2024 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 22–25, 2024, LONDON, UK

BOOLEAN LOW-RANK MATRIX COMPRESSED SENSING

Anonymous

Anonymous

ABSTRACT

In real-world datasets, leveraging the low-rank and sparsity properties enables developing efficient algorithms across a diverse array of data-related tasks, including compression, compressed sensing, matrix completion, etc. Notably, these two properties often coexist in certain real-world datasets such as Boolean datasets and quantized real-valued datasets. To harness the advantages of low-rank and sparsity simultaneously, we adopt a technique inspired by compressed sensing and Boolean matrix completion. Our approach entails compressing a low-rank sparse Boolean matrix by performing inner product operations with a randomly generated Boolean matrix. We then propose two decoding algorithms based on linear programming and message-passing techniques to recover the original matrix. Our experiments demonstrate superior recovery performance of our proposed algorithms compared to Boolean compressed sensing and Boolean matrix completion, with equal measurement requirements.

1. INTRODUCTION

In processing real-valued data, commonly used techniques such as compressed sensing [1] and matrix completion [2] leverage the inherent sparsity of data in order to improve the efficiency. Compressed sensing involves transforming structures such as 2D images into vectors [3, 4], potentially altering the original matrix's structure and information. Sparsity in this context means having many zero elements in a specific domain within the vector. In matrix completion, sparsity is related to low-rank properties, indicating sparsity in the singular value vector of the original matrix.

The literature on Boolean matrix factorization and completion, see, e.g., [5], tends to focus primarily on the low-rank property, even when dealing with datasets that exhibit both low-rank and sparsity. This approach often overlooks valuable information that could be harnessed effectively given the inherent sparsity of the data. In recommender systems and collaborative filtering, matrices often represent data quantized into a small finite set of integers, e.g., in the Netflix problem continuous values are quantized into integers between 1 and 5. However, the true underlying values may exceed this range, resulting in introducing quantization noise to the dataset [6].

Anonymous.

In matrix completion and one-bit matrix completion literature [7, 6], the focus shifts from predicting exact ratings to identifying whether a user is interested in a movie based on ratings above the global average or not. This often results in a Boolean preference matrix (interested/not interested). Yet, users often provide moderate ratings (e.g., 2, 3, 4) when they lack strong preferences, leading to the potential sparsity in the Boolean matrix. For instance, in Table 1, considering ratings of 1 as *disliked* and transforming higher ratings to 0 results in a sparse matrix with a density of 6.11% in the MovieLens-100K dataset. This phenomenon holds in the MovieLens-1M dataset as well. Importantly, such transformations maintain the low rank of the matrix, aligning with the assumptions in the matrix completion literature [2].

Motivated by the aforementioned applications, in this work, we provide a framework aimed at improving efficiency and accuracy in scenarios involving sparse data with low-rank structures by harnessing the concurrent low-rank and sparsity characteristics. More specifically, we consider the problem of compressing a Boolean matrix that is simultaneously sparse and low-rank by leveraging random Boolean matrices as sensing matrices. This problem is referred to as **Boolean matrix compressed sensing (BMCS)**. In Section 2, we formally formulate the problem and provide an upper bound on the number of sparse and low-rank matrices. We then propose a linear programming algorithm for recovery of a sparse Boolean matrix in Section 3 and a message-passing algorithm in Section 4 to address this problem. Various experiments are also provided throughout the paper.

2. PROBLEM FORMULATION AND UPPER BOUND

In our setup, all the arithmetic operations are over the Boolean semi-ring $(\mathbb{B}, \vee, \wedge)$, which means the addition "$+$" operator is replaced by the logical OR operator "$\vee$" and the multiplication "$\times$" operator is replaced by the logical AND operator "$\wedge$". With this pair of arithmetic operations, the matrix multiplication can be defined. For a $\boldsymbol{X} \in \{0,1\}^{m\times k}$ and a $\boldsymbol{Y} \in \{0,1\}^{k\times n}$, we use $\boldsymbol{X} \bullet \boldsymbol{Y} = \boldsymbol{Z}$ to denote Boolean matrix multiplication, which means $[\boldsymbol{Z}]_{i,j} = \bigvee_{\kappa\in[k]} [\boldsymbol{X}]_{i,\kappa} \wedge [\boldsymbol{Y}]_{\kappa,j}$. As in [8, 9], the Boolean rank is defined as $\text{rank}_{\mathcal{B}}(\boldsymbol{Z}) \triangleq \min\{k | \boldsymbol{Z} = \boldsymbol{X} \bullet \boldsymbol{Y}, \boldsymbol{X} \in \{0,1\}^{m\times k}, \boldsymbol{Y} \in \{0,1\}^{k\times n}\}$.

Specifically, our goal is to recover a sparse Boolean ma-

Table 1: Some characteristics of the real-world data sets.

Dataset	Size	Density(%)
Abstracts [1]	4894×12841	0.9
DBLP[2]	19×6980	12.95
20Newsgroups[3](> 0)	7505×61188	0.21
20Newsgroups(> 1)	7505×61188	0.06
Movie-Lense-100K[4](≤ 2)	100000	17.48
Movie-Lense-100K(≤ 1)	100000	6.11
Movie-Lense-1M(≤ 2)	1000000	16.37
Movie-Lense-1M(≤ 1)	1000000	5.62

trix $\boldsymbol{Z}$ with rank k ($k \ll m, n$) from d Boolean measurements. It is well-known that determining the Boolean rank of a matrix is NP-hard [10] and no efficient approximation algorithm has been discovered [11]. To impose a rank constraint of k on matrix $\boldsymbol{Z}$, a straightforward approach is to search for matrices $\boldsymbol{X} \in \{0,1\}^{m\times k}$ and $\boldsymbol{Y} \in \{0,1\}^{k\times n}$ such that $\boldsymbol{X} \bullet \boldsymbol{Y} = \boldsymbol{Z}$, i.e., solving the following optimization problem:

$$\min \|\boldsymbol{Z}\|_0 \tag{1a}$$
$$\text{s.t.} \quad \mathcal{A}(\boldsymbol{Z}) = \boldsymbol{u} \tag{1b}$$
$$\boldsymbol{X} \bullet \boldsymbol{Y} = \boldsymbol{Z} \tag{1c}$$
$$\boldsymbol{X} \in \{0,1\}^{m\times k}, \boldsymbol{Y} \in \{0,1\}^{k\times n}, \boldsymbol{u} \in \{0,1\}^{d}, \tag{1d}$$

where $\mathcal{A}$ is an affine transformation from $\{0,1\}^{m\times n}$ to $\{0,1\}^d$ with each element of the output vector defined as:

$$u_a = \langle \mathbf{H}_a, \boldsymbol{Z} \rangle := \bigvee_{(i,j)\in[m][n]} [\mathbf{H}_a]_{i,j} \wedge [\boldsymbol{Z}]_{i,j} \quad a \in [d]. \tag{2}$$

Here, $\mathbf{H}_a \in \{0,1\}^{m\times n}$ are random matrices, which we refer to as **sensing matrices**. Each element of the sensing matrix has a certain probability to be 1, and we refer to elements being 1 as **non-zero elements**.

