Generalized Matrix Local Low Rank Representation by Random Projection and Submatrix Propagation

Pengtao Dang
Purdue University
Indianapolis, IN, USA
dangp@purdue.edu

Haiqi Zhu
Indiana University
Bloomington, IN, USA
haiqzhu@indiana.edu

Tingbo Guo
School of Medicine, Indiana University
Indianapolis, IN, USA
guoti@iu.edu

Changlin Wan
Purdue University
West Lafayette, IN, USA
wan82@purdue.edu

Tong Zhao
Uber, Inc
Seattle, WA, USA
tongz@uber.com

Paul Salama
Purdue University
Indianapolis, IN, USA
psalama@purdue.edu

Yijie Wang
Indiana University
Bloomington, IN, USA
yijwang@iu.edu

Sha Cao*
School of Medicine, Indiana University
Indianapolis, IN, USA
shacao@iu.edu

Chi Zhang*
School of Medicine, Indiana University
Indianapolis, IN, USA
czhang87@iu.edu

ABSTRACT

Matrix low rank approximation is an effective method to reduce or eliminate the statistical redundancy of its components. Compared with the traditional global low rank methods such as singular value decomposition (SVD), local low rank approximation methods are more advantageous to uncover interpretable data structures when clear duality exists between the rows and columns of the matrix. Local low rank approximation is equivalent to low rank submatrix detection. Unfortunately, existing local low rank approximation methods can detect only submatrices of specific mean structure, which may miss a substantial amount of true and interesting patterns. In this work, we develop a novel matrix computational framework called RPSP (Random Probing based submatrix Propagation)[1] that provides an effective solution for the general matrix local low rank representation problem. RPSP detects local low rank patterns that grow from small submatrices of low rank property, which are determined by a random projection approach. RPSP is supported by theories of random projection. Experiments on synthetic data demonstrate that RPSP outperforms all state-of-the-art methods, with the capacity to robustly and correctly identify the low rank matrices when the pattern has a similar mean as the background, background noise is heteroscedastic and multiple patterns present in the data. On real-world datasets, RPSP also demonstrates its effectiveness in identifying interpretable local low rank matrices.

*Corresponding author
[1]https://github.com/ptdang1001/RPSP

CCS CONCEPTS

• **Computing methodologies → Machine learning algorithms**.

KEYWORDS

Local low rank matrix, Sub-matrix detection, Representation learning, Randomized matrix approximation, Random projection

ACM Reference Format:
Pengtao Dang, Haiqi Zhu, Tingbo Guo, Changlin Wan, Tong Zhao, Paul Salama, Yijie Wang, Sha Cao, and Chi Zhang. 2023. Generalized Matrix Local Low Rank Representation by Random Projection and Submatrix Propagation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, Long Beach, CA, USA, 12 pages. https://doi.org/10.1145/3580305.3599361

1 INTRODUCTION

Low rank representation of a matrix can reduce or eliminate the statistical redundancy among its components, and enable a lower dimensional representation without significant loss of information[18]. It has found wide-range utilities in the field of data mining including recommendation systems [22, 47], computer vision[4, 37], and signal processing [27, 36]. Mathematically, for a target matrix $X \in \mathbb{R}^{M\times N}$, the goal of low rank approximation is to find a low rank matrix $\hat{X}$, such that the residual matrix $E = X - \hat{X}$ follows certain tolerance criteria. Singular value decomposition (SVD) is the best-known method which provides the true matrix rank and gives the optimal approximation based on the Eckart-Young Theorem [8, 13, 34]. Many algorithms have been proposed to find matrix low-rank approximation, including randomization techniques such as randomized SVD [14, 24, 28], and rank regularization methods such as nuclear norm-based methods [17, 30]. However, all these methods assume that the whole matrix is low-rank, or known as

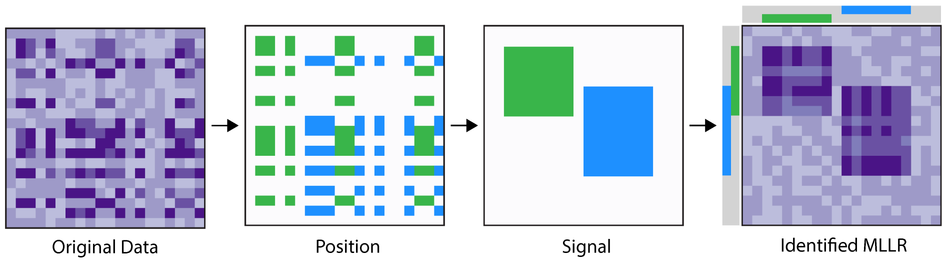

Figure 1: One example of the Matrix Local Low Rank Representation (MLLRR) Problem.

global low-rank, which often fails to detect local data structures, such as the local low-rank patterns suggested by [20, 25].

As illustrated in **Fig 1**, a locally low-rank matrix can be generally understood as the superposition of multiple matrices, or sum of a series of sparse and low-rank matrices, each consisting of a sparse set of rows and columns. In the example of movie rating data, global methods will decompose the rating matrix into a user factor matrix and a movie factor matrix. However, these approaches doesn't reveal the specific genres of movies that various groups of users are interested in. For example, a user shares similar tastes in drama movies with a certain group of users, while having similar preferences with a distinct group of users in horror movies. Another example of such 'locality' property is the online purchase behavior data, where a subset of items was purchased under a common reason by a subset of customers, while the identities of neither the items nor the users are known [7]. The locality of patterns also exists in biological single-cell RNA-sequencing data, where a subgroup of genes may be regulated by an unknown signal that is activated only in a subset of cells, forming a local low-rank gene co-regulation module [5, 42, 46]. In addition, shapes, numbers, and words in imaging data also carry strong locality characteristics [20]. In all these situations, **M**atrix **L**ocal **L**ow **R**ank **R**epresentation (MLLRR) is more advantageous to uncover more informative and interpretable patterns hidden in the data with its locality assumption.

Formally, for a given matrix $X \in \mathbb{R}^{M \times N}$, the MLLRR problem aims to identify submatrices $X_{I_k \times J_k}, k = 1...K$, each being low-rank, with I_k and J_k representing row and column indices of X. The low-rankness of $X_{I_k \times J_k}$ means that the number of its calculated singular values greater than a certain threshold is low, and could be further decomposed as the sum of a pattern submatrix and background submatrix, corresponding to the large singular values and negligible singular values respectively. The total number of possible $I_k \times J_k$ combinations is 2^{N+M}, making the MLLRR problem NP-hard [40]. Existing methods for MLLRR fall into three categories: (1) co-clustering approaches that identify submatrices with distinct mean compared to background [2, 10, 19]; (2) Sparse matrix decomposition-based methods that penalize the number of non-zero entries in the factor/singular matrices [16, 21, 38, 39, 43, 48]; and (3) Anchor-based methods that randomly selects some anchor-points, and then estimate local low-rank matrix approximation for each neighborhood of the anchor-point [20, 25, 49]. Although these methods have improved upon global low-rank methods to some extent, they can only identify the submatrices with a pattern mean that differs from the background [48]. For example, the following types of local structures cannot be detected: (1) a low rank submatrix having a similar pattern mean compared to its background; (2) the background submatrix having heterogeneous and even contaminated distributions, and (3) the submatrices are of small size.

In this study, we aim to solve the general MLLRR problem with a computationally efficient algorithm, namely RPSP (**R**andom **P**robing-based **S**ub-matrix **P**ropagation). RPSP first evaluates low-rankness of a large set of randomly sampled small submatrices, and gradually grows these low rank submatrices. RPSP adopts a random projection-based approach to approximate the singular values of submatrices, which drastically improved the computational efficiency compared to the conventional QR decomposition-based computation [18]. The approximated singular values are further used to evaluate the low-rankness of the submatrices. We systematically benchmarked RPSP with state-of-the-art (SOTA) methods on simulated data and four real-world datasets. RPSP outperformed all SOTA methods in different scenarios. RPSP is shown to have the unique capability to handle heteroscedastic error distributions, and distinguish a true local low matrix from background noise with or without spiked mean structure. Application of RPSP on real-world datasets demonstrated its capability in detecting context-meaningful local low-rank matrices.

The major contributions of this work include:

(1) **RPSP is the first solution for the general MLLRR problems**: Compared to existing methods, RPSP is the only method that can robustly solve the general MLLRR problem, especially when (i) the patterns are small, (ii) the mean of patterns is not necessarily distinct compared to the background, (iii) the background error is non-Gaussian or heterogeneous, and (iv) there are a large number of low-rank submatrices of different sizes and ranks.

(2) **A new perspective in detecting and embedding local low rank matrices**: RPSP is the first method that adopts random projection to solve the MLLRR problem. A new computational framework and mathematical formulation to efficiently compute, embed, and propagate local low-rankness of submatrices were developed.

(3) **A theoretical framework is developed by adopting the mathematical theories of random projection and random covering of unit R-sphere** that support: (i) the identifiability of local low-rankness, (ii) bound of sensitivity and specificity, and (iii) impact of errors and pattern sizes with respect to the setting of hyperparameters of RPSP.

2 PRELIMINARIES

2.1 Notations and Mathematical Backgrounds

We utilized the notation of matrix operation, matrix rank, and low-rankness by following [25, 31].

