The Permuted Striped Block Model and its Factorization - Algorithms with Recovery Guarantees

Michael Murray *†, Jared Tanner *†

April 10, 2020

Abstract

We introduce a novel class of matrices which are defined by the factorization $\mathbf{Y} := \mathbf{AX}$, where $\mathbf{A}$ is an $m \times n$ wide sparse binary matrix with a fixed number d nonzeros per column and $\mathbf{X}$ is an $n \times N$ sparse real matrix whose columns have at most k nonzeros and are *dissociated*. Matrices defined by this factorization can be expressed as a sum of n rank one sparse matrices, whose nonzero entries, under the appropriate permutations, form striped blocks - we therefore refer to them as Permuted Striped Block (PSB) matrices. We define the *PSB data model* as a particular distribution over this class of matrices, motivated by its implications for community detection, provable binary dictionary learning with real valued sparse coding, and blind combinatorial compressed sensing. For data matrices drawn from the PSB data model, we provide computationally efficient factorization algorithms which recover the generating factors with high probability from as few as $N = \mathcal{O}\left(\frac{n}{k}\log^2(n)\right)$ data vectors, where k, m and n scale proportionally. Notably, these algorithms achieve optimal sample complexity up to logarithmic factors.

1 Introduction

In many data science contexts, data is represented as a matrix and is often factorized into the product of two or more structured matrices so as to reveal important information. Perhaps the most famous of these factorizations is principle component analysis (PCA) [22]), in which the unitary factors represent dominant correlations within the data. Dictionary learning [35] is another prominent matrix factorization in which the data matrix is viewed to lie, at least approximately, on a union of low rank subspaces. These subspaces are represented as the product of an overcomplete matrix, known as a dictionary, and a sparse matrix. More generally a wide variety of matrix factorizations have been studied to solve a broad range of problems, for example missing data in recommender systems [27], nonnegative matrix factorization [23] and automatic separation of outliers from a low rank model via sparse PCA [14].

In this paper we introduce a new data matrix class which permits a particular factorization of interest. The members of this class are composed of a sum of n rank one matrices of the form $\mathbf{a}_i\tilde{\mathbf{x}}_i$, where $\mathbf{a}_i \in \{0,1\}^m$ is a *binary* column vector with exactly $d << m$ non-zeros and $\tilde{\mathbf{x}}_i \in \mathbb{R}^N$ is a *real* row vector. Unlike PCA, and more analogous to dictionary learning, we typically consider $n > \max\{N, m\}$. An example of a matrix in this class is the sum of n rank one matrices whose supports are striped blocks of size $d \times k$ that may or may not overlap. Here 'striped' refers to the entries in any given column of an associated block having the same coefficient value. More

*Mathematical Institute, University of Oxford, UK. Email: michael.murray@maths.ox.ac.uk

†Alan Turing Institute, London, UK. This work was supported by EPSRC grant EP/N510129/1.

generally, data matrices in this class can be expressed as the sum of n independently permuted striped blocks and as a result we will refer to them as *Permuted Striped Block (PSB) matrices*. We provide a visualization of such matrices in Figure 1.

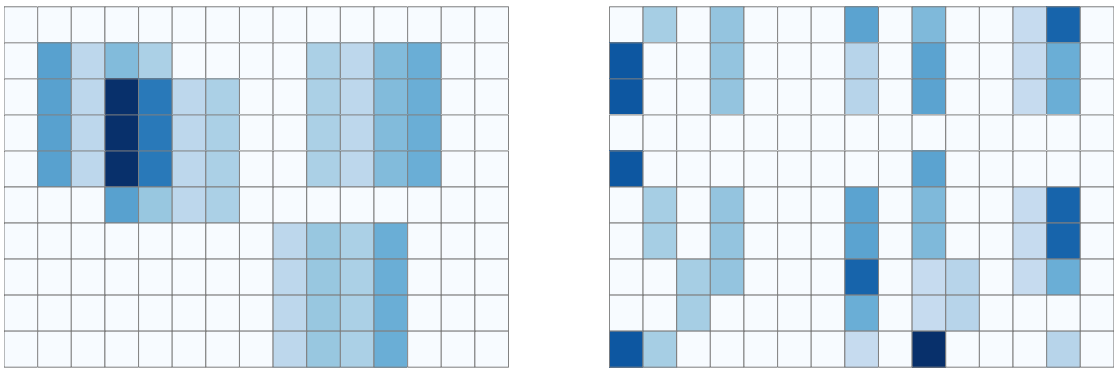

Figure 1: Visualization of two examples of PSB matrices. In both cases the data consists of a sum of four rank 1 matrices, each with support size 16. The left hand plot corresponds to the case where the support of each rank 1 matrix is arranged into a block. The right hand plot takes the same rank 1 matrices as in the left hand plot, but before summing them applies an independent random permutation to each. Note that white squares indicate a zero entry and that entries in the support (squares coloured a blue shade) that are in the same column have the same coefficient value (hence, aside from where there are overlaps, each column of a block is a single stripe of colour).

1.1 Data Model and Problem Definition

A PSB matrix $\mathbf{Y} \in \mathbb{R}^{m \times N}$ can be defined as $\mathbf{Y} := \mathbf{AX}$, where $\mathbf{A}$ is an $m \times n$ sparse binary matrix with exactly d nonzeros per column and $\mathbf{X}$ is an $n \times N$ column k sparse real matrix. In Definition 1 we present the *PSB data model*, which defines a particular distribution over the set of PSB matrices. The focus of this paper is to show that, under certain conditions and with high probability, data sampled from the PSB data model has a unique (up to permutation) factorization of the form discussed which can be computed efficiently and in dimension scalings that are near optimal. The details of PSB data model are given below in Definition 1. As a quick point to clarify our notation, bold upper case letters will be used to refer to deterministic matrices while nonbolded upper case letters will be used to refer to random matrices, i.e., a matrix drawn from a particular distribution (from the context it should be clear which distribution is being referred to).

Definition 1. ***Permuted Striped Block (PSB) data model:*** *given $d, k, m, n \in \mathbb{N}$, with $k < n$, $d < m$, define*

- *$\mathcal{A}_d^m \subset \{0,1\}^m$ as the set of binary vectors of dimension m with exactly d nonzeros per column and $\mathcal{A}_d^{m \times n}$ as the set of $m \times n$ matrices with columns $\mathbf{a}_i \in \mathcal{A}_d^m$ for all $i \in [n]$.*
- *$\mathcal{X}_k^n \subseteq \mathbb{R}^n$ as the set of real, dissociated (see Definition 2) and k sparse n dimensional vectors, and $\mathcal{X}_k^{n \times N}$ as the set of $n \times N$ matrices with columns $\mathbf{x}_i \in \mathcal{X}_k^n$ for all $i \in [N]$.*

We now define the following random matrices, note that the randomness is over their supports only.

- *$A := [A_1\ A_2 ...\ A_n]$ is a random binary matrix of size $m \times n$ where $A_i \in \mathcal{A}_d^m$ for $i \in [n]$. The distribution over the supports of these random vectors is defined as follows. The first*

$\lfloor m/d \rfloor$ columns A_i are formed by dividing a random permutation of $[m]$ into disjoint sets of size d and assigning each disjoint set as the support of an A_i. This process is repeated with independent permutations until n columns are formed. In this construction there are a fixed number d nonzeros per column and a maximum of $\lceil nd/m \rceil$ nonzeros per row.

- *$X := [X_1\ X_2 ...\ X_N]$ is a random real matrix of size $n \times N$ whose distribution is defined by concatenating N mutually independent and identically distributed random vectors X_i from $\mathcal{X}_k^n$; that is, the support of each X_i is chosen uniformly at random across all possible supports of size at most k.*

The PSB data model is the product of the aforementioned factors, generating the random matrix $Y := AX$.

The columns X_i comprising X in the PSB data model have nonzeros drawn so that partial sums of the nonzeros give unique nonzero values - a property which is referred to as dissociated.

Definition 2. *A vector $\mathbf{x} \in \mathbb{R}^N$ is said to be dissociated iff for any subsets $\mathcal{T}_1, \mathcal{T}_2 \subset supp(\mathbf{x})$ then $\sum_{j \in \mathcal{T}_1} x_j \neq \sum_{i \in \mathcal{T}_2} x_i$.*

The concept of dissociation comes from the field of additive combinatorics (Definition 4.32 in [33]). Although at first glance this condition appears restrictive it is fulfilled almost surely for isotropic vectors and more generally for any random vector whose nonzeros are drawn from a continuous distribution.

1.2 Motivation and Related Work

The PSB data model and the associated factorization task can be interpreted and motivated from a number of perspectives, three of which we now highlight.

- **A generative model for studying community detection and clustering:** consider the general problem of partitioning the nodes of a graph into clusters so that intra cluster connectivity is high relative to inter cluster connectivity. Given such a graph two basic questions of interest are 1) do these clusters or communities of nodes exist (community detection) and 2) can we actually recover them (clustering)? To study this question researchers study various generative models for random graphs, one of the most popular (particularly in machine learning and the network sciences) being the stochastic block model (for a recent survey see [1]). In this setting the observed data matrix is the adjacency matrix of a graph, generated by first sampling the cluster to which each node belongs and then sampling edges based on whether or not the relevant pair of nodes belong to the same cluster. The weighted stochastic block model, Aicher et al [3], generalizes this idea, allowing the weights of this adjacency matrix to be non-binary. The PSB data model can be viewed as an alternative data model for studying community detection and clustering. Indeed, the PSB data model can be interpreted as the adjacency matrix of a weighted bipartite graph. Recovering the factors A and X from Y is valuable as A encodes n clusters or groups in the set $[m]$ (where each group is a fixed size d) and each column of X encodes a soft clustering of an object in $[N]$ into k of n groups. The nonzero coefficients in a given column of X represent the strength of association between an object in $[N]$ and the clusters defined by A.

- **Dictionary learning with a sparse, binary dictionary:** for classes of commonly used data it is typically the case that there are known representations in which the object can be represented using few components, e.g. images are well represented using few coefficients in

wavelet or discrete cosine representations. That is, there is a known "dictionary" $\mathbf{A}$ for which data y_i from a specific class has the property that $\min_{x_i} \|Ax_i - y_i\|$ is small even while the number of nonzeros in $x_i \in \mathbb{R}^n$ is limited to $k << n$, meaning $\|x_i\|_0 \leq k$. Dictionary learning allows one to tailor a dictionary to a specific data set $\mathbf{Y} = [y_1 \ \cdots \ y_N]$ by solving $\min_{A,X} \|\mathbf{Y} - \mathbf{AX}\|$ subject to $\|x_i\|_0 \leq k$ for all $i \in [N]$. Alternatively, dictionary learning applied to a data matrix $\mathbf{Y}$ without prior knowledge of an initial dictionary $\mathbf{A}$ reveals properties of the data through the learned dictionary. Factorizing a matrix drawn from the PSB data model can be viewed as a specific instance of dictionary learning in which the dictionary is restricted to be in the class of overcomplete sparse, binary dictionaries. While this restricted class of dictionaries limits the type of data which the PSB data model can be used to describe, as we will show in Section 3 it does allow for a rigorous proof that with high probability it is possible to efficiently learn the dictionary and sparse code. This extends the growing literature on provable dictionary learning - see Table 1 for a summary of some recent results.

- **Learned combinatorial compressed sensing:** the field of compressed sensing [13, 17] studies how to efficiently reconstruct a sparse vector from only a few linear measurements. These linear measurements are generated by multiplying the sparse vector in question by a matrix, referred to as the sensing matrix. The sensing matrices most commonly studied are random and drawn from one of the following distributions: Gaussian, uniform projections over the unit sphere and partial Fourier. Combinatorial compressed sensing [10] instead studies the compressed sensing problem in the context of a random, sparse, binary sensing matrix. The problem of factorizing PSB matrices can therefore also be interpreted as recovering the sparse coding matrix $\mathbf{X}$ from N compressed measurements without access to the sensing matrix $\mathbf{A}$. Indeed, the factorization problem we study here can be motivated by the conjecture that, as combinatorial compressed sensing algorithms are so effective, the sparse code can be recovered even without knowledge of the sensing matrix.

The PSB data model, Definition 1, and the associated factorization task are most closely related to the literature on subspace clustering, multiple measurement combinatorial compressed sensing, and more generally the literature on provable dictionary learning. Each of these topics studies the model $\mathbf{Y} = \mathbf{AX}$, where $\mathbf{X}$ is assumed to be a sparse matrix with up to k nonzeros per column. These topics differ in terms of what further assumptions are imposed on the dictionary $\mathbf{A}$ and sparse coding matrix $\mathbf{X}$. The most general setup is that of dictionary learning, in which typically no additional structure is imposed on $\mathbf{A}$. In the literature on provable dictionary learning however, structure is often imposed on the factor matrices so as to facilitate the development of theoretical guarantees even if this comes at the expense of model expressiveness. Two popular structural assumptions are that the dictionary is complete (square and full rank) or that its columns are uncorrelated. Furthermore, as is the case for the PSB data model, in provable dictionary learning it is common to assume that the factor matrices are drawn from a particular distribution, e.g. Bernoulli-Subgaussian, so as to understand how difficult the factorization *typically* is to perform, rather than in the worst case. Table 1 lists some recent results and summarizes their modelling assumptions. Subspace clustering considers the further structural assumption that the columns of the sparse coding matrix can be partitioned into disjoint sets $\mathcal{S}_i$, where the number of nonzeros per row of each of the submatrices $\mathbf{X}_{\mathcal{S}_i}$ is less than the minimum dimension of $\mathbf{A}$ and $\mathbf{X}$. In this setting, the data $\mathbf{Y}$ is contained on a union of subspaces which are of lower dimension than the ambient dimension [19].

The PSB data model differs from the above literature most notably in the construction of A being sparse, binary and having exactly d nonzeros per column. The case of A being sparse and binary has been studied previously in [5], where, guided by the notion of reversible neural

Authors	A	X	k	N	Run time	Recovery
Spielman et al [29]	Complete	Bernoulli-Subgauss.	$\mathcal{O}(n^{1/2})$	$\Omega(n^2\log^2(n)$	poly(n)	ϵ-accuracy
Agarwal et al [2]	Incoherent	Bernoulli-Uniform	$\mathcal{O}(n^{1/4})$	$\mathcal{O}(n\log(n))$	poly(n)	ϵ-accuracy
Arora et al [6]	Incoherent	Bernoulli-Uniform	$\mathcal{O}(n^{1/2})$	$\mathcal{O}(n^2\log^2(n))$	poly(n)	ϵ-accuracy
Sun et al [31] [32]	Complete	Bernoulli-Gaussian	$\mathcal{O}(n)$	poly(n)	poly(n)	ϵ-accuracy
Barak et al [9]	σ-dictionary	(d,τ)-nice distribution	$\mathcal{O}(n)$	poly(n)	quasi-poly(n)	ϵ-accuracy
Arora et al [4]	Indiv. Recoverable	Bernoulli-{0,1}	$\mathcal{O}(n/\text{poly}(\log(n)))$	poly(n)	quasi-poly(n)	ϵ-accuracy
Arora et al [5]	Bernoulli-$\{-1,0,1\}$	Bernoulli-$\{0,1\}$	$^*\mathcal{O}(n/\log^2(n))$	$^*\mathcal{O}(\log^5(n))$	$^*\mathcal{O}(n^2N)$	Exact
M&T	Definition 1	Definition 1	$\mathcal{O}(n)$	$\mathcal{O}(\log^2(n)$	$\mathcal{O}(n^2N)$	Exact

Table 1: Summary of recent results from the provable dictionary learning literature. *The recovery results for [5] displayed here are presented under the assumption that $\rho = C/d$ for some large constant C (as stated in Theorem 1, pg. 5) and $d >> \log^2(n)$ (Theorem 1 of Appendix B, pg. 21), meaning $\rho << 1/\log^2(n)$. Furthermore, the run time for [5] is based only on step 1 of Algorithm 1 (pg. 8) which requires computing the row wise correlation of row vectors in $\mathbf{Y}$; as there are $\mathcal{O}(n^2)$ pairs of rows then if each row has $\mathcal{O}(N)$ nonzeros then this implies $\mathcal{O}(n^2N)$.

networks (that a neural network can be run in reverse, acting as a generative model for the observed data), the authors consider learning a deep network as a layerwise nonlinear dictionary learning problem. This work differs substantially from [5] in a number of respects: first the authors consider $Y = \sigma(AX)$, here σ is the elementwise unit step function which removes information about the nonzero coefficients of X and means that Y is binary. As a result of this, and because the authors are in the setup where each layer feeds into the next, then X is also assumed to be binary. The factors generating this thresholded model are challenging to recover due to the limited information available and as a consequence this model is also less descriptive for nonbinary data. In particular, in the three motivating examples previously covered: the communities would be unweighted, the dictionary learning data would be binary, and the learned combinatorial compressed sensing would only be for binary signals. Third and finally, in [5] the nonzeros of A are drawn independently of one another, the distribution defined in the PSB data model is a significant departure from this with both inter and intra column nonzero dependencies. We also emphasize that in terms of method the approach we take to factorize a matrix drawn from the PSB data model differs markedly from that adopted in [5]. In this prior work A is reconstructed (up to permutation) using a non-iterative approach, involving first the computation of all row wise correlations of Y from which pairs of nonzero entries are recovered. Using a clustering technique adapted from the graph square root problem, these pairs of nonzeros, which can be thought of as *partial supports* of the columns of $\mathbf{A}$ of length 2, are then combined to recover the columns in question. In contrast, our method iteratively generates large partial supports directly from the columns of $\mathbf{Y}$ which can readily be clustered while simultaneously revealing nonzero entries in $\mathbf{X}$. This process is then repeated on $\mathbf{Y} - \hat{\mathbf{A}}^{(t)}\hat{\mathbf{X}}^{(t)}$, which is the residual of $\mathbf{Y}$ after the approximations of $\mathbf{A}$ and $\mathbf{X}$ at the t^{th} iteration have been removed.

1.3 Main Contributions

The main contributions in this paper are an 1) the introduction of the Permuted Sparse Binary (PSB) data model, 2) an algorithm, Expander Based Factorization (EBF), for computing the factorization of PSB matrices and 3) recovery guarantees for EBF under the PSB data model, which we summarize in Theorem 1. Central to our method of proof is the observation that the random construction of A as in Definition 1 is with high probability the adjacency matrix of a left d regular (k,ϵ,d) bipartite expander graph. We define such graphs in Definition 3 and discuss their properties in Section 2.2. In what follows the set of left d regular (k,ϵ,d) bipartite expander graph

adjacency matrices of dimension $m \times n$ is denoted $\mathcal{E}_{k,\epsilon,d}^{m\times n}$.