Next, we derive an upper bound on the number of distinct sparse low-rank Boolean matrices. In [12], it was shown that every sparse Boolean matrix requires at least one sparse factorization. This result is presented in the following lemma.

Lemma 1 *[12] For every $n \times m$ Boolean matrix $\boldsymbol{Z}$ with $\text{rank}_{\mathcal{B}}(\boldsymbol{Z}) = k$, there exists a pair of $n \times k$ and $k \times m$ Boolean matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ such that $\boldsymbol{Z} = \boldsymbol{X} \bullet \boldsymbol{Y}$ and*

$$\|\boldsymbol{X}\|_0 + \|\boldsymbol{Y}\|_0 \leq 2\|\boldsymbol{Z}\|_0.$$

This lemma is used in the following theorem to set an upper limit on sparse low-rank Boolean matrices.

Theorem 1 *(Upper bound on the number of sparse low-rank matrices): Let $\Psi_{m,n}(k,s)$ ($s \geq k$, $s \leq \frac{k(m+n)}{4}$) be the number of Boolean matrices $\boldsymbol{Z} \in \{0,1\}^{m\times n}$ of rank exactly k and at most s non-zero element. Then we have*

$$\Psi_{m,n}(k,s) \leq \frac{s2^{k(m+n)H_B(\frac{2s}{k(m+n)})+1}}{k!},$$

where H_B is a binary entropy function.

The proof can be found in Appendix A in the full version of the paper [13]. Note that for the binary entropy function H_B, we need $\frac{2s}{k(m+n)} \leq \frac{1}{2}$, which implies $s \leq \frac{k(m+n)}{4}$ in [7]. In Section 4.3, we will compare this upper bound with a lower bound on the number of general low-rank Boolean matrices.

3. INTEGER PROGRAMMING AND LP RELAXATION

Boolean Compressed Sensing (BCS) [14, 15] aims at the reconstruction of sparse Boolean vectors from their corresponding Boolean measurements. Similarly, as it is a standard practice in image processing [3, 4], it often involves the transformation of a matrix into a vector, especially when some fundamental properties of the matrix, such as the rank, remain unknown. An integer programming characterization specifically tailored for BCS is provided in the following:

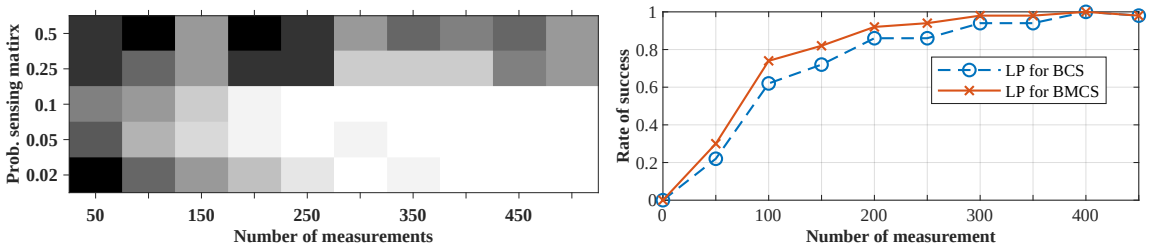
$$\min_{Z} \sum_{i,j} Z_{i,j} \tag{3a}$$
$$\text{s.t.} \sum_{(i,j)\in\omega_a} Z_{i,j} \geq u_a, \sum_{(i,j)\in\omega_{a'}} Z_{i,j} = 0 \quad a \in \mathcal{I}, a' \in \mathcal{J}, \tag{3b}$$
$$Z_{ij} \in \{0,1\} \quad i \in [n], j \in [m]. \tag{3c}$$

Nevertheless, given our access to the rank information, we can enhance recovery performance by imposing constraints on the space of feasible solutions. Similar to (1d), our goal is to determine $\boldsymbol{X}$ and $\boldsymbol{Y}$ such that the rank of $\boldsymbol{Z}$ is k. Therefore, the integer programming discussed above can be extended to the following:

$$\min_{X,Y,T,Z} \sum_{i,j} Z_{i,j} \tag{4a}$$
$$\text{s.t.} \sum_{(i,j)\in\omega_a} Z_{i,j} \geq u_a, \sum_{(i,j)\in\omega_{a'}} Z_{i,j} = 0 \; a \in \mathcal{I}, a' \in \mathcal{J}, \tag{4b}$$
$$T_{ilj} \leq Z_{ij} \leq \sum_{l=1}^{k} T_{ilj} \quad i \in [n], j \in [m], \ell \in [k], \tag{4c}$$
$$T_{i\ell j} \in MC\left(X_{i\ell}, Y_{\ell j}\right) \quad i \in [n], j \in [m], \ell \in [k], \tag{4d}$$
$$X_{i\ell}, Y_{\ell j}, Z_{ij} \in \{0,1\} \quad i \in [n], j \in [m], \ell \in [k]. \tag{4e}$$

Constrain (4b) constitutes the linear representation for Boolean measurements as described in (2), with $\mathcal{I} = \{a \mid u_a = 1\}$ and $\mathcal{J} = \{a \mid u_a = 0\}$. Here, $\omega_a = \{(i_1, j_1), (i_2, j_2), ...\}$ represents the coordinates of non-zero elements within H_a. In constrain (4d), we apply the McCormick envelope [16] as the linear representation for the Boolean product $X \wedge Y$:

$$MC(X,Y) = \{T \in \mathbb{R} : 0 \leq T, X + Y - 1 \leq T, T \leq X, T \leq Y\} \tag{5}$$



(a) $m = n = 30, k = 3$

(b) $m = n = 40, k = 2$

Fig. 1: Plot (1a) visualizes the performance of distinct sensing matrices, where the grayscale intensity signifies the percentage of successful recoveries. Specifically, black pixels denote instances where the method encountered failure in all 100 trials, while white pixels represent a flawless success in all 100 trials. Plots (1b) compares the performance of LP for BCS and LP for BMCS with matrices varying dimension m, n and rank k. Each point in these plots reflects the rate of success observed across 100 trials.