General notations of matrix and operations. Denote a real matrix of M rows and N columns as $X^{M\times N} \in \mathbb{R}^{M\times N}$, and its (i, j)-th entry as X_{ij}. We use $I_k \subset \{1, ..., M\}$ and $J_k \subset \{1, ..., N\}$ to denote row and column indices of the kth submatrix and $X_{I_k\times J_k}$ as the submatrix indexed by $I_k \times J_k$. Here I_k and J_k can be any subset of the row and column indices, and different submatrices $X_{I_k\times J_k}$ can be overlapped. Denote $||X||_p$, $||X||_*$ as the $\mathcal{L}_p$ and nuclear norm of a matrix, respectively. The nuclear norm is the sum of all singular values of X. Denote $Rank(X)$ as the rank of $X^{M\times N}$. $Rank(X) = r$ if and only if X has r non-zero singular values; and this is also equivalent to that the rows (or columns) of X are spanned by an r dimension vector space, which is called the row space (or column space) of X. To say that X has a low-rank property, we mean $r \ll min(M, N)$.

Low-rankness of real-world matrix. Random noise is inevitable in real-world data. Intuitively, the rank of a random noise-added real-world data matrix $X^{M\times N}$ is $min(M, N)$, due to the existence of many small and close-to-zero singular values. To characterize the low-rankness of a real-world matrix, we introduce numerical rank. Let $U^{M\times M}$ and $V^{N\times N}$ be the left and right singular vector matrices of $X^{M\times N}$, and $\Sigma^{N\times N}$ be the diagonal matrix of singular values. The numerical rank of X is defined as the number of singular values that are greater than a certain threshold associated with a tolerance parameter ϵ, denoted as $Rank(X, \epsilon)$, i.e., $Rank(X, \epsilon) = min_{\|X-\hat{X}\|_2\leq\epsilon} Rank(\hat{X})$ [13].

The global low-rank approximation of X is formulated as

$$X = \hat{X} + E \tag{2.1}$$

where $\hat{X}$ is a low rank matrix and E represents background noises. The low-rank property of X could be characterized by $\frac{||\hat{X}||_*}{||X||_*}$. Here a large ratio suggests that X can be well fitted by $\hat{X}$. When $\hat{X}$ is unknown, an alternative measure to characterize the low-rank property of X is by $\frac{\sum_{n=1}^{k}\Sigma_{ii}}{||X||_*}$ ($\forall k \leq r^*$), here r^* is the numerical rank of X or any other estimated rank of X [31, 44].

2.2 Problem statement

Global low-rank methods are generally effective in detecting global data structures, which relate each column to all rows, or vice versa. However, the global low-rank structure may not be informative when there exists a clear duality of the rows and columns. In other words, subsets of rows and columns within a matrix may be generated from distinct sparse subspace structures. To better capture the subspace structures in a matrix generation process, the local low-rank approximation of a matrix is defined as

$$X^{M\times N} = \sum_{k=1}^{K} \hat{X}_k^{M\times N} + E^{M\times N} \tag{2.2}$$

where $\hat{X}_k^{M\times N}$ corresponds to the k-th low-rank pattern, and takes zero values outside of $I_k \times J_k$, and E represents noises.

Compared to (2.1), (2.2) flexibly characterizes the subspace structures in X. Based on (2.2), we formally define the MLLRR problem from the perspective of local pattern detection.

Definition 1. Matrix Local Low Rank Representation (MLLRR). For a given matrix $X^{M\times N}$, MLLRR identifies K low rank submatrices $X_k := X_{I_k\times J_k}, I_k \subset \{1...M\}, J_k \subset \{1...N\}, k = 1, ..., K$, s.t. X_k are low rank matrices with small numerical ranks.

In Definition 1, K does need to be pre-given, and I_k and J_k do not need to be disjoint. One advantage of this definition is that the MLLRR problem does not rely on a specific distribution of background noise. In this study, we utilize the low-rankness measure given in section 2.1, $\frac{\Sigma_{11}}{||X_k||_*}$, i.e., the largest singular value divided by the nuclear norm of X_k. We further define a special case of MLLRR.

Definition 2. Local Constant Variation (LCV) LCV considers that the low rank submatrices are with spiked expected means ($\mathbb{E}$) compared to the background, i.e. $\mathbb{E}(X_{ij}) = u_k, \forall(i, j) \in I_k \times J_k$, $\mathbb{E}(X_{ij}) = u_0, \forall(i, j) \notin \{I_k \times J_k\}_{k=1}^{K}$.

Noted, LCV is a special case of MLLRR as by definition, each submatrix of LCV pattern, $X_{I_k\times J_k}$, is generated by a rank-1 matrix. We separate LCV from the general MLLRR problem because most existing methods only solve the LCV problem. In this study, we focus on solving the general MLLRR problem when local low rank matrices do not have a distinct mean difference from the background.

Let $E_k := X_k - \hat{X}_{k,I_k\times J_k}$ be the background noise of the kth submatrix, where $\hat{X}_{k,I_k\times J_k}$ denotes a submatrix of $\hat{X}_k$ indexed by $I_k\times J_k$. Importantly, real-world data is often noisy and heteroscedastic, meaning (1) the distribution for different entries in E_k may not be identical; (2) for $(i, j) \notin \{I_k \times J_k\}_{k=1}^{K}$, X_{ij} may not be identically distributed; and (3) certain entries in X_k may be outliers that could corrupt its low rank structure. Of note, in this study, we do not restrict the form of the background noise distribution.

2.3 Related works

Existing approaches of the MLLRR problem. Currently, there exist three types of methods for the MLLRR problem, namely co-clustering, sparse matrix decomposition, and anchor-based approaches. The main goal of co-clustering is to find a matrix partition such that the intra-co-cluster distance could be minimized, where the distance measure is defined as the Kullback–Leibler divergence in Bregman co-clustering [2], and Euclidean distance in the plaid model [19]. For matrix decomposition-based methods, they identify local low rank matrices by imposing sparsity constraints to the factor or singular matrices U, V [21, 38, 39, 43, 48]. For anchor-based methods, Lee et al. proposed the LLORMA method by using prior knowledge to select anchors of local low rank patterns and their nearby points with a smooth kernel function [20], and subsequently along the same line, more methods were proposed for anchor selection [25, 49]; Chen et al. proposed the WEMAREC method that builds upon the submatrices identified by co-clustering methods [6]. More details on the existing method formulations were provided in APPENDIX. Among the three types of methods, co-clustering only solves the LCV problem. Although both matrix decomposition and anchor-based methods focus on detecting local low rank submatrices, they can only detect those submatrices with a distinct spiked mean. In other words, only the local low rank submatrices exhibiting the LCV property could be detected. In addition, sparse

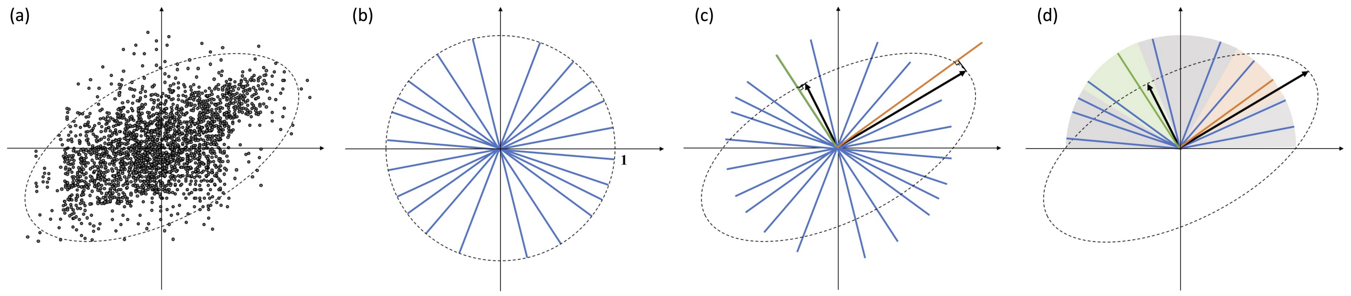

Figure 2: Mathematical considerations of random projection-based singular value computation.

matrix decomposition tends to detect large submatrix that may explain better the variance of the entire matrix, while sacrificing the locality of the submatrices [38, 48], and the high computational cost of anchor-based methods are not scalable to large matrix [6, 20]. And none of the existing methods is capable of handling outliers or heteroscedastic errors. In summary, there is lack of an effective and scalable solution for the general MLLRR problem, especially when the mean of the target pattern is not different from its background and the data contains outliers and/or heteroscedastic errors.

Random projection-based computation of SVD for global low-rank approximation. Singular value decomposition (SVD) is the best-known method for providing the true matrix rank and the optimal approximation based on the Eckart-Young Theorem [8, 13, 34]. Due to the computational limitations of classical SVD for large-scale matrices, numerous non-deterministic algorithms have been developed to address the low-rank approximation problem. Among them, randomized methods based on random projection [1, 3], which combine probability theory with numerical linear algebra to find a good approximation of the target matrix, achieve good theoretical guarantees and high computational efficiency [11, 12, 14, 23, 26, 35, 35]. These works underpin the theoretical foundation of our method. However, to the best of our knowledge, random projection has not been previously used to address the MLLRR problem.

3 RPSP AND ITS MATHEMATICAL BASIS

The biggest challenge with local low rank submatrix detection lies in the fact that neither the row nor column indices of the submatrix are known. As given in Definition 1, the low-rankness property of a submatrix is evaluated through the computation of its numerical rank, which, apparently, can't be evaluated until the submatrix has been revealed. However, it is computationally impossible to go through all the submatrices of an input matrix.

