9] C. Buchheim, R. Hübner, and A. Scobel, Ellipsoid bounds for convex quadratic integer programming, SIAM Journal on Optimization, 25 (2015), pp. 741–769.

[10] C. Buchheim and A. Wiegele, Semidefinite relaxations for non-convex quadratic mixed-integer programming, Mathematical Programming, 141 (2013), pp. 435–452.

[11] X.-W. Chang and G. H. Golub, Solving ellipsoid-constrained integer least squares problems, SIAM Journal on Matrix Analysis and Applications, 31 (2009), pp. 1071–1089.

[12] X.-W. Chang and Q. Han, Solving box-constrained integer least squares problems, IEEE Transactions on wireless communications, 7 (2008), pp. 277–287.

[13] U. Fincke and M. Pohst, A procedure for determining algebraic integers of given norm, in European Conference on Computer Algebra, Springer, 1983, pp. 194–202.

[14] U. Fincke and M. Pohst, Improved methods for calculating vectors of short length in a lattice, including a complexity analysis, Mathematics of computation, 44 (1985), pp. 463–471.

[15] C. A. Floudas, Nonlinear and mixed-integer optimization: fundamentals and applications, Oxford University Press, 1995.

[16] O. K. Gupta and A. Ravindran, Branch and bound experiments in convex nonlinear integer programming, Management science, 31 (1985), pp. 1533–1546.

[17] B. Hassibi and H. Vikalo, On the expected complexity of integer least-squares problems, in Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on, vol. 2, IEEE, 2002, pp. II–1497.

[18] B. Hassibi and H. Vikalo, On the sphere-decoding algorithm i. expected complexity, IEEE transactions on signal processing, 53 (2005), pp. 2806–2818.

[19] R. Hemmecke, M. Köppe, J. Lee, and R. Weismantel, Nonlinear integer programming, arXiv preprint arXiv:0906.5171, (2009).

[20] H. Kim and H. Park, Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method, SIAM journal on matrix analysis and applications, 30 (2008), pp. 713–730.

[21] J. Kim and H. Park, Fast nonnegative matrix factorization: An active-set-like method and comparisons, SIAM Journal of Scientific Computing, 33 (2011), pp. 3261–3281.

[22] D. D. Lee and H. S. Seung, Algorithms for non-negative matrix factorization, in Advances in neural information processing systems (NIPS), vol. 13, 2001, pp. 556–562.

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

[24] D. Li, X. Sun, and F. Wang, Convergent lagrangian and contour cut method for nonlinear integer programming with a quadratic objective function, SIAM Journal on Optimization, 17 (2006), pp. 372–400.

[25] M. M. Lin, Discrete Eckart-Young theorem for integer matrices, SIAM Journal on Matrix Analysis and Applications, 32 (2011), pp. 1367–1382.

[26] J. Park and S. Boyd, A semidefinite programming method for integer convex quadratic minimization, Optimization Letters, (2017), pp. 1–20.

[27] A. Saxena, P. Bonami, and J. Lee, Convex relaxations of non-convex mixed integer quadratically constrained programs: extended formulations, Mathematical Programming, 124 (2010), pp. 383–411.

[28] C.-P. Schnorr and M. Euchner, Lattice basis reduction: Improved practical algorithms and solving subset sum problems, Mathematical programming, 66 (1994), pp. 181–199.

[29] A. Schrijver, Theory of linear and integer programming, John Wiley & Sons, 1998.

[30] P. Teunissen, Least-squares estimation of the integer gps ambiguities, in Invited lecture, section IV theory and methodology, IAG general meeting, Beijing, China, 1993.

[31] S. A. Vavasis, On the complexity of nonnegative matrix factorization, SIAM Journal on Optimization, 20 (2009), pp. 1364–1377.