By restricting X and Y to binary values, i.e., $X, Y \in \{0, 1\}$, we observe that $MC(X, Y)$ exclusively contains the Boolean product of X and Y, which is $X \wedge Y \in \{0, 1\}$. Constrain (4c) denotes the linear formulation for the Boolean matrix multiplication. Since this is an integer programming problem, it is inherently NP-hard in general. To obtain a tractable linear programming (LP) relaxation, we substitute constrain (3c) with:

$$Z_{ij} \in [0, 1] \quad i \in [n], j \in [m].$$

and replace the constrain (4e) by

$$X_{i\ell}, Y_{\ell j}, Z_{ij} \in [0, 1] \quad i \in [n], j \in [m], \ell \in [k].$$

3.1. Experiment

The plot depicted in (1a) illustrates the impact of a few sensing matrices, denoted as $\mathbf{H}$, across a range of different parameters. Our experiment starts by generating a 30×3 matrix $\boldsymbol{X}$ and a 3×30 matrix $\boldsymbol{Y}$, where the entries of both $\boldsymbol{X}$ and $\boldsymbol{Y}$ are independently drawn from a Bernoulli distribution with a parameter 0.05. The corresponding multiplication of $\boldsymbol{X}$ and $\boldsymbol{Y}$ yields a 30×30 matrix $\boldsymbol{Z}$ of rank 3, and each of its entries is distributed according to the Bernoulli distribution with parameter $p(Z) = 1 - (1 - p(X)p(Y))^k = 0.00748$. Subsequently, we generated random sensing matrices $\mathbf{H}$ following a Bernoulli distribution with different probabilities, denoted as $p(H) \in \{0.02, 0.05, 0.1, 0.25, 0.5\}$, to compress the matrix $\boldsymbol{Z}$. In most cases, the linear programming (LP) procedures either effectively recover the ground truth or yield an all-zero matrix. The grayscale intensity depicted in Figure (1a) indicates the percentage of successful recoveries, with black pixels representing failures in 100 attempts, and white pixels representing consistent success in all 100 attempts. One can observe that when $p(H) \approx 0.1$, the LP exhibits the most robust recovery performance. This probability value will be utilized for the sensing matrix in the subsequent three experiments. Figures (1b) provide a comparison of the performance of Linear Programming (LP) for Boolean Compressed Sensing (BCS) and Binary Matrix Completion Sensing (BMCS). In these plots, matrices $\boldsymbol{Z}$ with varying dimensions and ranks are generated from $\boldsymbol{X}$ and $\boldsymbol{Y}$, each having entries drawn from a Bernoulli distribution with a probability of 0.05 with different sizes. Each data point in these plots represents the success rate over 100 trails. In each iteration, we begin by randomly creating the matrix $\boldsymbol{Z}$ and sensing matrices $\mathbf{H}$. Subsequently, we compute the measurement vector $\boldsymbol{u}$. Using this particular $\mathbf{H}$ and $\boldsymbol{u}$, we apply LP for BMCS directly, and we apply LP for BCS by converting the matrix into a vector. The plots illustrate that LP for BMCS consistently succeeds whenever LP for BCS succeeds in all iterations. Also note that, regardless of the parameter settings, LP for BMCS consistently exhibits a higher success rate. However, it can be observed that the difference in the performance between LP for BMCS and LP for BCS is not significant. This is not surprising, because, as in [17], our model is a structured model where convex optimization usually cannot achieve much better performance. In the next section, instead of formulating it as a non-convex recovery problem, we formulate it as a marginal inference problem in order to get better recovery performance.

4. MASSAGE PASSING

In this section, we frame our problem as a marginal inference task and use the max-sum Belief Propagation method to approximate it. Prior works have also tackled compressed sensing [18, 19] and matrix completion [20, 21, 22] as marginal inference problems, employing message-passing algorithms for efficient recovery.

4.1. Bayesian Formulation

Consider a communication channel, with d sensing matrices $\mathbf{H} \in \{0, 1\}^{m,n}$ in hand, where we sense the matrix $\boldsymbol{Z}$ d times independently. Assuming the results are passed through a bi-

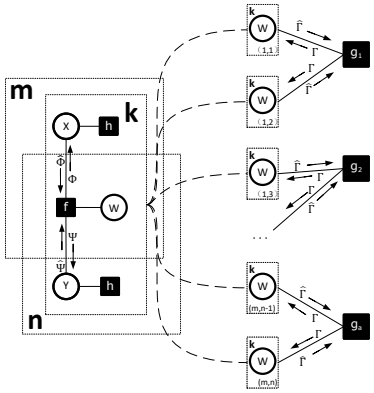


Fig. 2: Factor graph

nary symmetric channel (BSC) and we observe $\boldsymbol{u} \in \{0,1\}^d$, the objective is to recover a rank-k matrix $\boldsymbol{Z}$ from $\boldsymbol{u}$. Since each sensing matrix $\mathbf{H}$ is independent, we have:

$$p(\boldsymbol{u}|\boldsymbol{Z}, \mathbf{H}_1, ..., \mathbf{H}_d) = \prod_{a=1}^{d} p(u_a|\boldsymbol{Z}, \mathbf{H}_a) \tag{6}$$

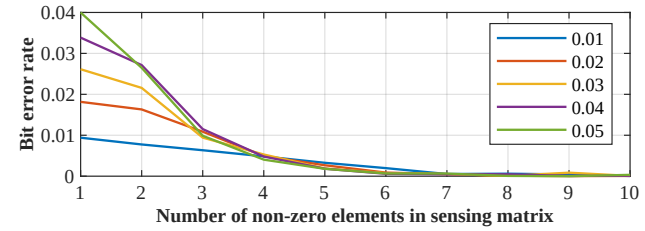
As discussed in Sectin 1, in order to recover $\boldsymbol{Z}$, we need to find $\boldsymbol{X} \in \{0,1\}^{m\times k}$ and $\boldsymbol{Y} \in \{0,1\}^{k\times n}$ such that $\boldsymbol{X} \bullet \boldsymbol{Y} = \boldsymbol{Z}$ to ensure that the rank of $\boldsymbol{Z}$ is k. Consistent with the typical Bayesian approach, we make the assumption that we possess knowledge of the prior probabilities $p(\boldsymbol{X})$ and $p(\boldsymbol{Y})$. Subsequently, the problem transforms into a Maximum A Posteriori (MAP) inference problem. $\operatorname{argmax}_{\boldsymbol{Z}} p(\boldsymbol{Z}|\boldsymbol{u}, \mathbf{H}_1, ..., \mathbf{H}_d) = \operatorname{argmax}_{\boldsymbol{X},\boldsymbol{Y}} p(\boldsymbol{X}, \boldsymbol{Y}|\boldsymbol{u}, \mathbf{H}_1, ..., \mathbf{H}_d)$, where

$$\begin{aligned} &p(\boldsymbol{X}, \boldsymbol{Y} \mid \boldsymbol{u}, \mathbf{H}_1, ..., \mathbf{H}_d) \\ &\propto p(\boldsymbol{X})p(\boldsymbol{Y})p(\boldsymbol{u} \mid \boldsymbol{X} \bullet \boldsymbol{Y}, \mathbf{H}_1, ..., \mathbf{H}_d) \\ &= p(\boldsymbol{X})p(\boldsymbol{Y}) \prod_{a=1}^{d} p(u_a|\boldsymbol{X} \bullet \boldsymbol{Y}, \mathbf{H}_a) \end{aligned} \tag{7}$$

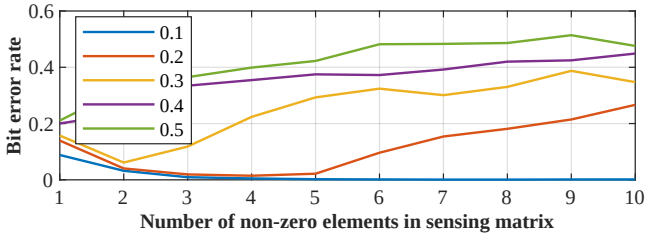
We proceed to create a graphical model and leverage the max-sum Belief Propagation algorithm to obtain an approximation for a solution, as discussed next. It is worth noting that the inference problem addressed in [7] only focuses on low-rank information, whereas our method also incorporates matrix sparsity, which constitutes our contribution in this work.