Theorem 1. *Let Y be drawn from the PSB data model (Definition 1) under the assumption that $A \in \{\mathcal{E}_{k,\epsilon,d}^{m\times n}\} \cap \{\epsilon \leq 1/6\}$. If EBF, Algorithm 1, terminates at an iteration $t_f + 1 \in \mathbb{N}$ then the following statements are true.*

1. ***EBF only identifies correct entries:*** *for all $t \leq t_f$ there exists a permutation $P^{(t)}$ such that $supp(\hat{A}^{(t)}(P^{(t)})^T) \subseteq supp(A)$, $supp(P^{(t)}\hat{X}^{(t)}) \subseteq supp(X)$ and all nonzeros in $P^{(t)}\hat{X}^{(t)}$ are equal to the corresponding entry in X.*
2. ***Uniqueness of factorization:*** *if $\hat{A}^{(t_f)}\hat{X}^{(t_f)} = AX$ then this factorization is unique up to permutation.*
3. ***Probability that EBF is successful:*** *suppose $k = \alpha_1 n + 1$ and $m = \alpha_2 n$ where $\alpha_1, \alpha_2 \in (0,1)$ and $\alpha_1 \leq 1 - \frac{d}{m}$. If $N \geq \nu \frac{4d}{\tau(n)} \frac{n}{k} \ln^2(n)$ where $\tau(n) = \mathcal{O}(1)$ (see Lemma 8 for a full definition) and $\nu > 1$ is a constant, then the probability that EBF recovers A and X up to permutation is greater than $1 - \mathcal{O}\left(n^{-\sqrt{\nu}+1}\log^2(n)\right)$.*

As each column of $\mathbf{Y}$ is composed of k columns from $\mathbf{A}$, which itself has n columns, a minimum number of $N \geq n/k$ columns are necessary in order for $\mathbf{Y}$ to have at least one contribution from each column in $\mathbf{A}$. To be clear, this is a necessary condition on N to identify all columns in $\mathbf{A}$. Theorem 1 states that $N = \mathcal{O}(\frac{n}{k}\log^2(n)$ is sufficient to recover both A and X from Y with high probability under the PSB data model. We believe this result is likely to be a sharp lower bound as one $\log(n)$ factor arises from a coupon collector argument inherent to the way X is sampled, and the second $\log(n)$ factor is needed to achieve the stated rate of convergence. As discussed in more detail in Section 4.2, assuming the same asymptotic relationships as in statement 3 of Theorem 1, EBF has a per iteration cost (in terms of the input data dimensions) of $\mathcal{O}(m^2N)$. The rest of this paper is structured as follows: in Section 2 we provide the algorithmic details of EBF, in Section 3 we prove Theorem 1 and in Section 4 we provide a first step improvement of EBF along with numerical simulations, demonstrating the efficacy of the algorithms in practice. The proofs of the key supporting lemmas are mostly deferred to the Appendix.

2 Algorithmic Ideas

In this section we present a simple algorithm which leverages the properties of expander graphs to try and compute the factorization of $\mathbf{Y} := \mathbf{AX}$ where $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ and $\mathbf{X} \in \mathcal{X}_k^{n\times N}$. We call this algorithm Expander Based Factorization (EBF). To describe and define this algorithm we adopt the following notational conventions. Note that all variables referenced to here will be defined properly in subsequent sections.

- For any $\alpha \in \mathbb{N}$ we define $[\alpha] := \{x \in \mathbb{N} : x \leq \alpha\}$ as the set of natural numbers less than or equal to α, for example, $[3] = \{1, 2, 3\}$.
- If $\mathbf{B}$ is an $m \times n$ matrix, $\mathcal{R} \subseteq [m]$ a subset of the row indices and $\mathcal{C} \subseteq [n]$ a subset of the column indices, then $\mathbf{B}(\mathcal{R}, \mathcal{C})$ is the submatrix of $\mathbf{B}$ formed with only the rows in $\mathcal{R}$ and the columns in $\mathcal{C}$.
- $t \in \mathbb{N}$ will be used as an the iteration index for the algorithms presented.
- $\hat{\mathbf{A}}^{(t)} \in \{0,1\}^{m\times n}$ is the estimate of $\mathbf{A}$, up to column permutation, at iteration t.

- $\hat{\mathbf{X}}^{(t)} \in \mathbb{R}^{n \times N}$ is the estimate of $\mathbf{X}$, up to row permutation, at iteration t.
- $\mathbf{R}^{(t)} := \mathbf{Y} - \hat{\mathbf{A}}^{(t)}\hat{\mathbf{X}}^{(t)} \in \mathbb{R}^{m \times N}$ is the residual of the data matrix $\mathbf{Y}$ at iteration t.
- $c(t)$ is the number of partial supports extracted from $\mathbf{R}^{(t-1)}$ at iteration t.
- $\mathbf{W}^{(t)} \in \{0,1\}^{m \times c(t)}$ is the matrix of partial supports $[\mathbf{w}_1^{(t)}, \mathbf{w}_2^{(t)} ... \mathbf{w}_{c(t)}^{(t)}]$, extracted from $\mathbf{R}^{(t-1)}$.
- $\mathcal{Q}^{(t)}$ is the set of singleton values (see Definition 4) extracted from $\mathbf{R}^{(t-1)}$ which appear in a column of $\mathbf{Y}$ more than $(1-2\epsilon)d$ times.
- $\mathcal{C}_h^{(t)} := \{p \in [c(t)] : \mathbf{w}_p^{(t)} \subset supp(\mathbf{a}_l)\}$ is the set of partial supports (see Definition 5) of $\mathbf{a}_l$ extracted from $\mathbf{R}^{(t-1)}$ used to update the hth column of the estimate $\hat{\mathbf{a}}_h^{(t)}$.
- A column $\mathbf{a}_l$ with $l \in [n]$ of $\mathbf{A}$ said to have been *recovered* iff there is some iteration for which for all subsequent iterates there exists a column $\hat{\mathbf{a}}_h^{(t)}$ where $h \in [n]$ such that $\mathbf{a}_l = \hat{\mathbf{a}}_h^{(t)}$.
- A column $\hat{\mathbf{a}}_h^{(t)}$ with $h \in [n]$ of $\hat{\mathbf{A}}^{(t)}$ is said to be *complete* at iteration t iff $|supp(\hat{\mathbf{a}}_h^{(t)})| = d$
- σ is the elementwise unit step function; $\sigma(z) = 1$ for $z > 0$ and $\sigma(z) = 0$ for $z \leq 0$.
- $\mathcal{H}^{(t)}$ is the set of column indices of $\hat{\mathbf{A}}^{(t)}$ which have d non-zeros, i.e., $\mathcal{H}^{(t)} := \{i \in [n] : |supp(\hat{\mathbf{a}}_i^{(t)})| = d\}$. Furthermore $\bar{\mathcal{H}}^{(t)} := [n] \backslash \mathcal{H}^{(t)}$.
- For consistency and clarity we will typically adopt the following conventions: $i \in [N]$ will be used as an index for the columns of $\mathbf{Y}$, $\mathbf{R}^{(t)}$, $\mathbf{X}$ and $\hat{\mathbf{X}}^{(t)}$, $j \in [m]$ will be used as an index for the rows of $\mathbf{Y}$, $\mathbf{R}^{(t)}$, $\mathbf{A}$ and $\hat{\mathbf{A}}^{(t)}$, $l \in [n]$ will be used as an index for the columns of $\mathbf{A}$ as well as the rows of $\mathbf{X}$, $h \in [n]$ will be used as an index for the columns of $\hat{\mathbf{A}}^{(t)}$ as well as the rows of $\hat{\mathbf{X}}^{(t)}$, and finally $p \in [c(t)]$ will be used as an index for the columns of $\mathbf{W}^{(t)}$.

2.1 The Expander Based Factorization Algorithm (EBF)

The key idea behind EBF is that if the binary columns of $\mathbf{A}$ are sufficiently sparse and have nearly disjoint supports, then certain entries in $\mathbf{X}$, which we will term singleton values (see Definition 4), can be directly identified from entries of $\mathbf{Y} := \mathbf{AX}$. Furthermore, it will become apparent that singleton values not only provide entries of $\mathbf{X}$ but also entries in $\mathbf{A}$ via the construction of partial supports. By iteratively combining the information gained about $\mathbf{A}$ and $\mathbf{X}$ from the singleton values and partial supports and then removing it from the residual, EBF can iteratively recover parts of $\mathbf{A}$ and $\mathbf{X}$ until either no further progress can be made or the factors have been fully recovered (up to permutation).

The remainder of Section 2 is structured as follows: in Section 2.2 we review the definition and properties of (k, ϵ, d) expander graphs so as to formalize a notion of the columns of $\mathbf{A}$ being sufficiently disjoint in support, in Section 2.3 we define and prove certain key properties of singleton values and partial supports, in Section 2.4 we prove that EBF never introduces erroneous nonzeros into the estimates of either $\mathbf{A}$ or $\mathbf{X}$ and review each step of Algorithm 1 in more detail. Finally, in Section **??**, for completeness we provide a sketch of a proof that in a certain parameter regime the matrix A in the PSB data model is with high probability the adjacency matrix of a (k, ϵ, d) expander graph.

Algorithm 1 EBF($\mathbf{Y}$, d, k, m, n, N, ϵ)

Init: $\mathbf{R}^{(0)} \leftarrow \mathbf{Y}$, $\hat{\mathbf{A}}^{(0)} \leftarrow \text{zeros}(m, n)$, $\hat{\mathbf{X}}^{(0)} \leftarrow \text{zeros}(n, N)$, $t \leftarrow 0$.
while $\|\hat{\mathbf{A}}^{(t)}\|_0 > \|\hat{\mathbf{A}}^{(t-1)}\|_0$ or $\|\hat{\mathbf{X}}^{(t)}\|_0 > \|\hat{\mathbf{X}}^{(t-1)}\|_0$ or $t = 0$ **do**
  Set $t \leftarrow t + 1$.
  Extract set of singleton values $\mathcal{Q}^{(t)}$ and associated partial supports $\mathbf{W}^{(t)}$ from $\mathbf{R}^{(t-1)}$.
  Update $\hat{\mathbf{A}}^{(t)}$ by inspecting $\mathbf{R}^{(t)}$ for singleton values identified in prior iterations.
  Cluster partial supports $\mathbf{w}_p^{(t)}$ $(p \in [c(t)])$ into sets $\mathcal{C}_h^{(t)}$ $(h \in [n])$.
  Update $\hat{\mathbf{X}}^{(t)}$: for all $r_{j,i} \in \mathcal{Q}^{(t)}$ if $r_{j,i}$ has partial support $\mathbf{w}_p^{(t)} \in \mathcal{C}_h^{(t)}$ set $\hat{x}_{h,i} \leftarrow r_{j,i}$.
  Update $\hat{\mathbf{A}}^{(t)}$: for all $l \in [n]$ set $\hat{\mathbf{a}}_h^{(t)} \leftarrow \sigma\left(\hat{\mathbf{a}}_h^{(t-1)} + \sum_{p \in \mathcal{C}_h^{(t)}} \mathbf{w}_p^{(t)}\right)$.
  Update residual: $\mathbf{R}^{(t)} \leftarrow \mathbf{Y} - \hat{\mathbf{A}}^{(t)}\hat{\mathbf{X}}^{(t)}$.
end while
Return $(\hat{\mathbf{A}}^{(t)}, \hat{\mathbf{X}}^{(t)})$

2.2 Background on Expander Graphs

The PSB data model in Definition 1 is chosen so as to leverage certain properties of expander graphs. Such graphs are both highly connected yet sparse and are an interesting object both from a practical perspective, playing a key role in applications, e.g. error correcting codes, as well in areas of pure maths (for a survey see Lubotzky [24]). We consider only left d-regular (k, ϵ, d) bipartite expander graphs and for ease we will typically refer to these graphs as (k, ϵ, d) expanders.

Definition 3 (**(k, ϵ, d) Expander Graph [28]**). *Consider a left d-regular bipartite graph $G = ([n], [m], E)$, for any $\mathcal{S} \subset [n]$ let $\mathcal{N}(\mathcal{S}) := \{j \in [m] : \exists l \in \mathcal{S} \text{ s.t. } (l, j) \in E\}$ be the set of nodes in $[m]$ connected to a node in $\mathcal{S}$. G is a (k, ϵ, d) expander iff*

$$|\mathcal{N}(\mathcal{S})| > (1 - \epsilon)d|\mathcal{S}| \quad \forall \quad \mathcal{S} \in [n]^{(\leq k)}. \tag{2.1}$$

Here $[n]^{(\leq k)}$ denotes the set of subsets of $[n]$ with cardinality at most k. A key property of such graphs is the Unique Neighbour Property.

Theorem 2 (**The Unique Neighbour Property [15, Lemma 1.1]**). *Suppose that G is an unbalanced, left d-regular bipartite graph $G = ([n], [m], E)$. Let $\mathcal{S}$ be any subset of nodes $\mathcal{S} \in [n]^{(\leq k)}$ and define*

$$\mathcal{N}_1(\mathcal{S}) = \{j \in \mathcal{N}(\mathcal{S}) : |\mathcal{N}(j) \cap \mathcal{S}| = 1\},$$

here $\mathcal{N}(j)$ is the set of nodes of $[n]$ connected to node $j \in [m]$. If G is a (k, ϵ, d) expander graph then

$$|\mathcal{N}_1(\mathcal{S})| > (1 - 2\epsilon)d|\mathcal{S}| \quad \forall \quad \mathcal{S} \in [n]^{(\leq k)}. \tag{2.2}$$

A proof of Theorem 2 in the notation used here is available in [25, Appendix A]. For our purposes it will prove more relevant to describe expander graphs using matrices. The adjacency matrix of a (k, ϵ, d) expander graph $G := ([m], [n], E)$ is an $m \times n$ binary matrix $\mathbf{A}$ where $a_{l,j} = 1$ if there is an edge between node $j \in [m]$ and $l \in [n]$ and is 0 otherwise[1]. Applying Definition 3 and Theorem 2 we make the following observations.

[1]We note that adjacency matrices of graphs are often defined to describe the edges present between *all* nodes in the graph, however we emphasize that in the definition adopted here the edges between nodes in the same group are not set to zero but rather are omitted entirely.

Corollary 1 (**Adjacency matrix of a (k, ϵ, d) Expander Graph**). *If* $\mathbf{A}$ *is the adjacency matrix of a (k,* ϵ*, d) Expander Graph* $G := ([m], [n], E)$ *then any submatrix* $\mathbf{A}([m], \mathcal{S})$ *of* $\mathbf{A}$*, where* $\mathcal{S} \in [n]^{(\leq k)}$*, satisfies the following properties.*

1. *By definition 3 there are more than* $(1-\epsilon)d|\mathcal{S}|$ *rows in* $\mathbf{A}([m], \mathcal{S})$ *that have* ***at least one*** *non-zero.*
2. *By Theorem 2 there are more than* $(1-2\epsilon)d|\mathcal{S}|$ *rows in* $\mathbf{A}([m], \mathcal{S})$ *that have* ***only one*** *non-zero.*

We will use $\mathcal{E}_{k,\epsilon,d}^{m\times n} \subset \{0,1\}^{m\times n}$ *to denote the set of (k,* ϵ*, d) expander graph adjacency matrices of dimension* $m \times n$*.*

2.3 Singleton values and partial supports

For now we will consider a generic vector $\mathbf{b} \in \mathbb{R}^m$ such that $\mathbf{b} = \mathbf{A}\mathbf{z}$ where $\mathbf{A} \in \{0,1\}_{k,\epsilon,d}^{m\times n}$ and $\mathbf{z} \in \mathbb{R}^n$. We will later apply the theory we develop here to the columns of the residual at *any* iteration t. Letting $\tilde{\mathbf{a}}_j$ denote the jth row of $\mathbf{A}$, then any given entry of $\mathbf{b}$ is a sum of some subset of the entries of $\mathbf{z}$. We now introduce two concepts which underpin much of what follows.

Definition 4 (**Singleton Value**). *Consider a vector* $\mathbf{b} = \mathbf{A}\mathbf{z}$ *where* $\mathbf{A} \in \{0,1\}^{m\times n}$ *and* $\mathbf{z} \in \mathbb{R}^n$*, a singleton value of* $\mathbf{b}$ *is an entry* b_j *such that* $|supp(\tilde{\mathbf{a}}_j) \cap supp(\mathbf{z})| = 1$*, hence* $b_j = x_l$ *for some* $l \in supp(\mathbf{z})$*.*

Definition 5 (**Partial Support**). *A partial support* $\mathbf{w}_p \in \{0,1\}^m$ *of a column* $\mathbf{a}_l \in \{0,1\}^m$ *of* $\mathbf{A} \in \{0,1\}^{m\times n}$ *is a binary vector satisfying* $supp(\mathbf{w}_p) \subseteq supp(\mathbf{a}_l)$*.*

Singleton values are of interest in the context of factorizing a matrix drawn from the PSB data model as once identified their contribution can be removed from the residual. Furthermore, using a singleton value one can construct a partial support by creating a binary vector with nonzeros where the singleton value appears. Therefore identifying singleton values allows for the recovery of nonzeros in both $\mathbf{A}$ and $\mathbf{X}$. To leverage this fact however we need a criteria or certificate for identifying them. Under the assumption that $\mathbf{A}$ is a (k, ϵ, d) expander and that $\mathbf{z}$ is dissociated (see Definition 2), then it is possible to derive a certificate based on a lower bound on the mode (or frequency) with which a value appears in $\mathbf{b}$.

Lemma 1 (**Identification of singleton values**). *Consider a vector* $\mathbf{b} = \mathbf{A}\mathbf{z}$ *where* $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ *and* $\mathbf{z} \in \mathcal{X}_k^n$*, then the frequency of any singleton value is at least* $2\epsilon d$*. To be precise,*

$$|\{j \in supp(\mathbf{b}) : \exists\, l \in supp(\mathbf{x}_i)\ s.t.\ b_j = z_l\}| \geq 2\epsilon d.$$

A proof of Lemma 1 is provided in Appendix A.1. This Lemma provides a sufficient condition for testing whether a value is a singleton or not. However, it does not provide any guarantees that in $\mathbf{b}$ there will be any singleton values. In Lemma 2, adapted from [25, Theorem 4.6], we show, under certain assumptions, that there always exist a positive number of singleton values which appear more than $(1-2\epsilon)d$ times in $\mathbf{b}$.

Lemma 2 (**Existence of singleton values, adapted from [25, Theorem 4.6]**). *Consider a vector* $\mathbf{b} = \mathbf{A}\mathbf{z}$ *where* $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ *and* $\mathbf{z} \in \mathcal{X}_k^n$*. For* $l \in supp(\mathbf{z})$*, let* Ω_l *be the set of row indices* $j \in [m]$ *of* $\mathbf{b}$ *for which* $b_j = z_l$*, i.e.,* $\Omega_l := \{j \in [m] : b_j = z_l\}$*. Defining* $\mathcal{T} := \{l \in supp(\mathbf{z}) : |\Omega_l| > (1-2\epsilon)d\}$ *as the set of singleton values which appear more than* $(1-2\epsilon)d$ *times in* $\mathbf{b}$*, then*

$$|\mathcal{T}| \geq \frac{|supp(\mathbf{z})|}{(1+2\epsilon)d}.$$

Therefore, so long as $\mathbf{b}$ *is nonzero it is always possible to extract at least 1 partial support of size greater than* $(1-2\epsilon)d$.

A proof of Lemma 2 is given in Appendix A.2.

Step 5 of Algorithm 1 relies on us being able to accurately and efficiently sort the partial supports extracted by their column of $\mathbf{A}$. Corollary 2 states that if $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ with $\epsilon \leq 1/6$ and if the partial supports are sufficiently large, then clustering can be achieved *without error* simply by computing pairwise inner products.