To overcome this challenge, RPSP considers the MLLRR problem from a different angle by identifying small low rank matrices and gradually growing them into larger patterns. Firstly, for a low rank matrix $X^{M\times N}$ of rank $r \ll min(M,N)$, the self consistency property suggests that any $M_0 \times N_0$ submatrix $(M_0, N_0 \geq r)$ randomly sample from X is most likely to have a rank of r (Lemma 1 in [31]). Secondly, for a given matrix, the total number of square submatrices of dimension M_0 grows exponentially with M_0. The first fact indicates that any submatrix of low rank is a collage or complete coverage of its own (smaller) submatrices, which are also of low rank. The second fact indicates that the only way for us to grow a local low rank submatrix is to start from the much smaller submatrices. In fact, for M_0 as small as 2, it is computationally feasible for us to obtain a full collection of $M_0 \times M_0$ submatrices that could densely cover $X^{M\times N}$. By teasing out all the $M_0 \times M_0$ submatrices of low rank, we could then gradually build them up into larger low rank submatrices. The evaluation of the low-rankness for a large number of submatrices now becomes computationally expensive.

In RPSP, our key contribution is the development of a new strategy to tackle the MLLRR problem by integrating two approaches, namely (1) a random projection-based assessment of low-rank submatrices and (2) a submatrix propagation approach to merge low rank submatrices into larger ones. Specifically, random projection can efficiently compute the low-rankness for a large set of small matrices. While the computational cost of random projection increases when the submatrices grow to larger ones, the number of submatrices that need to be assessed will substantially decrease because the ones without low-rankness are filtered out during the computation. The trade-off between the size and number of the to-be-evaluated submatrices ensures the efficiency of RPSP.

3.1 Efficient computation of singular values for a large set of small matrices

Conventionally, singular values are computed by QR decomposition of $O(n^3)$ complexity. In this study, we utilized an alternative approach to drastically increase the efficiency of computing singular values for a large amount of small matrices from the perspective of random projection. Without loss of generality, we illustrate our mathematical bases and the computation of singular values on square matrices. The efficient singular value computation is supported by the following three lemmas in the theory of Random Projection and Random Covering of Unit R-Sphere.

Lemma 1. *Let $\sigma_1, ..., \sigma_R$ denote the singular values of $X \in \mathbb{R}^{R\times R}$ in descending order. Let $P \in \mathbb{R}^{R\times N}$ be a random projection matrix whose columns consist of N randomly generated unit vectors, with the j-th column dented by $P_{\cdot j}$. Then $\lim_{N\to\infty} \max_{1\leq j\leq N} \|XP_{\cdot j}\|_2 = \sigma_1$ and $\lim_{N\to\infty} \min_{1\leq j\leq N} \|XP_{\cdot j}\|_2 = \sigma_R$.*

Lemma 1 indicates that the largest and smallest singular values of X could be computed through simple matrix multiplication operations between X and randomly generated projection vectors, and the computed values will converge to true ones as long as the number of projection vectors is big enough. Noted, **Lemma 1** is commonly used in random projection-based singular value estimation [29].

Lemma 2. *Let $\boldsymbol{v}_j$ be the singular vector corresponding to σ_r, $r = 1, ..., R$. Then $\lim_{N \to \infty} \max_{P_{\cdot,j} \in Sp(\boldsymbol{v}_1,...,\boldsymbol{v}_{r-1})^{\perp}, j \in 1,...,N} ||XP_{\cdot j}||_2 = \sigma_r$, for $r \in 2, ..., R$. Here, $Sp(\boldsymbol{v}_1, ..., \boldsymbol{v}_{r-1})^{\perp}$ denotes the null (or complemented) space of the linear span of $\boldsymbol{v}_1, ..., \boldsymbol{v}_{r-1}$.*

Lemma 2 illustrates an approximation methods for calculating subsequent singular values and singular vectors. Together, all the singular values and singular vectors could be well approximated by sampling a large number of projection vectors. We next show that our projection vectors are not arbitrarily selected, but rather designed to uniformly cover the space of $\mathbb{R}^{R\times 1}$, such that the sampling process could be highly efficient.

Lemma 3. *The minimum number of caps of half angle θ required to cover the unit Euclidean R-sphere is called the Random Covering of the Unit R − Sphere. Then*

$$N_c(R, \theta) = exp(R \cdot f_c(\theta)(1 + \epsilon_R(\theta))) \tag{3.1}$$

, where $\epsilon_R \longrightarrow 0$ as $R \longrightarrow \infty$ and $f_c(\theta) = -log \sin\theta$. Here $N_c(R, \theta)$ is the minimum number of caps with the given dimension R and half angle θ. This means, when R is large enough, if we randomly choose $exp(-Rlogsin\theta)$ caps, then the area of the uncovered surface of the R-sphere will be almost negligible[41, 45].

In **Lemma 1 and 2**, we first proved that the singular values of X can be estimated via simple operations against a set of randomly sampled unit vectors, including inner products, sum of squares, and max pooling, which can be efficiently computed on GPU. **Lemma 3** is derived from the theories in Random Covering of Unit R-Sphere and suggests the minimal requirement for densely covering a Unit R-dimensional Sphere. **Lemma 3** provides a bound of the cardinality of the random unit vectors to ensure that for any vector in $\mathbb{R}^R$ there almost surely exists at least one random unit vector, whose cosine similarity to the vector is larger than $\cos 2\theta$. The proof of **Lemma 1** and **Lemma 2** are given in APPENDIX and the **Lemma 3** was proven as a Corollary in the section *II* of [45].

Lemma 1, 2, and 3 together suggest that for a given matrix and the level of error to be tolerated, its singular values could be estimated by simple operations against a set of randomly generated unit vectors with a bounded size. Here we do not claim the mathematical novelty of the lemmas. However, they form important theoretical bases of **Algorithm 1** for an efficient approximation of singular values. **Fig 2** illustrates the idea of **Algorithm 1: Singular Value Approximation**. Its input includes a matrix X (**Fig 2a** illustrates a 2D example) and a set of randomly generated unit vectors P (**Fig 2b**, a 2D example). It projects X onto P and iteratively estimates top singular values (**Fig 2c** red line) and the null space (**Fig 2c** green line) of the approximated left singular vectors.

In **Algorithm 1**, $P_{(r)}$ and $\mathcal{P}_r^{nullsp}$ are estimated rth left singular vector and the null space of the linear space spanned by the first r left singular vectors, respectively. As randomly generated vectors cannot be stringently orthogonal, θ is a hyper-parameter that determines the randomly generated vectors that are in the null space of $P_{(r)}$. Noted, the null space of the linear space spanned by each P_j does not rely on X that can be computed before the random projection. Thus, the random projection and iterative computing of σ_r and $P_{(r)}$ only involve inner product, max, and sum of squares, which can be efficiently and parallelly computed on GPU for a very large set of small matrices. The max pooling step can be further optimized by first clustering the random unit vectors into groups of high cosine similarities (**Fig 2d**) and then computing random projection to the central vector of each group, as detailed in APPENDIX.

Algorithm 1: Singular Value Approximation (Based on Random Projection)

```
Inputs: $X^{R\times R}$, $N^R$ randomly generated unit vectors
    denoted as $P \in \mathbb{R}^{R\times N^R}$, cutoff $\theta$
Outputs: Estimated singular values $\sigma_1, \sigma_2, ..., \sigma_R$
Singular Value Approximation($X, P, \theta$):
$Q \leftarrow XP$
Generate vector $G$, $G_j \leftarrow \sqrt{\sum_{i=1}^{R} Q_{ij}^2}$
$\sigma_1 \leftarrow \max\{G_j | 1 \le j \le N^R\}$; $\sigma_R \leftarrow \min\{G_j | 1 \le j \le N^R\}$
$P_{(1)} \leftarrow \arg\max_{P_j} G_j$; $\mathcal{P}_0^{nullsp} \leftarrow \{P_j | P_j$ are columns of $P\}$
for $r$ in 1,...,$R-1$ do
    $\mathcal{P}_r^{nullsp} \leftarrow \{P_j | P_j \in \mathcal{P}_{r-1}^{nullsp}$,
      $\max\{\cos(P_j, P_{(1)}), ..., \cos(P_j, P_{(r)})\} < \cos(2\theta)\}$
    $\sigma_{r+1} \leftarrow \max\{G_j|$ the corresponding $P_j$ of
      $G_j \in \mathcal{P}_r^{nullsp}\}$
    $P_{(r+1)} \leftarrow \arg\max_{P_j} \{G_j | P_j \in \mathcal{P}_r^{nullsp}\}$
end
return $\{\sigma_1, \sigma_2, ..., \sigma_R\}$
```

3.2 The RPSP framework

Algorithm 2 and **Fig 3** illustrate the main framework of RPSP. The inputs of RPSP include a matrix $X^{M\times N}$ and hyper-parameters. The output are identified local low rank matrices, denoted as $\{X_{I_k \times J_k}\}, k = 1, ..., K$. The initialization step of RPSP generates random unit vectors for estimating singular values of small matrices. Specifically, N_t random vectors of length 2^t, denoted as P_t, $t = 1...T$ will be generated, where T is the number of layers for submatrices propagation.

