Noname manuscript No.
(will be inserted by the editor)

Integer Matrix Approximation and Data Mining

Bo Dong · Matthew M. Lin · Haesun Park

Received: date / Accepted: date

Abstract Integer datasets frequently appear in many applications in science and engineering. To analyze these datasets, we consider an integer matrix approximation technique that can preserve the original dataset characteristics. Because integers are discrete in nature, to the best of our knowledge, no previously proposed technique developed for real numbers can be successfully applied. In this study, we first conduct a thorough review of current algorithms that can solve integer least squares problems, and then we develop an alternative least square method based on an integer least squares estimation to obtain the integer approximation of the integer matrices. We discuss numerical applications for the approximation of randomly generated integer matrices as well as studies of association rule mining, cluster analysis, and pattern extraction. Our computed results suggest that our proposed method can calculate a more accurate solution for discrete datasets than other existing methods.

Keywords Data mining · Matrix factorization · Integer least squares problem · Clustering · Association rule · Pattern extraction

The first author's research was supported in part by the National Natural Science Foundation of China under grant 11101067 and the Fundamental Research Funds for the Central Universities. The second author's research was supported in part by the National Center for Theoretical Sciences of Taiwan and by the Ministry of Science and Technology of Taiwan under grants 104-2115-M-006-017-MY3 and 105-2634-E-002-001. The third author's research was supported in part by the Defense Advanced Research Projects Agency (DARPA) XDATA program grant FA8750-12-2-0309 and NSF grants CCF-0808863, IIS-1242304, and IIS-1231742.

Bo Dong
School of Mathematical Sciences, Dalian University of Technology, Dalian, Liaoning 116024, China.
E-mail: dongbo@dlut.edu.cn

Matthew M. Lin
Department of Mathematics, National Cheng Kung University, Tainan 701, Taiwan.
E-mail: mhlin@mail.ncku.edu.tw

Haesun Park
School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0765, USA.
E-mail: hpark@cc.gatech.edu

1 Introduction

The study of integer approximation has long been a subject of interest in the areas of communication networks, data organization, environmental chemistry, lattice design, and finance [3,21,35,37]. In this study, we consider the integer matrix approximation (IMA) of an integer matrix. If we let $\mathbb{Z}^{m\times n}$ be the set of $m \times n$ integer matrices, the IMA problem can be stated as follows:

> **(IMA Problem)** Given an integer matrix $A \in \mathbb{Z}^{m\times n}$ and a positive integer $k < \min\{m, n\}$, find $U \in \mathbb{Z}^{m\times k}$ and $V \in \mathbb{Z}^{k\times n}$ such that the residual function
>
> $$f(U, V) := \|A - UV\|_F^2 \tag{1}$$
>
> is minimized.

Note that in real applications, entries of U and V can be further restricted to a certain domain. We also address this scenario in our later discussion. Concretely, we can say that the IMA is a problem of representing an original integer dataset in a space spanned by integer based vectors using integer representations to minimize the residual function in (1). One example that characterizes this problem is the so-called *market basket transaction*. Table 1 shows a transaction example, in which there are orders from five customers $C_1, \ldots, C_5$. In practice, this would typically be a huge dataset with many

	Bread	Milk	Diapers	Eggs	Chips	Beer
C_1	2	1	3	0	2	5
C_2	2	1	1	0	2	4
C_3	0	0	4	2	0	2
C_4	4	2	2	0	4	8
C_5	0	0	2	1	0	1

Table 1 An integer representation of the transaction example

customers and shopping items. Similar to the discussion in [31,33], we want to find an association rule from matrix V such as "{2 diapers, 1 egg} ⇒ {1 beer}", meaning that if a customer has shopped for two diapers and one egg, then he/she has a higher possibility of shopping for "one" beer. Also, entries of U determine the weight associated with each representative transaction, i.e., the strength of associations between a customer and the representative transaction. To ensure that the number of items is an integer, the weight is specified as a nonnegative integer corresponding to the representative transaction. For example, we can approximate the original integer dataset, defined in Table 1, by two representative transactions $[2, 1, 1, 0, 2, 4]$ and $[0, 0, 2, 1, 0, 1]$ with the weight determined by a 5×2 integer matrix:

$$A = \begin{bmatrix} 2&1&3&0&2&5 \\ 2&1&1&0&2&4 \\ 0&0&4&2&0&2 \\ 4&2&2&0&4&8 \\ 0&0&2&1&0&1 \end{bmatrix} \approx \begin{bmatrix} 1&1 \\ 1&0 \\ 0&2 \\ 2&0 \\ 0&1 \end{bmatrix} \begin{bmatrix} 2&1&1&0&2&4 \\ 0&0&2&1&0&1 \end{bmatrix}. \tag{2}$$

This is a rank-two approximation that provides the possible shopping behaviors of the original five customers.

Another immediate application of the IMA might come to mind: if we reference each column in Table 1 to the ratings of a group of users, we have a broad record of a scenario such as the automobile manufacturing industry, with the existing ratings represented in a matrix. That is, assume we have five cars with six different types of attributes with integer ratings ranging from 0 to 8. By extending the concept of the IMA, we can solve this problem by representing the approximation as follows:

> Given an integer matrix $A \in \mathbb{Z}^{m\times n}$ and a positive integer $k < \min\{m, n\}$, find a binary matrix $U = [u_{ij}] \in \mathbb{Z}_2^{m\times k}$, for $i = 1, \ldots, m$, and $V \in \mathbb{Z}^{k\times n}$ such that the residual function
> $$f(U, V) := \|A - UV\|_F^2$$
> is minimized.

In this way, each row of the approximated matrix U represents the associations between a car and the possible features, and each column of V gives the strength of the associations between the features and an attribute. To serve a customer order more efficiently, it is desirable to initially build a few basic models to attract the most customers. Our task, here, is to use integer representatives to express the associations among cars, features, and attributes such that the interpretation can be carried out directly. As as example, in section 4.4, we provide a typical car evaluation dataset from the UCI repository [4].

Note that the goal of using the IMA is to analyze original integer (i.e., discrete) data in terms of integer representatives to enable the underlying information to be directly conveyed. Due to the data's discreteness, conventional techniques such as singular value decomposition (SVD) [19] and nonnegative matrix factorization (NMF) [11,13,26,27, 29,30] are inappropriate and cannot achieve this goal. This problem can be regarded more appropriately as a generalization of the so-called Boolean matrix approximation, where the entries of A, U, and V are limited to $\mathbb{Z}_2 = \{0, 1\}$, i.e., binaries. Refer to [31,33] for a nonorthogonal binary decomposition approach to the approximation of a given binary dataset. In fact, the need for IMA has been long-standing in the experimental sciences. It is applied to extract features/essences normally embedded in the original problems with basis elements required to fulfill discrete properties. Major discussions of this subject can be found in the fields of wireless communication with binary source signals [43]; gene expression data [34,42]; cellular differentiation from DNA methylation [22]; and clustering analysis [5,39]. See also [24,35,36,40,44] for a comprehensive study of its variants. In contrast to studies of binary datasets, in this paper, our main objective is to propose a means for analyzing general integer (discrete) datasets and to provide a direct interpretation.

This paper is organized as follows. In section 2, we investigate IMA in terms of the alternating least squares method and discuss some characteristics related to orthogonal constraints. In section 3, we review methods for solving integer least squares problems, apply them to solve the IMA, and present a corresponding convergence analysis. In section 4, we examine seven examples to illustrate the capacity and efficiency of our proposed IMA approach. In section 5, we make our concluding remarks.

2 Alternating Least Squares Method with Integer Constraint

The framework of the alternating least squares (ALS) method comprises the following two steps:

– Step 1. Provide an initial value $U \in \mathbb{Z}^{m\times k}$.
– Step 2. Solve the following problems iteratively until a stopping criterion is satisfied:

$$\min_{V\in\mathbb{Z}^{k\times n}} \|A - UV\|_F^2, \tag{3a}$$

$$\min_{U\in\mathbb{Z}^{m\times k}} \|A - UV\|_F^2. \tag{3b}$$

Alternatively, first, initialize V and iterate (3a) and (3b) in reverse order. In Step 2, each subproblem must find optimal matrices U or V such that $\|A-UV\|_F^2$ is minimized. We further observe that:

$$\|A - UV\|_F^2 = \sum_j \|A(:,j) - UV(:,j)\|_2^2 = \sum_i \|A(i,:) - U(i,:)V\|_2^2. \tag{4}$$

It follows that Eq. (4) is minimized if for each j,

$$\|A(:,j) - UV(:,j)\|_2$$

is minimized and for each i,

$$\|A(i,:) - U(i,:)V\|_2$$

is minimized. Let round(X) be a function that rounds each element of X to the nearest integer with the greatest magnitude. Because the 2-norm is invariant under orthogonal transformations, a result immediately follows .

Theorem 1 *Suppose* $\mathbf{a} \in \mathbb{Z}^{m\times 1}$ *and* $U \in \mathbb{Z}^{m\times k}$ *with* $m \geq k$ *and* $U^\top U = I_k$*, where* I_k *is the* $k \times k$ *identity matrix. Let*

$$\hat{\mathbf{v}} = U^\top \mathbf{a} \in \mathbb{Z}^{k\times 1}. \tag{5}$$

Then,

$$\|\mathbf{a} - U\hat{\mathbf{v}}\|_2 \leq \min_{\mathbf{v}\in\mathbb{Z}^{k\times 1}} \|\mathbf{a} - U\mathbf{v}\|_2.$$

With an obvious notation change, an analogous statement is true for $\mathbf{a} \in \mathbb{Z}^{1\times n}$ and $V \in \mathbb{Z}^{k\times n}$ having $n \geq k$ and $VV^\top = I_k$. Theorem 1 provides a means for obtaining the optimal matrix V by direct observation.

Theorem 2 *Suppose* $A \in \mathbb{Z}^{m\times n}$ *and* $U \in \mathbb{Z}^{m\times k}$ *with* $m \geq k$ *and* $U^\top U = I_k$*. Let*

$$\widehat{V} = U^\top A \in \mathbb{Z}^{k\times n}.$$

Then,

$$\|A - U\widehat{V}\|_F \leq \min_{V\in\mathbb{Z}^{k\times n}} \|A - UV\|_F.$$

Similarly, the optimal matrix $\widehat{U} = AV^\top \in \mathbb{Z}^{m\times k}$ can be immediately obtained, provided that V is given and $VV^\top = I_k$. However, a question remains of whether the optimal matrices U and V can be obtained simply by rounding the real optimal solutions $(AV^\top)(VV^\top)^{-1}$ and $(U^\top U)^{-1}(U^\top A)$, respectively. The answer is definitely no, due to the discreteness embedded in the integer matrix.

Example 1 Consider $\min_{\mathbf{v}\in\mathbb{Z}^{2\times 1}} \|\mathbf{a} - U\mathbf{v}\|_2$, where $U = \begin{bmatrix} 8 & 1 \\ 9 & 2 \end{bmatrix}$ and $\mathbf{a} = \begin{bmatrix} 16 \\ 17 \end{bmatrix}$. The optimal solution $\mathbf{v}_{\text{opt}} \in \mathbb{R}^{2\times 1}$ is

$$\mathbf{v}_{\text{opt}} = (U^\top U)^{-1}(U^\top \mathbf{a}) \approx \begin{bmatrix} 2.1429 \\ -1.1429 \end{bmatrix}.$$

Considering integer vectors $\mathbf{v}_1 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$, $\mathbf{v}_2 = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$, $\mathbf{v}_3 = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$ and $\mathbf{v}_4 = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$ which are near $\mathbf{v}_{\text{opt}}$, we see that $\|\mathbf{a} - U\mathbf{v}_i\|_2 = \sqrt{2}, \sqrt{13}, \sqrt{113}, 6\sqrt{2}$ for $i = 1, \ldots, 4$, respectively. However, if we take $\mathbf{v} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$, we have $\|\mathbf{a} - U\mathbf{v}\|_2 = 1 < \|\mathbf{a} - U\mathbf{v}_i\|_2$ for $i = 1, \ldots, 4$. This implies that the best approximation of the vector v cannot be obtained by rounding the real optimal solution. Figure 1 provides a geometric demonstration.