Corollary 2 (**Clustering partial supports**). *Any pair of partial supports* $\mathbf{w}_p$, $\mathbf{w}_q$ *satisfying* $|supp(\mathbf{w}_p)| > (1-2\epsilon)d$ *and* $|supp(\mathbf{w}_q)| > (1-2\epsilon)d$ *originate from the same column of* $\mathbf{A}$ *iff* $\mathbf{w}_p^T\mathbf{w}_q \geq (1-4\epsilon)d$.

A proof of Corollary 2, which follows directly from results derived in the proof of Lemma 2, is provided in Appendix A.3.

2.4 Algorithm 1 accuracy guarantees and summary

Based on the results of Section 2.3 it is now possible to provide the following accuracy guarantees for EBF.

Lemma 3 (**EBF only identifies correct entries**). *Let* $\mathbf{Y} := \mathbf{AX}$, *where* $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ *with* $\epsilon \leq 1/6$ *and* $\mathbf{X} \in \mathcal{X}_k^{n\times N}$. *Suppose that EBF terminates at iteration* $t_f+1 \in \mathbb{N}$, *then the following statements are true.*

1. *For all* $t \leq t_f$ *there exists a permutation* $\mathbf{P}^{(t)}$ *such that* $supp(\hat{\mathbf{A}}^{(t)}(\mathbf{P}^{(t)})^T) \subseteq supp(\mathbf{A})$, $supp(\mathbf{P}^{(t)}\hat{\mathbf{X}}^{(t)}) \subseteq supp(\mathbf{X})$ *and all nonzeros in* $\mathbf{P}^{(t)}\hat{\mathbf{X}}^{(t)}$ *are equal to the corresponding entry in* $\mathbf{X}$.
2. *EBF fails only if* $\mathbf{R}^{(t_f)} \neq \boldsymbol{0}_{m\times N}$.

For a proof of Lemma 3 see Appendix A.4. Lemma 3 states that EBF never makes a mistake in terms of introducing erroneous nonzeros to either $\hat{\mathbf{A}}^{(t)}$ or $\hat{\mathbf{X}}^{(t)}$, and hence fails to factorize $\mathbf{Y}$ only by failing to recover certain nonzeros in $\mathbf{A}$ or $\mathbf{X}$. We are now in a position to be able to summarize and justify how and why each step of Algorithm 1 works.

- **Steps 1**, **2** and **3** are self explanatory, **1** being the initialization of certain key variables, **2** defining a while loop which terminates only when neither of the estimates $\hat{\mathbf{A}}^{(t)}$ or $\hat{\mathbf{X}}^{(t)}$ are updated from the previous iterate and **3** being the update of the iteration index.
- **Steps 4** and **5** are in fact conducted simultaneously with each column of the residual being processed in parallel. The mode of each nonzero in a column is calculated and the value checked against singleton values identified in previous iterates. If a nonzero value appears more than $(1-2\epsilon)d$ times then its associated partial support is constructed. Note that although it may be possible to identify singleton values which appear at least $2\epsilon d$ times, unless they appear more than $(1-2\epsilon)d$ then it is not possible to guarantee that their associated partial support can be clustered correctly. Therefore, such a singleton value cannot be placed in a row of $\hat{\mathbf{X}}^{(t)}$ consistent with the other singleton values extracted, and its associated partial support cannot be used to augment $\hat{\mathbf{A}}^{(t)}$. If any nonzero value of a column of the residual matches a previously identified singleton value from the same column then it can also be used to augment $\hat{\mathbf{A}}^{(t)}$. Indeed, if $r_{j,i}^{(t)} = \hat{x}_{l,i}^{(t)}$ then by Lemma 3, and by the fact that $\mathbf{x}_i$ is dissociated it must be that $\hat{a}_{z,l}^{(t)} = 1$.

- **Steps 6**, **7** and **8** can be conducted on the basis of Corollary 2. A naive method would be to simply take each of the partial supports extracted from $\mathbf{R}^{(t)}$ in turn and compute its inner product with the nonzero columns of $\hat{\mathbf{A}}^{(t)}$. By Lemma 3 and the fact that each partial support has cardinality larger than $(1-2\epsilon)d$, then either a partial support will match with an existing column of $\hat{\mathbf{A}}^{(t)}$ or otherwise it can be used as the starting estimate for a new column of $\mathbf{A}$. Once a partial support, for which we know the column in $\mathbf{R}^{(t-1)}$ from which it was extracted, is assigned to a column of $\hat{\mathbf{A}}^{(t)}$, then the corresponding location in $\hat{\mathbf{X}}^{(t)}$ can be updated using its associated singleton value.

- **Step 9** - once all singleton values and partial supports extracted from $\mathbf{R}^{(t-1)}$ have been used to augment $\hat{\mathbf{A}}^{(t)}$ and $\hat{\mathbf{X}}^{(t)}$ then the residual can be updated and the algorithm loops.

2.5 The construction of A in the PSB data model: motivation and connections to (k, ϵ, d) expander graphs

The dictionary model A in Definition 1 is chosen so as to satisfy two properties that facilitate the identification of its columns from the data Y. First, the number of nonzeros per row of A is bounded so that each entry can be readily guaranteed to be recovered, and second, the supports of the d nonzeros per column are drawn so that the probability of substantial overlap is bounded. The remainder of this section is a further discussion of how these properties motivate the construction of A and give insight into other choices of A for which the methods presented in this paper could be extended. Some explicit examples of such extensions are given in Section 5.

In order that an entire column of A can be identified, the union of the associated partial supports extracted by EBF needs to be equal to the entire support of the column. That sufficiently many partial supports of a column will be observed at some iteration of EBF is achieved by ensuring that there are enough nonzeros per row of X, this is analyzed in Lemma 9 in Section 3. The inability to identify all entries in a column of A only occurs if the random combinations of k columns of A have a consistent overlap, resulting in an entry from the column in question not being present in any of the partial supports. The probability of such an overlap is analyzed in the proof of Lemma 8, where we show that bounding the maximum number of nonzeros per row of A is sufficient to ensure that such a consistent overlap has a probability that goes to zero exponentially fast in the number of partial supports available. This motivates the random construction of A in Definition 1, where the maximum number of nonzeros per row of A is given by $\lceil nd/m \rceil$.

The algorithmic approach of EBF, i.e., the extraction and clustering of singleton values and partial supports, follows from A being a (k, ϵ, d) expander. In [7] it was shown that if A were constructed such that the column supports have fixed cardinality d and were drawn *independently* and uniformly at random, then A would be a (k, ϵ, d) expander with high probability. More precisely, [7, 8] considered d and ϵ to be fixed with k, m, and n growing proportionally and showed that the probability that A would not be the adjacency matrix of a (k, ϵ, d) expander graph is bounded by a given function which decays exponentially in n. Without modification, the aforementioned proof and bounds in [7] prove that the construction of A in the PSB data model, Definition 1, is with high probability a (k, ϵ, d) expander graph. For brevity we do not review this proof in detail and instead only sketch why the proof is equally valid here. In [7] the large deviation probability bounds are derived by considering the submatrix $A_{\mathcal{S}}$ consisting of $|\mathcal{S}| = k$ columns from A. The support $\mathcal{S}$ is repeatedly divided into two disjoint supports, say $\mathcal{S}_1$ and $\mathcal{S}_2$, each of size approximately $k/2$ and then the number of neighbours $\mathcal{N}(\mathcal{S})$ is bounded in terms of $\mathcal{N}(\mathcal{S}_1)$ and $\mathcal{N}(\mathcal{S}_2)$. This process is repeated until the finest level partitions contain a single column, in which case $\mathcal{N}(j) = d$ for all $j \in [n]$. Bounds on the probability of the number of neighbours $\mathcal{N}(\mathcal{S}_i)$ is computed using the

mutual independence of the columns. These bounds also hold for the construction in Definition 1 since the columns either have independently drawn supports or if they are dependent then they are disjoint by construction.

3 Theoretical Guarantees

Analyzing and proving theoretical guarantess for EBF directly is challenging, so instead our approch will be to study a simpler surrogate algorithm which we can use to lower bound the performance of EBF. To this end, we introduce the Naive Expander Based Factorization Algorithm (NEBF) which, despite being suboptimal from a practical perspective, is still sufficiently effective for us to prove Theorem 1.

3.1 The Naive Expander Based Factorization Algorithm (NEBF)

NEBF, given in Algorithm 2, is based on the same principles as EBF developed in Section 2 - in short the extraction and clustering of partial supports and singleton values. However, there are a number of significant restrictions included in order to allow us to determine when and with what probability it will succeed. At each iteration of the while loop, lines 2-14, NEBF attempts to recover a column of $\mathbf{A}$, if it fails to do so then the algorithm terminates. Therefore, by construction, at the start of the hth iteration the first $h-1$ of $\hat{\mathbf{A}}^{(t)}$ columns are complete. On line 3 the subroutine PeelRes (for more details see Algorithm 4 in Appendix B.1) iteratively removes the contributions of complete columns of $\hat{\mathbf{A}}^{(t)}$ from the residual until none of the partial supports extracted match with a complete column. This means that the clusters of partial supports returned by PeelRes all correspond to as of yet not complete columns of $\hat{\mathbf{A}}^{(t)}$. To define PeelRes we need to introduce the notion of the *visibile set* $\mathcal{V}^{(t)} := \{h \in [n] : |\mathcal{C}_h^{(t)}| > 0\}$, which is the set of column indices of $\hat{\mathbf{A}}^{(t-1)}$ for which there exists at least one partial support extracted from $\mathbf{R}^{(t-1)}$. In step 4 NEBF identifies the cluster of partial supports with the largest cardinality and then in steps 6 and 7 attempts to use this cluster to compute and complete the hth column of $\hat{\mathbf{A}}^{(t)}$. If this step is successful then in lines 8-9 the residual and iteration index are updated and the algorithm repeates. If the construction of the hth column is unsuccessful, i.e., it's missing at least one entry, then the algorithm terminates.

We now present a few properties of NEBF: first and analagous to Lemma 3 NEBF does not make mistakes.

Lemma 4 (NEBF only identifies correct entries). *Let $\mathbf{Y} := \mathbf{AX}$, where $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ with $\epsilon \leq 1/6$ and $\mathbf{X} \in \mathcal{X}_k^{n\times N}$. Suppose NEBF terminates at an iteration $t_f + 1 \in \mathbb{N}$, then the following statements are true.*

1. *For all $t \leq t_f$ there exists a permutation matrix $\mathbf{P}^{(t)}$ such that*
   (a) *$\mathbf{R}^{(t)} = \mathbf{A}(\mathbf{X} - \mathbf{P}^{(t)}\hat{\mathbf{X}}^{(t)})$ where $(\mathbf{X} - \mathbf{P}^{(t)}\hat{\mathbf{X}}^{(t)}) \in \mathcal{X}_k^{n\times m}$.*
   (b) *From any nonzero $\mathbf{r}_i^{(t)}$, at least $|supp(\mathbf{x}_i - \mathbf{P}^{(t)}\hat{\mathbf{x}}_i^{(t)})|/(1+2\epsilon)d > 1$ singleton values and associated partial supports, each with more than $(1-2\epsilon)d$ nonzeros, can be extracted and clustered without error.*
   (c) *$supp(\hat{\mathbf{A}}^{(t)}(\mathbf{P}^{(t)})^T) \subseteq supp(\mathbf{A})$, $supp(\mathbf{P}^{(t)}\hat{\mathbf{X}}^{(t)}) \subseteq supp(\mathbf{X})$ and any nonzero in $\mathbf{P}^{(t)}\hat{\mathbf{X}}^{(t)}$ is equal to its corresponding entry in $\mathbf{X}$.*
2. *NEBF fails only if $\mathbf{R}^{(t_f)} \neq \mathbf{0}_{m\times N}$.*

Algorithm 2 NEBF($\mathbf{Y}$, d, k, m, n, N, ϵ)

Init: $\mathbf{R}^{(0)} \leftarrow \mathbf{Y}$, $\hat{\mathbf{A}}^{(0)} \leftarrow \text{zeros}(m, n)$, $\hat{\mathbf{X}}^{(0)} \leftarrow \text{zeros}(n, N)$, $t \leftarrow 1$, $h \leftarrow 1$, ERROR $\leftarrow$ FALSE
while $h \leq n$ **and** ERROR = FALSE **do**
  Iteratively peel residual: $(t, \hat{\mathbf{X}}^{(t)}, \{\mathcal{C}_{h'}^{(t)}\}_{h'=1}^{n}, \mathbf{W}^{(t)}) \leftarrow \text{PeelRes}(t, h-1, \mathbf{R}^{(t-1)}, \hat{\mathbf{X}}^{(t-1)}, \hat{\mathbf{A}}^{(t-1)})$
  Set $\mathcal{C}_h^{(t)}$ to be the cluster of partial supports in $\{\mathcal{C}_{h'}^{(t)}\}_{h'=1}^{n}$ with the largest cardinality.
  if $|\mathcal{C}_h^{(t)}| > 0$ **then**
    if $|supp(\sigma\left(\sum_{p \in \mathcal{C}_h^{(t)}} \mathbf{w}_p^{(t)}\right))| = d$ **then**
      Compute the hth column and update $\hat{\mathbf{A}}^{(t)}$: $\hat{\mathbf{a}}_h^{(t)} \leftarrow \sigma\left(\sum_{p \in \mathcal{C}_h^{(t)}} \mathbf{w}_p^{(t)}\right)$.
      Compute the residual: $\mathbf{R}^{(t)} \leftarrow \mathbf{Y} - \hat{\mathbf{A}}^{(t)}([m], [h])\hat{\mathbf{X}}^{(t)}([h], [N])$.
      Update column index: $h \leftarrow h + 1$
    else
      ERROR $\leftarrow$ TRUE
    end if
  end if
end while
Return $(\hat{\mathbf{A}}, \hat{\mathbf{X}}^{(t)})$

A proof of Lemma 4 can be found in appendix A.5. One of the key differences between NEBF and EBF is that at every iteration the residual is of the form $\mathbf{R}^{(t)} = \mathbf{A}\mathbf{Z}^{(t)}$ where $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m \times n}$ and $\mathbf{Z}^{(t)} \in \mathcal{X}_k^{n \times N}$: as a result Lemma 2 applies at every iteration. This is covered in more detail in Lemma 4 below.

3.2 Key supporting Lemmas

Before we can proceed to prove Theorem 1 it is necessary for us to introduce a number of key supporting lemmas. First, and taking inspiration from algorithms in the combinatorial compressed sensing literature (in particular Expander Recovery [21]), it holds for both EBF and NEBF that recovery of $\mathbf{A}$ is sufficient for the recovery of $\mathbf{X}$. This result will prove useful in what follows as it allows us to study the recovery of only $\mathbf{A}$ rather than both $\mathbf{A}$ and $\mathbf{X}$.

Lemma 5 (**Recovery of A is sufficient to guarantee success**). *Let $\mathbf{Y} := \mathbf{A}\mathbf{X}$, where $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m \times n}$ with $\epsilon \leq 1/6$ and $\mathbf{X} \in \mathcal{X}_k^{n \times N}$. If either EBF or NEBF recover $\mathbf{A}$ up to permutation at some iteration t, then both are guaranteed to recover $\mathbf{X}$ by iteration $t + k$. Therefore, for both algorithms recovery of $\mathbf{A}$ is both a necessary and sufficient condition for success and hence under the PSB data model the probability that Y is successfully factorized is equal to the probability that A is recovered.*

For a proof of Lemma 5 see Appendix A.6. Second, using Lemmas 3 and 4, we have the following uniqueness result concerning the factorization calculated by EBF and NEBF.

Lemma 6 (**Uniqueness of factorization**). *Let $\mathbf{Y} := \mathbf{A}\mathbf{X}$, where $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m \times n}$ with $\epsilon \leq 1/6$ and $\mathbf{X} \in \mathcal{X}_k^{n \times N}$. If either EBF or NEBF terminates at an iteration $t_f + 1$ such that $\mathbf{R}^{(t_f)} = \mathbf{0}_{m \times n}$ then the factorization is not only successful, but is also unique, up to permutation, in terms of factorizations of this form.*

For a proof of Lemma 6 see Appendix A.7. The next Lemma states that if NEBF succeeds in computing the factors of $\mathbf{Y} := \mathbf{A}\mathbf{X}$ then so will EBF. The key implication of this Lemma is that it is sufficient to study NEBF in order to lower bound the probability that EBF is successful.

Lemma 7 (**NEBF can be used to lower bound the performance of EBF**). *Assuming that* $\mathbf{A} \in \mathcal{E}_{k,\epsilon,d}^{m\times n}$ *with* $\epsilon \leq 1/6$ *and* $\mathbf{X} \in \mathcal{X}_k^{n\times N}$*, then if NEBF successfully computes the factorization of* $\mathbf{Y} := \mathbf{AX}$ *up to some permutation of the columns and rows of* $\mathbf{A}$ *and* $\mathbf{X}$ *respectively, then so will EBF. Furthermore, if the probability that NEBF successfully factorizes* Y*, drawn from the PSB data model, is greater than* $1-\delta$ *then the probability EBF successfuly factorizes* Y *is also greater than* $1-\delta$*.*

A proof of Lemma 7 can be found in Appendix A.8. Lemma 8 below provides a lower bound on the probability that a column A_l of the random matrix A from the PSB data model can be recovered from L of its partial supports.

Lemma 8 (**Column recovery from** L **partial supports**). *Under the construction of* A *in Definition 1, consider any* $l \in [n]$*. Let* $\mathcal{C}_l := \{i \in [L] : supp(W_i) \subseteq supp(A_l), |supp(W_i)| > (1-2\epsilon)d\}$ *be a set of* L *partial supports associated with* A_l*. Let* $\hat{A}_l := \sigma(\sum_{i\in\mathcal{C}_l} W_i)$*, where* σ *is the unit step function applied elementwise, be the reconstruction of* A_l *based on these partial supports. With* $\zeta := \lceil nd/m \rceil$*, then*

$$\mathbb{P}(\hat{A}_l \neq A_l) \leq d\left(1 - \binom{n-\zeta}{k-1}\binom{n-1}{k-1}^{-1}\right)^L.$$

Furthermore, if $k = \alpha_1 n + 1$ *and* $m = \alpha_2 n$*, where* $\alpha_1, \alpha_2 \in (0,1)$ *are constants and* $\alpha_1 \leq 1 - \frac{d}{m}$*, then this upper bound can be simplified as follows,*

$$\mathbb{P}(\hat{A}_l \neq A_l) \leq de^{-\tau(n)L}$$

where $\tau(n) := -\ln\left(1 - \left(1 - \frac{d}{\alpha_2(1-\alpha_1)n}\right)^{\alpha_1 n}\right)$ *is* $\mathcal{O}(1)$*.*

For a proof see Appendix A.9. The key takeaway of this lemma is that the probability that NEBF fails to recover a column decreases exponentially in L, the number of partial supports available to reconstruct it. Finally, Lemma 9 concerns the number of data points required so that each column of A is seen sufficiently many times. To be clear, we require N large enough so that the number of nonzeros per row of X is at least as large as some lower bound β with high probability.