RPSP first randomly samples L_1 number of 2×2 submatrices from X, whose singular values are estimated by **Algorithm 1** against P_1, as described in 3.1. Noted, 2×2 is the smallest unit submatrix that possess a low rank structure. The value $\frac{\sigma_1}{||P||_*}$ characterizes the low rank property of a 2×2 matrix P, where σ_1 and $||P||_*$ denote the first singular value and nuclear norm of P. The low rank property of the 2×2 submatrices is further propagated by a weighted sampling to generate L_2 number of 4×4 submatrices. Specifically, a pair of none overlapped 2×2 matrices were randomly sampled by a probability weighted by the average of their $\frac{\sigma_1}{||P||_*}$ values. The row and column indices of the two samples matrices form a new 4×4 submatrix. The singular values of the 4×4 submatrices will be estimated by random projection and orthogonal pooling

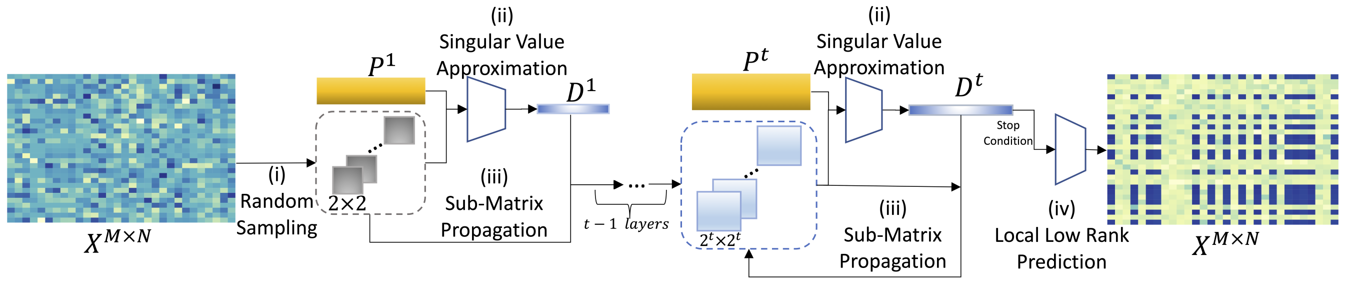

Figure 3: The framework of the RPSP algorithm.

against P_2. This procedure will be iteratively conducted T times, by which L_T number of $2^T \times 2^T$ submatrices will be randomly sampled weighted by the low-rankness propagated through the T layers, whose singular values will be estimated. For each $t = 1, ..., T$, RPSP also computes a $M \times N$ scoring matrix S^t, in which S^t_{ij} stores the frequency of observing a large value of $\frac{\sigma_1}{||P||_*}$ among all the sampled $2^t \times 2^t$ submatrices that hits to X_{ij}. S^T_{ij} can be viewed as an approximation of the probability that X_{ij} is contained by a MLLRR submatrix with a size of $2^T \times 2^T$ or larger. The local low rank submatrices in X can be further identified by a co-clustering over S^T. RPSP (**Algorithm 2**) consists the following sub algorithms:

(i) **Singular Value Approximation** (**Algorithm 1**) computes singular values for the $2^t \times 2^t$ submatrices, as described in 3.1.

(ii) **Submatrix Propagation** generates $2^{t+1} \times 2^{t+1}$ submatrices by randomly sample pairs of non-overlapped $2^t \times 2^t$ submatrices with a probability weighted by the average of the low-rankness score. This approach enable the propagation of the low-rankness of two small submatrices to a larger one if the two small submatrices truly hit one local low rank submatrix (Detailed in APPENDIX).

(iii) **Local Low Rank Prediction** reconstructs the local low rank matrices in X based on the scoring matrix S^T (Detailed in APPENDIX).

In the **Algorithm 2**, P_t denote the sets of randomly generated unit vectors; $\mathcal{R}_t$ denote the sets of the $2^t \times 2^t$ submatrices randomly sampled ($t = 1$) or weighted sampled ($t = 2, ..., T$) by **Submatrix Propagation**; $\mathcal{R}_t[j]$ denotes the jth submatrix in $\mathcal{R}_t$; $D^{t,t\times L_t}$ store the estimated singular values; $LowRankScore^T$ is a vector storing the top singular value divided by the nuclear norm of each $2^T \times 2^T$ submatrix; and S^T denotes the scoring matrix, where S^T_{ij} is the frequency of observing $LowRankScore^T > C$ for all the submatrices that contain X_{ij}. Noted, the hyper-parameters T and L_t can be easily determined based on the computational capability while C and N_t can be determined based on the level of errors that can be tolerated (see details in APPENDIX).

Algorithm 2: RPSP

```
Inputs: $X^{M\times N}$, hyper-parameters
  $T, N_t, L_t, C, K, t = 1, ..., T, k = 1, ..., K$
Outputs: The indices set $\{\mathcal{I} \times \mathcal{J}\}$, where $I_k \in \mathcal{I}, J_k \in \mathcal{J}$,
  $X_{I_k \times J_k}$ is a local low rank matrix.
RPSP($X, T, N_t, L_t, C, K$):
for $t$ in 1,...,$T$ do
    $P_t \leftarrow \{N_t$ randomly generated unit vectors of length $2^t\}$
end
$\mathcal{R}_1 \leftarrow \{L_1\ 2\times 2$ submatrices randomly sampled from $X\}$
$D^{1,2\times L_1} \leftarrow$ Singular Value Approximation($\mathcal{R}_1, P_1$)
for $t$ in 2,...,$T$ do
    $\mathcal{R}_t \leftarrow$ Submatrix Propagation($X, \mathcal{R}_{t-1}, D^{t-1}, L_t$)
    $D^{t,2^t\times L_t} \leftarrow$ Singular Value Approximation($\mathcal{R}_t, P_t$)
end
for $j$ in 1,...,$L_T$ do
    $LowRankScore^T[j] \leftarrow \frac{D^T_{1,j}}{\sum_{i=1}^T D^T_{i,j}}$
end
for $i$ in 1,...,$M$ do
    for $j$ in 1,...,$N$ do
        $S^T_{ij} \leftarrow$ frequency of $LowRankScore^T[k] > C$ for all
          $\mathcal{R}_t[k]$ contains $X_{ij}$
    end
end
$\mathcal{I}\times\mathcal{J} \leftarrow$ Local Low Rank Prediction($S^T$)
return $\mathcal{I}\times\mathcal{J}$
```

3.3 Computational cost

RPSP contains four major steps, namely (1) randomly generating unit vectors, (2) **Singular Value Approximation**, (3) **Submatrix Propagation**, and (4) **Local Low Rank Prediction**. The computational cost of **Submatrix Propagation** is determined by the number of sampled submatrix pairs, which could be optimized based on the computational resource. The computational complexity of **Local Low-Rank Prediction** is $O(max\{M, N\}^3)$. The most time and computationally consuming part is the generation of random unit vectors and the **Algorithm 1: Singular Value Approximation**, which takes more than 95% running time of RPSP.

We further evaluated the computational efficiency and the accuracy of **Algorithm 1** versus conventional QR decomposition-based SVD on both GPU and CPU servers (see details in APPENDIX[2]). We tested the two methods 50 times on 10^5, 10^6, 10^7 and 10^8 number of 2×2, 10^5 number of 4×4, and 10^5 number of 8×8 matrices. The averaged normalized root mean squared error between estimated and true singular values is 0.07. We observed that **Algorithm 1**

[2]Full APPENDIX at https://github.com/ptdang1001/RPSP

used $10^{-5} - 10^{-2}$ seconds, which is consistently about 10^5 times faster than QR decomposition-based SVD (see details in APPENDIX and APPENDIX Table 2[2]).

Based on the parameters of GPU machine, the computational cost of RPSP could be optimized by changing its hyperparameters to control the running time within a few seconds to a few minutes, which is comparable to SOTA methods (see EXPERIMENTS).

4 EXPERIMENTS ON SYNTHETIC DATA

We evaluated the overall performance and computational cost of RPSP in different scenarios of MLLRR problem and compared RPSP with SOTA methods on a comprehensive setup of synthetic datasets.

4.1 Experimental setup

We simulate $X \in \mathbb{R}^{M \times N}$ as $X = \sum_{k=1}^{K} X^k + E$. Here, entries in X^k is padded by zero except for those indexed by I_k and J_k, corresponding to a local low rank submatrix with m_k rows and n_k columns. E is background noise simulated by $E_{i,j} \sim N(0, \alpha_k * sd), \forall i \in I_k, j \in J_k$; and the rest of the entries in E follows $N(0, sd)$. In evaluating the algorithm's scalability, we allow M and N to have three different values. Otherwise, we let $M = N = 1000$. To simulate $X^k_{I_k \times J_k}$, we first simulated Y_k as $Y_k = U_k V_k^T$, where $U_k \in \mathbb{R}^{m_k \times r_k}$ and $V_k \in \mathbb{R}^{n_k \times r_k}$, and entries in U_k, V_k all follow $U(0,1)$. Then $X^k_{I_k, J_k}$ is simulated as $Y_k - \overline{Y_k} + \mu_k$. Here, $\overline{Y_k}$ denotes the element-wise mean of matrix Y_k, and hence μ_k mimics the overall mean of the k-th pattern matrix, and α_k mimics the relative noise level of the local low rank matrix to the overall background noise matrix. In total, we obtained 284 different simulation scenarios, each has 5 repetitions, which include:

(1) **Perturbed pattern mean:** pattern mean $\mu_k = \beta_k * sd$, where β_k is a sequence from 0 to 3 with step size 0.1; relative noise level $\alpha_k = \{0, 0.1\}$; pattern size $m_k = n_k = \{200, 500\}$.

(2) **Perturbed background error:** pattern mean $\mu_k = \beta_k * sd$, where $\beta_k = \{0, 0.1\}$; relative noise level α_k is a sequence from 0 to 3 with step size 0.1; pattern size $m_k = n_k = \{200, 500\}$.

(3) **Perturbed pattern size:** pattern mean $\mu_k = \beta_k * sd$, where $\beta_k = \{0, 0.1\}$; relative noise level $\alpha_k = \{0, 0.1\}$; pattern size $m_k = n_k$ is a sequence from 100 to 500 with step size of 20.