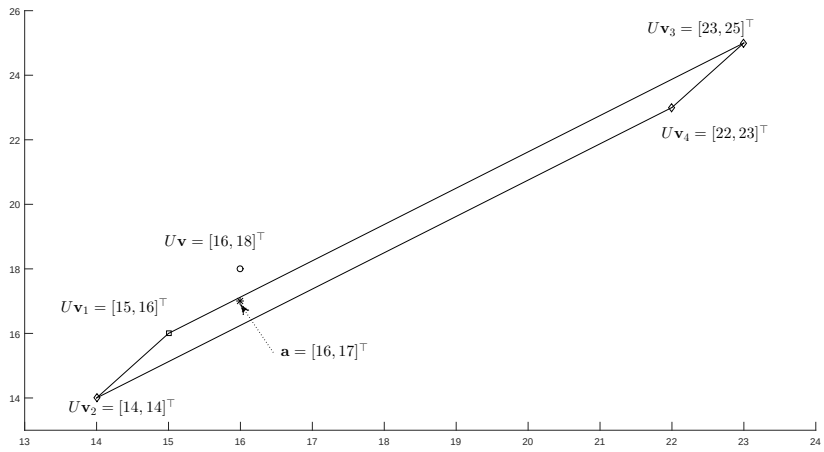


Fig. 1 A counterexample for the rounding approach

Whenever each column/row is fixed, the major mechanism of the ALS method is the computation of the global minimum at each iteration. To this end, we must study the approach for solving integer least squares problems and apply it to solve the IMA column-by-column/row-by-row to generate a sequence of descent residuals.

3 Integer Least Squares Problems

Given a vector $y \in \mathbb{R}^m$ and a matrix $H \in \mathbb{R}^{m\times n}$ with full column rank, we can define the integer least squares (ILS) problem as follows:

$$\min_{\mathbf{x}\in\mathbb{Z}^n} \|H\mathbf{x} - \mathbf{y}\|_2^2. \tag{6}$$

Although solving ILS problems is known to be NP-hard [16], this approach is applied in the fields of communications, lattice design, cryptography, radar imaging, and global positioning systems [3,21,38]. We are particularly interested in an ILS problem with box constraints

$$\min_{\mathbf{x}\in\mathcal{B}} \|H\mathbf{x}-\mathbf{y}\|_2^2, \tag{7}$$

where $\mathcal{B}=\mathcal{B}_1\times\cdots\times\mathcal{B}_n$ with $\mathcal{B}_i=\{x_i\in\mathbb{Z}:\ell_i\leq x_i\leq u_i\}$, due to its ability to solve IMAs with entries limited to a bounded domain. Note that ILS problems (6) are equivalent to the constrained problem (7), whenever $-\ell_i=u_i=\infty$ for all i. Therefore, in the following discussion, we use the definition in (7) to represent ILS problems generally.

Solving ILS problems typically involves two stages: reduction (or preprocessing) and search. Reduction involves transforming an ILS problem into a reduced form

$$\min_{\mathbf{z}\in\mathcal{B}} \|R\mathbf{z}-\hat{\mathbf{y}}\|_2^2, \tag{8}$$

such that

$$\begin{bmatrix} R \\ 0 \end{bmatrix} = Q^\top HP,\ \hat{\mathbf{y}}=Q_1^\top\mathbf{y},\ \mathbf{z}=P^{-1}\mathbf{x}, \tag{9}$$

where $Q=\begin{bmatrix} Q_1 & Q_2 \end{bmatrix}\in\mathbb{R}^{m\times m}$ is an orthogonal matrix, $P\in\mathbb{Z}_2^{n\times n}$ is a permutation matrix, and $R=[r_{i,j}]\in\mathbb{R}^{n\times n}$ is an upper triangular matrix [8]. As a result of this transformation, it follows that:

$$\min_{\mathbf{x}\in\mathcal{B}} \|\mathbf{y}-H\mathbf{x}\|_2^2 = \min_{\mathbf{z}\in\mathcal{B}} \|\hat{\mathbf{y}}-R\mathbf{z}\|_2^2 + \|Q_2^\top\mathbf{y}\|_2^2.$$

Hence, finding the optimal solution to an ILS problem is equivalent to finding the optimal solution to the reduced triangular form (8). For this purpose, we consider the following inequality

$$\|R\mathbf{z}-\hat{\mathbf{y}}\|_2^2<\beta, \tag{10}$$

where $\beta>0$, $\hat{\mathbf{y}}=[\hat{y}_i]\in\mathbb{R}^n$, $\mathbf{z}=[z_i]\in\mathbb{Z}^n$, and $R=[r_{i,j}]\in\mathbb{R}^{n\times n}$. We start the search process with a search ellipsoid denoted by (10) that has at least one point $\mathbf{z}\in\mathcal{B}$ within it. We then look for a process to identify a point $\mathbf{z}\in\mathcal{B}$ satisfying (10) and update the parameter β by taking $\beta=\|R\mathbf{z}-\hat{\mathbf{y}}\|_2^2$. This process is continued until there is no more valid point $\mathbf{z}\in\mathcal{B}$ lying inside the search ellipsoid, i.e., the ellipsoid can be shrunk no further. We then output the last integer point $\mathbf{z}\in\mathcal{B}$ identified as the optimal solution of (8). More precisely, if we consider $\mathbf{z}$ to be a solution for (10) and define the following:

$$c_n=\frac{\hat{y}_n}{r_{n,n}},\quad c_k=\frac{\hat{y}_k-\sum_{j=k+1}^{n} r_{k,j}z_j}{r_{k,k}}, \tag{11}$$

for $k=n-1,\ldots,1$, then (10) is equivalent to

$$\sum_{i=1}^{n} r_{i,i}^2(z_i-c_i)^2<\beta, \tag{12}$$

which implies that

$$r_{n,n}^2(z_n-c_n)^2<\beta, \tag{13a}$$

$$r_{k,k}^2(z_k-c_k)^2<\beta-\sum_{i=k+1}^{n} r_{i,i}^2(z_i-c_i)^2, \tag{13b}$$

for $k = n-1, \ldots, 1$. The well-known search process for ILS problems with box constraints is the Schnorr-Euchner enumerating strategy in [7,8]. This process begins by first selecting $z_n = \text{round}(\text{c}_\text{n})_{\mathcal{B}_\text{n}}$ and letting $k = n-1$. Next, if (13a) is satisfied, we move forward to level $n-1$ and choose $z_{n-1} = \text{round}(\text{c}_{\text{n}-1})_{\mathcal{B}_{\text{n}-1}}$. At this level, if z_{n-1} satisfies (13) with $k = n-1$, we move forward to level $n-2$; otherwise, we move back to level n, but select z_n as the next nearest integer in $\mathcal{B}_n$ to c_n. This process continues until we reach level 1 with a valid integer $z_1 = \text{round}(\text{c}_\text{k})_{\mathcal{B}_1}$ that satisfies (13b) with $k = 1$. Third, we update the parameter β by defining $\beta = \|R\mathbf{z} - \hat{\mathbf{y}}\|_2^2$, where $\mathbf{z} = [z_1, \ldots, z_n]^\top$. At this point, no more integer z_1 can be found to satisfy (13), so we must move back to level 2. Then, we choose z_2 as the next nearest integer in $\mathcal{B}_2$ to c_2 ("next" means "next to" the second entry in the most up-to-date vector $\mathbf{z}$). If z_2 satisfies (13b) with $k = 2$, we move forward to level 1 to update the value of z_1; otherwise we move backward to level 3 to update the value of z_3 as the next nearest integer in $\mathcal{B}_3$ to c_3 and so on. Finally, we fail to update the value of z_n to satisfy (13a) for the current β. We then output the latest found integer point $\mathbf{z}$ as the optimal solution for (8). Here, we use Algorithm 1 to illustrate the search process for the optimal solution of (8). To make this search process meaningful, the initial bound should be carefully considered. The simplest choice for the initial bound is infinity. In this work, we generate the initial bound immediately after completing the reduction process. See either Algorithm 2 or Algorithm 3 below for the integer point $\hat{\mathbf{z}}$. This point $\hat{\mathbf{z}}$, known as the *Babai integer point*, would satisfy the constraint $\mathcal{B}$ and is commonly used as a suboptimal solution for this initial bound β, i.e., $\beta = \|R\hat{\mathbf{z}} - \hat{\mathbf{y}}\|_2^2$.

On the other hand, when solving ILS problems with box constraints, i.e., ℓ_i or u_i is finite, the search process should take those constraints into account. That is, during the search process, we must search for an integer vector that simultaneously satisfies (13) and the box constraint in (7). Although this constraint complicates our search process, the complexity serves to shrink the search range. This observation has been discussed in [8] and can be summarized as follows. Note that since $z_i \in \mathcal{B}_i$ for each i,

$$\hat{y}_k - \sum_{i=k}^{n} \max(r_{k,i}\ell_i, r_{k,i}u_i) \le \hat{y}_k - \sum_{i=k}^{n} r_{k,i}z_i \le \hat{y}_k - \sum_{i=k}^{n} \min(r_{k,i}\ell_i, r_{k,i}u_i), \tag{14}$$

it follows that

$$(\hat{y}_k - \sum_{i=k}^{n} r_{k,i}z_i)^2 \ge \delta_k.$$

Here, we define

$$\delta_k = \min\{(\hat{y}_k - \sum_{i=k}^{n} \max(r_{k,i}\ell_i, r_{k,i}u_i))^2, (\hat{y}_k - \sum_{i=k}^{n} \min(r_{k,i}\ell_i, r_{k,i}u_i))^2\},$$

if the upper and lower bounds in (14) have the same sign; otherwise, take $\delta_k = 0$. Let

$$t_n = 0, \quad t_k = \sum_{i=k+1}^{n} r_{i,i}^2 (z_i - c_i)^2, \quad k = 1, \ldots, n-1,$$

$$\gamma_1 = 0, \quad \gamma_k = \sum_{i=1}^{k-1} \delta_i, \quad k = 2, \ldots, n.$$

Algorithm 1: (Search algorithm) $[\mathbf{z}_{\rm opt}] = \textsc{Search}(R, \hat{\mathbf{y}}, \mathcal{B}, \beta)$

Input: A matrix $R = [r_{i,j}] \in \mathbb{R}^{n\times n}$, a vector $\hat{\mathbf{y}} = [\hat{y}_i] \in \mathbb{R}^n$, the box constraints $\mathcal{B} = \mathcal{B}_1 \times \cdots \times \mathcal{B}_n$ with $\mathcal{B}_i = \{x_i \in \mathbb{Z} : \ell_i \le x_i \le u_i\}$, an initial upper bound $\beta \in \mathbb{R}$.
Output: An optimal solution $\mathbf{z}_{\rm opt} \in \mathbb{Z}^n$.

