# Efficient Identification of Butterfly Sparse Matrix Factorizations

Léon Zheng, Elisa Riccietti, Rémi Gribonval

# Efficient Identification of Butterfly Sparse Matrix Factorizations*

Léon Zheng[‡†], Elisa Riccietti[†], and Rémi Gribonval[†]

**Abstract.** Fast transforms correspond to factorizations of the form $\mathbf{Z} = \mathbf{X}^{(1)} \dots \mathbf{X}^{(J)}$, where each factor $\mathbf{X}^{(\ell)}$ is sparse and possibly structured. This paper investigates *essential uniqueness* of such factorizations, i.e., uniqueness up to unavoidable scaling ambiguities. Our main contribution is to prove that any $N \times N$ matrix having the so-called *butterfly* structure admits an essentially unique factorization into $J$ butterfly factors (where $N = 2^J$), and that the factors can be recovered by a hierarchical factorization method, which consists in recursively factorizing the considered matrix into two factors. This hierarchical identifiability property relies on a simple identifiability condition in the two-layer and fixed-support setting. This approach contrasts with existing ones that fit the product of butterfly factors to a given matrix via gradient descent. The proposed method can be applied in particular to retrieve the factorization of the Hadamard or the discrete Fourier transform matrices of size $N = 2^J$. Computing such factorizations costs $\mathcal{O}(N^2)$, which is of the order of dense matrix-vector multiplication, while the obtained factorizations enable fast $\mathcal{O}(N \log N)$ matrix-vector multiplications and have the potential to be applied to compress deep neural networks.

**Key words.** Identifiability, matrix factorization, sparsity, hierarchical factorization, butterfly factorization

**AMS subject classifications.** 15A23, 65F50, 94A12

## 1. Introduction.

Sparse matrix factorization with $J \geq 2$ factors is the problem of approximating a given matrix $\mathbf{Z}$ by a product of $J$ sparse factors $\mathbf{X}^{(1)}\mathbf{X}^{(2)} \dots \mathbf{X}^{(J)}$. Such a factorization is desired to reduce time and memory complexity for numerical methods involving the linear operator associated to $\mathbf{Z}$, e.g., in large-scale linear inverse problems [16, 55, 34, 35]. It can also be applied for compressing deep neural networks [12, 13, 41, 11, 6]: indeed, dense weight matrices of very large size in modern neural network architectures, like in transformers [14, 15] or MLP-mixers [56], can be replaced by products of sparse and possibly structured matrices in order to reduce the number of parameters of the model and the number of FLOPs.

The generic sparse matrix factorization problem is formulated as follows:

$$\min_{\mathbf{X}^{(1)},\dots,\mathbf{X}^{(J)}} \|\mathbf{Z} - \mathbf{X}^{(1)}\mathbf{X}^{(2)} \dots \mathbf{X}^{(J)}\|_F, \quad \text{such that } \mathbf{X}^{(\ell)} \text{ is sparse for all } \ell = 1, \dots, J, \tag{1.1}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. There are typically two kinds of sparsity constraints:

1. *Classical sparsity constraints*: they are encoded by a family of *allowed* supports that force the factors to have some prescribed sparsity patterns, like $k$-sparsity by row and/or column (i.e., each factor has at most $k$ nonzero entries per row and/or column).
2. *Fixed-support constraints*: the supports (i.e., the set of indices corresponding to nonzero entries in a matrix) are known a priori and each factor $\mathbf{X}^{(\ell)}$ is constrained to have a support included in a given prescribed support $\mathbf{S}^{(\ell)}$, $\ell = 1, \dots, J$.

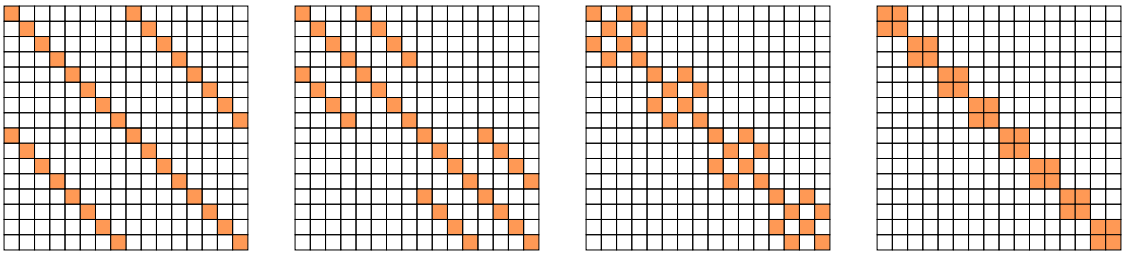
*Submitted to the editors on April 1st, 2022, revision sent on July 27th, 2022. This work is an extension of [32].