We evaluated the method performance of RPSP and selected SOTA methods on these synthetic datasets, based on how well the identified patterns hit the true ones, and avoid the background noise. We label the entries hitting true patterns as "positive" and the rest as "negative", and the True Positive (TP), False Negative (FP), False Positive (FP), and True Negative (TN) occurrences are defined as the number of "positive" entries that are identified as pattern (TP) or background (FP), or the number of "negative" entries that are identified as pattern (FP) or background (TN). The overall prediction accuracy is defined as $\frac{TP+TN}{TP+TN+FP+FN}$.

4.2 Performance evaluation of RPSP

We benchmarked RPSP with five SOTA methods, namely Bregman co-clustering (CC) [2] and Plaid [19], two sparse matrix decomposition methods (SSVD [48] and SPCA [50]), and one anchor-based method LLORMA [20]. Detailed parameter settings of RPSP and other methods are provided in APPENDIX[2].

Accuracy in solving the MLLRR problem under different scenarios. Fig 4a-c illustrated the accuracy (y-axis) of RPSP (red) and other methods for solving the MLLRR problem with and without LCV property in different scenarios. Overall, RPSP achieved higher than 0.8 accuracies under most settings, which is consistently higher than all baseline methods. On the dense data, SPCA failed to identify any pattern while CC detects the whole matrix as one pattern, hence these two methods were excluded from further analysis. RPSP is the only method that can identify local low rank patterns when its mean is close to the background mean (**Fig 4a**), i.e., the MLLRR problem without the LCV property. As expected, we observed the prediction accuracy of the baseline methods to increase as the mean difference becomes larger (**Fig 4a**), and all methods to have decreased performance with the increase of the noise level (**Fig 4b**). RPSP and LLORMA are more robust to high noise levels compared to SSVD and Plaid. The size test suggested that RPSP can accurately identify the pattern when its size is even smaller than 100×100 in a 1000×1000 matrix (**Fig 4c**). When the pattern size increases, the prediction accuracy of RPSP and LLORMA also increases, but not SSVD or Plaid. An explanation is that a larger pattern is easier to be hit by the randomly sampled submatrices in RPSP or the anchoring in LLORMA, while SSVD and Plaid rely on the pattern sparsity assumption and are less sensitive to large patterns.

Power in detecting multiple patterns and the submatrices of different ranks. Where there exists more than 1 local rank-1 sub matrices, RPSP again achieved high performance (**Fig 4d1-2**). The way RPSP detects local low rank matrices is from the scoring matrices, which are less impacted by the number of patterns. It is noteworthy that we focus on the general MLLRR problem in a dense matrix, especially when the LCV property does not hold, while LLORMA and Plaid are more efficient on the LCV problem in a sparse matrix. On the dense matrix, all the baseline methods failed to identify the local low rank pattern when the mean difference between the pattern and the background is low. We also evaluated RPSP on identifying local low rank patterns of different dimensions (**Fig 4d4**). Our results demonstrated that RPSP has a high robustness in detecting patterns of different dimensions. Noted, the specificity of RPSP is always bounded by $1 - a$, where a is the probability of the presence of a local low rank matrix in a noise matrix.

Running time. We evaluated the time consumption of the methods on dense matrices of three sizes, $M = N = 10^2, 10^3, 10^4$ (**Fig 4d3**). The running time of RPSP, Plaid and LLORMA are at a similar level. Detailed experimental results and parameters of the GPU and CPU server for the experiment are provided in APPENDIX[2].

Robustness of sub-algorithms. As each sub-algorithm is necessary for RPSP, no ablation experiment can be conducted. We evaluated the robustness of sub-algorithms with respect to their hyperparameters, including the cutoff θ in **Algorithm 1**, the low-rankness cutoff C, the dimension of submatrices T, the number of randomly sampled unit vectors N_t, and the number of randomly generated or propagated submatrices L_t in **RPSP**. We found the performance of **Algorithm 1** is highly robust if $cos(2\theta)$ is larger than 0.6 (APPENDIX[2], Table 4 and 5). Our experiment also suggested that *RPSP* is robust if $C > 0.75$ is set. Perturbing C in our synthetic data-based experiment suggested $C = 0.95$ for $T = 2$ and $C = 0.8$ for $T = 4, 8, 16$ achieved optimal detection accuracy, which

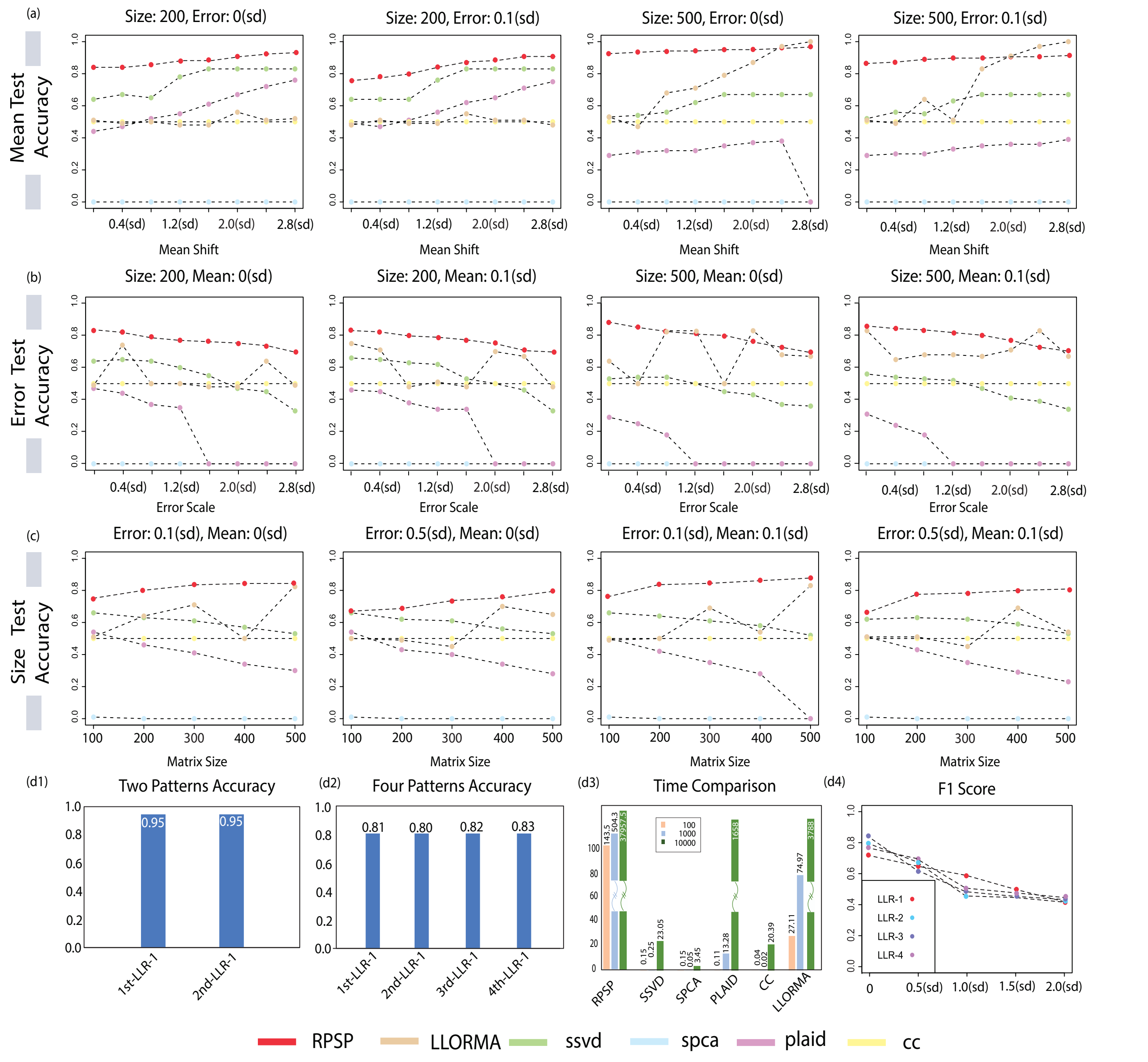

Figure 4: Benchmark of RPSP on Synthetic Data.

was used in both synthetic and real-world experiments. Changing $T = \{2, 4, 8, 16\}$ to $\{3, 6, 12, 24\}$ or $\{3, 9, 27\}$ drastically increased the computational consumption but did not improve the algorithm performance. N_t is determined by **Lemma 3**. L_t is determined by input matrix size and computational capacity. A larger L_t will achieve better performance. Here we set $L_1 = 10^7$ and $L_t = L_{t-1}/10$.