1 **begin**
  `/* Select` z_n`. */`
  $c_n = \dfrac{\hat{y}_n}{r_{n,n}}$; $z_n \leftarrow \text{round}(\text{c}_\text{n})_{\mathcal{B}_\text{n}}$; $k = n-1$;
2
3 **if** (13) *is satisfied* **then**
4 **if** $k > 1$ **then**
  `/* Once` (13b) `is satisfied, move to level` $k-1$ `and select` z_{k-1}`.*/`
  $k \leftarrow k-1$; $c_k = \dfrac{\hat{y}_k - \sum_{j=k+1}^{n} r_{k,j} z_j}{r_{k,k}}$; $z_k \leftarrow \text{round}(\text{c}_\text{k})_{\mathcal{B}_\text{k}}$; Go to line 3;
5
6 **else**
  `/* A valid point is found in level 1. Update` β `and move back to level` $k = 2$`.*/`
7 $\mathbf{z}_{\rm opt} \leftarrow \mathbf{z}$; $\beta \leftarrow \|R\mathbf{z} - \hat{\mathbf{y}}\|_2^2$; Go to line 3;
8 **end**
9 **else**
10 **if** k=n **then**
  `/* For` $k = n$`, terminate the iteration and output` $\mathbf{z}$`,` once (13a) `is not satisfied.*/`
11 Output the integer point $\mathbf{z}_{\rm opt}$ as the optimal solution of (8).
12 **else**
  `/* Once` (13) `is not satisfied, move to` $k+1$`.`
13 $k \leftarrow k+1$
14 **end**
15 **while** *All integers in* $\mathcal{B}_k$ *have been selected* **do**
  `/* If all integers in` $\mathcal{B}_k$ `have been selected, move further back to` $k+1$ `and so on, until` $k+1 = n$`.*/`
16 Go to line 10.
17 **end**
  `/* Update` z_k`*/`
18 Select z_k as the next nearest integer in $\mathcal{B}_k$ to c_k; Go to line 3;
19 **end**
20 **end**

Since $(\hat{y}_k - \sum_{j=k}^{n} r_{k,j} z_j)^2 = r_{k,k}^2 (z_k - c_k)^2$, (12) implies that

$$\beta > \sum_{i=1}^{n} r_{i,i}^2 (z_i - c_i)^2 = \sum_{i=1}^{n} (\hat{y}_i - \sum_{j=i}^{n} r_{i,j} z_j)^2 \ge \gamma_k + r_{k,k}^2 (z_k - c_k)^2 + t_k,$$

that is,

$$r_{n,n}^2 (z_n - c_n)^2 < \beta - \gamma_n, \tag{15a}$$

$$r_{k,k}^2 (z_k - c_k)^2 < \beta - \gamma_k - t_k, \quad k = 1, \ldots, n-1. \tag{15b}$$

We then have an upper bound that is at least as tight as that in (13) for solving ILS problems with box constraints. That is, the condition (13) in Algorithm 1 can be replaced by (15) to accelerate the search process. This reduced constraint (15) is then applied to solve our IMA problems in section 4.

Note that (13) implies that the reduced form (8) must not be obtained simply through a regular QR algorithm with permutations. That is, to enhance the efficiency

of the search process, the reduced form (8) should strive for diagonal entries arranged in a nondecreasing order,

$$|r_{1,1}| \leq |r_{2,2}| \leq \cdots \leq |r_{n,n}|. \tag{16}$$

The purpose here is to reduce the search range of z_k for $k = n-1, \ldots, 1$ once the right hand side of (13) is provided. Note that the search range is also affected by the right hand side of (13). We want each $r_{i,i}^2(z_i - c_i)^2$ to be as large as possible. However, this consideration may cause a problem because z_i can be very similar in value or even equal to c_i. Therefore, even if $r_{i,i}$ is very large, $r_{i,i}^2(z_i - c_i)^2$ can be very small so that a small $r_{i,i}$, rather than a large one, may be chosen as the i-th column because it induces larger $r_{i,i}^2(z_i - c_i)^2$. Since the product $|r_{1,1}r_{2,2}\cdots r_{n,n}| = (\det(H^\top H))^{\frac{1}{2}}$ is fixed, this implies that a small $r_{i,i}$ resides in a column with a large index i, which does not comply with our requirement in (16). To strike a balance, Chang and Han [8] proposed a column reordering approach (CH algorithm) to select z_i as the second nearest integer to c_i, i.e., $|z_i - c_i| \geq 0.5$. With this setting, once $|r_{i,i}(z_i - c_i)|$, i.e., $r_{i,i}^2(z_i - c_i)^2$, is very large, $|r_{i,i}|$ is also typically very large. The entire reduction process can be summarized as follows:

Step 1. Compute the QR decomposition of H,

$$H = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix},$$

and define $P = I_n$, $\hat{\mathbf{y}} = Q_1^\top \mathbf{y}$, $\bar{\mathbf{y}} = \hat{\mathbf{y}}$, and $k = n$, with notations defined in (9).

Step 2. If $k < n$, compute $\bar{\mathbf{y}}(1:k) \leftarrow \bar{\mathbf{y}}(1:k) - R(1:k, k+1)\hat{z}_{k+1}$. Otherwise, let $\widetilde{R} = R(1:k, 1:k)$, $\tilde{\mathbf{y}} = \bar{\mathbf{y}}(1:k)$. Calculate the value $|r_{k,k}(z_k - c_k)|$ by doing the following computations:

$$z_k^{(0)} = \text{round}(c_k)_{\mathcal{B}_k},\ z_k = \text{round}(c_k)_{\mathcal{B}_k \setminus z_k^{(0)}},\ \text{dist}_k = |r_{k,k}(z_k - c_k)|,$$

where $c_k = \bar{y}(k)/r_{k,k}$ and $\bar{\mathbf{y}} = [\bar{y}_i]$. In this work, "$\text{round}(c_k)_{\mathcal{B}_i}$" and "$\text{round}(c_k)_{\mathcal{B}_i \setminus z_i^{(0)}}$" denote the closest and second closest integers, respectively, in $\mathcal{B}_i$ to c_k for some i. To select the (k,k)-entry of R, we consider the following two process.

– First, we swap columns i, with $i = 1, \ldots, k-1$, for column k of $\widetilde{R}$ by a permutation matrix $P_{i,k}$. After each swap, we return $\widetilde{R}$ to upper triangular form by first using $k - i$ Givens rotations $G_{k-1,k}, G_{k-2,k-1}, \cdots, G_{i,i+1}$ to eliminate the last $k - i$ elements in the i-th column of the permuted matrix $\widetilde{R}$, where

$$G_{j,j+1} = \begin{bmatrix} I_{j-1} & & & \\ & c & s & \\ & -s & c & \\ & & & I_{n-j-1} \end{bmatrix}, \quad c^2 + s^2 = 1.$$

Then, the i-th column of $\widetilde{R}$ has entries $r_{i+1,i} = \cdots = r_{k,i} = 0$. However, the subdiagonal entries $r_{k,k-1}, r_{k-1,k-2}, \cdots, r_{i+2,i+1}$ are nonzero. To eliminate these $k-i-1$ entries of $\widetilde{R}$, another $k-i-1$ Givens rotations $G'_{i+1,i+2}, G'_{i+2,i+3}, \cdots, G'_{k-1,k}$ to obtain an updated upper triangular matrix $\widetilde{R}$. To simplify our notation, we define the multiplication of these $2k - 2i - 1$ Givens rotations as follows:

$$G_i = G'_{k-1,k} \cdots, G'_{i+2,i+3} G'_{i+1,i+2} G_{i,i+1} \cdots G_{k-2,k-1} G_{k-1,k}.$$

Also, we use G_i to upate $\tilde{\mathbf{y}}$. In other words, we have

$$\overline{R} \leftarrow G_i \widetilde{R} P_{i,k},$$
$$\bar{\mathbf{y}} \leftarrow G_i \tilde{\mathbf{y}}.$$

– Second, compute

$$z_i^{(0)} = \text{round}(c_k)_{\mathcal{B}_i},\ z_i = \text{round}(c_k)_{\mathcal{B}_i \setminus z_i^{(0)}},\ \text{dist}_i = |\bar{r}_{k,k}(z_i - c_k)|, \tag{17}$$

where $c_k = \bar{y}_k / \bar{r}_{k,k}$, $\overline{R} = [\bar{r}_{i,j}]$, and $\bar{\mathbf{y}} = [\bar{y}_i]$.

Step 3. If $\text{dist}_j = \max_{1 \le i \le k} \text{dist}_i$, let $\hat{z}_k = z_j^{(0)}$. Interchange columns j and k of P, $\mathcal{B}_j$ and $\mathcal{B}_k$, and update R and $\hat{\mathbf{y}}$ by defining

$$R(1:k, 1:k) \leftarrow G_j R(1:k, 1:k) P_{j,k},$$
$$R(1:k, k+1:n) \leftarrow G_j R(1:k, k+1:n),$$
$$\hat{\mathbf{y}}(1:k) \leftarrow G_j \hat{\mathbf{y}}(1:k).$$

Move back to Step 2 by replacing the index k with $k-1$, unless $k = 2$.

Step 4. For $k = 1$, update $\bar{\mathbf{y}}(1) \leftarrow \bar{\mathbf{y}}(1) - R(1,2)\hat{z}_2$ and compute $c_1 = \bar{y}(1)/r_{1,1}$ and $\hat{z}_1 = \text{round}(c_1)_{\mathcal{B}_1}$. Based on these four steps, we write down the details of the entire process in Algorithm 2.

It should be noted that in Algorithm 2 the computation from lines 11 to 13 is cumbersome. This is because after we swap columns i and k, $k-i$ Givens rotations are required to eliminate the last $k-i$ elements in the i-th column and $k-i-1$ more Givens rotations are required to eliminate the subdiagonal entries from columns $i+1$ to k. To simplify this permutation and triangularization process, Breen and Chang [6] suggest rotating the i-th column to the last column and shifting columns $i, i+1, \ldots, k$ to the left by one position. This transformation can be performed by multiply $\widetilde{R}$ on the right by a permutation matrix, denoted by $\widehat{P}_{k,j}$ in Algorithm 3, where we obtain $\widehat{P}_{k,j}$ by rotating the j-th column of an identity matrix of appropriate size to the last column and shifting columns $j, j+1, \ldots, k$ to the left one position. After this transformation, only $k-i-1$ Givens rotations are required to perform the triangularization. In Algorithm 3, we use the notation $\widehat{G}_{k,j}$ to denote these $k-i-1$ Givens rotations.

Later, Su and Wassell [41] proposed a geometric approach for efficiently computing c_k, provided that H is nonsingular, which only required the matrices H in the ILS problems (6) and (7) have full column rank and can be non-square. Once c_k is known, $z_i^{(0)}$ and dist_i can be obtained immediately; in other words, we can have the following two formulae for $z_i^{(0)}$ and dist_i:

$$z_i^{(0)} = \text{round}(\mathrm{c_k})_{\mathcal{B}_\mathrm{i}} = \text{round}(\mathrm{e_k^\top \overline{R}^{-1} \bar{y}})_{\mathcal{B}_\mathrm{i}} = \text{round}(\mathrm{e_i^\top \widetilde{R}^{-1} \tilde{y}})_{\mathcal{B}_\mathrm{i}}, \tag{18a}$$

$$\text{dist}_i = |\bar{r}_{k,k} z_i - \bar{y}_k| = |\bar{r}_{k,k}||z_i - \frac{\bar{y}_k}{\bar{r}_{k,k}}| = \frac{|z_i - c_k|}{\|\overline{R}^{-\top} e_k\|_2} = \frac{|z_i - c_k|}{\|\widetilde{R}^{-\top} e_i\|_2}. \tag{18b}$$

This implies that we can simplify the CH algorithm in terms of the formulae given in (18). Also, the Givens rotations G_i and permutation $P_{i,k}$ can be formed implicitly, since $z_i^{(0)}$ and dist_i could be computed only by the matrix $\widetilde{R}$ without doing any permutation. Note that a similar but more complicated discussion can be found in [6].

Algorithm 2: (LLL reduction) $[R, P, \hat{\mathbf{y}}, \ell, \mathbf{u}, \hat{\mathbf{z}}] = \mathrm{CH}(H, \mathbf{y}, \ell, \mathbf{u})$

Input: A matrix $H \in \mathbb{R}^{m\times n}$, a vector $\mathbf{y} \in \mathbb{R}^n$, a lower bound vector $\ell \in \mathbb{Z}^n$ and an upper bound vector $\mathbf{u} \in \mathbb{Z}^n$ collected from the box constraint $\mathcal{B} = \mathcal{B}_1 \times \cdots \times \mathcal{B}_n$ with $\mathcal{B}_i = \{x_i \in \mathbb{Z} : \ell_i \le x_i \le u_i\}$.