4.2. Factor graph

The factor graph used for inference via the belief propagation algorithm is illustrated in Figure 2. The left section of the factor graph resembles the one utilized for Boolean matrix completion in existing literature [7]. However, while the inference problem discussed in [7] solely emphasizes low-rank information, our approach additionally integrates matrix sparsity, which constitutes our contribution in this work. For each entry of $\boldsymbol{X} \in \{0,1\}^{m\times k}$ and $\boldsymbol{Y} \in \{0,1\}^{k\times n}$, a variable node is considered. In total, there are $K \times (M + N)$



(a) Density ranging from 0.01 to 0.05



(b) Density ranging from 0.1 to 0.5

Fig. 3: Evaluation of synthetic matrices varying in density

variable nodes. Note that there are also d random sensing matrices $\mathbf{H}_1, \mathbf{H}_2, ..., \mathbf{H}_d$. Suppose each $\mathbf{H}_a$ has t_a non-zero elements whose coordinates are denoted by $\omega_a = \{(i_1, j_1), (i_2, j_2), ..., (i_{t_a}, j_{t_a})\}$. To simplify the notation, we denote $w_{a,1} = (i_1, j_1), ..., w_{i,t} = (i_{t_a}, j_{t_a})$. Then, the entries of $\boldsymbol{u}$ can be expressed as:

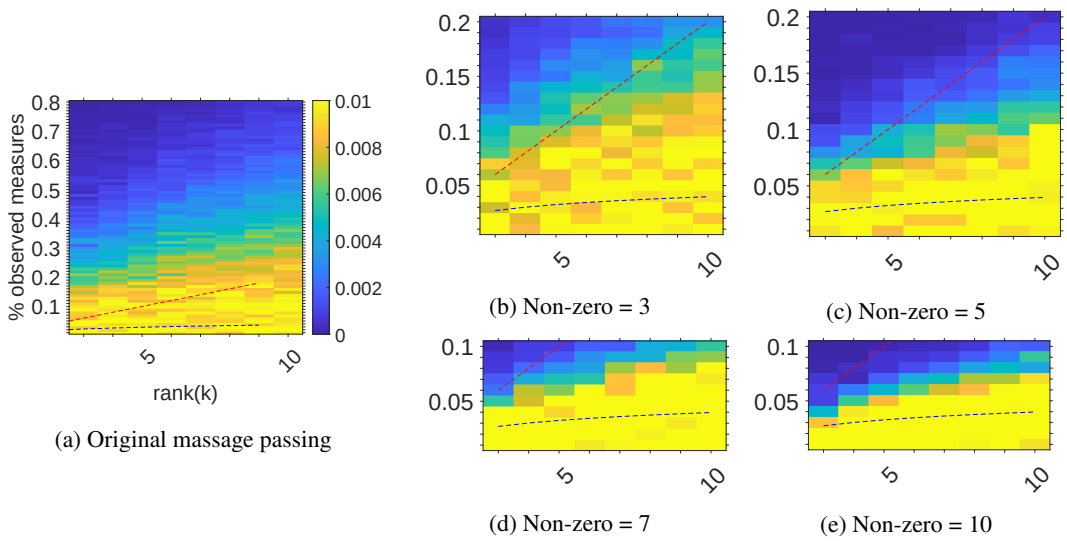
$$\begin{aligned} u_a = \langle \mathbf{H}_a, \boldsymbol{Z} \rangle &:= \bigvee_{(i,j)\in[m][n]} [\mathbf{H}_a]_{i,j} \wedge [\boldsymbol{Z}]_{i,j} = \bigvee_{l=1}^{t_a} \boldsymbol{Z}_{\omega_{a,l}} \\ &= \bigvee_{l=1}^{t_a} \bigvee_{k=1}^{K} \boldsymbol{X}_{\omega_{a,l,1},k} \wedge \boldsymbol{Y}_{k,\omega_{a,l,2}} \quad a \in [d] \end{aligned} \tag{8}$$

Therefore, $k \times m \times n$ auxiliary variables $W_{m,n,k} = X_{m,k} \wedge Y_{k,n}$ are needed to represent the Boolean product of $X_{m,k}$ and $Y_{k,n}$. Hence, $M \times N \times K$ hard constraint factors $\mathrm{f}_{m,n,k}(X_{m,k}, Y_{k,n}, W_{m,n,k}) = \mathbb{I}(W_{m,n,k} = X_{m,k} \wedge Y_{k,n})$, are required, where $\mathbb{I}(.)$ is referred to as the identity function where $\mathbb{I}_{\text{max-sum}}(true) = 0$ and $\mathbb{I}_{\text{max-sum}}(false) = -\infty$. There are also $m \times k$ and $n \times k$ local factor nodes to represent the logarithm of priors of X and Y, i.e., $h_{m,k}(X_{m,k}) = \log(p(X_{m,k}))$ and $h_{k,n}(Y_{k,n}) = \log(p(Y_{k,n}))$ over auxiliary variables. At last, there are d factors to represent the constrain imposed by Eq.(8) :

$$\begin{aligned} &\mathrm{g}_{\omega_a}\left(\{W_{\omega_{a,l},k}\}_{1\le k\le K, 1\le l\le t_a}\right) \\ &= \log\left(p^{O}_{\omega_a}\left(u_a \mid \bigvee_{l=1}^{t_a}\bigvee_{k=1}^{K} W_{\omega_{i,l},k}\right)\right) \end{aligned}$$

If we take the logarithm of the posterior, Eq.(7) becomes:

$$\begin{aligned} &\log(p(\boldsymbol{X}, \boldsymbol{Y} \mid \boldsymbol{u}, \mathbf{H}_1 ..., \mathbf{H}_d)) \\ &= \sum_{m,k} \mathrm{h}_{m,k}(X_{m,k}) + \sum_{k,n} \mathrm{h}_{k,n}(Y_{k,n}) \end{aligned}$$



(a) Original massage passing

(b) Non-zero = 3

(c) Non-zero = 5

(d) Non-zero = 7

(e) Non-zero = 10

Fig. 4: Plot (4a) displays the recovery bit error rate using the message-passing algorithm from [7]. Plots (4b), (4c), (4d), and (4e) demonstrate performance enhancement with increasing non-zero elements in the sensing matrix. The dashed red line represents the information-theoretic upper bound from [7], while the dashed black line corresponds to the bound in Theorem 1.