Lemma 9 (**Nonzeros per row in** X). *Under the construction of* X *in Definition 1, with* $N \geq \beta\left(\mu\frac{n}{k}\ln(n) + 1\right)$ *for some* $\mu > 1$*, then the probability that* X *has at least* β *non-zeros per row is at least* $\left(1 - n^{-(\mu-1)}\right)^\beta$*.*

For a proof of Lemma 9 see Appendix A.10. With these Lemmas in place we are ready to proceed to the proof of Theorem 1.

3.3 Proof of Theorem 1

Statements 1 and 2 of Theorem 1 are immediate consequences of Lemmas 3 and 6 respectively, so all that is left is to prove statement 3. To quickly recap, our objective is to recover up to permutation the random factor matrices A and X, as defined in the PSB data model in Definition 1, from the random matrix $Y := AX$. Our strategy at a high level is as follows: using Lemmas 5 and 7 we lower bound the probability that EBF factorizes Y by lower bounding the probability that NEBF recovers A. NEBF recovers A up to permutation iff at each iteration of the while loop, lines 2-14 of Algorithm 2, a new column of A is recovered. We lower bound the probability of this using Lemma 8 by first conditioning on there being a certain number of nonzeros per row of X using Lemma 9, and then using a pigeon hole principle argument to ensure that $|\mathcal{C}_h^{(t)}| \geq L(n)$. Here $L(n)$ is chosen to ensure the desired rate. In what follows we adopt the following notation.

- Λ^*_{EBF} and Λ^*_{NEBF} are the events that EBF and NEBF respectively recover A and X up to permutation, meaning there exists an iteration t_f and a permutation $P^{(t_f)}$ such that $\hat{A}^{(t_f)}(P^{(t_f)})^T = A$ and $P^{(t_f)}\hat{X}^{(t_f)} = X$.
- $\Lambda_0 := \{A \in \mathcal{E}^{m\times n}_{k,\epsilon,d}\} \cap \{\epsilon \leq 1/6\}$ is the event that A is the adjacency matrix of a (k, ϵ, d) expander graph with expansion parameter $\epsilon \leq 1/6$.
- For $h \in [n]$ let Λ_h denote the event that a column of A is recovered at the hth iterate of the while loop on lines 2-14 of Algorithm 2 at some iteration t_h of NEBF. Note that by construction $t_1 < t_2 ... < t_h$ where $h \in [n]$ is the index of the last column completed before NEBF terminates.
- Λ_{n+1} is the event that each row of X has at least some quantity $\beta(n)$ (yet to be specified) nonzeros per row which is a function of n.

Proof. As NEBF recovers A iff at every iteration of the while loop (lines 2-14 of Algorithm 2) a column of A is recovered, then

$$\mathbb{P}\left(\Lambda^*_{NEBF} \mid \Lambda_0\right) = \mathbb{P}\left(\bigcap_{h=1}^{n} \Lambda_h \mid \Lambda_0\right)$$

We now apply Bayes' Theorem and condition on Λ_{n+1},

$$\begin{aligned}
\mathbb{P}\left(\Lambda^*_{NEBF} \mid \Lambda_0\right) &= \mathbb{P}\left(\bigcap_{h=1}^{n} \Lambda_h \mid \Lambda_0\right) \\
&= \frac{\mathbb{P}\left(\bigcap_{h=1}^{n} \Lambda_h, \Lambda_0\right)}{\mathbb{P}\left(\Lambda_0\right)} \\
&\geq \frac{\mathbb{P}\left(\bigcap_{h=1}^{n} \Lambda_h, \Lambda_0 \mid \Lambda_{n+1}\right)\mathbb{P}\left(\Lambda_{n+1}\right)}{\mathbb{P}\left(\Lambda_0\right)} \\
&= \frac{\mathbb{P}\left(\bigcap_{h=1}^{n} \Lambda_h \mid \Lambda_0, \Lambda_{n+1}\right)\mathbb{P}\left(\Lambda_0 \mid \Lambda_{n+1}\right)\mathbb{P}\left(\Lambda_{n+1}\right)}{\mathbb{P}\left(\Lambda_0\right)} \\
&= \mathbb{P}\left(\bigcap_{h=1}^{n} \Lambda_h \mid \Lambda_0, \Lambda_{n+1}\right)\mathbb{P}\left(\Lambda_{n+1}\right) \\
&= \mathbb{P}\left(\Lambda_{n+1}\right)\prod_{h=1}^{n}\mathbb{P}\left(\Lambda_h \mid \bigcap_{l=0}^{h-1}\Lambda_l, \Lambda_{n+1}\right).
\end{aligned}$$

In the above, line 2 follows as a result of Bayes Theorem, line 3 is an application of the law of total probability and line 4 is derived using the probability chain rule. The equality on line 5 follows as A and X are drawn independently of one another, therefore, given that Λ_0 is a property of A and Λ_{n+1} a property of X, $\mathbb{P}(\Lambda_0 \mid \Lambda_{n+1}) = \mathbb{P}(\Lambda_0)$. Finally, line 6 is once again an application of the probability chain rule. As an immediate consequence of Lemma 9 it holds that if $N \geq \beta(n)\left(\mu\frac{n}{k}\ln(n) + 1\right)$ then

$$\mathbb{P}\left(\Lambda_{n+1}\right) \geq \left(1 - n^{-(\mu-1)}\right)^{\beta(n)}.$$

With $\beta(n) := (1 + 2\epsilon)dL(n)$, where $L(n) \in \mathbb{N}$ is yet to be determined, we now prove using Lemma 8 that for all $i \in [n]$ it holds that

$$\mathbb{P}\left(\Lambda_h \mid \bigcap_{l=0}^{h-1}\Lambda_l, \Lambda_{n+1}\right) > 1 - de^{-\tau(n)L(n)},$$

where $\tau(n) := -\ln\left(1-\left(1-\frac{d}{\alpha_2(1-\alpha_1)n}\right)^{\alpha_1 n}\right)$. Suppose then that NEBF is starting iteration $h \in [n]$ of the while loop (lines 2-14 of Algorithm 2), meaning that the first $h-1$ columns of $\hat{A}^{(t_{h-1})}$ are complete and, by Lemma 4, $h-1$ columns of A have been recovered. By 1b) of Lemma 4 and given that h and k are finite then there exists an iteration $t_h > t_{h-1}$ such that $[h-1] \cap \mathcal{V}^{(t_h)} = \emptyset$. To be clear, all partial supports belonging to complete columns of $\hat{A}^{(t_{h-1})}$ that it is possible to remove from the residual will have been removed and as a result PeelRes will terminate. Denoting the partial supports extracted from $\mathbf{r}_i^{(t_h)}$ on line 6 of PeelRes as $\mathcal{T}_i$, then using Lemma 4 we can lower bound the number of partial supports extracted at iteration t_h as follows.

$$\begin{aligned}
\sum_{i\in[N]} |\mathcal{T}_i| &> \frac{1}{(1+2\epsilon)d} \sum_{i\in[N]} |supp(X_i) \cap [h-1]| \\
&\geq \frac{1}{(1+2\epsilon)d} \sum_{j\geq h} |supp(\tilde{X}_j)| \\
&\geq \frac{(n-h+1)\beta(n)}{(1+2\epsilon)d} \\
&= (n-h+1)L(n).
\end{aligned}$$

Here the first inequality follows directly from 1b) of Lemma 4, the second inequality holds by 1a) of Lemma 4 and that the rows h to n of $P^{(t_h)}\hat{X}^{(t_h)}$ are zero, finally the third inequality and fourth equality hold as we are conditioning on Λ_{n+1}, meaning there are at least $\beta(n) = (1+2\epsilon)dL(n)$ nonzeros per row of X. Because there are more than $(n-h+1)L(n)$ partial supports, all of which belong to one of the $n-h+1$ columns of A not yet reconstructed, then by the pigeon hole principle there must be a cluster which has at least $L(n)$ partial supports. Therefore, as $\hat{A}_h^{(t_h)} = \sigma\left(\sum_{p\in\mathcal{C}_h^{(t_h)}} W_p\right)$ and $|\mathcal{C}_h^{(t_h)}| \geq L$, then using Lemma 8 it holds for any $h \in [n]$ that

$$\mathbb{P}\left(\Lambda_h \mid \bigcap_{j=0}^{l-1} \Lambda_j, \Lambda_{n+1}\right) > 1 - de^{-\tau(n)L(n)}.$$

Here, from Lemma 8, note that $\tau(n)$ is $\mathcal{O}(1)$. Define $L(n) := \lceil \frac{\gamma \ln(n)}{\tau(n)} \rceil$ for some $\gamma > 1$, and let $N \geq \beta(n)\left(\mu\frac{n}{k}\ln(n)+1\right)$ for some $\mu > 1$. Under these assumptions then using Lemma 8 and Lemma 9 we have

$$\mathbb{P}\left(\Lambda^*_{NEBF} \mid \Lambda_0\right) > \left(1-n^{-(\mu-1)}\right)^{\beta(n)}\left(1-de^{-\tau(n)L(n)}\right)^n.$$

Analyzing the right-hand side of this lower bound

$$\begin{aligned}
\left(1-de^{-\tau(n)L(n)}\right)^n &= \left(1-d'e^{\ln(1/n^\gamma)}\right)^n \\
&= \left(1-\frac{d'}{n^\gamma}\right)^n \\
&= \sum_{i=0}^{\infty}\binom{n}{i}(-1)^i\left(\frac{d'}{n^\gamma}\right)^i \\
&= 1-\frac{d'}{n^{\gamma-1}} + \sum_{i=2}^{\infty}\binom{n}{i}(-1)^i\left(\frac{d'}{n^\gamma}\right)^i \\
&= 1-\mathcal{O}\left(n^{-\gamma+1}\right).
\end{aligned}$$

In the above d' is a constant compensating for the ceil function in the definition of $L(n)$ and the equality on the third line is simply an application of the Binomial series expansion. Analyzing now the left-hand side derived from Lemma 9, then as $\beta(n) = (1+2\epsilon)d\lceil\frac{\gamma\ln(n)}{\tau(n)}\rceil$ and $\tau(n) = \mathcal{O}(1)$ then $\beta(n) = \mathcal{O}(\log(n))$. As a result

$$\begin{aligned}\left(1-n^{-(\mu-1)}\right)^{\beta(n)} &= \sum_{i=0}^{\infty}\binom{\beta(n)}{i}(-1)^i\left(n^{-\mu+1}\right)^i \\ &= 1-\frac{\beta(n)}{n^{\mu-1}}+\sum_{i=2}^{\infty}\binom{\beta(n)}{i}(-1)^i\left(n^{-\mu+1}\right)^i \\ &= 1-\mathcal{O}\left(n^{-\mu+1}\log(n)\right).\end{aligned}$$

Therefore asymptotically

$$\mathbb{P}\left(\Lambda^*_{NEBF} \mid \Lambda_0\right) > \left(1-\mathcal{O}\left(n^{-\mu+1}\log(n)\right)\right)\left(1-\mathcal{O}\left(n^{-\gamma+1}\right)\right).$$

We simplify our result to make it more interpretable by letting $N \in \mathbb{N}$ satisfy $N \geq \nu\frac{4d}{\tau(n)}\frac{n}{k}\ln^2(n)$ where $\nu = \alpha\gamma$ is some constant. Note that if this is satisfied then $N \geq \beta(n)\left(\mu\frac{n}{k}\ln(n)+1\right)$ and so the above inequality holds. Without loss of generality if we let $\gamma = \mu$ then

$$\mathbb{P}\left(\Lambda^*_{NEBF} \mid \Lambda_0\right) > 1-\mathcal{O}(n^{-\sqrt{\nu}+1}\log(n))$$

as claimed. Therefore, by Lemma 7, under the same conditions it also holds that

$$\mathbb{P}\left(\Lambda^*_{EBF} \mid \Lambda_0\right) > 1-\mathcal{O}(n^{-\sqrt{\nu}+1}\log(n))$$

as claimed in statement 3 of Theorem 1. This concludes the proof. □

4 An improved factorization algorithm and numerics

EBF was designed with the primary goal of recoverying the sparse binary factor $\mathbf{A}$, with identification of the sparse coding matrix $\mathbf{X}$ not making use of algorithmic advances in combinatorial compressed sensing. This section presents a first step improvement to EBF with the aim of countering this and is structured as follows: in Section 4.1 we introduce a simple improvement to EBF by augmenting the sparse coding recovery using the ℓ_0-decoding algorithm [25, Algorithms 1 & 2]. Sections 4.2 and 4.3 respectively analyze the computational complexity and describe how to choose the expansion parameter ϵ, Section 4.4 then concludes with numerical experiments, demonstrating the efficacy of EBF and ℓ_0-EBF in practice on synthetic problems.

4.1 An improved factorization algorithm: ℓ_0-EBF

EBF, Algorithm 1, can readily be improved[2] by augmenting the recovery of the sparse coding factor using algorithms designed in the combinatorial compressed sensing literature; for example: Sparse Matching Putsuit (SMP) [12], Sequential Sparse Matching Pursuit (SSMP) [11], Left Degree Dependent Signal Recovery (LDDSR) [34] and Expander Recovery (ER) [21]. From amongst these we consider ℓ_0-decode [25, Algorithms 1 & 2], as this algorithm is also based on the model that the columns of $\mathbf{X}$ satisfy the dissociated property, and as can be observed in [25] is able to recover $\mathbf{X}$ from $\mathbf{Y}$ given $\mathbf{A}$ even with large k, i.e., $k \approx m/3$.

[2]NEBF, Algorithm 2, is introduced to establish Theorem 1 and is not recommended for computing the factors in practice.

Algorithm 3 ℓ_0-EBF($\mathbf{Y}$, d, k, m, n, N, ϵ)

1: **Init:** $\mathbf{R}^{(0)} \leftarrow \mathbf{Y}$, $\hat{\mathbf{A}}^{(0)} \leftarrow \text{zeros}(m, n)$, $\hat{\mathbf{X}}^{(0)} \leftarrow \text{zeros}(n, N)$, $t \leftarrow 0$.
2: **while** $\|\hat{\mathbf{A}}^{(t)}\|_0 > \|\hat{\mathbf{A}}^{(t-1)}\|_0$ or $\|\hat{\mathbf{X}}^{(t)}\|_0 > \|\hat{\mathbf{X}}^{(t-1)}\|_0$ or $t = 0$ **do**
3: Set $t \leftarrow t + 1$.
4: Extract set of singleton values $\mathcal{Q}^{(t)}$ and associated partial supports $\mathbf{W}^{(t)}$ from $\mathbf{R}^{(t-1)}$.
5: Update $\hat{\mathbf{A}}^{(t)}$ by inspecting $\mathbf{R}^{(t)}$ for singleton values identified in prior iterations.
6: Cluster partial supports $\mathbf{w}_p^{(t)}$ ($p \in [c(t)]$) into sets $\mathcal{C}_h^{(t)}$ ($h \in [n]$).
7: Update $\hat{\mathbf{X}}^{(t)}$: for all $r_{j,i} \in \mathcal{Q}^{(t)}$ if $r_{j,i}$ has partial support $\mathbf{w}_p^{(t)} \in \mathcal{C}_h^{(t)}$ set $\hat{x}_{h,i} \leftarrow r_{j,i}$.
8: Update $\hat{\mathbf{A}}^{(t)}$: for all $l \in [n]$ set $\hat{\mathbf{a}}_h^{(t)} \leftarrow \sigma\left(\hat{\mathbf{a}}_h^{(t-1)} + \sum_{p \in \mathcal{C}_h^{(t)}} \mathbf{w}_p^{(t)}\right)$.
9: Compute full column residual: $\mathbf{R}' \leftarrow \mathbf{Y} - \hat{\mathbf{A}}^{(t)}([m], \mathcal{H}^{(t)})\hat{\mathbf{X}}^{(t)}(\mathcal{H}^{(t)}, [N])$.
10: Update rows of $\hat{\mathbf{X}}^{(t)}$ in $\mathcal{H}^{(t)}$: $\hat{\mathbf{X}}^{(t)}(\mathcal{H}, [N]) \leftarrow \ell_0\text{-decode}(\mathbf{R}', \hat{\mathbf{A}}^{(t)}([m], \mathcal{H}^{(t)}), \alpha)$.
11: Update residual: $\mathbf{R}^{(t)} \leftarrow \mathbf{Y} - \hat{\mathbf{A}}^{(t)}\hat{\mathbf{X}}^{(t)}$.
12: **end while**
13: **Return** $(\hat{\mathbf{A}}^{(t)}, \hat{\mathbf{X}}^{(t)})$

Relative to EBF, ℓ_0-EBF involves two additional steps, corresponding to Steps 9 and 10 of Algorithm 3. In order that ℓ_0-decode can be incorporated into step 9 without any adaptations, it needs to act only on the columns of $\mathbf{A}$ that are complete i.e., those whose indices are in $\mathcal{H}^{(t)}$. For this reason we compute the full column residual, which we denote $\mathbf{R}'$, using just the complete columns of $\hat{\mathbf{A}}^{(t)}$, denoted $\hat{\mathbf{A}}^{(t)}([m], \mathcal{H}^{(t)})$ and pass both to the ℓ_0-decode subroutine in step 10. ℓ_0-decode acts on each columns of $\mathbf{R}'$ independently, identifying if there is a value ω such that $\|\mathbf{r}'_i\|_0 - \|\mathbf{r}'_i - \omega\hat{\mathbf{a}}_l^{(t)}\|_0 \geq \alpha$ for an appropriate α, and if so then $\omega\hat{\mathbf{a}}_l^{(t)}$ is removed from $\mathbf{r}'_i$ and x_i is updated by $\hat{x}_{l,i}^{(t)} \leftarrow \hat{x}_{l,i}^{(t)} + \omega$. If $supp(\mathbf{x}_j) \subseteq \mathcal{H}^{(t)}$, that is $\mathbf{r}_i$ is generated using only the columns of $\mathbf{A}$ which have been recovered, then so long as $\alpha \geq (1 - 2\epsilon)d$ then ℓ_0-decode will recover the correct values in x_i, see [25, Theorems 4.6 & 4.7].

Corollary 3 (**Guarantees for ℓ_0-EBF**). *Theorem 1 holds for ℓ_0-EBF provided $\alpha \geq (1 - 2\epsilon)d$.*

A proof of this corollary is given in Appendix A.11. Although $\alpha \geq (1 - 2\epsilon)d$ gives us guaranted recovery, the practical benefit of ℓ_0-EBF over EBF is that it typically performs well with the α parameter smaller than the theory suggests. This relaxes the condition on the frequency with which an entry appears for it be accepted as a singleton value and hence typically allows one to recover $\mathbf{X}$ for both larger values of k and with less computational effort. This benefit is evident in Figures 2 & 3.

4.2 Computational complexity

The computational complexity of each step in Algorithm 3 is as follows, note that many of the steps are fully parallelizable.