Output: An upper triangular matrix $R = [r_{i,j}] \in \mathbb{R}^{n\times n}$, a permutation $P \in \mathbb{R}^{n\times n}$, and a vector $\hat{\mathbf{y}} \in \mathbb{R}^n$ for the computation of (8) and (9) with updated upper and lower bounds $\ell \leftarrow P^\top \ell$ and $\mathbf{u} \leftarrow P^\top \mathbf{u}$, respectively, and the Babai integer point $\hat{\mathbf{z}} = [\hat{z}_i]$.

begin

`/* Step 1: Use Householder transformations to Compute the QR decomposition. */`

Compute the QR decomposition of H to obtain the reduced form in (8), i.e., $H = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix}$;

$P \leftarrow I_n$; $\hat{\mathbf{y}} \leftarrow Q_1^\top \mathbf{y}$; $\bar{\mathbf{y}} \leftarrow \hat{\mathbf{y}}$;

for $k \leftarrow n$ **to** 2 **do**

`/* Step 2: Truncate the vector y first. */`

if $k < n$ **then**

$\bar{\mathbf{y}}(1:k) \leftarrow \bar{\mathbf{y}}(1:k) - R(1:k, k+1)\hat{z}_{k+1}$

end

`/* Compute` $|r_{k,k}(z_k - c_k)|$ `before doing the permutation */`

$c_k \leftarrow \bar{y}(k)/r_{k,k}$; $z_k^{(0)} \leftarrow \mathrm{round}(\mathrm{c_k})_{\mathcal{B}_\mathrm{k}}$; $z_k \leftarrow \mathrm{round}(\mathrm{c_k})_{\mathcal{B}_\mathrm{k}\setminus \mathrm{z_k^{(0)}}}$;

$\mathrm{dist}_k \leftarrow |r_{k,k}(z_k - c_k)|$;

`/* Temporary variables for computing` c_k

$\widetilde{R} \leftarrow R(1:k, 1:k)$; $\tilde{\mathbf{y}} \leftarrow \bar{\mathbf{y}}(1:k)$;

for $i \leftarrow 1$ **to** $k-1$ **do**

`/* Step 2a: Swap columns` i `and` k `of` $\widetilde{R}$ `with a permutation matrix` $P_{i,k}$ `and return` $\widetilde{R}$ `to upper triangular form with Givens rotations` G_i`. */`

$\overline{R} \leftarrow G_i \widetilde{R} P_{i,k}$;

`/* Step 2b: Let` $\overline{R} = [\bar{r}_{i,j}]$ `and` $\bar{\mathbf{y}} = [\bar{y}_i]$`, and compute` $|r_{k,k}(z_k - c_k)|$`. */`

$\bar{\mathbf{y}}(1:k) \leftarrow G_i \tilde{\mathbf{y}}(1{:}k)$;

$c_k \leftarrow \bar{y}_k/\bar{r}_{k,k}$; $z_i^{(0)} \leftarrow \mathrm{round}(\mathrm{c_k})_{\mathcal{B}_\mathrm{i}}$; $z_i \leftarrow \mathrm{round}(\mathrm{c_k})_{\mathcal{B}_\mathrm{i}\setminus \mathrm{z_i^{(0)}}}$;

$\mathrm{dist}_i \leftarrow |\bar{r}_{k,k}(z_i - c_k)|$;

`/* Step 3: Find out the best way to do the permutation. */`

if $dist_i > dist_k$ **then**

$\mathrm{dist}_k \leftarrow \mathrm{dist}_i$, $j \leftarrow i$;

`/* Define the Babai integer point for the initial` β `in Algorithm 1. */`

$\hat{z}_k \leftarrow z_j^{(0)}$;

end

`/* Collect the information the upper triangular matrix` $R \in \mathbb{R}^{n\times n}$`, the permutation matrix` $P \in \mathbb{Z}^{n\times n}$`, the vector` $\hat{\mathbf{y}} \in \mathbf{R}^n$`, and the permuted intervals` $\mathcal{B}_i$`, for` $i = 1, \ldots, n$`. */`

Interchange columns j and k of P and entries j and k of ℓ and $\mathbf{u}$;

$R(1:k, 1:k) \leftarrow G_j R(1:k, 1:k) P_{j,k}$; $R(1:k, k+1:n) \leftarrow G_j R(1:k, k+1:n)$;

$\hat{\mathbf{y}}(1:k) \leftarrow G_j \hat{\mathbf{y}}(1:k)$;

end

end

`/* Step 4 Information of` $k = 1$ `*/`

$\bar{\mathbf{y}}(1) \leftarrow \bar{\mathbf{y}}(1) - R(1,2)\hat{z}_2$;

$c_1 \leftarrow \bar{y}(1)/r_{1,1}$; $\hat{z}_1 \leftarrow \mathrm{round}(\mathrm{c_1})_{\mathcal{B}_1}$;

end

Algorithm 3: (Boxed-LLL reduction) $[R, P, \hat{\mathbf{y}}, \ell, \mathbf{u}, \hat{\mathbf{z}}] = \mathrm{MCH}(H, \mathbf{y}, \ell, \mathbf{u})$

Input: A matrix $H \in \mathbb{R}^{m\times n}$, a vector $\mathbf{y} \in \mathbb{R}^n$, a lower bound vector $\ell \in \mathbb{Z}^n$ and an upper bound vector $\mathbf{u} \in \mathbb{Z}^n$ collected from the box constraint $\mathcal{B} = \mathcal{B}_1 \times \cdots \times \mathcal{B}_n$ with $\mathcal{B}_i = \{x_i \in \mathbb{Z} : \ell_i \le x_i \le u_i\}$.

Output: An upper triangular matrix $R \in \mathbb{R}^{n\times n}$, a permutation matrix $P \in \mathbb{Z}^{n\times n}$, a vector $\hat{\mathbf{y}} \in \mathbb{R}^m$ for the computation of (8) and (9) with updated lower and upper bounds $\ell \leftarrow P^\top \ell$ and $\mathbf{u} \leftarrow P^\top \mathbf{u}$, respectively, and the Babai integer point $\hat{\mathbf{z}} = [\hat{z}_i]$.

begin

`/* Compute the QR decomposition. */`

Compute the QR decomposition of H to obtain the reduced form in (8), i.e., $H = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix}$;

$S \leftarrow R^{-\top}$; $\hat{\mathbf{y}} \leftarrow Q_1^\top \mathbf{y}$;

`/* Initialize matrices G, generated from a series of Givens rotations, and P, generated from a series of permutations. */`

$G \leftarrow I_n$; $P \leftarrow I_n$;

`/* Information of original` R `and` $\hat{\mathbf{y}}$ `*/`

$\check{R} \leftarrow R$; $\check{\mathbf{y}} \leftarrow \hat{\mathbf{y}}$;

for $k \leftarrow n$ **to** 2 **do**

`/* Compute` $|c_k - z_k|/\|S(k,k)\|_2$ `without doing the permutation*/`

$c_k \leftarrow \hat{\mathbf{y}}(k)^\top S(k,k)$; $z_k^{(0)} \leftarrow \mathrm{round}(\alpha)_{\mathcal{B}_k}$; $z_k \leftarrow \mathrm{round}(\alpha)_{\mathcal{B}_k \setminus z_k^{(0)}}$;

$\mathrm{dist}_k \leftarrow |c_k - z_k|/\|S(k,k)\|_2$;

for $i \leftarrow 1$ **to** $k-1$ **do**

`/* Use (18) to evaluate dist`$_i$`. */`

$c_k \leftarrow \hat{\mathbf{y}}(i:k)^\top S(i:k,i)$; $z_i^{(0)} \leftarrow \mathrm{round}(\alpha)_{\mathcal{B}_i}$; $z_i \leftarrow \mathrm{round}(\alpha)_{\mathcal{B}_i \setminus z_i^{(0)}}$;

$\mathrm{dist}_i \leftarrow |c_k - z_i|/\|S(i:k,i)\|_2$;

`/* Find out the best way to do the permutation. */`

if $dist_i > dist_k$ **then**

$\mathrm{dist}_k \leftarrow \mathrm{dist}_i$; $j \leftarrow i$;

`/* Define the Babai integer point for the initial` β `in Algorithm 1. */`

$\hat{\mathbf{z}}_k \leftarrow z_j^{(0)}$;

end

end

$\hat{\mathbf{y}} \leftarrow \hat{\mathbf{y}} - R(:,j) z_j^{(0)}$;

if $j \neq k$ **then**

`/* Permute columns` k `and` j `by a permutation matrix` $\widehat{P}_{k,j}$ `with` $\widehat{P}_{k,j}$ `obtained by rotating the` j`-th column of an identity matrix of appropriate size to the last column and shifting columns` $j, j+1, \ldots, k$ `to the left one position. */`

$R \leftarrow R\widehat{P}_{k,j}$; $S \leftarrow S\widehat{P}_{k,j}$; $P(1:k,1:k) \leftarrow P(1:k,1:k)\widehat{P}_{k,j}$;

`/* Triangularization */`

$R \leftarrow \widehat{G}_{k,j} R$; $S \leftarrow \widehat{G}_{k,j} S$; $\hat{\mathbf{y}} \leftarrow \widehat{G}_{k,j}\hat{\mathbf{y}}$;

`/* Update` ℓ `and` u`, i.e., exchange constraints in` B_j `and` B_k

$\ell(1:k) \leftarrow \widehat{P}_{k,j}^\top \ell(1:k)$; $\mathbf{u}(1:k) \leftarrow \widehat{P}_{k,j}^\top \mathbf{u}(1:k)$;

`/* Collect Givens rotations */`

$G(1:k,1:k) \leftarrow \widehat{G}_{k,j} G(1:k,1:k)$;

end

`/* Remove the last rows of` R`,` S `and` $\hat{\mathbf{y}}$`, and the last columns of` R `and` S`. */`

$\hat{\mathbf{y}} \leftarrow \hat{\mathbf{y}}(1:k-1)$; $R(:,k) \leftarrow []$; $S(:,k) \leftarrow []$; $R(k,:) \leftarrow []$; $S(k,:) \leftarrow []$;

end

`/* Recover and output` R `and` $\hat{\mathbf{y}}$`, and define` $\hat{z}_1$`. */`

$R \leftarrow G\check{R}Z$; $\hat{\mathbf{y}} \leftarrow G\check{\mathbf{y}}$; $\hat{z}_1 \leftarrow \mathrm{round}(\hat{\mathbf{y}}(1)^\top \mathrm{S}(1,1))_{\mathcal{B}_1}$;

end

Additionally, the algorithm in [6] focuses more on how to perform the column reordering without deriving the final upper triangular matrix R, which is required information for solving ILS problems. Then, we combine the strategies in [8] and [6] and record the process with the details in Algorithm 3.

Combining these reduction and search processes, we propose an ALS approach in Algorithm 4 to solve the IMA. It should be emphasized that the reduction and search algorithms defined in Algorithm 1 and Algorithm 3 work for both unconstrained and contrained ILS problems. However, while solving IMAs with box constraints, the inequality (13) used in Algorithm 1 can be replaced by the inequality (15) to enhance the speed of convergence. This strategy is indeed applied in our numerical experiments in section 4.

Algorithm 4: (IMA) $[U, V] = \text{IMA}(A, V)$

Input: A matrix $A \in \mathbb{Z}^{m\times n}$ and an initial matrix $U \in \mathbb{Z}^{m\times k}$ for the computation of (3a).

Output: $U \in \mathbb{Z}^{m\times k}$ and $V \in \mathbb{Z}^{k\times n}$ as a minimizer of (1).

begin
  repeat
    `/* Update` U`. */`
    for $i = 1, \ldots, m$ **do**
      $[R, Z, \hat{\mathbf{y}}] = \text{MCH}(V^\top, A(i,:)^\top)$;
      $[\mathbf{z}] = \text{SEARCH}(R, \hat{\mathbf{y}})$;
      `/* For this SEARCH algorithm, replace` (13) `by` (15) `for IMA with box constraints. */`
      $U(i,:) \leftarrow (Z\mathbf{z})^\top$
    end
    `/* Update` V`. */`
    for $j = 1, \ldots, n$ **do**
      $[R, Z, \hat{\mathbf{y}}] = \text{MCH}(U, A(:,j))$;
      $[\mathbf{z}] = \text{SEARCH}(R, \hat{\mathbf{y}})$;
      `/* For this SEARCH algorithm, replace` (13) `by` (15) `for IMA with box constraints. */`
      $V(:,j) \leftarrow Z\mathbf{z}$;
    end
  until *a convergence is attained* ;
end

3.1 Properties of Convergence

Now, we have an ALS approach for solving the IMA. Let $\{(U_i, V_i)\}$ be a sequence of optimal matrices obtained from the computation of (3) using Algorithm 4. In this section, we show the sequence $\{\|A - U_iV_i\|_F\}$ is nonincreasing and that

$$\lim_{i\to\infty} \|A - U_iV_i\|_F = \|A - UV\|_F \tag{19}$$

for some particular $U \in \mathbb{Z}^{m\times k}$ and $V \in \mathbb{Z}^{k\times n}$.