5 EXPERIMENTS ON REAL-WORLD DATA

We benchmarked RPSP on four real-world datasets of different density rates (proportion of non-zero entities in the overall input matrix), error distributions, and local low rank patterns, namely (1) the MovieLens data, (2) two single-cell RNA-seq data, and (3) a spatial transcriptomics data. Details of data processing and algorithm settings are given in APPENDIX[2]. Four metrics were utilized to evaluate the performance of each method, namely (1) the low-rankness, calculated as the averaged $\frac{\sigma_1}{||P||_*}$, (2) the averaged Size of the identified submatrices P, (3) the total Coverage Rate(CR), defined as the total number of entries in the top-k patterns divided by the size of the input matrix, and (4) the Running Time. Strong

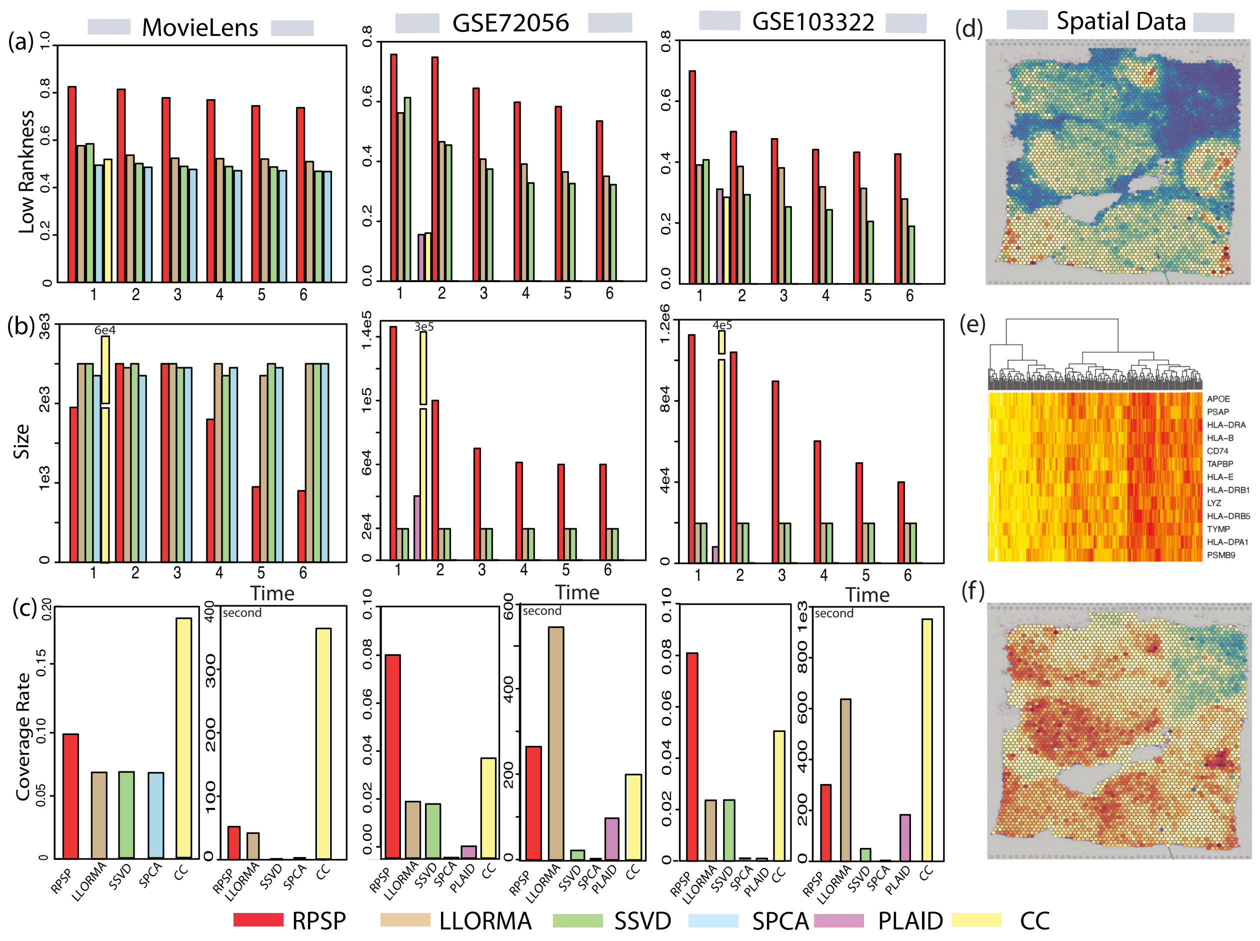

Figure 5: Experiment on real-world data.

low-rankness, high averaged size and coverage rate, and short running time suggest high performance. We refrained from using the F-1 score and accuracy metrics because the accurate pattern in real-world data remains unknown. In addition, the context-specific meanings of detected local low rank patterns were evaluated based on prior knowledge. **Table 6**[2] and **Fig 5** summarize the major results. Complete discussions of the real-world data experiments are given in APPENDIX[2].

In summary, RPSP outperforms baseline methods on the four real-world datasets, MovidLens data[15], single cell RNA-seq data[33] and spatial transcriptomics data, in terms of the low-rankness, size, coverage rate, and contextual interpretability of detected local low rank matrices. The running time of RPSP is at a similar level to baseline methods. For a deeper dive into the real-world datasets used and further experimental details, please refer to the APPENDIX[2].

6 CONCLUSION

In this work, we provided a new computational framework, namely RPSP, to detect local low rank matrices. RPSP is supported by rigorously derived mathematical theories. While existing methods mainly solve the LCV sub-problem of MLLRR, our developed RPSP is the first method capable of handling the general MLLRR problem. RPSP utilizes a random projection and GPU-based method to efficiently compute singular values and low-rankness for a large set of small matrices. RPSP further propagates the low-rankness identified from small matrices to identify larger local low rank matrices of coherent patterns. In both synthetic and real-world experiments, we demonstrated that RPSP outperforms all baseline methods on the general MLLRR problems for data of different sparsity levels and error distributions. Particularly, RPSP could detect low rank submatrices even when its mean structure is not distinguishable from the background, or when the error distribution is heteroscedastic. The source code, analysis, testing data and a comprehensive supplementary of RPSP are available via https://github.com/ptdang1001/RPSP.

7 ACKNOWLEDGMENTS

This research was supported in part by NSF [IIBR 2047631, IIS 2145314]; American Cancer Society [RSG-22-062-01-MM]; NIGMS [1R01GM131399]; NIH [R35 GM147241-01].

REFERENCES

[1] Dimitris Achlioptas. 2003. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of computer and System Sciences* 66, 4 (2003), 671–687.

[2] Arindam Banerjee, Inderjit Dhillon, Joydeep Ghosh, Srujana Merugu, and Dharmendra S Modha. 2007. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. *Journal of Machine Learning Research* 8, Aug (2007), 1919–1986.

[3] Ella Bingham and Heikki Mannila. 2001. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining.*

[4] Thierry Bouwmans and El Hadi Zahzah. 2014. Robust PCA via principal component pursuit: A review for a comparative evaluation in video surveillance. *Computer Vision and Image Understanding* 122 (2014), 22–34.

[5] Wennan Chang, Changlin Wan, Yong Zang, Chi Zhang, and Sha Cao. 2020. Supervised clustering of high dimensional data using regularized mixture modeling. *arXiv preprint arXiv:2007.09720* (2020).

[6] Chao Chen, Dongsheng Li, Yingying Zhao, Qin Lv, and Li Shang. 2015. WEMAREC: Accurate and scalable recommendation through weighted and ensemble matrix approximation. In *Proceedings of the 38th ACM SIGIR.* 303–312.

[7] Yao Cheng, Liang Yin, and Yong Yu. 2014. LorSLIM: low rank sparse linear methods for top-n recommendations. In *2014 IEEE International Conference on Data Mining.* IEEE, 90–99.

[8] Biswa Nath Datta. 2010. *Numerical linear algebra and applications.* Vol. 116. Siam.

[9] Inderjit S Dhillon. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining.* 269–274.

[10] Chris Ding, Tao Li, Wei Peng, and Haesun Park. 2006. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining.* 126–135.

[11] Eleni Drinea, Petros Drineas, and Patrick Huggins. 2001. A randomized singular value decomposition algorithm for image processing applications. In *Proceedings of the 8th panhellenic conference on informatics.* Citeseer, 278–288.

[12] Petros Drineas and Michael W Mahoney. 2016. RandNLA: randomized numerical linear algebra. *Commun. ACM* 59, 6 (2016), 80–90.

[13] Gene H Golub and Charles F Van Loan. 2013. *Matrix computations.* JHU press.

[14] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review* 53, 2 (2011), 217–288.

[15] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015).

[16] Sepp Hochreiter, Ulrich Bodenhofer, Martin Heusel, Andreas Mayr, Andreas Mitterecker, Adetayo Kasim, Tatsiana Khamiakova, Suzy Van Sanden, Dan Lin, Willem Talloen, et al. 2010. FABIA: factor analysis for bicluster acquisition. *Bioinformatics* 26, 12 (2010), 1520–1527.

[17] Zhanxuan Hu, Feiping Nie, Rong Wang, and Xuelong Li. 2021. Low rank regularization: A review. *Neural Networks* 136 (2021), 218–232.

[18] N Kishore Kumar and Jan Schneider. 2017. Literature survey on low rank approximation of matrices. *Linear and Multilinear Algebra* 65, 11 (2017), 2212–2244.

[19] Laura Lazzeroni and Art Owen. 2002. Plaid models for gene expression data. *Statistica sinica* (2002), 61–86.

[20] Joonseok Lee, Seungyeon Kim, Guy Lebanon, Yoram Singer, and Samy Bengio. 2016. LLORMA: Local low-rank matrix approximation. *JMLR* 17 (2016), 442–465.

[21] Mihee Lee, Haipeng Shen, Jianhua Z Huang, and JS Marron. 2010. Biclustering via sparse singular value decomposition. *Biometrics* 66, 4 (2010), 1087–1095.

[22] Jingjing Li, Ke Lu, Zi Huang, and Heng Tao Shen. 2019. On both cold-start and long-tail recommendation with social data. *IEEE Transactions on Knowledge and Data Engineering* 33, 1 (2019), 194–208.

[23] Mu Li, Wei Bi, James T Kwok, and Bao-Liang Lu. 2014. Large-scale Nyström kernel matrix approximation using randomized SVD. *IEEE transactions on neural networks and learning systems* 26, 1 (2014), 152–164.

[24] Edo Liberty, Franco Woolfe, Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. 2007. Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences* 104, 51 (2007), 20167–20172.