- Steps 4 & 5: these steps can be done simultaneously at a cost of $\mathcal{O}(d^2k^2)$ operations per column, so for the whole of $\mathbf{R}^{(t)}$ the computational complexity is $\mathcal{O}(d^2k^2N)$. We note that the processing of each column for both these steps can be conducted in parallel.
- Step 6 & 7: clustering the partial supports can be achieved by computing the inner products between each $\mathbf{w}_i$ as well as with the columns of $\hat{\mathbf{A}}^{(t)}$. Computationally this is equivalent to computing $\left(\mathbf{W}^{(t)}\right)^T \hat{\mathbf{A}}^{(t)}$, given each column of $\mathbf{R}^{(t)}$ provides $\mathcal{O}(k)$ partial supports this has

a computational complexity of $\mathcal{O}(d^2knN)$. The update of $\hat{\mathbf{X}}^{(t)}$ can be done in parallel with the clustering at a cost of $\mathcal{O}(kN)$ since once a partial support has been clustered, then the corresponding singleton value can be used to update the relevant entry in $\hat{\mathbf{X}}^{(t)}$. We note that the clustering of each column cannot always be performed in parallel.

- Step 8: updating the reconstruction of $\hat{\mathbf{A}}^{(t)}$ requires that for each of its n columns one computes $\mathcal{O}(N)$ additions of $\mathcal{O}(d)$ sparse partial supports, which has a total complexity of $\mathcal{O}(dnN)$. Updating $\mathcal{H}^{(t)}$ requires $\mathcal{O}(d)$ operations per column, so is overall $\mathcal{O}(dn)$. Once clustering has been performed the update of each column is fully parallelizable.

- Step 9 & 10: the computational complexity of both these steps is dominated by a matrix multiplication. The inner product between a row of $\hat{\mathbf{A}}^{(t)}$ and a column of $\hat{\mathbf{X}}^{(t)}$ is $\mathcal{O}(dk)$, given that there are $\mathcal{O}(nN)$ of these inner products then the overall complexity of both of these steps is $\mathcal{O}(dknN)$.

- Step 11: the computational complexity of running ℓ_0-decode on a single column of the residual is $\mathcal{O}(dkn)$ [25, Theorems 4.6 & 4.7]. As there are $\mathcal{O}(N)$ nonzero columns in the residual $\mathbf{R}^{(t)}$ then the cost is $\mathcal{O}(dknN)$, which is also fully parallelizable.

In summary, although ℓ_0-EBF requires some additional computational overheads when compared with EBF, asymptotically both algorithms have a per while loop iteration complexity of $\mathcal{O}(d^2k^2N)+\mathcal{O}(d^2knN)$, albeit with different constants. Under the asymptotic regime outlined in Theorem 1 with d fixed, $k=\alpha_1 n+1$ and $m=\alpha_2 n$ where $\alpha_1,\alpha_2\in(0,1)$, then in terms of the dimensions of the input matrix $\mathbf{Y}$ this gives a per while loop iteration cost of $\mathcal{O}(m^2N)$.

4.3 Choosing a value of ϵ in practice

The expansion parameter ϵ plays a key role in all the Algorithms discussed so far, both in extracting partial supports as well as clustering them. However, the value of this parameter is generally not known; for instance, even if we know that $\mathbf{A}\in\mathcal{E}^{m\times n}_{k,\epsilon,d}$ it is in general NP hard to compute its expansion parameter. From Corollary 2, if $\epsilon>1/6$ then we can provide no guarantees that the clustering step in any of the Algorithms will be successful, therefore in this situation even if we know ϵ it is of little value to us, at least under the current framework. The key takeaway from this section is that so long as $\epsilon\leq 1/6$ then knowing the exact value of ϵ is not important - we can simply use $1/6$ as the ϵ parameter in Algorithms 1-3 without any appreciable loss in performance. We now justify this claim, note that in what follows ϵ' is our estimate of the true parameter ϵ.

- *Identifying singleton values and extracting partial supports:* if we underestimate the true expansion parameter, i.e., $\epsilon'<\epsilon$, then we run the risk of identifying non-singletons as singletons as $2\epsilon' d<2\epsilon d$. If we overestimate the expansion parameter, i.e., $\epsilon'\geq\epsilon$, this is not true since $2\epsilon' d\geq 2\epsilon d$. Therefore overestimating introduces no false positives in terms of identifying singleton values. Furthermore, given Corollary 2, if we just want to identify the partial supports appearing more than $(1-2\epsilon)d$ then if we overestimate ϵ then $(1-2\epsilon)d>(1-2\epsilon')d$. Therefore, by overestimating we also still identify all the partial supports with size greater than $(1-2\epsilon)d$ which we would have identified if we had used the true ϵ value instead.

- *Clustering partial supports:* we assign two columns to the same cluster iff their inner product is greater than $(1-4\epsilon')d$. Since $(1-4\epsilon)d>(1-4\epsilon')d$, then any pair of partial supports that do originate from the same column are still clustered correctly. Furthermore, if ϵ' is an overestimate then partial supports originating from different columns are still never assigned

to the same cluster. Indeed, with $\epsilon' \leq 1/6$ and $\epsilon' \geq \epsilon$ then $(1-4\epsilon')d \geq 2\epsilon d$. Therefore, since by Theorem 2 the overlap of two full columns of $\mathbf{A}$ is less than $2\epsilon d$, if the supports of two partial supports overlaps by more than $(1-4\epsilon')d$ then they must belong to the same column. Therefore, two partial supports belong to the same cluster iff their supports overlap by more than $(1-4\epsilon')d$.

In conclusion, so long as $\epsilon \leq \frac{1}{6}$ then we are able to extract and accurately cluster all partial supports which appear more than $(1-2\epsilon)d$ times by using $\epsilon' = \frac{1}{6}$. The drawback of using this ϵ' is that by overestimating we potentially ignore certain singleton values that appear with frequency between $2\epsilon d$ and $(1-2\epsilon)d$. However, since we have no guarantees in terms of how to cluster these partial supports we lose little by this overestimation.

4.4 Numerical experiments

In this section we demonstrate the efficacy of EBF and ℓ_0-EBF in practice on synthetic problems. When generating the supports of both test matrices $\mathbf{A}$ and $\mathbf{X}$ we follow the same procedure as outlined in Definition 1. The coefficients of $\mathbf{X}$ are sampled uniformly at random from $[-b_2, -b_1] \cap [b_1, b_2]$ where $0 < b_1 < b_2$ (in our experiments we used $b_1 = 1$ and $b_2 = 5$).

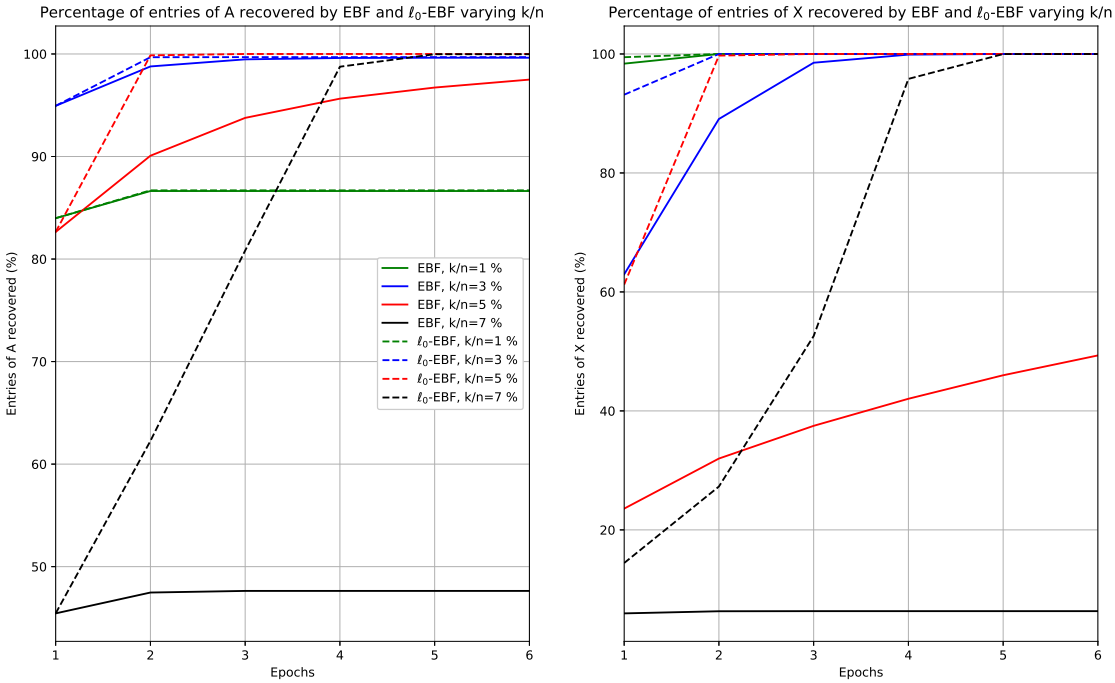

Figure 2: Recovery, up to permutation, of ($\mathbf{A}$,$\mathbf{X}$), generated as described in Definition 1, from $\mathbf{Y} = \mathbf{AX}$ using EBF and ℓ_0-EBF for different values of k. Experiment parameters; $N = 200$, $n = 1200$, $m = 900$, $d = 10$, $\epsilon = 1/6$, $\alpha = 3$ with k varied between 1% and 7% of n. We emphasize that the green curve in the left hand plot is below 100% only due to there being 149 columns of $\mathbf{A}$ that do not appear in the data matrix $\mathbf{Y}$. All columns which are present in the data are recovered perfectly.

Figure 2 shows the recovery of entries of $\mathbf{A}$ and $\mathbf{X}$ by epoch (single pass through all data available) for various values of k. Note that as k grows there is a tradeoff between the number of nonzeros per row of $\mathbf{X}$ and the degree of overlap between sets of k columns of $\mathbf{A}$. For very low sparsities, i.e. $k/n = 1\%$, the extraction of singleton values and hence partial supports is not

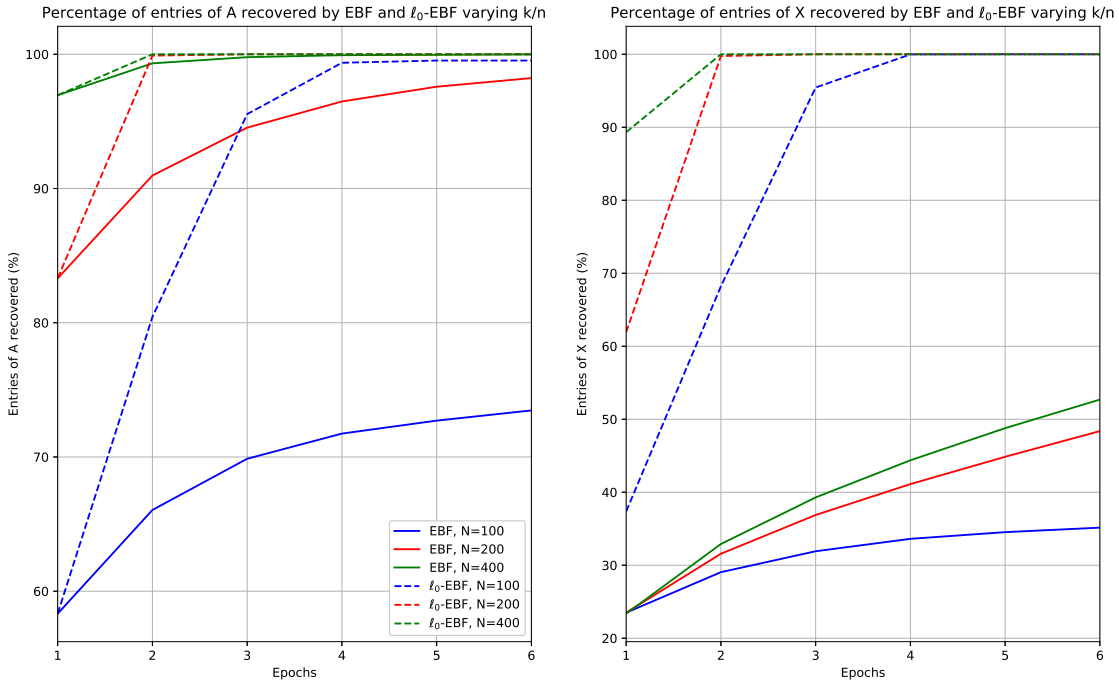

Figure 3: Recovery, up to permutation, of $(\mathbf{A},\mathbf{X})$, generated as described in Definition 1, from $\mathbf{Y} = \mathbf{AX}$ using EBF and ℓ_0-EBF for different values of N. Experiment parameters; n = 1200, m = 900, d = 10, k = 60, $\epsilon = 1/6$, $\alpha = 3$ (ℓ_0-EBF parameter only) with $N = 100$, 200 and 400. Recovery figures for ℓ_0-EBF include innacuracies of coefficient values in $\leq 0.1\%$ of entries of $\mathbf{X}$ (support is recovered with complete accuracy).

challenging since the degree of overlap between only a few columns of $\mathbf{A}$ is small. As a result, both algorithms recover $\mathbf{X}$ with ease by the end of the second epoch. However, recovering the entirety of $\mathbf{A}$ is not possible for either algorithm since there are a number of rows of $\mathbf{X}$, 149 to be exact, that have no nonzeros. This means that there are 149 columns of $\mathbf{A}$ which never appear or contribute to the data and hence cannot be recovered from it. We emphasize that the failure to recover the entirety of $\mathbf{A}$ in this case is due to insufficient data in regard to the way the supports of $\mathbf{X}$ are sampled, not due to the way either EBF or ℓ_0-EBF operate. Indeed, both algorithms perfectly recover all columns of $\mathbf{A}$ which do appear in the data. From the perspective of unsupervised learning then the issue arising here can be interpreted as the training data set being too small to accurately represent the underlying data distribution. For $k/n = 3\%$ there are only 3 zero rows in $\mathbf{X}$ and both algorithms recover all but these 3 columns of $\mathbf{A}$ and all of $\mathbf{X}$. Note however that recovery, at least for EBF, takes more epochs since the level of overlap between subsets of k columns has increased. For $k/n = 5\%$ there are no zero rows in $\mathbf{X}$. However, in only six epochs EBF has not yet succeeded in factorizing $\mathbf{Y}$ fully. By contrast, ℓ_0-EBF recovers both factors after 2 epochs, highlighting the benefit of the added ℓ_0 decode step. Finally, for $k/n = 7\%$, the degree of overlap between subsets of k columns of $\mathbf{A}$ is becoming significant, therefore far fewer partial supports of size larger than $(1-2\epsilon)d$ can be extracted and as a result EBF fails to converge. ℓ_0-EBF is still successful however, since by setting $\alpha = 3$ partial supports whose support cardinality is smaller than $(1-2\epsilon)d$ can still be matched (although not necessarily with theoretical guarantees) with completed columns and hence their contribution removed from the residual. We emphasize that although in practice ℓ_0-EBF, when compared with EBF, performs the factorization more efficiently and for larger values of k, this relies on relaxing the α parameter potentially below the threshold

suggested in Corollary 3. Indeed, when $\alpha < (1-2\epsilon)d$ then ℓ_0-EBF does on occasion make mistakes and this is evident in our experiments, albeit only in less than 0.1 percent of entries.

Figure 3 shows the recovery of $\mathbf{A}$ and $\mathbf{X}$ keeping k fixed at $n/20$ and for the three choices of $N = 100$, 200, and 400. The trend for both algorithms is the same: more data ensures more partial supports per iteration and hence a greater chance of success and faster convergence. We highlight that despite all values of N considered here being far less than $n = 1200$, ℓ_0-EBF is able to compute almost all the factorizations in just 5 epochs ($\mathbf{A}$ for N=100 being the exception and here only $< 1\%$ of entries are missing). After only 6 epochs EBF has not had enough iterations to converge and it seems unlikely that it will successfully compute any of the factorizations. This is due to its overly pessimistic criteria for selecting singleton values, resulting in few entries of $\mathbf{X}$ being recovered.

5 Conclusions and future extensions

The PSB data model, Definition 1, and its associated factorization algorithms introduce a new variant of community detection, dictionary learning, and show that combinatorial compressed sensing reconstructions can be achieved without knowledge of the sensing matrix. Notably, the Expander Based Factorization (EBF) algorithms presented here achieve near optimal sampling complexity $N = \mathcal{O}\left(\frac{n}{k}\log^2(n)\right)$, as highlighted in Theorem 1. Moreover, these algorithms also have a low per iteration cost of $\mathcal{O}(m^2N)$ (see Section 4.2), much of which is trivially parallelizable and furthermore experiments on synthetic data demonstrate that the factorization is typically achieved in only a few iterations. The main limitation of the data model and the results presented are related to the modelling assumptions. A few potential future extensions are as follows:

- **Stability to noise and projections of arbitrary matrices onto the PSB model:** More developed matrix factorizations, such as PCA, subspace clustering, and dictionary learning, are all known to be effective for matrices which only approximately achieve the modelling assumptions in question. A first extension of the PSB data model would be to show that the factors are stable to additive noise and remain efficient and reliable to compute. Stability to additive noise one might expect to achieve via related work in combinatorial compressed sensing, such as[26]. This could be further augmented by having repeated samples contained in Y. In order for the PSB data model to be more generally applicable, approximation rates are required, in particular a derivation of the distance between an arbitrary matrix $\mathbf{Y}$ and its projection onto the PSB data model - which would presumably show better approximation as the number of columns n of A is increased. To be clear, this would entail bounds of the form $\|\mathbf{Y} - P_{PSB(n)}\mathbf{Y}\| \leq f(n)$ where $P_{PSB(n)}(\cdot)$ projects to the nearest matrix in the PSB data model and $f(n)$ is a rapidly decreasing function of n and or potentially d and k.

- **Relaxing the $\epsilon \leq 1/6$ condition:** Corollary 2 guarantees that partial supports can be trivially clustered if the binary factor matrix A is a (k, ϵ, d) expander graph with expansion parameter $\epsilon \leq 1/6$. This expansion parameter is guaranteed over all $\binom{n}{k}$ sets of k columns in A. As Y contains at most N such sets of k columns, and as N would typically be exponentially smaller than this binomial coefficient, it is expected that similar bounds could be derived under a model for which it holds only that with high probability a subset of k columns of A satisfies the expansion bound. Moreover, the numerical experiments conducted in Section 4 demonstrate that successful factorization is possible in parameter regimes where ϵ is likely to exceed 1/6. As a result, it seems reasonable that more robust extensions of the current clustering method could be conceived.

- **Broadening the PSB data model:** two assumptions that restrict the expressivity of the PSB data model are that there is a fixed number d nonzeros per column of A and that the columns of X are dissociated. Relaxing these conditions would clearly aid in the applicability of the model. One would expect it to be possible to derive similar results for the case in which the number of nonzeros per column of A is bounded from above and below. However, this would likely increase the computational complexity of the factorization task as the number of nonzeros per column would also need to be learned. Robustness to the entries in X being dissociated, at least with high probability, can also be expected, however, adversarial choices of nonzero values in X may require combinatorial searches on certain entries of Y and could result in Y no longer having a unique factorization.