To prove that the sequence $\{\|A - U_i V_i\|_F\}$ is nonincreasing, we show that

$$\|A - U_i V_i\|_F \geq \|A - U_{i+1} V_i\|_F \geq \|A - U_{i+1} V_{i+1}\|_F$$

for each i. This amounts to showing that Algorithm 4 computes the optimal solutions of

$$\min_{V(:,j) \in \mathbb{Z}^{k \times 1}} \|A(:,j) - UV(:,j)\|_F$$

and

$$\min_{U(i,:) \in \mathbb{Z}^{1 \times k}} \|A(i,:) - U(i,:)V\|_F$$

for each j and each i, respectively, which can be written as follows.

Theorem 3 *The two processes of the reduction strategy in Algorithm 3 and search strategy in Algorithm 1 provide a global optimal solution to the ILS problem with box constraints.*

Proof We use the notations in section 3 and let $\hat{\mathbf{z}}$ be the vector obtained using the above reduction and search processes. If there exists an integer vector $\mathbf{z} \in \mathcal{B}$ that satisfies the following:

$$\|R\mathbf{z} - \hat{\mathbf{y}}\|_2^2 < \|R\hat{\mathbf{z}} - \hat{\mathbf{y}}\|_2^2,$$

it follows that the integer vector $\mathbf{z}$ satisfies (13) (respectively, (15) with box constraints). That is, the search process can be continued from level $k = 1$ with this new vector $\mathbf{z}$, which contradicts the search strategy.

This theorem indicates that if $A = UV$ and the initial value V_0 of (3a) is equal to V, then

$$\|A - U_1 V_0\|_F = 0$$

after one iteration. The same result holds if we first initialize $U_0 = U$ and iterate (3b). In addition, Theorem 3 implies that $\{\|A - U_i V_i\|_F\}$ is a nonincreasing sequence and hence, converges. The remaining issue is whether the sequence $\{(U_i, V_i)\}$ has at least one limit point. In the optimization analysis, this property does not necessarily hold unless a further constraint is added, such as the boundedness of the feasible region. We might make an a priori assumption that the U_i's and V_i's are restricted to the bounded domains $\mathcal{B}_U$ and $\mathcal{B}_V$ of subsets of $\mathbb{Z}^{m \times k}$ and $\mathbb{Z}^{k \times n}$, respectively. Then, there exists a subsequence $\{U_{i_k}\}$ of $\{U_i\}$ such that $\lim_{k \to \infty} U_{i_k} = U$ for some $U \in \mathbb{Z}^{m \times k}$ and similarly, a subsequence $\{V_{i_{k_\ell}}\}$of $\{V_{i_k}\}$ such that $\lim_{\ell \to \infty} V_{i_{k_\ell}} = V$, for some $V \in \mathbb{Z}^{k \times n}$. This implies that

$$\lim_{i \to \infty} \|A - U_i V_i\|_F = \lim_{\ell \to \infty} \|A - U_{i_{k_\ell}} V_{i_{k_\ell}}\|_F = \|A - UV\|_F.$$

Therefore, (19) holds for some $U \in \mathcal{B}_U$ and $V \in \mathcal{B}_V$. The above analysis also facilitates an agreeable interpretation of the convergence results. That is, once the limit of (19) equals zero, every limit point of the subsequence $\{U_i, V_i\}$ is an exact decomposition of the original matrix A. In this regard, we emphasize that even if the sequences of U_i and V_i converge to some particular U and V, this does not mean that we obtain a local/global minimum for the objective function (1), because we are considering discrete datasets. It would be difficult to define the local minimum in the same terms as those for continuous datasets, and further investigation of this issue is merited.

4 Numerical Experiments

In this section, we conduct seven experiments on integer datasets. In the first test, we randomly generate integer datasets and assess the low-rank approximation in terms of the IMA approach with different initial values. In the second test, we compare our results with those from SVD and NMF approaches by rounding the obtained approximations. In the remaining tests, we illustrate the capacity of our algorithms to perform association rule mining, cluster analysis, and practical dataset analysis.

We performed all computations in this section using MATLAB/version 2016b on MacBook Air with a 1.7 GHZ Intel Core i7 processor and 8 GB of memory. We set the terminating condition of our algorithm as

$$\|V_{\text{new}} - V_{\text{old}}\|_F < n \cdot k \cdot \mathbf{u},$$

where V_{new} and V_{old} represents two successive iterations, m and n correspond to the size of the input matrix, k represents the low rank defined in (1), and $\mathbf{u} = 2^{-52} \cong 2.22e-16$ is the machine zero.

4.1 Random test

Let

$$A = \begin{bmatrix} 16 & 9 & 7 & 12 & 13 \\ 20 & 12 & 8 & 14 & 14 \\ 22 & 12 & 10 & 17 & 19 \\ 22 & 14 & 10 & 16 & 17 \\ 28 & 17 & 13 & 21 & 23 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 1 \\ 2 & 3 & 2 \\ 3 & 4 & 2 \end{bmatrix} \begin{bmatrix} 4 & 1 & 1 & 3 & 3 \\ 2 & 2 & 2 & 2 & 3 \\ 4 & 3 & 1 & 2 & 1 \end{bmatrix}.$$

Using the IMA approach with the constraints $1 \leq u_{i,j}, v_{i,j} \leq 4$, the initial matrix

$$V_0 = \begin{bmatrix} 3 & 1 & 2 & 4 & 2 \\ 4 & 4 & 2 & 2 & 3 \\ 4 & 3 & 1 & 3 & 3 \end{bmatrix},$$

yields the following computed solution:

$$U^* = \begin{bmatrix} 1 & 1 & 2 \\ 3 & 2 & 1 \\ 2 & 1 & 3 \\ 2 & 2 & 2 \\ 3 & 2 & 3 \end{bmatrix}, V^* = \begin{bmatrix} 3 & 1 & 1 & 1 & 3 \\ 4 & 4 & 2 & 3 & 1 \\ 4 & 2 & 2 & 4 & 4 \end{bmatrix},$$

and $\|A - U^*V^*\|_F^2 = 9$.

However, if the initial matrix is as follows:

$$V_0 = \begin{bmatrix} 2 & 3 & 2 & 4 & 1 \\ 3 & 2 & 2 & 1 & 2 \\ 2 & 1 & 4 & 3 & 3 \end{bmatrix},$$

then the computed solution is

$$U^* = \begin{bmatrix} 2\ 2\ 1 \\ 1\ 4\ 1 \\ 3\ 3\ 1 \\ 2\ 4\ 1 \\ 3\ 4\ 2 \end{bmatrix}, V^* = \begin{bmatrix} 3\ 1\ 1\ 3\ 3 \\ 3\ 2\ 1\ 2\ 2 \\ 4\ 3\ 3\ 2\ 3 \end{bmatrix},$$

and $\|A - U^*V^*\|_F^2 = 7$. This is an entirely different result. Although we can see that the sequence $\{\|A - U_iV_i\|_F^2\}$ is convergent, we cannot guarantee that the convergent sequence provides the optimal solution of (1). Figure 2 shows the distribution of the residual $\|A - U^*V^*\|_F^2$ obtained by randomly choosing 100 initial points.

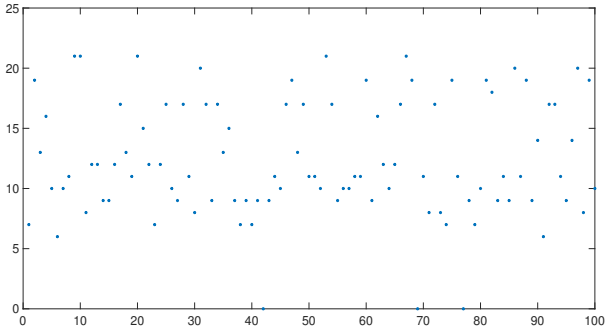


Fig. 2 Distribution of the residuals

From the figure, we can see that although almost all residuals are located within the interval $[5, 20]$, the zero residual indicates a good initial matrix can exactly recover matrix A. Also, the average iteration number is 2.89, the largest iteration number is 6, and the average elapsed time is 0.0125 seconds.

Note that our algorithm is designed for matrices with full column rank. Therefore, in the subsequent iterative processes, if U (or $V^\top$) is not full column rank, we add a small perturbation, say $10^{-12}I$, to the matrix U (or $V^\top$), where I is an identity matrix of appropriate size. However, if matrices U and V are not of full rank, transforming an undetermined ILS problem into an equivalent overdetermined ILS problem [9] might be one way to handle this difficulty and deserves further investigation.

We know that the IMA problem is not a convex problem. The dependence of the initial values is inevitable. In our next example, we show that our algorithm can find a better approximation than those provided by the SVD and NMF approaches.

4.2 Comparison of methods

For a given matrix $A = UV$, where $U = [u_{i,j}] \in \mathbb{Z}^{n\times r}$ and $V = [v_{i,j}] \in \mathbb{Z}^{r\times n}$ are two random integer matrices, and $1 \le u_{i,j}, v_{i,j} \le 4$, we want to find a low-rank

approximation

$$A \approx WH,$$

where $W \in \mathbb{Z}^{n\times r}$ and $H \in \mathbb{Z}^{r\times n}$. Ideally, we want to obtain the solution $W = U$ and $H = V$, but this ideal result can hardly be achieved due to the choice of the initial point.

In Table 2, the values in the IMA column are the results we obtained using our IMA method, and the values in the rSVD and rNMF columns are the results obtained by rounding the real solutions using the MATLAB built-in commands *svd* and *nnmf*, respectively. We obtain the values of rSVD by taking the square roots of the desired singular values and assigning them equally to the corresponding left and right singular vectors before performing the rounding. In this table, we also present the interval (Interval) that contains all of the residual values and the average residual value (Aver.) obtained by randomly choosing 100 initial points, the average number of iterations (it. ♯) performed by our algorithm, and the percentages (Per.) by which the results of our method are superior to those of existing methods with the same initial value.

n	IMA			rSVD	rNMF		Per.
	it. ♯	Interval	Aver.		Interval	Aver.	
20	5.97	[0,446]	292.71	1524	[237912,245144]	244488.32	100%
30	9.35	[0,1491]	1121.96	7091	[1353513,1379591]	1378844.46	100%
40	11.93	[1484,2558]	2141.3	16170	[42200308,4253713]	4251914.85	100%
50	14.44	[3126,4587]	3882.87	28613	[10021329,10106866]	10103425.27	100%

Table 2 Performance of different algorithms for fixed $r = n/5$

Note that the average rounding residual values resulting from SVD and NMF are much greater than those obtained using our IMA approach or even greater than the worst residual value in our experiments, which clearly shows that our method is more accurate than the conventional methods in handling IMA problems.

In Table 3, the second and forth columns list the average elapsed times of our IMA method and the NMF approach, respectively. The third column lists the elapsed times of the SVD approach. We can see that the IMA method cannot perform as fast as the SVD and NMF approaches. But, as far as we know, the IMA method is the only way to solve IMA problems with higher accuracy. In the next five examples, we demonstrate the application of the IMA technique to association rule mining, cluster analysis, and pattern extraction.

n	IMA	rSVD	rNMF
20	0.0930	0.0046	0.0066
30	0.417	0.0057	0.0094
40	1.0142	0.0049	0.0122
50	2.3313	0.0060	0.0160

Table 3 The elapsed times (seconds)

4.3 Association rule mining

Association rule mining is a well-studied method for discovering the embedded relations between variables in a given dataset. For example, in Table 1, we want to predict whether a customer who buys two diapers and one egg will go on to buy a beer. Since each shopping item is inseparable, we record customer shopping behaviors in a discrete system. Conventional approaches for analyzing this dataset use a Boolean expression [31,33] to the effect that we can find a rule such as "$\{\text{diaper}, \text{egg}\} \Rightarrow \{\text{beer}\}$" in the sales data, but we cannot determine how the quantity affects the marketing activities. Therefore, here, we demonstrate how to apply the IMA technique to perform association rule learning with quantity analysis. We use the toy dataset given in Table 1 as an example. The same idea can be applied to an extended file of applications, including Web usage mining, cheminformatics, and intrusion detection with large datasets.