$$+\sum_{a} \mathrm{g}_{\omega_a}\left(\left\{W_{\omega_{a,l},k}\right\}_{1\leq k\leq K, 1\leq l\leq t_a}\right)$$

To improve Belief Propagation (BP) efficacy, we focus on reducing loops in the factor graph. We enforce constraints to ensure each entry in $\mathbf{Z}$ is sensed at most once, and each sensing matrix $\mathbf{H}_a$ has an equal number of non-zero elements. This limits total measurements, e.g., if each $\mathbf{H}$ matrix has exactly two non-zero elements, max measurements d is at most half the matrix size ($\frac{1}{2} \times m \times n$). This stems from needing to sense every $\mathbf{Z}$ entry once for full capacity. Consequently, no loops exist between variable nodes $\mathbf{W}_{m,n,k}$ and factor node g, with each factor node g having degree $t \times k$. We use Max-Sum BP algorithm to approximate MAP assignment with details provided in Appendix B in [13].

4.3. Experiment

Figure 3 depicts the performance of our Belief Propagation (BP) algorithm with respect to data matrices of varying densities. The matrices considered here maintain a fixed size of 200×200 and a rank of 5, with density being the only varying parameter. Density refers to the probability of a matrix entry being equal to 1. To achieve a matrix with a density of $p(\mathbf{Z})$, we employ a process where we initially generate two matrices, $\mathbf{X}$ and $\mathbf{Y}$, with entries drawn independently from a Bernoulli distribution with probabilities $p(\mathbf{X}) = p(\mathbf{Y}) = \sqrt{1 - \sqrt[k]{1 - p(\mathbf{Z})}}$. We maintain a consistent measurement percentage of 10%, resulting in a measurement length $\mathbf{d} = 0.1 \times 200 \times 200 = 4000$, limiting the maximum non-zero elements in the sensing matrix to 10.

Plot (3b) shows that when the matrix $\mathbf{Z}$ density falls below 10%, our algorithm improves the bit error rate by increasing the non-zero elements in the sensing matrix. Additionally, Plot (3a) demonstrates that our algorithm can achieve near-zero bit error rates when the non-zero elements in the sensing matrix exceed 5, and the matrix density remains below 5%.

Figure 4 presents a comparative evaluation between the message-passing algorithm for matrix completion proposed in [7] and our proposed message-passing algorithm for BMCS. Here, we maintain a fixed matrix size of 200×200 and a density of 0.01, while varying the rank. The number of measurements is constrained at 20% when there are 5 non-zero elements in the sensing matrix and at 10% when there are 10 non-zero elements. As a result, the Y-axis scale in Plot (4d) and Plot (4e) is half that of Plot (4b) and Plot (4c).

Our observations reveal that as the number of non-zero elements in the sensing matrix increases, our algorithm consistently demonstrates enhanced performance in terms of bit error rate. Remarkably, when the number of non-zero elements exceeds 5, the message-passing algorithm for BMCS surpasses the bound presented in [7] and approaches the bound characterized in Theorem 1. Comparison of the message-passing algorithm for BMCS with other algorithms for Boolean matrix factorization using real-world datasets is presented in Appendix 8 in the full version of the paper [13].

5. CONCLUSION

This paper unveils a novel pattern within Boolean matrices, illustrating that certain Boolean matrices possess simultaneous sparsity and low-rank properties. These properties may also be present in other datasets, such as image data, warranting further study. Additionally, studying recovery algorithms, such as non-convex optimization-based ones, could provide improved performance and theoretical guarantees.

References

[1] D.L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[2] Emmanuel Candes and Benjamin Recht, "Exact matrix completion via convex optimization," *Communications of the ACM*, vol. 55, no. 6, pp. 111–119, 2012.

[3] Lu Gan, "Block compressed sensing of natural images," in *2007 15th International Conference on Digital Signal Processing*, 2007, pp. 403–406.

[4] Zhonghao Zhang, Yipeng Liu, Jiani Liu, Fei Wen, and Ce Zhu, "Amp-net: Denoising-based deep unfolding for compressive image sensing," *IEEE Transactions on Image Processing*, vol. 30, pp. 1487–1500, 2021.

[5] Pauli Miettinen, "Matrix decomposition methods for data mining : Computational complexity and algorithms," 2009.

[6] Mark A. Davenport, Yaniv Plan, Ewout van den Berg, and Mary Wootters, "1-Bit matrix completion," *Information and Inference: A Journal of the IMA*, vol. 3, no. 3, pp. 189–223, 07 2014.

[7] Siamak Ravanbakhsh, Barnabas Poczos, and Russell Greiner, "Boolean matrix factorization and noisy completion via message passing," in *Proceedings of The 33rd International Conference on Machine Learning*, 2016, pp. 945–954.

[8] K.H. Kim, *Boolean Matrix Theory and Applications*, Monographs and textbooks in pure and applied mathematics. Dekker, 1982.

[9] Derek Desantis, Erik Skau, Duc P. Truong, and Boian Alexandrov, "Factorization of binary matrices: Rank relations, uniqueness and model selection of boolean decomposition," *ACM Trans. Knowl. Discov. Data*, vol. 16, no. 6, jul 2022.

[10] Dana S Nau, George Markowsky, Max A Woodbury, and D Bernard Amos, "A mathematical analysis of human leukocyte antigen serology," *Mathematical Biosciences*, vol. 40, no. 3-4, pp. 243–270, 1978.

[11] Pauli Miettinen and Stefan Neumann, "Recent developments in boolean matrix factorization," 2020.

[12] Pauli Miettinen, "Sparse boolean matrix factorizations," in *2010 IEEE International Conference on Data Mining*, 2010, pp. 935–940.

[13] "Boolean low-rank matrix compressed sensing," *https://github.com/MLSPsubmission/mlsp2024/blob/main/full_version.pdf*, 2024.

[14] George K. Atia and Venkatesh Saligrama, "Boolean compressed sensing and noisy group testing," *IEEE Transactions on Information Theory*, vol. 58, no. 3, pp. 1880–1901, 2012.

[15] Dmitry Malioutov and Mikhail Malyutov, "Boolean compressed sensing: Lp relaxation for group testing," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 3305–3308.

[16] Reka A. Kovacs, Oktay Gunluk, and Raphael A. Hauser, "Binary matrix factorisation and completion via integer programming," 2021.

[17] Samet Oymak, Amin Jalali, Maryam Fazel, Yonina C. Eldar, and Babak Hassibi, "Simultaneously structured models with application to sparse and low-rank matrices," *IEEE Transactions on Information Theory*, vol. 61, no. 5, pp. 2886–2908, 2015.

[18] Mohsen Bayati and Andrea Montanari, "The dynamics of message passing on dense graphs, with applications to compressed sensing," *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 764–785, 2011.