A Proofs deferred to Appendix

A.1 Proof of Lemma 1: Identification of Singleton Values

Proof. This result follows by upper bounding the frequency with which any nonsingleton value appears. Consider the pairwise overlap between any two columns $\mathbf{a}_l$ and $\mathbf{a}_h$ of $\mathbf{A} \in \mathcal{E}^{m\times n}_{(k,\epsilon,d)}$ where $l, h \in [n]$ and $l \neq h$, from Definition 3 it follows that

$$|supp(\mathbf{a}_l) \cup supp(\mathbf{a}_h)| > (1-\epsilon)2d.$$

Using the inclusion-exclusion principle then

$$|supp(\mathbf{a}_l)| + |supp(\mathbf{a}_h)| - |supp(\mathbf{a}_l) \cap supp(\mathbf{a}_h)| > (1-\epsilon)2d.$$

Given that $|supp(\mathbf{a}_l)| = |supp(\mathbf{a}_h)| = d$ then a simple rearrangement shows that

$$|supp(\mathbf{a}_l) \cap supp(\mathbf{a}_h)| < 2d - (1-\epsilon)2d = 2\epsilon d.$$

Therefore for any $\mathcal{S} \subset [n]$ with $|\mathcal{S}| \geq 2$ and for any $l, h \in \mathcal{S}$

$$|\bigcap_{l\in\mathcal{S}} supp(\mathbf{a}_l)| \leq |supp(\mathbf{a}_l) \cap supp(\mathbf{a}_h)| < 2\epsilon d.$$

Now consider the entries of $\mathbf{b} = \mathbf{A}\mathbf{z}$ where $\mathbf{z}$ is dissociated and note that any nonsingleton value will be the sum of two or more entries in $\mathbf{z}$. Therefore, for any subset of the support $\mathcal{S} \subset supp(\mathbf{z})$ satisfying $|\mathcal{S}| \geq 2$, then using the inequality derived above the frequency with which nonsingleton values appear can be bounded from above as follows,

$$|\{j \in supp(\mathbf{b}) : y_j = \sum_{l\in\mathcal{S}} z_l \}| = |\bigcap_{l\in\mathcal{S}} supp(\mathbf{a}_l)| < 2\epsilon d.$$

Therefore any value that appears more than $2\epsilon d$ times is a singleton value. Note that since by assumption $\mathbf{z}$ is dissociated, any singleton value cannot be formed by a linear combination of more than one element of $\mathbf{z}$. □

A.2 Proof of Lemma 2 - Existence of Singleton Values

Proof. For typographical ease let $\gamma := \lceil (1-2\epsilon)d \rceil$ and define $U := supp(\mathbf{z}) \backslash \mathcal{T}$, which is the set of singleton values which appear at most $(1-2\epsilon)d$ times. Additionally the following graph inspired notation is adopted,

- $\mathcal{N}_1(\mathbf{b}) := \bigcup_{l \in supp(\mathbf{z})} \Omega_l$ is the set of row indices of $\mathbf{b}$ corresponding to singleton values, by Theorem 2 it holds that $|\mathcal{N}_1(\mathbf{b})| \geq \gamma |supp(\mathbf{z})|$.
- $\mathcal{N}_1^{\mathcal{T}}(\mathbf{b}) := \bigcup_{l \in \mathcal{T}} \Omega_l$ is the set of row indices of $\mathbf{b}$ corresponding to singleton values that appear more than $(1-2\epsilon)d$ times. Therefore $\gamma|\mathcal{T}| \leq |\mathcal{N}_1^{\mathcal{T}}(\mathbf{b})| \leq d|\mathcal{T}|$.
- $\mathcal{N}_1^{\mathcal{U}}(\mathbf{b}) := \bigcup_{l \in \mathcal{U}} \Omega_l$ is the set of row indices of $\mathbf{b}$ corresponding to singleton values that appear at most $(1-2\epsilon)d$ times. Note that by definition $|\mathcal{N}_1^{\mathcal{U}}(\mathbf{b})| \leq (\gamma - 1)|\mathcal{U}|$.

By definition $|\mathcal{N}_1(\mathbf{b})| = |\mathcal{N}_1^{\mathcal{T}}(\mathbf{b})| + |\mathcal{N}_1^{\mathcal{U}}(\mathbf{b})|$, it hence follows that

$$\begin{aligned} |\mathcal{N}_1(\mathbf{b})| &\geq \gamma |supp(\mathbf{z})| \\ &= \gamma(|\mathcal{T}| + |\mathcal{U}|) \\ &= \gamma|\mathcal{T}| + |\mathcal{U}| + (\gamma - 1)|\mathcal{U}| \\ &\geq \gamma|\mathcal{T}| + |\mathcal{U}| + |\mathcal{N}_1^{\mathcal{U}}(\mathbf{b})|. \end{aligned}$$

Subsituting $|\mathcal{N}_1(\mathbf{b})| = |\mathcal{N}_1^{\mathcal{T}}(\mathbf{b})| + |\mathcal{N}_1^{\mathcal{U}}(\mathbf{b})|$ it holds that

$$\begin{aligned} |\mathcal{N}_1^{\mathcal{T}}(\mathbf{b})| &\geq \gamma|\mathcal{T}| + |\mathcal{U}| \\ &= |\mathcal{T}| + |\mathcal{U}| + (\gamma - 1)|\mathcal{T}| \\ &= |supp(\mathbf{z})| + (\gamma - 1)|\mathcal{T}|. \end{aligned}$$

Since $|\mathcal{N}_1^{\mathcal{T}}(\mathbf{b})| \leq d|\mathcal{T}|$ then $d|\mathcal{T}| \geq |supp(\mathbf{z})| + (\gamma - 1)|\mathcal{T}|$, rearranging gives

$$|\mathcal{T}| \geq \frac{|supp(\mathbf{z})|}{d - \gamma + 1}.$$

Since $\gamma \geq d - 2\epsilon d$ then

$$|\mathcal{T}| \geq \frac{|supp(\mathbf{z})|}{1 + 2\epsilon d}.$$

Given that $|\mathcal{T}| \in \mathbb{Z}_{\geq 0}$ and $\frac{|supp(\mathbf{z})|}{1+2\epsilon d} > 0$, then so long as $|supp(\mathbf{z})| \geq 1$ and $\mathbf{b}$ is nonzero it is always possible to extract at least one partial support with cardinality greater than $(1-2\epsilon)d$. □

A.3 Proof of Corollary 2 - clustering partial supports

Proof. From the proof of Lemma 1, if $\mathbf{w}_p$ and $\mathbf{w}_q$ arise from different columns of $\mathbf{A}$, say $\mathbf{a}_l$ and $\mathbf{a}_h$ respectively where $l \neq h$, then

$$\mathbf{w}_p^T \mathbf{w}_q \leq \mathbf{a}_l^T \mathbf{a}_h < 2\epsilon d.$$

If $\mathbf{w}_p$ and $\mathbf{w}_q$ belong to the same column, say $\mathbf{a}_l$, then since each has more than $(1-2\epsilon)d$ nonzeros of $\mathbf{a}_l$ then they must differ by less than $4\epsilon d$ nonzeros. Therefore

$$\mathbf{w}_p^T \mathbf{w}_q \geq (1 - 4\epsilon)d.$$

For $(1-4\epsilon)d \geq 2\epsilon d$ to hold then rearranging gives $\epsilon \leq 1/6$. Therefore, when $\epsilon \leq 1/6$, if $\mathbf{w}_p$ and $\mathbf{w}_q$ arise from different columns then $\mathbf{w}_p^T \mathbf{w}_q < (1-4\epsilon)d$. Hence $\mathbf{w}_p^T \mathbf{w}_q \geq (1-4\epsilon)d$ iff $\mathbf{w}_p$ and $\mathbf{w}_q$ arise from the same column of $\mathbf{A}$. □

A.4 Proof of Lemma 3 - EBF only identifies correct entries

Proof. Statement 2 of 3 is an immediate consequence of statement 1 since if statement 1 holds and $\mathbf{R}^{(t_f)} = \mathbf{0}_{m\times n}$ then EBF must have successfully recovered $\mathbf{A}$ and $\mathbf{X}$ up to permutation. Statement 1 can be proven by induction: the base case $t = 0$ is trivial as $\hat{\mathbf{X}}^{(0)} = \mathbf{0}_{n\times N}$ and $\hat{\mathbf{A}}^{(0)} = \mathbf{0}_{m\times n}$. Assume then that statement 1 is true at iteration $t-1$ and consider any column of the residual $i \in [n]$, then there exists a permutation $\mathbf{P}^{(t-1)}$ such that

$$\begin{aligned}\mathbf{r}_i^{(t-1)} &= \mathbf{y}_i - \hat{\mathbf{A}}^{(t-1)}\hat{\mathbf{x}}_i^{(t-1)} \\ &= \mathbf{y}_i - \hat{\mathbf{A}}^{(t-1)}(\mathbf{P}^{(t-1)})^T\mathbf{P}^{(t-1)}\hat{\mathbf{x}}_i^{(t-1)} \\ &= \mathbf{y}_i - \mathbf{A}'\mathbf{x}_i' \\ &= \mathbf{y}_i - \mathbf{y}_i'\end{aligned}$$

where $\mathbf{y}_i' := \mathbf{A}'\mathbf{x}_i'$. Consider a nonsingleton value $s = \sum_{l\in\mathcal{S}} x_{l,i}$ which is the sum of some subset $\mathcal{S} \subseteq supp(\mathbf{x}_i)$, $|\mathcal{S}| \geq 2$ entries of $\mathbf{x}_i$. Note by the dissociated property that there does not exist a $\mathcal{T} \subset supp(\mathbf{x}_i)$, $\mathcal{T} \neq \mathcal{S}$ such that $s = \sum_{l\in\mathcal{T}} x_{l,i}$. From the proof of Lemma 1 then $\mathcal{S} \subseteq supp(\tilde{\mathbf{a}}_j) \cap supp(\mathbf{x}_i)$ is true for less than $2\epsilon d$ row indices $j \in [m]$. Therefore, by the dissociated property, for a nonsingleton value to appear more than $2\epsilon d$ times in $\mathbf{r}_i^{(t)}$ it must be introduced by the subtraction of $\mathbf{y}_i'$ into new locations, meaning that for some $j \in [m]$ for which $\mathcal{S} \nsubseteq supp(\tilde{\mathbf{a}}_j) \cap supp(\mathbf{x}_i)$ then $\mathcal{S} \subseteq supp(\tilde{\mathbf{a}}_j') \cap supp(\mathbf{x}_i')$. Since it is assumed that statement 1 is satisfied at iteration $t-1$, $\mathbf{A}' \in \{0,1\}^{m\times n}$ satisfies $supp(\mathbf{A}') \subseteq supp(\mathbf{A})$, $\mathbf{x}_i' \in \mathcal{X}_k^n$ satisfies $supp(\mathbf{x}_i') \subseteq supp(\mathbf{x}_i)$ and if $x_{j,i}' \neq 0$ then $x_{j,i}' = x_{i,j}$. Therefore, for all $j \in [m]$ it holds that $(supp(\tilde{\mathbf{a}}_j')\cap supp(\mathbf{x}_i')) \subseteq (supp(\tilde{\mathbf{a}}_j)\cap supp(\mathbf{x}_i))$ and hence if $\mathcal{S} \nsubseteq supp(\tilde{\mathbf{a}}_j) \cap supp(\mathbf{x}_i)$ then $\mathcal{S} \nsubseteq supp(\tilde{\mathbf{a}}_j') \cap supp(\mathbf{x}_i')$. It then follows that the subtraction of $\mathbf{y}_i'$ does not introduce more instances of a nonsingleton value and therefore the mode of nonsingleton values in the residuals at any iteration is bounded from above by $2\epsilon d$. Since it is assumed that $\epsilon \leq 1/6$ then $2\epsilon d \leq (1-2\epsilon)d$ and hence, for any $t \leq t_f$ and $i \in [n]$, any value appearing in $\mathbf{r}_i^{(t)}$ more than $(1-2\epsilon)d$ times cannot be a nonsingleton. Since EBF only identifies an entry as a singleton value if it appears more than $(1-2\epsilon)d$ times then no nonsingleton values are identified as singletons. Consequently, the extraction of singleton values and partial supports from $\mathbf{r}_i^{(t)}$ and their subsequent clustering by Corollary 2 only identifies correct entries. As this is true for all $i \in [n]$ then there exists a permutation $\mathbf{P}^{(t)}$ such that $supp(\hat{\mathbf{A}}^{(t)}(\mathbf{P}^{(t)})^T) \subseteq supp(\mathbf{A})$, $supp(\mathbf{P}^{(t)}\hat{\mathbf{X}}^{(t)}) \subseteq supp(\mathbf{X})$ and all nonzeros in $\mathbf{P}^{(t)}\hat{\mathbf{X}}^{(t)}$ are equal to the corresponding entry in $\mathbf{X}$. Therefore by induction statement 1 is true for all $t \leq t_f$. □

A.5 Proof of Lemma 4 - NEBF only identifies correct entries

Proof. Statements 1a)-c) of Lemma 4 can be proven for all $t \leq t_f$ using induction and the associated statement 2 is a byproduct of this. The base case $t = 0$ is trivial for 1a) since $\hat{\mathbf{X}}^{(0)} = \mathbf{0}_{n\times N}$ and $\mathbf{R}^{(0)} = \mathbf{Y} = \mathbf{A}\mathbf{X}$, so by definition $\mathbf{X} \in \mathcal{X}_k^{n\times N}$ and $\mathbf{P}^{(0)}$ can be any permutation. Given 1a) holds true for $t = 0$ then by Lemma 2 there will be at least $|supp(\mathbf{x}_i)|/(1+2\epsilon d)$ singleton values and associated partial supports, each with more than $(1-2\epsilon)d$ nonzeros, per column of $\mathbf{R}^{(0)}$. Since it is assumed that $\epsilon \leq 1/6$ then $2\epsilon d \leq (1-2\epsilon)d$ and so, by Lemma 1, any nonsingleton value cannot appear in a column more than $(1-2\epsilon)d$ times. Therefore singleton values appearing more than $(1-2\epsilon)d$ times in a column can be identified by checking their column mode and are necessarily correct entries in of $\mathbf{X}$. Additionally, by Corollary 2, these partial supports can also be clustered without error. Therefore 1b) holds for any permutation at $t = 0$. Finally, since $\hat{\mathbf{X}}^{(0)} = \mathbf{0}_{n\times N}$ and $\hat{\mathbf{A}}^{(0)} = \mathbf{0}_{m\times n}$ then trivially 1c) is also true for any permutation at $t = 0$. Assuming that 1a)-c) are true at iteration $t-1$ consider iteration t. Firstly, if $\mathbf{R}^{(t-1)} = \mathbf{0}_{m\times n}$ then no partial supports

will be extracted and NEBF will terminate without making any further updates and therefore $t = t_f$. Therefore, as assumption 1c) is true at iteration $t-1$ by assumpption, then NEBF must have recovered $\mathbf{A}$ and $\mathbf{X}$ up to permutation and hence statement 2 must be true. Suppose then that $\mathbf{R}^{(t-1)} \neq \mathbf{0}_{m\times n}$, therefore by the assumption that 1a) and 1b) are true at iteration $t-1$, and by applying Lemmas 1, 2 and Corollary 2 then a nonzero number of singleton values and partial supports, each with support cardinality larger than $(1-2\epsilon)d$, are extracted and clustered without error on lines 1 and 2 (or 8 and 9 depending on the previous iterate) of PeelRes (Algorithm 4 Appendix B.1). It suffices to study two subcases, if $[h-1]\cap\mathcal{V}^{(t)} \neq \emptyset$ then at iteration t an iteration of the while loop of PeelRes occurs. If $[h-1]\cap\mathcal{V}^{(t)} = \emptyset$ then PeelRes terminates and NEBF attempts to update a column. The following argument shows that 1a)-c) hold true in either case.

If $[h-1]\cap\mathcal{V}^{(t)} \neq \emptyset$ then on line 5 of PeelRes $\mathbf{X}^{(t)}$ is computed without error by updating entries in the first $h-1$ rows of $\mathbf{X}^{(t-1)}$, corresponding to the complete columns in $\mathbf{A}^{(t)}$. Combined with the fact that $\mathbf{A}^{(t)}$ is not updated, this is highlight explicitly on line 5 of Algorithm 4 in Appendix B.1, and that 1c) is true at iteration $t-1$, then 1c) is also true at iteration t. Given 1c) is true then there exists a permutation $\mathbf{P}^{(t)}$ such that on line 6 of PeelRes

$$\begin{aligned}\mathbf{R}^{(t)} &= \mathbf{Y} - \hat{\mathbf{A}}^{(t)}([m],[h-1])\hat{\mathbf{X}}^{(t)}([h-1],[N])\\ &= \mathbf{AX} - \hat{\mathbf{A}}^{(t)}([m],[h-1])(\mathbf{P}^{(t)})^T\mathbf{P}^{(t)}\hat{\mathbf{X}}^{(t)}([h-1],[N])\\ &= \mathbf{AX} - \mathbf{A}([m],[h-1])\mathbf{P}^{(t)}\hat{\mathbf{X}}^{(t)}([h-1],[N])\\ &= \mathbf{A}\left(\mathbf{X} - \mathbf{P}^{(t)}\hat{\mathbf{X}^{(t)}}\right).\end{aligned}$$

Here the equality on line 3 follows due to 1c) being true at iteration t and the first $h-1$ columns of $\hat{\mathbf{A}}^{(t)}$ are complete. The final equality follows from rows h to n of $\mathbf{X}^{(t)}$ being zero. Again since 1c) is true at iteration t then as $\mathbf{X} \in \mathcal{X}_k^{m\times n}$ it follows that $\left(\mathbf{X} - \mathbf{P}^{(t)}\hat{\mathbf{X}^{(t)}}\right) \in \mathcal{X}_k^{m\times n}$. Therefore 1a) is also true at iteration t, and as a result, by Lemmas 1, 2 and Corollary 2, 1b) is also true at iteration t.

If $[h-1]\cap\mathcal{V}^{(t)} = \emptyset$ then PeelRes terminates and returns the clusters of partial supports extracted by NEBF. As it is assumed that $\mathbf{R}^{(t)} \neq \mathbf{0}_{m\times n}$ then the if statement condition on line 5 of Algorithm 2 will be satisfied. Now if on line 6 of Algorithm 2 $|supp\left(\sigma\left(\sum_{p\in\mathcal{C}_h^{(t)}}\mathbf{w}_p^{(t)}\right)\right)| < d$ then NEBF terminates. As a result nothing will have changed from iteration $t-1$ and therefore 1a)-c) hold true at iteration t also. If $|supp\left(\sigma\left(\sum_{p\in\mathcal{C}_h^{(t)}}\mathbf{w}_p^{(t)}\right)\right)| = d$ then as the singleton values and partial supports have been extracted and clustered without error, then there will exist an $l \in [n]$ such that $\hat{\mathbf{a}}_h^{(t)} = \mathbf{a}_l$. As a result, and also since $\mathbf{X}^{(t)}$ has not been updated from $\mathbf{X}^{(t-1)}$, then 1c) holds at iteration t. Given 1c) is true then there exists a permutation $\mathbf{P}^{(t)}$ such that on line 8 of NEBF

$$\begin{aligned}\mathbf{R}^{(t)} &= \mathbf{Y} - \hat{\mathbf{A}}^{(t)}([m],[h])\hat{\mathbf{X}}^{(t)}([h],[N])\\ &= \mathbf{AX} - \hat{\mathbf{A}}^{(t)}([m],[h])(\mathbf{P}^{(t)})^T\mathbf{P}^{(t)}\hat{\mathbf{X}}^{(t)}([h],[N])\\ &= \mathbf{AX} - \mathbf{A}([m],[h])\mathbf{P}^{(t)}\hat{\mathbf{X}}^{(t)}([h],[N])\\ &= \mathbf{A}\left(\mathbf{X} - \mathbf{P}^{(t)}\hat{\mathbf{X}^{(t)}}\right).\end{aligned}$$

Here again the equality on line 3 follows from 1c) being true at iteration t and also the first h columns of $\hat{\mathbf{A}}^{(t)}$ are complete. The fourth equality follows from rows $h+1$ to n of $\mathbf{X}^{(t)}$ being zero. Again as 1c) is true at iteration t and because $\mathbf{X} \in \mathcal{X}_k^{m\times n}$ then $\left(\mathbf{X} - \mathbf{P}^{(t)}\hat{\mathbf{X}^{(t)}}\right) \in \mathcal{X}_k^{m\times n}$. Therefore 1a) is also true at iteration t, and as a result, by Lemmas 1, 2 and Corollary 2, 1b) is also true at iteration t. Therefore by induction the result claimed is true. □

A.6 Proof of Lemma 5 - recovery of A is sufficient for factorization.