To begin, we choose the two most frequent entries in each column of A as the initial input matrix, for instance:

$$V = \begin{bmatrix} 2 & 1 & 2 & 0 & 2 & 4 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

such that the rows of the matrix can be decomposed according to the pattern that occurs most frequently in A. Upon using our IMA algorithm with box constraints $0 \leq u_{i,j} \leq 2, 0 \leq v_{i,j} \leq 4$, respectively, we can obtain the following approximation:

$$U = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 2 \\ 2 & 0 \\ 0 & 1 \end{bmatrix}, \quad V = \begin{bmatrix} 2 & 1 & 1 & 0 & 2 & 4 \\ 0 & 0 & 2 & 1 & 0 & 1 \end{bmatrix}$$

with the residual $\|A - UV\|_F^2 = 1$ and 0.1389 seconds of computation time. In this case, we can see that not only do we obtain the best approximation, but the obtained results are also more interpretable than those approximated using unconstrained optimization techniques. For example, from the first row of matrix V, we can have information such as the following:

$$\{2 \text{ bags of bread, } 1 \text{ milk, } 1 \text{ diaper}\} \Rightarrow \{2 \text{ chips, } 4 \text{ beers}\}.$$

However, unconstrained optimization techniques with a probability of 1 provide real (or nonnegative) representatives, which makes it hard to address customer' behaviors. In reality, we seldom (or never) sell eggs in fractions, but the fractional representatives in V would confuse the issue of marketing buyer behaviors and consumer needs. Although the recently-proposed Boolean matrix approximation [31,33] can identify the relationships among different products, our IMA method can further provide the quantities embedded in the shopping behaviors.

4.4 Clustering integer data

Cluster analysis has long played an important role in many fields, including pattern recognition, information retrieval, machine learning, bioinformatics, and data mining; see [2,14,15,17,20,28] and the references therein. Its purpose is to divide a set of

observations into subsets, called clusters, such that observations in the same cluster capture a similar structure in the data.

Car evaluation dataset: In this experiment, we demonstrate the application of our IMA algorithm to naturally produce clusters and their corresponding integer representatives for the Car Evaluation dataset from the UCI repository [4], which consists of 1728 instances. Each instance is characterized by seven attributes corresponding to buying price (buying), price of maintenance (maintenance), number of doors (doors), person capacity in terms of the number of persons it can carry (persons), size of the luggage trunk (luggage), safety, and class values (satisfaction). Each attribute corresponds to a specific quantity. To perform the cluster analysis, each attribute is indexed to some *ordinal* value. For example, very high, high, medium, or low prices are designated by the numbers 4, 3, 2, or 1, respectively. Big, medium, or small luggage trunks are designated by the numbers 3, 2, or 1, respectively. Cars with high, medium, or low safety, are numbered 3, 2, or 1, respectively. If the car has five or more doors or has capacity for five or more persons, the number 5 is used to denote these characteristics. The class values, very good, good, acceptable, or unacceptable, are numbered 4, 3, 2, or 1, respectively. See Table 4 for the information of the attributes and the corresponding ordinal values. In total, our input data A comprises a 1728×7 integer matrix. The last column of this matrix is particularly noticeable as it indicates whether the car is unacceptable (unacc), acceptable (acc), good, or very good (vgood), and the number (percentage) of instances per class are 1210 (70.02%), 384 (22.22%), 69 (3.99%), and 65 (3.76%), respectively.

attributes	ordinal values (in brackets)
buying	low(1), medium(2), high(3), very high(4)
maintenance	low(1), medium(2), high(3), very high(4)
doors	2(2), 3(3), 4(4), 5more(5)
persons	2(2), 4(4), more(5)
luggage	small(1), medium(2), large(3)
safety	low(1), medium(2), high(3)
satisfication	unacc(1), acc(2), good(3), vgood(4)

Table 4 Table of attributes and ordinal values

In our experiment, we applied the IMA algorithm to the 1728×6 submatrix after removing the last column from A. We divide A into four clusters by randomly picking four rows of A from each class as our initial value and, to satisfy the desired upper and lower bounds for each class, we require the obtained matrices $U = [u_{i,j}]$ and $V = [v_{i,j}]$ with entries $u_{i,j} \in \{0, 1\}$ and $1 \leq v_{i,j} \leq 5$, $i = 1, \ldots, m$ and $j = 1, \ldots, n$. The elapsed times required for this computation is around 1.7538 seconds.

To gauge the effectiveness of our algorithm, we applied two parameters, Precision and Recall [10,12], as follows:

$$\mathsf{Precision} := \frac{\sharp \text{ of (relevant data} \cap \text{retrieved data)}}{\sharp \text{ of retrieved data}},$$

$$\mathsf{Recall} := \frac{\sharp \text{ of (relevant data} \cap \text{retrieved data)}}{\sharp \text{ of relevant data}}.$$

These two parameters served as measures of statistical variability. In other words, Precision determined the ratio in which relevant data, which was mixed in with the

other class values, was recalled in the retrieved data, i.e., the precision of the recovery. Recall determined the ratio in which the retrieved data was relevant to a desired search (for example, looking for cars with a very good class value).

These clusters can then be interpreted in terms of the so-called *confusion matrix* in Table 5 with respect to whether the car was unacceptable, acceptable, good, or very good. As we can see, each row of U contains only one nonzero entry and the four representatives $\mathbf{v}_1, \ldots, \mathbf{v}_4$, and $\mathbf{v}_1$ in particular, have high precision with respect to the unacceptable class. In this experiment, we randomly identified an initial value by choosing representatives from four different class values. Our purpose was to obtain four representatives that could represent the original dataset effectively without knowing beforehand the distribution of the dataset. As we can see in Table 5, the Recall values suggest the possibility that the third cluster (model) will give rise to higher "good" customer expectations and the four cluster (model) to higher "very good" customer expectations. The zero values shown in Table 5 suggest that if automobile manufacturers want to build a car that will attract and interest customers, they must avoid models similar to cluster $\mathbf{v}_2$. Finally, we stress that there is also unknown information hidden in the obtained clusters that merits further investigation.

	Retrieved dataset			
	$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$	$\mathbf{v}_4$
unacc	329	339	285	257
acc	109	51	101	123
good	8	0	41	20
vgood	13	0	14	38
Cardinality of $\mathbf{v}_i$	459	390	441	438
Precision of unacc	0.7168	0.8692	0.6463	0.5868
Recall of unacc	0.2719	0.2802	0.2355	0.2124
Precision of acc	0.2375	0.1308	0.2290	0.2808
Recall of acc	0.2839	0.1328	0.2630	0.3203
Precision of good	0.0174	0	0.0930	0.0457
Recall of good	0.1159	0	0.5942	0.2899
Precision of vgood	0.0283	0	0.0317	0.0868
Recall of vgood	0.2000	0	0.2154	0.5846

Table 5 Precision and Recall of car evaluation

4.5 Pattern extraction

In this section, we report the effectiveness of our method in extracting integer patterns by one artificial dataset and two image datasets.

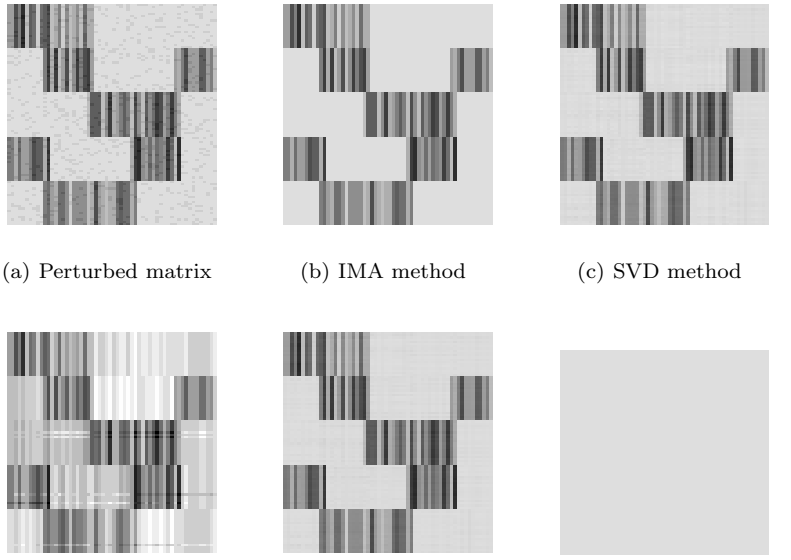
Artificial dataset: We generate one sampling matrix $PerA$ by perturbing a given block integer matrix $A = [a_{i,j}]$ with a random binary matrix $ErrA$. The given integer matrix consists of five groups of rows, and each group contains one or two blocks. In total, there are seven blocks, and each block, having 20 rows, is generated with the same random row vector, which is called a *generation vector* and obtained by the MATLAB command round(10*rand(1,column_number)). Here, we list these seven

generation vectors from top to bottom and from left to right as follows:

$$\begin{cases} 4\ 4\ 9\ 6\ 11\ 2\ 8\ 7\ 1\ 7\ 8\ 9\ 2\ 3\ 6\ 1\ 5\ 2\ 3\ 5\ 2\ 7\ 3\ 3 \\ 6\ 10\ 3\ 10\ 2\ 4\ 9\ 2\ 5\ 10\ 6\ 9\ 10\ 4 \\ 4\ 3\ 6\ 9\ 4\ 4\ 4\ 8\ 8\ 4\ 6\ 2 \\ 2\ 8\ 8\ 8\ 2\ 5\ 10\ 3\ 4\ 6\ 8\ 6\ 2\ 9\ 5\ 2\ 8\ 5\ 7\ 10\ 9\ 4\ 6\ 10\ 6\ 3 \\ 6\ 6\ 3\ 7\ 4\ 4\ 6\ 9\ 8\ 8\ 3\ 10 \\ 3\ 10\ 11\ 5\ 4\ 7\ 4\ 7\ 6\ 5\ 10\ 7\ 2\ 11 \\ 6\ 7\ 4\ 3\ 7\ 7\ 5\ 1\ 5\ 5\ 5\ 5\ 2\ 3\ 9\ 4\ 3\ 2\ 3\ 3\ 7\ 8\ 7\ 7\ 3\ 6. \end{cases}$$

Our goal in this test is to reveal the blocks embedded in this 100×58 matrix with the computed matrices $U = [u_{i,j}] \in \mathbb{Z}^{100\times 5}$ and $V = [v_{i,j}] \in \mathbb{Z}^{5\times 58}$ satisfying $1 \leq u_{i,j}, v_{i,j} \leq \max_{i,j} a_{i,j}$. As the discussion in [32], the essence of our work is to discover patterns hidden in the original data. However, we employ an entirely different strategy to reveal the underlying patterns in integer entries, whereas only binary patterns can be captured by the algorithm in [32]. We obtain the perturbed matrix $PerA$ using the following MATLAB command:

$$ErrA = round(rand(100, 58)/1.7); \quad PerA = A + ErrA.$$



(a) Perturbed matrix (b) IMA method (c) SVD method

(d) SVD method (integer) (e) NMF method (f) NMF method (integer)

Fig. 3 Approximations provided by different approaches

Figure 3(a) shows the sparsity of the matrix under consideration. Figure 3(b) shows the rank-5 approximation provided by our method. As shown in the figure, our method can discover all seven blocks in the matrix as well as the vector contained in each group. The Frobenius norm of the error of the approximation is 885, which is the

exact Frobenius norm of the matrix $ErrA$. This result shows that not only can our method find all the blocks, but it can also recover the values of the entries in the block. Figure 3(c) shows the rank-5 approximation obtained using the SVD approach. This result is optimal in the sense of real numbers, and the residual is 642.50, which is less than that provided by our method. However, a smaller residual does not imply better recovery. In fact, the Frobenius norm of the $ErrA$ is 885 and the residual of a desired approximation should be 885. More importantly, the approximation obtained using the SVD approach is not sufficiently useful for applications due to the limitations of integers. One available means of obtaining the integer approximation is to round the first five dominant components. The residual obtained in this manner is 9084, which is far greater than that provided by our method. From Figure 3(d), we see that this approach cannot reveal the blocks in the matrix. Similar to the SVD method, the residual of the approximation using the real NMF approach is 666.21, but if we add the integer limitation, this residual increases to 103805. From Figures 3(d) and 3(f), we see that after performing the rounding procedures, the SVD and NMF approaches, respectively, cannot successfully separate the blocks. In fact, if we apply the MATLAB command *nnmf* to obtain the real approximation of UV, we find all entries of the matrix V located in the interval $[0, 0.5)$. Therefore, when we use rounding to obtain the integer approximation, the rounded matrix V is the zero matrix and the approximated matrix is the zero matrix, as shown in Figure 3(f). Also, the required elapsed times of the IMA, SVD, and NMF methods are 0.6166, 0.1397, and 0.0979, respectively.