[25] Huafeng Liu, Liping Jing, Yuhua Qian, and Jian Yu. 2019. Adaptive local low-rank matrix approximation for recommendation. *ACM Transactions on Information Systems (TOIS)* 37, 4 (2019), 1–34.

[26] Michael W Mahoney et al. 2011. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning* 3, 2 (2011), 123–224.

[27] Ivan Markovsky. 2008. Structured low-rank approximation and its applications. *Automatica* 44, 4 (2008), 891–909.

[28] Per-Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. 2011. A randomized algorithm for the decomposition of matrices. *Applied and Computational Harmonic Analysis* 30, 1 (2011), 47–68.

[29] Per-Gunnar Martinsson and JA Tropp. 2020. Randomized numerical linear algebra: foundations & algorithms (2020). *arXiv preprint arXiv:2002.01387* (2020).

[30] Bamdev Mishra, Gilles Meyer, Francis Bach, and Rodolphe Sepulchre. 2013. Low-rank optimization with trace norm penalty. *SIAM Journal on Optimization* 23, 4 (2013), 2124–2149.

[31] Art B Owen and Patrick O Perry. 2009. Bi-cross-validation of the SVD and the nonnegative matrix factorization. (2009).

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[33] Sidharth V Puram, Itay Tirosh, Anuraag S Parikh, Anoop P Patel, et al. 2017. Single-cell transcriptomic analysis of primary and metastatic tumor ecosystems in head and neck cancer. *Cell* 171, 7 (2017), 1611–1624.

[34] Benjamin J Sapp and Written Preliminary Examination II. 2011. Randomized algorithms for low rank matrix decomposition. *Computer and Information Science, University of Pennsylvania* 24 (2011).

[35] Tamas Sarlos. 2006. Improved approximation algorithms for large matrices via random projections. In *2006 47th annual IEEE symposium on foundations of computer science (FOCS'06).* IEEE, 143–152.

[36] Romain Serizel, Marc Moonen, Bas Van Dijk, and Jan Wouters. 2014. Low-rank approximation based multichannel Wiener filter algorithms for noise reduction with application in cochlear implants. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22, 4 (2014), 785–799.

[37] Moein Shakeri and Hong Zhang. 2016. COROLA: A sequential solution to moving object detection using low-rank approximation. *Computer Vision and Image Understanding* 146 (2016), 27–39.

[38] Haipeng Shen and Jianhua Z Huang. 2008. Sparse principal component analysis via regularized low rank matrix approximation. *JMA* 99, 6 (2008), 1015–1034.

[39] Martin Sill, Sebastian Kaiser, Axel Benner, and Annette Kopp-Schneider. 2011. Robust biclustering by sparse singular value decomposition incorporating stability selection. *Bioinformatics* 27, 15 (2011), 2089–2097.

[40] Larry J Stockmeyer. 1975. *The set basis problem is NP-complete.* IBM Thomas J. Watson Research Division.

[41] Aaron R Voelker, Jan Gosmann, and Terrence C Stewart. 2017. Efficiently sampling vectors and coordinates from the n-sphere and n-ball. *Centre for Theoretical Neuroscience-Technical Report* 1 (2017).

[42] Changlin Wan, Wennan Chang, Yu Zhang, Fenil Shah, Xiaoyu Lu, Yong Zang, Anru Zhang, Sha Cao, Melissa L Fishel, Qin Ma, et al. 2019. LTMG: a novel statistical modeling of transcriptional expression states in single-cell RNA-Seq data. *Nucleic acids research* 47, 18 (2019), e111–e111.

[43] Daniela M Witten, Robert Tibshirani, and Trevor Hastie. 2009. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics* 10, 3 (2009), 515–534.

[44] Svante Wold. 1978. Cross-validatory estimation of the number of components in factor and principal components models. *Technometrics* 20, 4 (1978), 397–405.

[45] Aaron D Wyner. 1967. Random packings and coverings of the unit n-sphere. *The Bell System Technical Journal* 46, 9 (1967), 2111–2118.

[46] Chun-Qiu Xia, Ke Han, Yong Qi, Yang Zhang, and Dong-Jun Yu. 2017. A self-training subspace clustering algorithm under low-rank representation for cancer classification on gene expression data. *IEEE/ACM transactions on computational biology and bioinformatics* 15, 4 (2017), 1315–1324.

[47] Yang Xu, Lei Zhu, Zhiyong Cheng, Jingjing Li, and Jiande Sun. 2020. Multi-feature discrete collaborative filtering for fast cold-start recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 270–278.

[48] Dan Yang, Zongming Ma, and Andreas Buja. 2014. A sparse singular value decomposition method for high-dimensional data. *Journal of Computational and Graphical Statistics* 23, 4 (2014), 923–942.

[49] Xuejiao Yang and Bang Wang. 2019. Local matrix approximation based on graph random walk. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 1037–1040.

[50] Hui Zou, Trevor Hastie, and Robert Tibshirani. 2006. Sparse principal component analysis. *Journal of computational and graphical statistics* 15, 2 (2006), 265–286.

A APPENDIX

A.1 Sub-Algorithms of RPSP

A.1.1 submatrix Propagation. The **Algorithm 3 submatrix Propagation** conducts weighted sampling to generate $2^t \times 2^t$ submatrices based on the low-rankness of $2^{t-1} \times 2^{t-1}$ submatrices. The input of **Algorithm 3 submatrix Propagation** include the input matrix X, sampled $2^{t-1} \times 2^{t-1}$ submatrices, their singular values D^t, and the number of to be generated $2^t \times 2^t$ submatrices L_t.

A.1.2 Local Low Rank Prediction. RPSP further utilizes **Algorithm 4 Local Low Rank Prediction** to identify and

Algorithm 3: Submatrix Propagation

```
Inputs: $X, \mathcal{R}_{t-1}, D^{t-1}, L_t$
Outputs: $\mathcal{R}_t$
submatrix Propagation($X, \mathcal{R}_{t-1}, D^{t-1}, L_t$):
$\mathcal{R}_t \leftarrow \varnothing$
while $|R_t| < L_t$ do
    $Prob \sim U(0,1)$
    Randomly pick two non-overlapping submatrices from
      $\mathcal{R}_{t-1}$, denoted as $\mathcal{R}_{t-1}[i]$ and $\mathcal{R}_{t-1}[j]$
    $W \leftarrow \frac{D^{t-1}_{1,i} D^{t-1}_{1,j}}{\sum_{k=1}^{2^{t-1}} D^{t-1}_{k,i} \sum_{k=1}^{2^{t-1}} D^{t-1}_{k,j}}$
    if $Prob<W$ then
        $I$ ←Row indices of $\mathcal{R}_{t-1}[i]$ and $\mathcal{R}_{t-1}[j]$
        $J$ ←Column indices of $\mathcal{R}_{t-1}[i]$ and $\mathcal{R}_{t-1}[j]$
        $append(\mathcal{R}_t, X_{I\times J})$
end
return $\mathcal{R}_t$
```

reconstruct the local low rank matrices. RPSP computes a $M \times N$ scoring matrix S^T, in which S^T_{ij} stores the frequency of observing a large value of $\frac{\sigma_1}{||P||_*}$ among all the sampled $2^T \times 2^T$ submatrices that hits to X_{ij}. Hence, S^T_{ij} can be viewed as an approximation of the probability that X_{ij} is contained by a MLLRR submatrix with a size of $2^T \times 2^T$ or larger. Here we applied the **Spectral Co-Clustering** method developed by Dhillon et al [9] and the python library provided by scikit-learn [32] on S^T to identify local low rank submatrices. With the indices of each possible local low rank matrix identified, the local patterns were ranked by the level of their top singular values normalized by the sum of all singular values. Here the local patterns of the top K significant low rank property or with the top singular values large than a certain threshold form the final output of RPSP.

Algorithm 4: Local Low Rank Prediction

```
Inputs: $S^{T,M\times N}$
Outputs: $\mathcal{I} \times \mathcal{J}$
Local Low Rank Prediction($S^{T,M\times N}$):
$\mathcal{I} \times \mathcal{J} \leftarrow$ Spectral Co-Clustering($S^T$)
return $\mathcal{I} \times \mathcal{J}$
```

A.1.3 Optimize the max pooling with respect to null space in ***Algorithm 1****.* The max pooling with respect to null space can be further optimized by clustering the random unit vectors into groups of high cosine similarities. Specifically, the randomly sampled unit vectors were first clustered by using the K-mean of their cosine distance (1-cosine similarity). Then after $P_{(1)}, ..., P_{(r)}$ were identified, the null space of the linear span of $\{P_{(1)}, ..., P_{(r)}\}$ was estimated by the union of the clusters whose center P^C satisfies $\max\{\cos(P^C, P_{(1)}), ..., \cos(P^C, P_{(r)})\} < \cos(\theta)\}$. This approach effectively reduces the number of cosine distances needed to be computed.

A.1.4 Assessment of hyper-parameters of RPSP. RPSP has five hyper-parameters T, C, L_t, K, and N_t. L_t (number of randomly sampled or propagated submatrices) can be determined based on the input matrix size and computational capacity. T (number of layers for submatrix propagation) is set as 4 for efficient computation. C (threshold of $LowRankScore$) can be computed by randomly sampling $2^T \times 2^T$ submatrices of pure noise from randomly shuffled X and generating an empirical null distribution of $LowRankScore$. N_t (number of random unit vectors) can be determined by **Lemma 2**.