[19] David L Donoho, Arian Maleki, and Andrea Montanari, "Message-passing algorithms for compressed sensing," *Proceedings of the National Academy of Sciences*, vol. 106, no. 45, pp. 18914–18919, 2009.

[20] Riccardo Rossetti and Galen Reeves, "Approximate message passing for the matrix tensor product model," 2023.

[21] Byung-Hak Kim, Arvind Yedla, and Henry D. Pfister, "Imp: A message-passing algorithmfor matrix completion," 2010.

[22] Mahdi Soleymani, Qiang Liu, Hessam Mahdavifar, and Laura Balzano, "Matrix completion over finite fields: Bounds and belief propagation algorithms," in *2023 IEEE International Symposium on Information Theory (ISIT)*, 2023, pp. 1166–1171.

[23] Ranjeet Kumar, Utpreksh Patbhaje, and A. Kumar, "An efficient technique for image compression and quality retrieval using matrix completion," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 4, pp. 1231–1239, 2022.

[24] Xiao Peng Li, Lei Huang, Hing Cheung So, and Bo Zhao, "A survey on matrix completion: Perspective of signal processing," 2019.

[25] Dong Yang, Guisheng Liao, Shengqi Zhu, Xi Yang, and Xuepan Zhang, "Sar imaging with undersampled data via matrix completion," *IEEE Geoscience and Remote Sensing Letters*, vol. 11, no. 9, pp. 1539–1543, 2014.

[26] John Wright and Yi Ma, *High-Dimensional Data Analysis with Low-Dimensional Models: Principles, Computation, and Applications*, Cambridge University Press, 2022.

[27] Madeleine Udell, Corinne Horn, Reza Zadeh, and Stephen Boyd, "Generalized low rank models," 2015.

6. PROOF OF THEOREM 1

Proof For every Boolean matrices of rank exactly k and s non-zero element, as lemma 1, can be factorized to two matrix $\boldsymbol{X} \in \{0,1\}^{m\times k}$ and $\boldsymbol{Y} \in \{0,1\}^{k\times n}$ where $\|\boldsymbol{X}\|_0 + \|\boldsymbol{Y}\|_0 \leq 2s$.

In other words, for Boolean matrices in $\{0,1\}^{m\times n}$ of rank exactly k and at most s non-zero element, there exist at least one factorization $\boldsymbol{X} \in \{0,1\}^{m\times k}$ and $\boldsymbol{Y} \in \{0,1\}^{k\times n}$ that satisfy $\|\boldsymbol{X}\|_0 + \|\boldsymbol{Y}\|_0 \leq 2s$. By enumerating the possible combinations of $\boldsymbol{X}$ and $\boldsymbol{Y}$, we can derive an upper bound for the total. Note that $\boldsymbol{X}$ and $\boldsymbol{Y}$ must be full rank, i.e., $\mathrm{rank}_{\mathcal{B}}(\boldsymbol{X}) = \mathrm{rank}_{\mathcal{B}}(\boldsymbol{Y}) = k$. This implies that every column of $\boldsymbol{X}$ and every row of $\boldsymbol{Y}$ must be distinct. Therefore, there exist $k!$ permutation matrices $\boldsymbol{U}$, such that:

$$\boldsymbol{X} \neq \boldsymbol{X} \bullet \boldsymbol{U}, \boldsymbol{Y} \neq \boldsymbol{U}^T \bullet \boldsymbol{Y}$$

$$\|\boldsymbol{X}\|_0 = \|\boldsymbol{X} \bullet \boldsymbol{U}\|_0, \|\boldsymbol{Y}\|_0 = \|\boldsymbol{U}^T \bullet \boldsymbol{Y}\|_0$$

and

$$\boldsymbol{Z} = (\boldsymbol{X} \bullet \boldsymbol{U}) \bullet (\boldsymbol{U}^T \bullet \boldsymbol{Y})$$

That means each $\boldsymbol{Z}$ has at least $k!$ different factorization. Then we have:

$$\begin{aligned}\Psi_n(k,s) &< \frac{\sum_{l=1}^{2s}\binom{k(m+n)}{l}}{k!} \\ &\leq \frac{\sum_{l=1}^{2s} 2^{k(m+n)H_B(\frac{l}{k(m+n)})}}{k!} \\ &\leq \frac{s2^{k(m+n)H_B(\frac{2s}{k(m+n)})+1}}{k!}\end{aligned}$$

□

7. MASSAGE UPDATE

We adopt the Max-Sum Belief Propagation algorithm, which has proven to be effective in the context of Boolean matrix factorization, as outlined in [7]. This method serves as a message-passing mechanism for approximating the MAP assignment in our research. In accordance with the factor graph representation depicted in Figure 2, the arrangement of variable nodes X, Y, and W, as well as factor nodes h and f follows the structure presented in [7]. The only difference is in the arrangement of factor node g. Consequently, there is no need to modify the update rules for the messages $\Phi, \hat{\Phi}, \Psi$ and $\hat{\Psi}$ as these remain unchanged. Our focus is only on modifying the update rules for Γ.

As a result, we only need to update Equation (6d) of Algorithm 1 in [7]. The complete message-passing algorithm for our specific problem is detailed in Section 1.

7.1. Detailed derivation of the update rules of messages Γ

Suppose each sensing matrix $\mathbf{H}$ has t_a non-zero elements whose coordinates are denoted by $\omega_a = \{(i_1,j_1),(i_2,j_2),...,(i_{t_a},j_{t_a})\}$. To simplify the notation, we denote $w_{a,1} = (i_1,j_1),...,w_{a,t_a} = (i_{t_a},j_{t_a})$. The naive form of max-sum BP for the messages leaving this factor to each of $t_a \times K$ neighboring variables can be expressed as $\left\{W_{\omega_{i,j},k}\right\}_{1\leq k\leq K, 1\leq j\leq t_a}$.

$$\begin{aligned}&\mathrm{m}_{\mathrm{g}_{\omega_i}\to W_{\omega_{i,j},k}}\left(W_{\omega_{i,j},k}\right)^{(\mathrm{t}+1)} \\ &= \max_{\left\{W_{\omega_{i,\alpha},\beta}\right\}_{(\omega_{i,\alpha},\beta)\neq(\omega_{i,j},k)}} \left(\mathrm{g}_{\omega_i}\left(\left\{W_{\omega_{i,\alpha'},\beta'}\right\}_{\alpha',\beta'}\right)\right. \\ &\quad \left. + \sum_{(\omega_{i,j'},k')\neq(\omega_{i,j},k)} \mathfrak{m}_{W_{\omega_{i,j'},k'}\to \mathrm{g}_{w_i}}\left(W_{w_i,k'}\right)^{(\mathrm{t})}\right)\end{aligned} \tag{12}$$

In evaluating $\mathrm{g}_{\omega_i}\left(\left\{W_{\omega_{i,\alpha'},\beta'}\right\}_{\alpha',\beta'}\right)$ two scenarios are conceivable

1)$\bigvee_j \bigvee_k W_{\omega_{i,j},k} = 1$, that means at least one of $\left\{W_{\omega_{i,\alpha'},\beta'}\right\}_{1\leq\alpha'\leq t,\beta'}$ is non-zero and $\mathrm{g}_{\omega_i}\left(\left\{W_{\omega_{i,\alpha'},\beta'}\right\}_{\alpha',\beta'}\right)$ evaluates to $p_i\left(u_i \mid 1\right)$.