Image datasets: In image science, images are generally treated as a 2-dimensional array of pixels values. For example, in a gray scale image, every element of a digital image is represented by an integer between 0 and 255 (or a nonnegative decimal number between $[0, 1]$). In these experiments, let $A \in \mathbf{Z}^{m\times n}$ denote a collection of m images each of which is represented by a column of n pixels. We test our IMA algorithm on the Swimmer dataset [13] and the MIT CBCL faces dataset [1] and show that the integer factorization $A = UV$ can provide sparse (or even part-based) representations. Thus, the integer factorization $A = UV$ provides us a way to capture and classify the "intrinsic" characteristics embedded in the observed objects. Unfortunately, the advantages of the IMA algorithm on large datasets come at the price of much elapsed times. But, we believe that the IMA algorithm can be applied to other extended fields, such as the quantitative structure-activity relationship (QSAR) discovery in chemoinformatics with the aid of parallel computing techniques.

Like the discussions in [18,23], we apply the conventional sparsity [18]

$$s(A) = \frac{\sharp \text{ of zeros}(A)}{mn} \in [0, 1]$$

for an $m \times n$ matrix A, and the sparsity

$$sh(\mathbf{x}) = \frac{\sqrt{n} - \|\mathbf{x}\|_1/\|\mathbf{x}\|_2}{\sqrt{n} - 1} \in [0, 1],$$

proposed and defined by Hoyer [23], for any nonzero n dimensional vector $\mathbf{x}$ to measure the sparsity of the extracted patterns.

The first example is the **Swimmer dataset** characterized in [13] consisting of a set of black-and-white stick figures satisfying the so called *separable factorial articulation criteria*. Each figure, placed in a frame of 32×32 pixels, displays a figure with one static part (torso) and four moving parts (limbs). Each limb points to one of four different directions and thus gives rise to 256 figures in this collection. Figure 4, for example,

shows 50 sample images from the Swimmer dataset. After vectorizing each image in the Swimmer dataset into a column, we have the target matrix A of size 1024×256 with entires of only two integer values, 0 and 255. We convert without loss of generality the color values, 0 and 255, into 0 and 1. The question is whether we are able to compress images by inducing sparse representatives, i.e., representatives with relatively many zeros. An intuitive thought is to decompose each figure by a sparse, part-based representation, that is, each representation should be able to identify original image immediately and clearly, not by superposition of multiple features. Taking this thought and numerical rank of the original matrix A (which is 13) into account, we require, for example, the approximate matrices U and V to be in $Z_2^{256\times 5}$ and $Z_2^{5\times 1024}$. The 5 rows of the initial matrix V for the IMA and NMF approaches are randomly selected from rows of the matrix A.

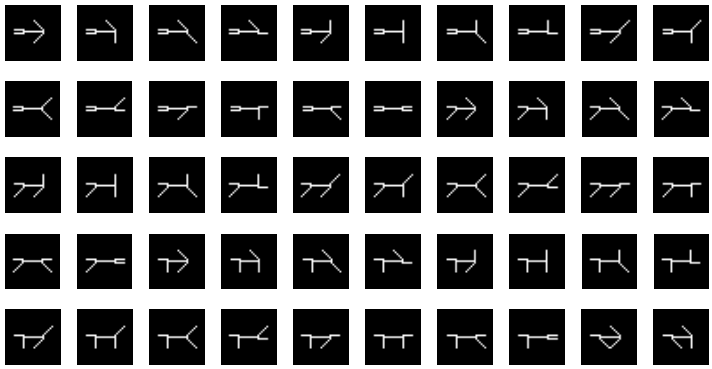


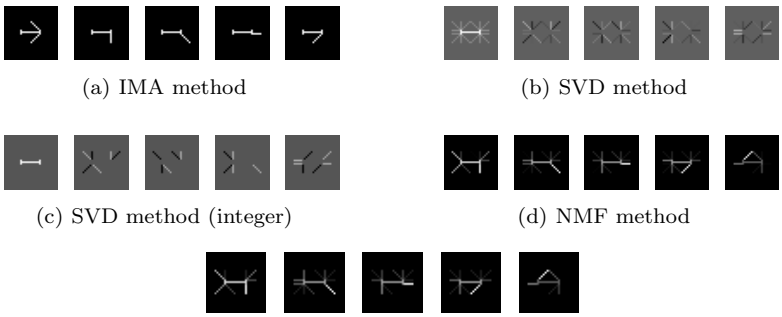
Fig. 4 50 sample images from the swimmer database.

After using the IMA, NMF, and SVD approaches, we reshape each row of the resulting matrices V into 32×32 matrices. Figure 5 displays basis images obtained by the IMA, NMF and SVD with or without rounding procedures. Also, the first column of Table 6 reports for the five methods the relative error (Rel) of the computed solutions:

$$\text{Rel} := \frac{\|A - UV\|_F}{\|A\|_F}$$

and the remaining four columns of Table 6 reports the corresponding sparsity measures, $s(\cdot)$ and $sh(\cdot)$, of factors U and V.

In Table 6, we see that the error caused by the IMA approach is less than those by the SVD and NMF approaches with rounding procedure, but higher than those without rounding procedure. However, if we use the MATLAB command *imshow* to display the extracted features, we see that the IMA approach clearly display 13 part-based images, while the SVD and NMF approaches can only provide blurring ones. See Figure 5 for the five images extracted by the IMA, SVD, and NMF approaches, respectively. Also, Table 6 shows that our IMA approach not only provides clear representatives, but compresses the input data very well; that is, our IMA approach has lower $s(\cdot)$



(a) IMA method (b) SVD method

(c) SVD method (integer) (d) NMF method

(e) NMF method (integer)

Fig. 5 Approximations provided by different approaches

values and higher $sh(\cdot)$ values. The zero value and NaN value shown in the row of the NMF(integer) are due to the fact that the matrix V by the NMF approach with rounding procedure is a zero matrix.

	Rel	$s(U)$	$s(V)$	$sh(U)$	$sh(V)$
IMA	0.63874	0.0215	0.2000	0.8655	0.5687
SVD	0.52705	0.8187	1	0.7889	0.1777
SVD(integer)	1.0159	0.0256	0.2469	0.8584	0.5176
NMF	0.54072	0.0674	0.7445	0.8086	0.3488
NMF(integer)	1	0.0596	0	0.8087	NaN

Table 6 The relative error and sparsity measures for the problem $\|A - UV\|_F$

As we know, 17 basic parts make up these 256 swimmers. Taking these basic parts into account, we should expect the possibility of recovering these 17 parts. However, if we start with the initial matrix V_0 of size $\mathbb{Z}^{17\times 256}$, the IMF and NMF approaches will be in trouble. This is because V_0 is not full rank anymore. Thus, unlike previous rank-5 approximation, we expect to extract each part one by one through rank-one approximation; that is, we search for 17 column and row vectors $\mathbf{u}_i$ and $\mathbf{v}_i$, $i = 1, \ldots, 17$, to approximate the data matrix A such that the error

$$\|A - \sum_{i=1}^{17} \mathbf{u}_i \mathbf{v}_i\|_F$$

is minimized and the column vectors $\mathbf{u}_i$, $i = 1, \ldots, 17$, can depict well the original patterns. For example, after the first approximation, we have two vectors $\mathbf{u}_1$ and $\mathbf{v}_1$. We then subtract $\mathbf{u}_1\mathbf{v}_1$ from the matrix A, and continue another rank-one approximation to the remaining matrix $A - \mathbf{u}_1\mathbf{v}_1$, until 17 column and row vectors $\mathbf{u}_i$ and $\mathbf{v}_i$, $i = 1, \ldots, 17$, are found. In our IMA algorithm, the computations of vectors $\mathbf{u}_k$ and $\mathbf{v}_k$, for $k \in \{1, \ldots, 17\}$, are done by selecting the initial vector V_0 as the row vector with the maximal sum in $A - \sum_{i=1}^{k-1} \mathbf{u}_i\mathbf{v}_i$. Figure 6 depicts that our IMA approach can recover all 16 limbs and one torso, which are completely disjointed from each other. However, the SVD approaches provide only blurring images, and the NMF approach even fail to

obtain $\mathbf{u}_k$ and $\mathbf{v}_k$, for some $k < 17$, since $A - \sum_{i=1}^{k-1} \mathbf{u}_i\mathbf{v}_i$ is not nonnegative anymore. Also, it must be emphasized that the vectors $\mathbf{u}_i$ and $\mathbf{v}_i$, $i = 1 \ldots, 17$, obtained by our IMA algorithm satisfying $A = \sum_{i=1}^{17} \mathbf{u}_i\mathbf{v}_i$ exactly.

(a) IMA method (b) SVD method

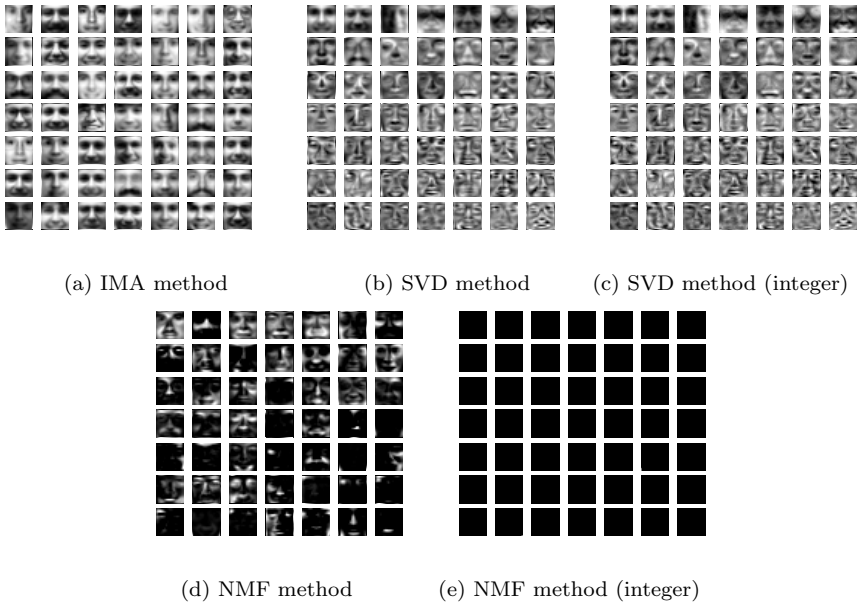
Fig. 6 Approximations provided by vectors $\mathbf{v}_i$'s

The second example is the **MIT CBCL face dataset**, consisting of $m = 2429$ gray-level facial images, each placed in a frame of $n = 19 \times 19$ pixels, and resulting in an $m \times n$ matrix A. This dataset has been widely investigated at the Center for Biological and Computational Learning at MIT. In the NMF and SVD approaches, the approximate decompositions of $A \approx UV$ provide such interpretation that each face (row of A) is approximated by the basic elements (rows of V) with the strength of the approximation identified by the corresponding row of U. In the IMA approach, the approximation $A \approx UV$ is achieved by an integer-valued pixels, whereas the NMF and SVD approaches cannot guarantee the satisfaction of the integer constraints. Therefore, unlike NMF and SVD approaches, we would like to discover a set of basic images, each of which is a whole face. We thus assume entries $u_{i,j}$ of U and entries $v_{i,j}$ of V satisfying $0 \leq u_{i,j} \leq 1$ and $0 \leq v_{i,j} \leq 255$. To start our iterations, we randomly select 49 rows from the data matrix A.