A.2 Mathematical Derivations And Considerations

A.2.1 Truncated SVD. Let $X^{M\times N} = U\Sigma V^T (M \geq N)$ be the SVD of X, where $U^{M\times N}$ and $V^{N\times N}$ are left and right singular vector matrices, $\Sigma^{N\times N}$ is a diagonal matrix of singular values. Define $\Sigma^{(r)}$ such that $\Sigma^{(r)}_{ii} = \Sigma_{ii}, i \leq r$; $\Sigma^{(r)}_{ii} = 0, i > r$, i.e., only keeps the top r singular values > 0. The truncated SVD of X of rank r is defined as $tSVD(X,r) = U\Sigma^{(r)}V^T$. Noted, $Rank(X) \leq r$ if and only if $X = tSVD(X,r)$.

A.2.2 Mathematical considerations of the MLLRR problem. The biggest challenge with local low rank submatrix detection lies in that neither the row or column indices of the submatrix are known. As given in Definition 2, the low-rankness property of a submatrix is evaluated through the computation of its singular values, which apparently can't be evaluated until the submatrix has been presented. However, it is computationally impossible to go through all the submatrices of an input matrix. RPSP grows a submatrix of low rank from smaller ones, which utilizes two facts. Firstly, for a low rank matrix $X^{M\times N}$ of rank $r \ll min(M,N)$, the self consistency property suggests that any $M_0 \times N_0$ submatrix $(M_0, N_0 \geq r)$ randomly sample from X is most likely to have a rank of r (Lemma 1 in [31]). Secondly, for a given matrix, the total number of square submatrices of dimension M_0 grows exponentially with M_0. The first fact indicates that any submatrix of low rank is a collage or complete coverage of its own (smaller) submatrices, which are also of low rank. The second fact indicates that the only way for us to grow a local low rank submatrix is to start from the much smaller submatrices. In fact, for M_0 as small as 2, it is computationally feasible for us to obtain a full collection of $M_0 \times M_0$ submatrices that could densely cover $X^{M\times N}$. By teasing out all the $M_0 \times M_0$ submatrices of low rank, we could then gradually build them up into larger low rank submatrices. The evaluation of the low-rankness for a large number of submatrices now becomes computationally expensive. In RPSP, our biggest contribution is that we have developed a singular value approximation method using random projection to efficiently evaluate the low-rankness of any given submatrix, making it possible for us to build a submatrix from its parts.

A few examples can illustrate why a global search cannot effectively solve the MLLRR problem. We consider the following square matrices:

(1) $X^{M\times M}$ in which $X_{ij} \sim N(0,1)$ *i.i.d.*. Here X is a matrix of standard Gaussian error. The largest singular value of X is about $2\sqrt{M}$.
(2) $Y^{M\times M}$ in which $Y_{i,\cdot} \equiv Y'$, $Y'[i] \sim N(0,1)$ *i.i.d.*. Here Y is a matrix of rank=1 that has the same level of mean and standard deviation as X. Noted, the largest singular value of Y is about M.

Table 1: Existing methods of MLLRR

Methods	Examples	Formulation	Tasks	Assumption
Co-clustering	Bregman; Plaid	$\min_{I_k,J_k,\mu_k} \sum_k \sum_{i\in I_k, j\in J_k} d(x_{ij},\mu_k)$	LCV	Matrix partition
Matrix decomposition	SSVD; SPCA	$\min_{U,V}(\lVert X-UV^T\rVert_F^2+\lambda_u\lVert U\rVert_1+\lambda_v\lVert V\rVert_1)$,	LCV MLLRR with LCV	Sparse patterns
Anchor based methods	LLORMA WEMAREC	$\min_{\hat{I},\hat{J},\hat{X}}(K_{X[\hat{I},\hat{J}]}\odot P_{X[\hat{I},\hat{J}]}(X-\hat{X}))$	MLLRR with LCV	submatrix detection

(3) $Z^{M\times M}$ in which $Z_{ij}\equiv a$. The largest singular value of Y is $M\times a$.

Hence for a low rank sub-matrix of size $\sqrt{2M}\times\sqrt{2M}$ or smaller, whose mean and the standard deviation are not different from the background's, it is less likely to be identified by a global SVD as the largest singular value of the sub-matrix is about the same level of the largest singular value of the background noise matrix. However, if the low rank sub-matrix has a spiked mean, its largest singular value will be amplified by the spiked mean and the top singular vector of the whole matrix is naturally sparse. Hence, an LCV problem is more likely to be solved by a global search while the MLLRR problem of the insignificant mean difference between pattern and background is less likely to be detected by a global search. So, it is necessary to think of an alternative approach to solving the general MLLRR problem. Noted, the idea of screening a large set of small submatrices and propagating the low rank property of smaller ones to bigger submatrices only involves the computing of singular values of local patterns. Hence, we do not expect that the RPSP method may have disparate performances in solving LCV and LRR problems.

A.2.3 Mathematical formulations of SOTA methods. Co-clustering methods simultaneously cluster rows and columns of a two-dimensional data matrix. The general assumption is that the targeted submatrix has a larger or small mean value compared to the background noise. The Bregman co-clustering method generates a matrix partition I_k, J_k by preserving the maximum information of data X within the partitions. The approximation error $M(I,J)-M(\tilde{I},\tilde{J})$ represents the difference between the preserved information and original data, here $M(I,J)$ is the mutual information and $M(I,J)-M(\tilde{I}-\tilde{J}) = KL(dist_1(I,J)\|dist_2(I,J))$. Laura et al. proposed the Plaid model to detect the submatrix by fitting each entry X_{ij} with K layers and make sure the summation of all layers $\sum_{k=1}^{K}\mu_k I_k J_k$ approximate the original value. Sparse SVD-based methods identify local low rank matrices by adding L1 sparse penalty to a global truncated SVD fitting. However, this type of method still demands distinct mean differences between pattern and background and trend to detect large low rank patterns that may explain the variance of the whole matrix. Lee et al. proposed the LLORMA method by using prior knowledge to select anchors of local low rank patterns. As listed in table 1, K_{Ω}^{h} is the kernel function with bandwidth h to smooth the projection value $P_{\Omega}(\cdot)$ near the anchor points Ω. However, this type of method, highly depends on prior knowledge that cannot solve the general MLLRR problem.

A.2.4 Proofs of Lemma 1 and Lemma 2. **Lemma 1 and 2** can be expressed and proved together. The two lemmas describe the following properties of random projection. For a given dimension R, denote $X^{R\times R}$ as an input matrix and $P\in\mathbb{R}^{R\times N^R}$ as a matrix of N^R randomly generated unit vectors in $\mathbb{R}^R$. $Y=XP$ denotes a random projection of X, then $\lim_{N^R\to\infty}\max_{1\le j\le N^R}\sqrt{\sum_{i=1}^{R}Y_{ij}^2}=\sigma_1$ and $\lim_{N^R\to\infty}\min_{1\le j\le N^R}\sqrt{\sum_{i=1}^{R}Y_{ij}^2}=\sigma_R$, here σ_1 and σ_R are the largest and smallest singular values of X. Denote $P_{(1)}=\arg\max_{P_{\cdot,j}}\sqrt{\sum_{i=1}^{R}Y_{ij}^2}$, $\lim_{N^R\to\infty}\max_{P_{\cdot,j}\in Sp(P_{(1)})^{\perp}}\sqrt{\sum_{i=1}^{R}Y_{ij}^2}=\sigma_2$, where $Sp(P_{(1)})^{\perp}$ denotes the null (or complemented) space of the linear space spanned by $P_{(1)}$. Similarly, define $P_{(r)}=\arg\max_{P_{\cdot,j}\in Sp(P_{(1)},\ldots,P_{(r-1)})^{\perp}}\sqrt{\sum_{i=1}^{R}Y_{ij}^2}$, $\lim_{N^R\to\infty}\max_{P_{\cdot,j}\in Sp(P_{(1)},\ldots,P_{(r-1)})^{\perp}}\sqrt{\sum_{i=1}^{R}Y_{ij}^2}=\sigma_r$, for $r\in 3,\ldots,R-1$.

Proof. Noted, $\sum_{i=1}^{R}Y_{ij}^2$ is the norm of the projection of X onto $Y_{\cdot,j}$. $Y_{\cdot,j}=XP_{\cdot,j}=U\Sigma V^T P_{\cdot,j}=\sum_{k=1}^{R}U_{\cdot,k}\Sigma_{kk}V_{k,\cdot}^T P_{\cdot,j}$, here $U\Sigma V^T$ is the SVD of X. Then we have $\sum_{i=1}^{R}Y_{ij}^2=\sum_{i=1}^{R}(\sum_{k=1}^{R}U_{ik}\Sigma_{kk}V_{k,\cdot}^T P_{\cdot,j})^2=\sum_{k=1}^{R}\sum_{i=1}^{R}(U_{ik})^2(\Sigma_{kk})^2(V_{k,\cdot}^T P_{\cdot,j})^2=\sum_{k=1}^{R}(\Sigma_{kk})^2(V_{k,\cdot}^T P_{\cdot,j})^2$ as U is orthogonal, both Σ_{kk} and $V_{k,\cdot}^T P_{\cdot,j}$ are scalars. Hence the largest and smallest $\sum_{i=1}^{R}Y_{ij}^2$ is Σ_{11} and Σ_{RR}, which are achieved when $P_{\cdot,j}$ is $V_{1,\cdot}^T$ and $V_{R,\cdot}^T$, respectively. As $Sp(Y_{(1)},\ldots,Y_{(r-1)})^{\perp}$ is the null space of the linear span of $Y_{(1)},\ldots,Y_{(r-1)}$, the largest projection of X onto this space is σ_r when $P_{\cdot,j}$ is $V_{r,\cdot}^T$. $\square$

For more detailed information about our experiments, please refer to the comprehensive appendix available at https://github.com/ptdang1001/RPSP.