2)$\bigvee_j \bigvee_k W_{\omega_{i,j},k} = 0$, $g(W_{\omega_i,k})$ evaluates to $p_i\left(u_i \mid 0\right)$.

For simplicity, let $\mathfrak{m}_{1,1}\left(W_{1,1}\right),\ldots,\mathfrak{m}_{1,K}\left(W_{1,K}\right),\ldots,\mathfrak{m}_{t,1}\left(W_{t,1}\right),\ldots,\mathfrak{m}_{t,K}\left(W_{t,K}\right)$ denote $\mathfrak{m}_{W_{\omega_{i,1},1}\to \mathrm{g}_{\omega_i}}\left(W_{\omega_{i,1},1}\right)^{(\mathrm{t})},\ldots,\mathfrak{m}_{W_{\omega_{i,1},1}\to \mathrm{g}_{\omega_i}}\left(W_{\omega_{i,1},1}\right)^{(\mathrm{t})}$ respectively. We assume the objective is to calculate the outgoing message to the first variable $\mathfrak{m}'_{1,1}\left(W_{1,1}\right) = \mathfrak{m}_{\mathrm{g}_{\omega_i}\to W_{\omega_{i,1},1}}\left(W_{\omega_{i,1},1}\right)^{(\mathrm{t})}$ The Equation (12) can be rewrote using this notation:

$$\begin{aligned}\mathfrak{m}'_{1,1}\left(W_{1,1}\right) = \max_{W_{1,2}\ldots W_{t,K}} &\left(\mathrm{g}_{\omega_i}\left(\left\{W_{\omega_{i,\alpha'},\beta'}\right\}_{\alpha',\beta'}\right)\right. \\ &\left. + \sum_{(\alpha,\beta)\neq(1,1)} \mathfrak{m}_{\alpha,\beta}\left(W_{\alpha,\beta}\right)\right)\end{aligned}$$

For $W_{1,1} = 1$, regardless of assignments to other nodes, we have $\bigvee_j \bigvee_k W_{\omega_{i,j},k} = 1$ and therefore the maximization above simplifies to

$$\begin{aligned}&\mathfrak{m}'_{1,1}(1) \\ &= \max_{W_{1,2}\ldots W_{t,K}}\left(\log\left(p_i\left(u_i \mid 1\right)\right) + \sum_{(\alpha,\beta)\neq(1,1)} \mathfrak{m}_{\alpha,\beta}\left(W_{\alpha,\beta}\right)\right) \\ &= \log\left(p_i\left(u_i \mid 1\right)\right) + \sum_{(\alpha,\beta)\neq(1,1)} \max\left(\mathfrak{m}_{\alpha,\beta}(0), \mathfrak{m}_{\alpha,\beta}(1)\right).\end{aligned}$$

For $W_{1,1} = 0$, as it in

$$(\alpha^*,\beta^*) = \arg_{(\alpha,\beta)\neq(1,1)} \max\left(\mathfrak{m}_{(\alpha,\beta)}(1) - \mathfrak{m}_{(\alpha,\beta)}(0)\right)$$

Algorithm 1 message passing for Boolean matrix compressed sensing

Input: 1) observed measurements $\boldsymbol{u} \in \{0,1\}^d$;
2) d sensing matrices $\mathbf{H} \in \{0,1\}^{M\times N}$, coordinates of each sensing $\mathbf{H}_i$ are $\omega_i = \{(i_1,j_1),(i_2,j_2),...,(i_{t_a},j_{t_a})\}$;
3) rank $K < M, N$; priors $p^X_{m,k}, p^Y_{n,k}$ $\forall m,n,k$;
Output: $\boldsymbol{Z} \in \{0,1\}^{M\times N}$
Initialization: $\Phi^{(\mathrm{t})}_{m,n,k}, \Psi^{(\mathrm{t})}_{m,n,k}, \hat{\Phi}^{(\mathrm{t})}_{m,n,k}, \hat{\Psi}^{(\mathrm{t})}_{m,n,k}, \hat{\Gamma}^{(\mathrm{t})}_{m,n,k}$ and $\Gamma^{(\mathrm{t})}_{m,n,k}$ $\forall m,n,k$
while $t < T_{max}$ **and** *not converged for all* m,n,k **do**

$$\Phi^{(\mathrm{t}+1)}_{m,n,k} := \left(\Gamma^{(\mathrm{t})}_{m,n,k} + \hat{\Psi}^{(\mathrm{t})}_{m,n,k}\right)_+ - \left(\hat{\Psi}^{(\mathrm{t})}_{m,n,k}\right)_+ \tag{9a}$$

$$\Psi^{(\mathrm{t}+1)}_{m,n,k} := \left(\Gamma^{(\mathrm{t})}_{m,n,k} + \hat{\Phi}^{(\mathrm{t})}_{m,n,k}\right)_+ - \left(\hat{\Phi}^{(\mathrm{t})}_{m,n,k}\right)_+ \tag{9b}$$

$$\hat{\Phi}^{(\mathrm{t}+1)}_{m,n,k} := \log\left(\frac{p^X_{m,k}(1)}{p^X_{m,k}(0)}\right) + \sum_{n'\neq n} \Phi^{(\mathrm{t})}_{m,n',k} \tag{9c}$$

$$\hat{\Psi}^{(\mathrm{t}+1)}_{m,n,k} := \log\left(\frac{p^Y_{n,k}(1)}{p^Y_{n,k}(0)}\right) + \sum_{m'\neq m} \Psi^{(\mathrm{t})}_{m',n,k} \tag{9d}$$

$$\hat{\Gamma}^{(\mathrm{t}+1)}_{m,n,k} := \min\left\{\hat{\Phi}^{(\mathrm{t})}_{m,n,k} + \hat{\Psi}^{(\mathrm{t})}_{m,n,k}, \hat{\Phi}^{(\mathrm{t})}_{m,n,k}, \hat{\Psi}^{(\mathrm{t})}_{m,n,k}\right\} \tag{9e}$$

$$\Gamma^{(\mathrm{t}+1)}_{\omega_{i,j},k} := \min\left\{\left(-\max_{(j',k')\neq(j,k)} \hat{\Gamma}^{(\mathrm{t})}_{\omega_{i,j'},k'}\right)_+, \tag{9f}$$

$$\sum_{(j',k')\neq(j,k)} \left(\hat{\Gamma}^{(\mathrm{t})}_{\omega_{i,j'},k'}\right)_+ + \log\left(\frac{p_i(u_i \mid 1)}{p_i(u_i \mid 0)}\right)\right\} \tag{9g}$$