Proof. By definition the recovery of $\mathbf{A}$ up to permutation is a necessary condition for the successful factorization of $\mathbf{Y}$, it therefore suffices to prove, from statement 2 of Lemmas 3 and 4, that if at some iteration t' there exists a $\mathbf{P}^{(t')}$ such that $\hat{\mathbf{A}}^{(t')}(\mathbf{P}^{(t')})^T = \mathbf{A}$, then in some subsequent iteration $t_f \geq t'$ the residual $\mathbf{R}^{(t_f)} = \mathbf{0}_{m\times N}$. In what follows each algorithm is analyzed in turn under this assumption.

Considering first EBF, then since by Lemma 3 EBF only introduces correct entries, then $\mathbf{R}^{(t)}$ fails to converge to 0 iff at some iteration of the while loop, lines 2-10 of Algorithm 1, no partial supports can be extracted or clustered. However, from line 8 of Algorithm 2, at each iteration $t \geq t'$ the residual is updated as

$$\begin{aligned}\mathbf{R}^{(t)} &= \mathbf{Y} - \hat{\mathbf{A}}^{(t)}\hat{\mathbf{X}}^{(t)} \\ &= \mathbf{AX} - \hat{\mathbf{A}}^{(t)}(\mathbf{P}^{(t)})^T\mathbf{P}^{(t)}\hat{\mathbf{X}}^{(t)} \\ &= \mathbf{AX} - \mathbf{AP}^{(t)}\hat{\mathbf{X}}^{(t)} \\ &= \mathbf{A}\left(\mathbf{X} - \mathbf{P}^{(t)}\hat{\mathbf{X}^{(t)}}\right).\end{aligned}$$

As $\mathbf{X} \in \mathcal{X}_k^{n\times N}$ and Lemma 3 holds true, then $\left(\mathbf{X} - \mathbf{P}^{(t)}\hat{\mathbf{X}^{(t)}}\right) \in \mathcal{X}_k^{n\times N}$. Therefore by Lemma 2 each nonzero column of $\mathbf{R}^{(t)}$ provides some nonzero number of singleton values and associated partial supports each with support larger than $(1-2\epsilon)d$. Therefore, at each iteration for which $\mathbf{R}^{(t)} \neq \mathbf{0}_{m\times N}$ there are always singleton values and partial supports that can be extracted and clustered correctly. Since each column of the residual is a sum of at most k columns of $\mathbf{A}$ then it therefore follows that $t_f \leq t' + k$.

Now considering NEBF, since all columns of $\hat{\mathbf{A}}^{(t)}$ are complete and fixed for $t \geq t'$, then for all $t \geq t'$ it holds that $\mathcal{V}^{(t)} \cap [n] \neq \emptyset$. Therefore all subsequent iterations occur within PeelRes (Algorithm 4 in Appendix B.1). Since, by Lemma 4, NEBF only includes correct entries, then $\mathbf{R}^{(t)}$ fails to converge to 0 iff at some iteration of the while loop, lines 3-10 of PeelRes, no partial supports can be extracted and matched with a column of $\hat{\mathbf{A}}^{(t')}$. However, this contradicts statement 1b) of Lemma 4. As a result, and once again since each column of the residual is a sum of at most k columns of $\mathbf{A}$, it therefore follows that $t_f \leq t' + k$.

Finally, if $\mathcal{Y}^*_{NEBF} \subseteq \mathbb{R}^{m\times N}$ denotes the subset of real matrices which NEBF can successfully factorize, and $\mathcal{Y}^{\mathcal{A}}_{NEBF} \subseteq \mathbb{R}^{m\times N}$ denotes the subset of real matrices for which NEBF recovers the generating factor A up to permutation, then $\mathcal{Y}^*_{NEBF} = \mathcal{Y}^{\mathcal{A}}_{NEBF}$ and so for any measure $\mathbb{P}$ it holds that $\mathbb{P}(\mathcal{Y}^*_{NEBF}) = \mathbb{P}(\mathcal{Y}^*_{NEBF})$. The same clearly holds true also for EBF and so, for either algorithm, under the PSB data model the probability that Y is successfully factorized is equal to the probability that A is recovered up to permutation. □

A.7 Proof of Lemma 6 - uniqueness of factorization

Proof. If $\mathbf{R}^{(t_f)} = \mathbf{0}_{m\times n}$ then EBF and NEBF successfully recover $\mathbf{A}$ and $\mathbf{X}$ up to permutation by Lemmas 3 and 4 respectively. To prove that this factorization is unique up to permutation suppose that $\mathbf{Y} = \mathbf{AX} = \mathbf{BZ}$, where $\mathbf{B} \in \mathcal{E}^{m\times n}_{k,\epsilon,d}$ with $\epsilon \leq 1/6$ and $\mathbf{Z} \in \mathcal{X}_k^{n\times N}$. Since, again by Lemmas 3 and 4, EBF and NEBF only identify and include correct partial supports and cluster the partial supports correctly, then for any partial support $\mathbf{w}_p^{(t)}$, $p \in c(t)$, extracted at any iteration $t \leq t_f$, there exists $l, h \in [n]$ such that $supp(\mathbf{w}_p^{(t)}) \subseteq supp(\mathbf{a}_l)$ and $supp(\mathbf{w}_p^{(t)}) \subseteq supp(\mathbf{b}_h)$. Consider then the set of partial supports extracted from $\mathbf{Y}$ that belong to $\mathbf{a}_l$, these all overlap with one another by at least $(1-4\epsilon)d$ elements and hence must also overlap $\mathbf{b}_h$ by at least $(1-4\epsilon)d$ elements. As a result, the set of partial supports belonging to $\mathbf{a}_l$ must also all belong to $\mathbf{b}_h$, and by the same

logic any partial support belonging to $\mathbf{b}_h$ must also belong to $\mathbf{a}_l$. Therefore $\mathbf{a}_l$ and $\mathbf{b}_h$ are formed from the same partial supports and so $\mathbf{a}_l = \mathbf{b}_h$. Applying the same reasoning to the other columns of $\mathbf{A}$ then clearly there exists a permutation matrix $\mathbf{P}$ such that $\mathbf{B} = \mathbf{A}\mathbf{P}^T$. By Lemma 5 it holds that recovery of $\mathbf{A}$ up to permutation is sufficient to recover $\mathbf{X}$ up to permutation. As a result, if $\mathbf{Y} = \mathbf{A}\mathbf{X} = \mathbf{B}\mathbf{Z}$ then there exists a $\mathbf{P}$ such that $\mathbf{B} = \mathbf{A}\mathbf{P}^T$ and $\mathbf{Z} = \mathbf{P}\mathbf{X}$ and so the factorization is unique up to permutation as claimed. □

A.8 Proof of Lemma 7: NEBF successful implies EBF will be successful

Proof. From Lemma 5 it suffices to prove that whenever $\mathbf{A}$ is recovered by NEBF then it is also recovered by EBF. This can be proven via induction, proving that every any iteration t any partial support extracted by NEBF is also extracted by EBF and hence any column recovered by NEBF by iterate t is also recovered by EBF by iterate t. Starting with the base case at $t = 1$ then neither algorithm has recovered any columns of $\mathbf{A}$ and for both algorithms the residual is $\mathbf{Y}$, therefore they extract exactly the same singleton values and hence partial supports. As a result, if NEBF is able to reconstruct a column at $t = 1$ then so will EBF. Now assume that at iteration t any partial support extracted by NEBF has also been extracted by EBF, and likewise any column recovered by NEBF has also been recovered by EBF. Consider iteration $t + 1$, by the assumption on the tth iterate, anything removed from the residual by NEBF will have also been removed by EBF. Therefore any partial support extracted at the $t + 1$th iterate by NEBF will also be extracted by EBF. As a result if NEBF is able to recover another column of $\mathbf{A}$ using these partial supports then so will EBF. Therefore by induction if NEBF is successful in recovering $\mathbf{A}$ by some iterate t then so will EBF. If $\mathcal{Y}^*_{NEBF} \subset \mathbb{R}^{m\times N}$ denotes the subset of real matrices for which NEBF is successful and $\mathcal{Y}^*_{NEBF} \subset \mathbb{R}^{m\times N}$ denotes the subset of real matrices for which EBF is successful, it therefore follows that $\mathcal{Y}^*_{NEBF} \subseteq \mathcal{Y}^*_{EBF}$ and hence for any measure $\mathbb{P}$ it holds that $\mathbb{P}(\mathcal{Y}^*_{EBF}) \geq \mathbb{P}(\mathcal{Y}^*_{NEBF})$. Therefore, under the PSB data model if NEBF is successful with probability at least $1 - \delta$ then so is EBF. □

A.9 Proof of Lemma 8 - Column recovery from L partial supports

Proof. To quickly recap the construction in Definition 1, the support of each column of X is independent and identically distributed, chosen uniformly at random across all possible supports of cardinality k. The support of each column of A is also drawn uniform at random across all possible supports of cardinality d, furthermore each column of A is dependent on at most $\frac{m}{d} - 1$ other columns and independent of all others. Note also that each column of A has d non-zeros per column and at most $\lceil \frac{nd}{m} \rceil$ non-zeros per row. In what follows and for clarity the index of each partial support will correspond to the column of Y from which it originates, for example W_i was extracted from Y_i (or more generally the residual) and the corresponding column of X is X_i. Note that each column of Y can only provide a maximum of one partial support per column of A, therefore this indexing choice does not result in any ambiguity. First note that

$$\begin{aligned}
\mathbb{P}(\hat{A}_l \neq A_l) &= \mathbb{P}\left(\left(\bigcup_{i=1}^{L} supp(W_i)\right) \neq supp(A_l)\right) \\
&= \mathbb{P}\left(\bigcup_{j\in supp(A_l)} \left(j \notin \left(\bigcup_{i=1}^{L} supp(W_i)\right)\right)\right)
\end{aligned}$$

Applying the union bound and given that $|supp(\mathbf{a}_l)| = d$ then

$$\begin{aligned}\mathbb{P}(\hat{A}_l = A_l) &\leq d\mathbb{P}\left(j \notin \left(\bigcup_{i=1}^{L} supp(W_i)\right) \mid j \in supp(A_l)\right) \\ &= d\mathbb{P}\left(\bigcap_{i=1}^{L} (j \notin supp(W_i)) \mid j \in supp(A_l)\right).\end{aligned}$$

If $j \notin supp(W_i)$ then there exists an $r \in supp(X_i)\backslash\{l\}$ such that $j \in supp(A_r)$. To be clear, if $j \in supp(A_l)$ but $j \notin supp(W_i)$ then there must exist another column A_r in the submatrix $A_{supp(X_i)}$ which has a 1 in its jth entry, resulting in $Y_{j,i}$ not being a singleton value. Letting $\zeta := \lceil nd/m \rceil$ then

$$\begin{aligned}\mathbb{P}\left(\bigcap_{i=1}^{L} (j \notin supp(W_i)) \mid j \in supp(A_l)\right) &= \mathbb{P}\left(\bigcap_{i=1}^{L}\left(j \in \bigcup_{r\in supp(X_i)\backslash\{l\}} supp(A_r)\right) \mid j \in supp(A_l)\right) \\ &\leq \mathbb{P}\left(\bigcap_{i=1}^{L}\left(j \in \bigcup_{r\in supp(X_i)\backslash\{l\}} supp(A_r)\right)\right) \\ &\leq \mathbb{P}\left(\bigcap_{i=1}^{L}\left(j \in \bigcup_{r\in supp(X_i)\backslash\{l\}} supp(A_r)\right) \mid |supp(\tilde{A}_j)| = \zeta\right) \\ &= \prod_{i=1}^{L}\mathbb{P}\left(\left(j \in \bigcup_{r\in supp(X_i)\backslash\{l\}} supp(A_r)\right) \mid |supp(\tilde{A}_j)| = \zeta\right).\end{aligned}$$

The first equality follows from the discussion in the paragraph above. The inequality on the second line follows from the number of non-zeros per row of $\mathbf{A}$ being bounded by its construction, hence the events $j \in supp(A_l)$ and $j \in supp(A_r)$ for $r \neq l$ are negatively correlated. The inequality on the third line of the above follows by considering j to achieve the bound ζ of nonzeros per row. Lastly, the equality on the fourth line follows from this conditioning and the independence of the supports of the columns of X. Indeed, the event $j \in \bigcup_{r\in supp(X_i)\backslash\{l\}} supp(A_r)$ reduces to the event $(supp(X_i) \cap supp(\tilde{A}_j))\backslash\{l\} \neq \emptyset$. Since the supports of the X_i are sampled independently from one another, then by conditioning in this manner these events are independent. Note that the probability that none of the supports of the columns of A with index in $supp(X_i)\backslash\{l\}$ contain j is simply a count of the number of draws of $supp(X_i)\backslash\{l\}$ that contain none of the column indices in $supp(\tilde{A}_j)\backslash\{l\}$, divided by the total number of possible draws of $supp(X_i)\backslash\{l\}$. Therefore, assuming that $n \geq k(1-\frac{d}{m})^{-1}$, meaning $\alpha_1 \leq 1 - \frac{d}{m}$ as required in Theorem 1, it holds that

$$\mathbb{P}\left(j \in \left(\bigcup_{r\in supp(X_i)\backslash\{l\}} supp(A_r)\right) \mid |supp(\tilde{A}_j)| = \zeta\right) = 1 - \binom{n-\zeta}{k-1}\binom{n-1}{k-1}^{-1}$$

and so

$$\mathbb{P}(\hat{A}_l \neq A_l) \leq d\left(1 - \binom{n-\zeta}{k-1}\binom{n-1}{k-1}^{-1}\right)^{L}$$

as claimed. To simplify the final probability bound in Lemma 8, we bound the binomial coefficients as follows:

$$\begin{aligned}\binom{n-\zeta}{k-1}\binom{n-1}{k-1}^{-1} &= \frac{(n-\zeta)!(n-k)!}{(n-\zeta-k+1)!(n-1)!} \\ &= \frac{(n-\zeta)(n-\zeta-1)...(n-\zeta-k+2)}{(n-1)(n-2)...(n-k+1)} \\ &\leq \left(\frac{n-\zeta-k+2}{n-k+1}\right)^{k-1}.\end{aligned}$$

As a result

$$\mathbb{P}(\hat{A}_l \neq A_l) \leq d\left(1-\left(\frac{n-\zeta-k+2}{n-k+1}\right)^{k-1}\right)^L$$

as claimed. To quickly recap, with k and m are proportional to n, to be specific $k = \alpha_1 n + 1$ and $m = \alpha_2 n$ with $\alpha_1, \alpha_2 \in (0,1)$, $\alpha_1 \leq 1 - \frac{d}{m}$, then

$$\begin{aligned}\left(\frac{n-\zeta-k+2}{n-k+1}\right)^{k-1} &= \left(1-\frac{d}{\alpha_2(1-\alpha_1)n}\right)^{\alpha_1 n} \\ &= \left(1-\frac{b}{\alpha_1 n}\right)^{\alpha_1 n}\end{aligned}$$

where $b = \frac{d\alpha_1}{\alpha_2(1-\alpha_1)}$ is a constant. Taking the exponential of the logarithm of the upper bound on $\mathbb{P}(\hat{A}_l \neq A_l)$ gives

$$\begin{aligned}\mathbb{P}(\hat{A}_l \neq A_l) &\leq d\left(1-\left(1-\frac{b}{\alpha_1 n}\right)^{\alpha_1 n}\right)^{L(n)} \\ &= de^{-\tau(n)L(n)}\end{aligned}$$

where $\tau(n) := -\ln\left(1-\left(1-\frac{b}{\alpha_1 n}\right)^{\alpha_1 n}\right)$. For $n > \frac{d}{\alpha_2(1-\alpha_1)}$ (which implies $n > \frac{k-1}{m-d}$), $\tau(n)$ is a monotonically increasing, furthermore given that $\lim_{n\to\infty}\tau(n) = -\ln(1-e^{-b})$ then $\tau(n) = \mathcal{O}(1)$. □

A.10 Proof of Lemma 9 - Nonzeros per row in X.

Proof. In what follows $\mathbb{P}$ will refer to the probability measure induced by the random variable in question rather than a specific measure. Let $X(k,\beta,r^c)$ denote a random binary matrix generated by concatenating i.i.d. column vectors, with support drawn uniform at random from all possible $\binom{n}{k}$ possible supports, until there are at least β non-zeros per row. Here r^c denotes *without replacement* in regard to how the support of an individual column is chosen - in contrast let $X(k,\beta,r)$ denote a random binary matrix generated by concatenating i.i.d. column vectors *with replacement*, by which it is meant that the support of each column is generated by taking k i.i.d samples uniform from $[n]$ with replacement, until there are at least β non-zeros per row. Let $\eta[X(k,\beta,r^c)]$ count the number of columns of $X(k,\beta,r^c)$ - here square brackets are adopted purely for typographical clarity. The supports of the random matrix X in Definition 1 are drawn in the same manner as those of the random matrix $X(k,\beta,r^c)$ - the difference between them being that the number of columns N of X is fixed in advance instead of, as is the case for $X(k,\beta,r^c)$, drawing enough columns until there are at least β non-zeros per row. In the latter then the number of columns is a random variable. Clearly then

$$\mathbb{P}\left(\text{“X has at least } \beta \text{ non-zeros per row”}\right) = \mathbb{P}\left(\eta[X(k,\beta,r^c)] \leq N\right).$$

The approach taken to prove this lemma here then is to lower bound $\mathbb{P}\left(\eta[X(k,\beta,r^c)] \leq N\right)$. First note that this problem can be interpreted as a generalization of the coupon collector and dixie cup problems [18][30][20], these correspond to analyzing the expectation of $\eta[X(1,1,r^c)]$ and $\eta[X(\beta,1,r^c)]$ respectively. For context, in the classic coupon collector problem, objects are drawn one at a time from a set of size n, typically uniformly at random with replacement until each object has been seen once. Note that the r^c in our notation means sampling without replacement *within* the same column. The generalization studied here is equivalent to analyzing the number of draws of k objects, uniform at random with replacement, until each object has been picked at least β times with high probability. The distribution of this random variable is challenging to analyze directly, so the strategy adopted here will be to instead bound it indirectly via the classic coupon collector distribution, for which Chernoff style bounds can easily be derived.