On the sparsity of the matrix U, from Table 7, we can find our IMA approach does provide a smaller sparsity measure $s(U)$ comparing with other approaches. On the sparsity of the matrix V, the IMA approach and the SVD approach both provide larger sparsity measure $s(V)$, comparing with the NMF approach. However, if we check the original data matrix, we can see that all entries of A are not zero. Therefore, a dense matrix V is acceptable. On the relative error, our IMA approach is larger than the SVD approach but smaller than the NMF approach, however, if we reshape the reconstructed row vectors of the matrix V to images, we can see that a set of whole-face images is discovered by the IMA approach, while fewer complete images and several incomplete, part-based representations are extracted by the SVD and NMF approaches. That might suggest that the SVD and NMF approaches are incapable of capturing the representative images immediately, let alone derive integer representatives; see Figure 7(c) and (e) for example.

	Rel	$s(U)$	$s(V)$	$sh(U)$	$sh(V)$
IMA	0.18948	0.0259	1	0.8416	0.0753
SVD	0.07515	1	1	0.3555	0.3422
SVD(integer)	0.09893	0.6261	0.8566	0.3874	0.3460
NMF	0.28535	0.6960	0.5746	0.3906	0.4659
NMF(integer)	0.99845	0.6942	1.6960e-04	0.3906	0.9945

Table 7 The relative error and sparsity measures for the problem $\|A - UV\|_F$



(a) IMA method (b) SVD method (c) SVD method (integer)

(d) NMF method (e) NMF method (integer)

Fig. 7 Approximations provided by reconstructed row vectors of the matrix V

5 Conclusions

Matrix factorization has been a long-standing technique used in data analysis due to its capacity to extract useful information, support decision-making, and assist in drawing conclusions from a given dataset. Conventional techniques developed thus far focus mostly on continuous datasets with real or nonnegative entries and cannot be directly applied to handle discrete datasets. Using the IMA technique, the principal contribution of this work is to offer an effective approach for examining in detail the constitution or structure of discrete information. The results of our numerical experiments suggest that our IMA approach works very well in the low-rank approximation of integer datasets. However, as a newly developed technique, we believe many interesting open issues still remain to be unsolved, not only those for real-life applications, but also from theoretical and algorithmic perspectives. For example, if each column in Table 1 is referenced to the ratings of an item among a group of users, we have a record for the broad scenarios of recommender systems, such as Netfix or MovieLens, where the existing ratings are represented in a matrix. That is, assume we have 5 users and 6 items with integer ratings ranging from 1 to 8, a zero value, which is usually denoted by a hyphen in recommender systems, indicates that the user has not rated the movie. By extending from the concepts of the IMA, we can directly solve this problem in terms of integer representatives.

Also, our approach is based on column-by-column/row-by-row approximation, with the calculation of each column/row being independent of the others. This implies that we can utilize parallel computation, as applied in [25], to speed up calculations when analyzing large-scale data matrices.

References

1. CBCL face database ♯ 1, MIT center for biological and computation learning (2000). URL http://www.ai.mit.edu/projects/cbcl
2. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data. Data Mining and Knowledge Discovery **11**(1), 5–33 (2005)
3. Agrell, E., Eriksson, T., Vardy, A., Zeger, K.: Closest point search in lattices. IEEE Transactions on Information Theory **48**(8), 2201–2214 (2002). DOI 10.1109/TIT.2002.800499
4. Asuncion, A., Newman, D.: UCI machine learning repository (2007). URL http://www.ics.uci.edu/∼mlearn/MLRepository.html
5. Banerjee, A., Krumpelman, C., Ghosh, J., Basu, S., Mooney, R.J.: Model-based overlapping clustering. In: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, KDD '05, pp. 532–537. ACM, New York, NY, USA (2005). DOI 10.1145/1081870.1081932. URL http://doi.acm.org/10.1145/1081870.1081932
6. Breen, S., Chang, X.W.: Column reordering for box-constrained integer least squares problems. http://arxiv.org/abs/1204.1407 (2012)
7. Chang, X.W., Golub, G.H.: Solving ellipsoid-constrained integer least squares problems. SIAM J. Matrix Anal. Appl. **31**(3), 1071–1089 (2009). DOI 10.1137/060660680. URL http://dx.doi.org/10.1137/060660680
8. Chang, X.W., Han, Q.: Solving box-constrained integer least squares problems. IEEE Transactions on Wireless Communications **7**(1), 277–287 (2008). DOI 10.1109/TWC.2008.060497
9. Chang, X.W., Yang, X.: An efficient regularization approach for underdetermined mimo system decoding (2007). URL http://www.cs.mcgill.ca/ chang/pub/ChaY07a.pdf
10. Chowdhury, G.G.: Introduction to modern information retrieval. 2nd ed. facet publishing (2004)
11. Chu, M.T., Lin, M.M.: Low-dimensional polytope approximation and its applications to nonnegative matrix factorization. SIAM J. Sci. Comput. **30**(3), 1131–1155 (2008). DOI 10.1137/070680436. URL http://dx.doi.org.prx.library.gatech.edu/10.1137/070680436
12. Cover, T.M., Thomas, J.A.: Elements of information theory, second edn. Wiley-Interscience [John Wiley & Sons], Hoboken, NJ (2006)
13. Donoho, D., Stodden, V.: When does nonnegative matrix factorization give a correct decomposition into parts? In: Proc. 17th Ann. Conf. Neural Information Processing Systems. NIPS, Stanford University, Stanford, CA, 2003 (2003)
14. Eisen, M.B., Spellman, P.T., Brown, P.O., Botstein, D.: Cluster analysis and display of genome-wide expression patterns. Proceedings of the National Academy of Sciences **95**(25), 14,863–14,868 (1998)
15. Elhamifar, E., Vidal, R.: Sparse subspace clustering: Algorithm, theory, and applications. IEEE transactions on pattern analysis and machine intelligence **35**(11), 2765–2781 (2013)
16. van Emde-Boas, P.: Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Report. Department of Mathematics. University of Amsterdam. Department, Univ. (1981). URL http://books.google.com.tw/books?id=tCQiHQAACAAJ
17. Filipovych, R., Resnick, S.M., Davatzikos, C.: Semi-supervised cluster analysis of imaging data. NeuroImage **54**(3), 2185–2197 (2011)
18. Gillis, N., Glineur, F.: Using underapproximations for sparse nonnegative matrix factorization. Pattern Recognition **43**(4), 1676 – 1687 (2010). DOI https://doi.org/10.1016/j.patcog.2009.11.013. URL http://www.sciencedirect.com/science/article/pii/S0031320309004324
19. Golub, G.H., Van Loan, C.F.: Matrix computations, fourth edn. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD (2013)
20. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of machine learning research **3**, 1157–1182 (2003)
21. Hassibi, A., Boyd, S.: Integer parameter estimation in linear models with applications to gps. In: Decision and Control, 1996., Proceedings of the 35th IEEE Conference on, vol. 3, pp. 3245–3251 vol.3 (1996). DOI 10.1109/CDC.1996.573639
22. Houseman, E.A., Accomando, W.P., Koestler, D.C., Christensen, B.C., Marsit, C.J., Nelson, H.H., Wiencke, J.K., Kelsey, K.T.: Dna methylation arrays as surrogate measures of cell mixture distribution. BMC Bioinformatics **13**(1), 86 (2012). DOI 10.1186/1471-2105-13-86. URL http://dx.doi.org/10.1186/1471-2105-13-86

23. Hoyer, P.O.: Non-negative matrix factorization with sparseness constraints. The Journal of Machine Learning Research **5**, 1457–1469 (2004)
24. Kabán, A., Bingham, E.: Factorisation and denoising of 0–1 data: a variational approach. Neurocomputing **71**(10), 2291–2308 (2008)
25. Kannan, R., Ishteva, M., Park, H.: Bounded matrix low rank approximation. In: Data Mining (ICDM), 2012 IEEE 12th International Conference on, pp. 319–328 (2012). DOI 10.1109/ICDM.2012.131
26. Kawamoto, T., Hotta, K., Mishima, T., Fujiki, J., Tanaka, M., Kurita, T.: Estimation of single tones from chord sounds using non-negative matrix factorization. Neural Network World **3**, 429–436 (2000)
27. Kim, H., Park, H.: Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. SIAM J. Matrix Anal. Appl. **30**(2), 713–730 (2008)
28. Kim, H., Park, H., Drake, B.L.: Extracting unrecognized gene relationships from the biomedical literature via matrix factorizations. BMC Bioinformatics **8(Suppl 9)**, S6 (2007)
29. Kim, J., He, Y., Park, H.: Algorithms for nonnegative matrix and tensor factorizations: a unified view based on block coordinate descent framework. J. Global Optim. **58**(2), 285–319 (2014). DOI 10.1007/s10898-013-0035-4. URL http://dx.doi.org/10.1007/s10898-013-0035-4
30. Kim, J., Park, H.: Fast nonnegative matrix factorization: an active-set-like method and comparisons. SIAM J. Sci. Comput. **33**(6), 3261–3281 (2011). DOI 10.1137/110821172. URL http://dx.doi.org/10.1137/110821172
31. Koyutürk, M., Grama, A.: Proximus: a framework for analyzing very high dimensional discrete-attributed datasets. In: KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 147–156. ACM, New York, NY, USA (2003). DOI http://doi.acm.org/10.1145/956750.956770
32. Koyuturk, M., Grama, A., Ramakrishnan, N.: Compression, clustering, and pattern discovery in very high-dimensional discrete-attribute data sets. IEEE Transactions on Knowledge and Data Engineering **17**(4), 447–461 (2005). DOI 10.1109/TKDE.2005.55
33. Koyutürk, M., Grama, A., Ramakrishnan, N.: Nonorthogonal decomposition of binary matrices for bounded-error data compression and analysis. ACM Trans. Math. Software **32**(1), 33–69 (2006). DOI 10.1145/1132973.1132976. URL http://dx.doi.org/10.1145/1132973.1132976
34. Liao, J.C., Boscolo, R., Yang, Y.L., Tran, L.M., Sabatti, C., Roychowdhury, V.P.: Network component analysis: Reconstruction of regulatory signals in biological systems. Proceedings of the National Academy of Sciences **100**(26), 15,522–15,527 (2003). DOI 10.1073/pnas.2136632100. URL http://www.pnas.org/content/100/26/15522.abstract
35. Lin, M.M.: Discrete eckart-young theorem for integer matrices. SIAM Journal on Matrix Analysis and Applications **32**(4), 1367–1382 (2011). DOI 10.1137/10081099X. URL http://epubs.siam.org/doi/abs/10.1137/10081099X
36. Meeds, E., Ghahramani, Z., Neal, R.M., Roweis, S.T.: Modeling dyadic data with binary latent factors. Advances in neural information processing systems **19**, 977 (2007)
37. Morgan, S.D.: Cluster analysis in electronic manufacturing. Ph.D. dissertation, North Carolina State University, Raleigh, NC 27695. (2001)
38. Mow, W.H.: Universal lattice decoding: a review and some recent results. In: Communications, 2004 IEEE International Conference on, vol. 5, pp. 2842–2846 Vol.5 (2004). DOI 10.1109/ICC.2004.1313048
39. Segal, E., Battle, A., Koller, D.: Decomposing gene expression into cellular processes. In: In Proceedings of the 8th Pacific Symposium on Biocomputing (2003)
40. Slawski, M., Hein, M., Lutsik, P.: Matrix factorization with binary components. In: Advances in Neural Information Processing Systems, pp. 3210–3218 (2013)
41. Su, K., Wassell, I.: A new ordering for efficient sphere decoding. In: Communications, 2005. ICC 2005. 2005 IEEE International Conference on, vol. 3, pp. 1906–1910 Vol. 3 (2005). DOI 10.1109/ICC.2005.1494671
42. Tu, S., Chen, R., Xu, L.: Transcription network analysis by a sparse binary factor analysis algorithm. J. Integrative Bioinformatics **9** (2012)
43. van der Veen, A.J.: Analytical method for blind binary signal separation. IEEE Transactions on Signal Processing **45**(4), 1078–1082 (1997). DOI 10.1109/78.564198
44. Zhang, Z., Li, T., Ding, C., Zhang, X.: Binary matrix factorization with applications. In: Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on, pp. 391–400. IEEE (2007)