end while
calculate log-ratio of the posterior marginals

$$\Xi_{m,k} := \log\left(\frac{p^X_{m,k}(1)}{p^X_{m,k}(0)}\right) + \sum_n \Phi^{(\mathrm{t})}_{m,n,k} \tag{10a}$$

$$\Upsilon_{k,n} := \log\left(\frac{p^Y_{k,n}(1)}{p^Y_{k,n}(0)}\right) + \sum_m \Psi^{(\mathrm{t})}_{m,n,k} \tag{10b}$$

calculate $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}$

$$X_{m,k} := \begin{cases} 1, & \text{if } \Xi_{m,k} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{11a}$$

$$Y_{k,n} := \begin{cases} 1, & \text{if } \Upsilon_{k,n} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{11b}$$

$$Z_{m,n} = \bigvee_k X_{m,k} \wedge Y_{k,n} \tag{11c}$$

return $\boldsymbol{Z}$

Therefore,

$$\mathfrak{m}'_{1,1}(0) = \max\Big(\log p_i(u_i \mid 0) + \sum_{(\alpha,\beta)\neq(1,1)} \mathfrak{m}_{\alpha,\beta}(0), \Big(p_i(u_i \mid 1) + \mathfrak{m}_{\alpha^*,\beta^*} + \sum_{(\alpha,\beta)\neq(1,1),(\alpha^*,\beta^*)} \max\left(\mathfrak{m}_{\alpha,\beta}(0), \mathfrak{m}_{\alpha,\beta}(1)\right)\Big)\Big)$$

As before, let us assume that the incoming messages are normalized such that $\forall_{\alpha,\beta}\mathfrak{m}_{\alpha,\beta}(0) = 0$, and therefore $\hat{\Gamma}_{m,n,k'} = \mathfrak{m}_{k'}(1)$

$$\begin{aligned}
&\Gamma_{\omega_{i,1},1} \\
&= \mathfrak{m}'_{1,1}(1) - \mathfrak{m}'_{1,1}(0) \\
&= \log\left(p_i(u_i \mid 1)\right) + \sum_{(\alpha,\beta)\neq(1,1)} \max\left(0, \mathfrak{m}_{\alpha,\beta}(1)\right) - \\
&\quad \max\Big(\log p_i(u_i \mid 0) + \sum_{(\alpha,\beta)\neq(1,1)} \mathfrak{m}_{\alpha,\beta}(0), \\
&\quad \log\Big(p_i(u_i \mid 1) + \mathfrak{m}_{\alpha^*,\beta^*} \\
&\quad + \sum_{(\alpha,\beta)\neq(1,1),(\alpha^*,\beta^*)} \max\left(0, \mathfrak{m}_{\alpha,\beta}(1)\right)\Big)\Big) \\
&= \min\Big(\log p_i(u_i \mid 1) - \log p_i(u_i \mid 0) \\
&\quad + \sum_{(\alpha,\beta)\neq(1,1)} \max\left(0, \mathfrak{m}_{\alpha,\beta}(1)\right), \max(-\mathfrak{m}_{\alpha^*,\beta^*}(1), 0)\Big) \\
&= \min\Big(\sum_{(\alpha,\beta)\neq(1,1)} \max\left(0, \hat{\Gamma}^{(t)}_{\omega_{i,\alpha},\beta}(1)\right) \\
&\quad + \log\left(\frac{p_i(u_i \mid 1)}{pi^O(u_i \mid 0)}\right), \max\Big(0, -\max_{(\alpha,\beta)\neq(1,1)} \hat{\Gamma}^{(\mathrm{t})}_{\omega_{i,\alpha},\beta}\Big)\Big)
\end{aligned}$$

Therefore, Equation (6f) in algorithm 1 of [7] becomes

$$\Gamma^{(\mathrm{t}+1)}_{\omega_{i,j},k} := \min\left\{\left(-\max_{(j',k')\neq(j,k)} \hat{\Gamma}^{(\mathrm{t})}_{\omega_{i,j'},k'}\right)_+, \sum_{(j',k')\neq(j,k)} \left(\hat{\Gamma}^{(\mathrm{t})}_{\omega_{i,j'},k'}\right)_+ + \log\left(\frac{p_i(u_i \mid 1)}{p_i(u_i \mid 0)}\right)\right\}$$

Since we only consider noiseless case, then:

$$p_i(u_i \mid 1) = \begin{cases} 1, & \text{if } u_i = 1 \\ 0, & \text{if } u_i = 0 \end{cases}$$

$$p_i(u_i \mid 0) = \begin{cases} 1, & \text{if } u_i = 0 \\ 0, & \text{if } u_i = 1 \end{cases}$$

8. REAL-WORLD APPLICATIONS

We employ our BMCS approach to compress the MovieLens-100K dataset and subsequently utilize the message-passing

	5%	10%	20%	50%	95%
message passing	0.0804	0.0736	0.0679	0.0532	0.0440
message passing-2	0.0779	0.0682	0.0545	0.0469	-
message passing-5	0.0626	0.0569	0.0447	-	-
message passing-10	0.0528	0.0425	-	-	-
GLRM	0.0746	0.0623	0.0612	0.0523	0.0515
1-bit completion	0.0611	0.0611	0.0611	0.0611	0.0611

Table 2: Bit error rate of different recovery algorithm for MovieLense dataset

algorithm outlined in Section 4 for recovery. Measurements in our proposed algorithm for BMCS are projections from matrices to lower-dimensional vectors while measurements in matrix completion involves projecting measurements from the complete set of matrix indices onto a subset of these indices. Therefore, partially observing a matrix can directly be regarded as a way of compression. It is worth noting that although generally matrix completion algorithms are typically employed for collaborative filtering and aimed at predicting missing values, some literature[23, 24, 25] has considered them as a method to recover compressed data. This perspective arises from the notion inherent in matrix completion that a small subset of entries can hold the most information of the matrix [26].

In this particular dataset, the ratings are expressed ordinarily within the range of 1 to 5. As shown in Table 1 , we transform the dataset into a sparse Boolean representation by setting ratings greater than 1 to 0. Table 2 presents an overview of the recovery bit error rates for various methods. It is assumed that the rank is fixed at 3, and different values of $\alpha \in \{.01, .05, .1, .2, .5, .95\}$ represent the portion of measurements. In addition to the message-passing algorithm described in [7], two conventional matrix completion algorithms, namely GLRM [27] and 1-bit matrix completion [6] are included for comparison.

In our proposed message-passing algorithm for BMCS, results are presented for different settings of the number of non-zero elements in the sensing matrix, with values chosen from the set $\{2, 5, 10\}$. The experiment results illustrate that as the number of non-zero elements in the sensing matrix increases, the recovery performance improves. Notably, when the number of non-zero elements exceeds 5, our algorithm consistently outperforms all other methods.