Using the notation described above, let $\left(X^{(i)}(k,1,r^c)\right)_{i=1}^{\beta}$, be mutually independent and identically distributed random matrices and define

$$\tilde{X}(k,\beta,r^c) := [X^{(1)}(k,1,r^c),\ X^{(2)}(k,1,r^c)...\ X^{(\beta)}(k,1,r^c)].$$

Since each of the submatrices $X^{(i)}(k,1,r^c)$ may contain rows with more than 1 nonzero in them then clearly it holds that

$$\mathbb{P}(\eta[X(k,\beta,r^c)] \leq N) \geq \mathbb{P}(\eta[\tilde{X}(k,\beta,r^c)] \leq N).$$

Observe that

$$\bigcap_{i=1}^{\beta}\{\eta[X^{(i)}(k,1,r^c)] \leq \frac{N}{\beta}\} \implies \{\eta[\tilde{X}(k,\beta,r^c)] \leq N\},$$

$$\{\eta[\tilde{X}(k,\beta,r^c)] \leq N\} \not\implies \bigcap_{i=1}^{\beta}\{\eta[X^{(i)}(k,1,r^c)] \leq \frac{N}{\beta}\},$$

therefore $\bigcap_{i=1}^{\beta}\{\eta[X^{(i)}(k,1,r^c)] \leq \frac{N}{\beta}\} \subseteq \{\eta[\tilde{X}(k,\beta,r^c)] \leq N\}$. As a result

$$\begin{aligned}
\mathbb{P}(\eta[X(k,\beta,r^c)] \leq N) &\geq \mathbb{P}(\eta[\tilde{X}(k,\beta,r^c)] \leq N) \\
&\geq \mathbb{P}\left(\bigcap_{i=1}^{\beta}\{\eta[X^{(i)}(k,1,r^c)] \leq \frac{N}{\beta}\}\right) \\
&= \prod_{i=1}^{\beta}\mathbb{P}\left(\eta[X^{(i)}(k,1,r^c)] \leq \frac{N}{\beta}\right) \\
&= \left[\mathbb{P}\left(\eta[X(k,1,r^c)] \leq \frac{N}{\beta}\right)\right]^{\beta}
\end{aligned}$$

where the equalities on the third and fourth lines follow from the assumptions of mutual independence and identical distribution respectively. If the support of a given column is sampled with replacement then this will potentially decrease its cardinality (relative to if it were sampled without replacement), it follows that

$$\mathbb{P}\left(\eta[X(k,1,r^c)] \leq \frac{N}{\beta}\right) \geq \mathbb{P}\left(\eta[X(k,1,r)] \leq \frac{N}{\beta}\right).$$

What follows is based on the observation that sampling k elements of the support of a column *with replacement* is equivalent to unioning the supports of k i.i.d. vectors, each with only one element

in their support. To this end consider now the random matrix $X(1,1)$ (note that the r argument has been dropped since there is no difference between sampling with or without replacement when there is only one nonzero), the columns of this matrix are i.i.d. random vectors of dimension n with a single nonzero whose location is drawn uniformly at random. Furthermore, sufficiently many of these columns are drawn so as to ensure that $X(1,1)$ has at least one nonzero per row. Denote the ith column of $X(1,1)$ as Z_i. Now let X' be a random binary matrix which is a deterministic function of $X(1,1)$, to be specific let $supp(X'_l) := \bigcup_{i=(l-1)k+1}^{lk} supp(Z_i)$ for all $l \in [N]$. Note that if $\eta[X(1,1)]$ is not a multiple of k then additional columns can be drawn, each mutually independent with one nonzero whose location is drawn uniform at random, so that the last column of X' is still the union of k such columns. As a result $\eta[X'] = \lceil \eta[X(1,1)]/k \rceil$. Recalling that sampling k elements of the support of a column *with replacement* is equivalent to unioning the supports of k i.i.d. vectors, each with only one element in their support, then X' has the same distribution as $X(k,1,r)$. Therefore, given that for any $x \in \mathbb{R}$ it holds that $\lceil x \rceil \leq x+1$, then

$$\begin{aligned}
\mathbb{P}\left(\eta[X(k,1,r^c)] \leq \frac{N}{\beta}\right) &\geq \mathbb{P}\left(\eta[X(k,1,r)] \leq \frac{N}{\beta}\right) \\
&= \mathbb{P}\left(\eta[X'] \leq \frac{N}{\beta}\right) \\
&= \mathbb{P}\left(\lceil \eta[X(1,1)]/k \rceil \leq \frac{N}{\beta}\right) \\
&\geq \mathbb{P}\left(\eta[X(1,1)] \leq k\left(\frac{N}{\beta}-1\right)\right).
\end{aligned}$$

It should be clear that $\eta[X(1,1)]$ is equivalent to the number of draws required to see each coupon at least once in the classic coupon collector problem, for which the following bound is well known bound (see e.g., [16]),

$$\mathbb{P}(\eta[X(1,1)] > n\ln(n) + tn) < e^{-t}.$$

Letting $k\left(\frac{N}{\beta}-1\right) = n\ln(n)+tn$ then rearranging gives $t = \frac{k(N-\beta)-\beta n\ln(n)}{\beta n}$. Assuming $n > 1$, then for some $\mu > 1$ if $N = \beta\left(\mu\frac{n}{k}\ln(n)+1\right)$ then $t = (\mu-1)\ln(n)$ and as a result

$$\begin{aligned}
\mathbb{P}\left(\eta[X(1,1)] \leq k\left(\frac{N}{\beta}-1\right)\right) &= 1 - \mathbb{P}\left(\eta[X(1,1)] > k\left(\frac{N}{\beta}-1\right)\right) \\
&> 1 - e^{-(\mu-1)\ln(n)} \\
&= 1 - n^{-(\mu-1)}.
\end{aligned}$$

Hence, if X has $N \geq \beta\left(\mu\frac{n}{k}\ln(n)+1\right)$ columns whose supports are sampled as described in Definition 1, then

$$\mathbb{P}\left(\text{“X has at least } \beta \text{ non-zeros per row”}\right) \geq \left(1-n^{-(\mu-1)}\right)^{\beta}$$

as claimed. □

A.11 Proof of Corollary 3 - Guarantees for ℓ_0-EBF

Clearly, if Lemma 3 applies to ℓ_0-EBF, meaning that the ℓ_0-decode introduces only correct entries into $\hat{\mathbf{X}}^{(t)}$ (note ℓ_0-decode does not alter $\hat{\mathbf{A}}^{(t)}$), then the performance of ℓ_0-EBF is lower bounded by that of EBF. If this is the case then the machinery developed in Section 3 and used to prove Theorem 1 for EBF can be used in exactly the same manner to prove the same result for ℓ_0-EBF.

It suffices then to prove the analogue of Lemma 3. The proof of this follows exactly as in Appendix A.4, but with the added condition that at each step of the induction it must hold that the ℓ_0-decode subroutine does not introduce any errors into $\hat{\mathbf{X}}^{(t)}$. For any iteration $t \leq t_f$, assuming that ℓ_0-decode does not introduce any erroneous errors at the previous iterate (note the following argument can be applied to the base case), then the full column residual is of the form

$$\mathbf{R}' = \sum_{l \in \bar{\mathcal{H}}^{(t)}} \mathbf{a}_l \tilde{\mathbf{x}}_l + \sum_{l \in \mathcal{H}^{(t)}} \mathbf{a}_l (\tilde{\mathbf{x}}_l - \tilde{\hat{\mathbf{x}}}^{(t)}_{\pi_l})$$

where π_l is the column of $\hat{\mathbf{A}}^{(t)}$ corresponding to the lth column of $\mathbf{A}$. Therefore the full column residual is of the form $\mathbf{R}' = \mathbf{A}\mathbf{Z}$ where $\mathbf{A} \in \mathcal{E}^{m \times n}_{k,\epsilon,d}$ and $\mathbf{Z}$ is equal to $\mathbf{X}$ with certain nonzero entries set to 0. Therefore, since any subset of entries of a dissociated column is also dissociated (see Definition 2), then $\mathbf{Z} \in \mathcal{X}^{n \times N}_k$. Now suppose, with $\alpha \geq (1-2\epsilon)d$, that ℓ_0-decode removes $\omega \mathbf{a}_l$ from a column $\mathbf{r}'_i$ and sets $\hat{x}^{(t)}_{\pi_l,i} \leftarrow \hat{x}^{(t)}_{\pi_l,i} + \omega$. Therefore ω must appear in $\mathbf{r}'_i$ at least $(1-2\epsilon)d$ times and therefore by Lemma 1 must be a singleton value. Furthermore, its associated partial support must also overlap $\mathbf{a}_l$ by at least $(1-2\epsilon)d$, therefore by Corollary 2 this partial support must originate from $\mathbf{a}_l$ and so $x_{l,i} = \omega$. Since this must be the first iteration for which this partial support has been identified (since otherwise it would have already been removed), then the update at this iteration is simply $\hat{x}^{(t)}_{\pi_l,i} \leftarrow \omega$. Furthermore, this entry will not be updated again since $\epsilon \leq 1/6$ implies there are fewer than $(1-2\epsilon)d$ row locations in which $x_{l,i}$ contributes to the entry value. As a result $\hat{x}^{(t)}_{\pi_l,i} = 0$ or $\hat{x}^{(t)}_{\pi_l,i} = x_{l,i}$ for all $t \leq t_f$ and so ℓ_0-decode does not introduce any erroneous nonzeros. Therefore Lemma 3 also applies to ℓ_0-EBF from which it follows, from the same argument provided in Section 3, that Theorem 1 also applies to ℓ_0-EBF.

B Algorithm subroutines

B.1 PeelRes - line 3 of Algorithm 2

Algorithm 4 PeelRes$(t, h, \mathbf{R}^{(t-1)}, \hat{\mathbf{X}}^{(t-1)}, \hat{\mathbf{A}}^{(t-1)})$

Extract set of singleton values $\mathcal{Q}^{(t)}$ and associated partial supports $\mathbf{W}^{(t)}$ from $\mathbf{R}^{(t-1)}$.
Cluster partial supports $\mathbf{w}^{(t)}_p$ $(p \in [c(t)])$ into sets $\mathcal{C}^{(t)}_{h'}$ $(h' \in [n])$, compute $\mathcal{V}^{(t)}$.
while $\mathcal{V}^{(t)} \cap [h] \neq \emptyset$ **do**
  Compute $\hat{\mathbf{X}}^{(t)}$: if $r^{(t-1)}_{j,i} \in \mathcal{Q}^{(t)}$ has partial support $\mathbf{w}^{(t)}_p \in \mathcal{C}^{(t)}_{h'}$ where $h' \leq h$, set $\hat{x}^{(t)}_{h',i} \leftarrow r^{(t-1)}_{j,i}$.
  Estimate of $\mathbf{A}$ is unchanged: $\hat{\mathbf{A}}^{(t)} \leftarrow \hat{\mathbf{A}}^{(t-1)}$
  Compute the residual: $\mathbf{R}^{(t)} \leftarrow \mathbf{Y} - \hat{\mathbf{A}}^{(t)}([m],[h])\hat{\mathbf{X}}^{(t)}([h],[N])$.
  Update iteration index: $t \leftarrow t+1$.
  Extract set of singleton values $\mathcal{Q}^{(t)}$ and associated partial supports $\mathbf{W}^{(t)}$ from $\mathbf{R}^{(t-1)}$.
  Cluster partial supports $\mathbf{w}^{(t)}_p$ $(p \in [c(t)])$ into sets $\mathcal{C}^{(t)}_{h'}$ $(h' \in [n])$, compute $\mathcal{V}^{(t)}$.
end while
Return $(t, \hat{\mathbf{X}}^{(t-1)}, \{\mathcal{C}^{(t)}_l\}_{l=1}^n, \mathbf{W}^{(t)})$

PeelRes, defined in Algorithm 4, iteratively removes the contributions of complete columns from the residual until no further partial supports matching the current set of complete columns can be found. To define PeelRes we need to introduce the notion of the *visibile set* $\mathcal{V}^{(t)} := \{h \in [n] : |\mathcal{C}^{(t)}_h| > 0\}$, which is the set of column indices of $\hat{\mathbf{A}}^{(t-1)}$ for which there exists at least one partial support

extracted from $\mathbf{R}^{(t-1)}$. On lines 1, 2, 8 and 9 singleton values and partial supports are extracted and clustered as they were for EBF. On line 4 observe that only entries of $\hat{\mathbf{X}}^{(t)}$ corresponding to complete columns in $\hat{\mathbf{A}}^{(t)}$ are updated. As is explicitly emphasized and stated on line 5, $\mathbf{A}^{(t)}$ is not updated at all during PeelRes. Additionally, the residual update on line 6 only involves the removal of contributions from complete columns of $\hat{\mathbf{A}}^{(t)}$. This subroutine terminates when no further partial supports can be identified that match with a complete column of $\hat{\mathbf{A}}^{(t)}$.

References

[1] E. Abbe, *Community detection and stochastic block models: Recent developments*, Journal of Machine Learning Research, 18 (2018), pp. 1–86.

[2] A. Agarwal, A. Anandkumar, and P. Netrapalli, *A clustering approach to learn sparsely-used overcomplete dictionaries*, IEEE Transcations on Information Theory, (2016).

[3] C. Aicher, A. Z. Jacobs, and A. Clauset, *Adapting the stochastic block model to edge-weighted networks*, ICML Workshop on Structured Learning, (2013).

[4] S. Arora, A. Bhaskara, R. Ge, and T. Ma, *More algorithms for provable dictionary learning*, CoRR, abs/1401.0579 (2014).

[5] ——, *Provable bounds for learning some deep representations*, in Proceedings of the 31st International Conference on Machine Learning, E. P. Xing and T. Jebara, eds., vol. 32 of Proceedings of Machine Learning Research, Bejing, China, 22–24 Jun 2014, PMLR, pp. 584–592.

[6] S. Arora, R. Ge, and A. Moitra, *New algorithms for learning incoherent and overcomplete dictionaries*, Journal of Machine Learning Research, 35 (2013).

[7] B. Bah and J. Tanner, *Vanishingly sparse matrices and expander graphs, with application to compressed sensing*, IEEE Transactions on Information Theory, 59 (2013), pp. 7491–7508.

[8] B. Bah and J. Tanner, *On the construction of sparse matrices from expander graphs*, Frontiers in Applied Mathematics and Statistics, 4 (2018), p. 39.

[9] B. Barak, J. A. Kelner, and D. Steurer, *Dictionary learning and tensor decomposition via the sum-of-squares method*, in Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing, STOC 15, New York, NY, USA, 2015, Association for Computing Machinery, p. 143151.

[10] R. Berinde, A. C. Gilbert, P. Indyk, H. Karloff, and M. J. Strauss, *Combining geometry and combinatorics: A unified approach to sparse signal recovery*, in 2008 46th Annual Allerton Conference on Communication, Control, and Computing, Sep. 2008, pp. 798–805.

[11] R. Berinde and P. Indyk, *Sequential sparse matching pursuit*, in Proceedings of the 47th Annual Allerton Conference on Communication, Control, and Computing, Allerton 09, IEEE Press, 2009, p. 3643.

[12] R. Berinde, P. Indyk, and M. Ruzic, *Practical near-optimal sparse recovery in the l1 norm*, in 2008 46th Annual Allerton Conference on Communication, Control, and Computing, 2008, pp. 198–205.

[13] E. Candes and T. Tao, *Decoding by linear programming*, Information Theory, IEEE Transactions on, 51 (2006), pp. 4203 – 4215.

[14] E. J. Candès, X. Li, Y. Ma, and J. Wright, *Robust principal component analysis?*, J. ACM, 58 (2011).

[15] M. Capalbo, O. Reingold, S. Vadhan, and A. Wigderson, *Randomness conductors and constant-degree lossless expanders*, in Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, ACM, 2002, pp. 659–668.

[16] B. Doerr, *Probabilistic Tools for the Analysis of Randomized Optimization Heuristics*, Springer International Publishing, Cham, 2020, pp. 1–87.

[17] D. L. Donoho, *Compressed sensing*, IEEE Transactions on Information Theory, 52 (2004), pp. 1289–1306.

[18] A. Doumas and V. Papanicolaou, *The coupon collectors problem revisited: generalizing the double dixie cup problem of newman and shepp*, ESAIM: Probability and Statistics, 20 (2016).

[19] E. Elhamifar and R. Vidal, *Sparse subspace clustering: Algorithm, theory, and applications*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 35 (2013), pp. 2765–2781.

[20] M. Ferrante and A. Tagliavini, *On the coupon-collector's problem with several parallel collections*, arXiv e-prints, (2016), p. arXiv:1609.04174.

[21] S. Jafarpour, W. Xu, B. Hassibi, and A. R. Calderbank, *Efficient and robust compressed sensing using optimized expander graphs*, IEEE Transactions on Information Theory, 55 (2009), pp. 4299–4308.

[22] I. T. Jolliffe and J. Cadima, *Principal component analysis: a review and recent developments*, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374 (2016), p. 20150202.

[23] T. Li and C.-c. Ding, *'Nonnegative Matrix Factorizations for Clustering: A Survey' in Aggarwal, Charu C. and Reddy, Chandan K. 'Data Clustering: Algorithms and Applications'*, ch. 7, pp. 149–176.

[24] A. Lubotzky, *Expander graphs in pure and applied mathematics*, Bulletin of the American Mathematical Society, 49 (2011).

[25] R. Mendoza-Smith and J. Tanner, *Expander 0-decoding*, Applied and Computational Harmonic Analysis, 45 (2018), pp. 642 – 667.

[26] R. Mendoza-Smith, J. W. Tanner, and F. Wechsung, *A robust parallel algorithm for combinatorial compressed sensing*, IEEE Transactions on Signal Processing, 66 (2018), pp. 2167–2177.

[27] L. T. Nguyen, J. Kim, and B. Shim, *Low-rank matrix completion: A contemporary survey*, IEEE Access, 7 (2019), pp. 94215–94237.

[28] M. Sipser and D. A. Spielman, *Expander codes*, IEEE Transactions on Information Theory, 42 (1996), pp. 1710–1722.

[29] D. Spielman, H. Wang, and J. Wright, *Exact recovery of sparsely-used dictionaries*, IJCAI International Joint Conference on Artificial Intelligence, 23 (2012).

[30] W. Stadje, *The collector's problem with group drawings*, Advances in Applied Probability, 22 (1990).

[31] J. Sun, Q. Qu, and J. Wright, *Complete dictionary recovery over the sphere I: Overview and the geometric picture*, IEEE Transactions on Information Theory, 63 (2017), pp. 853–884.

[32] ——, *Complete dictionary recovery over the sphere II: Recovery by riemannian trust-region method*, IEEE Transactions on Information Theory, 63 (2017), pp. 885–914.

[33] T. Tao and V. H. Vu, *Additive Combinatorics*, Cambridge Studies in Advanced Mathematics, Cambridge University Press, 2006.

[34] W. Xu and B. Hassibi, *Efficient compressive sensing with deterministic guarantees using expander graphs*, in 2007 IEEE Information Theory Workshop, 2007, pp. 414–419.

[35] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang, *A survey of sparse representation: Algorithms and applications*, IEEE Access, 3 (2015), pp. 490–530.