**Funding:** This project was supported in part by the AllegroAssai ANR project ANR-19-CHIA-0009 and the CIFRE fellowship N°2020/1643.

[†]Univ. de Lyon, ENS de Lyon, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France. (leon.zheng@ens-lyon.fr, elisa.riccietti@ens-lyon.fr, remi.gribonval@inria.fr). [‡]valeo.ai, Paris, France.

Problem (1.1) is known to be difficult in general, but even specific instances have been shown to be NP hard. On the one hand, problem (1.1) with classical sparsity constraints and $J = 2$ factors is the sparse coding problem [20, 16], when the dictionary $\mathbf{X}^{(1)}$ is known and the factor $\mathbf{X}^{(2)}$ is constrained to be $k$-sparse by column. This specific problem is shown to be NP-hard in [20, Theorem 2.17]. On the other hand, even in the case with $J = 2$ factors and fixed-support constraints, problem (1.1) has been recently shown to be NP-hard, without further assumptions on the prescribed fixed supports [31].

In this paper, we identify a particular choice of fixed-support constraint that makes problem (1.1) both well-posed and tractable, in the sense that problem (1.1) in the noiseless setting admits a unique[1] solution that can be computed with an algorithm of polynomial complexity. This fixed-support constraint is the so-called *butterfly* structure (see Definition 3.2 below). As illustrated in Figure 1 for the case with $J = 4$ factors, this fixed-support constraint enforces the factors $\mathbf{X}^{(\ell)} \in \mathbb{C}^{N\times N}$ ($N = 2^J$, $\ell = 1, \ldots, J$) to have at most two nonzero entries per row and per column, with a block diagonal structure for the factors $\mathbf{X}^{(2)}, \ldots, \mathbf{X}^{(J)}$.



(a) $\mathbf{S}_{\mathtt{bf}}^{(1)}$ (b) $\mathbf{S}_{\mathtt{bf}}^{(2)}$ (c) $\mathbf{S}_{\mathtt{bf}}^{(3)}$ (d) $\mathbf{S}_{\mathtt{bf}}^{(4)}$

Figure 1: Butterfly supports (see Definition 3.1 below) of size $N = 16$. Zero entries (white) *vs* entries that are allowed to be nonzero (colored).

The butterfly structure is interesting for machine learning applications mainly for two reasons. Firstly, finding the butterfly factors $\mathbf{X}^{(\ell)}$ ($\ell = 1, \ldots, J$) from the product $\mathbf{Z} := \mathbf{X}^{(1)} \ldots \mathbf{X}^{(J)}$ enables *fast* $\mathcal{O}(N \log N)$ matrix-vector multiplication by $\mathbf{Z}$. Indeed, there are $\mathcal{O}(\log N)$ factors in the butterfly factorization of a matrix of size $N \times N$, and each factor has $\mathcal{O}(N)$ nonzero entries [12]. Secondly, the butterfly structure is *expressive*: the composition of matrices with a butterfly structure can accurately approximate any given matrix [12, 13]. In practice, the potential of the butterfly structure in terms of expressiveness has been demonstrated in recent empirical works [12, 41, 13, 6, 11], where it is shown that replacing dense weight matrices in deep neural networks by butterfly-based structured matrices can reduce the model's complexity for faster training and inference time *without* harming its performance. The expressiveness property can be clearly illustrated by using the so-called kaleidoscope matrix representation [13]. Define $\mathcal{B}$ as the class of matrices that admit an exact butterfly factorization; $\mathcal{B}\mathcal{B}^*$ as the class of matrices of the form $\mathbf{M}^{(1)}\mathbf{M}^{(2)*}$, with $\mathbf{M}^{(1)}, \mathbf{M}^{(2)} \in \mathcal{B}$, and where $*$ denotes the conjugate transpose; $(\mathcal{B}\mathcal{B}^*)^W$ as the class of matrices of the form

[1] up to unavoidable scaling ambiguities.

$\mathbf{M}^{(1)} \dots \mathbf{M}^{(W)}$ with $\mathbf{M}^{(w)} \in \mathcal{B}\mathcal{B}^*$ for $w = 1, \dots, W$. Many structured linear maps common in machine learning are shown to be *tightly* captured by this so-called kaleidoscope hierarchy, in the sense that the matrices associated to these linear transforms are in $(\mathcal{B}\mathcal{B}^*)^W$ with *small* width $W$, resulting in a representation of the linear transform with nearly-optimal memory and time complexity [13]. For instance, the kaleidoscope hierarchy can express:

- **Classical fast linear transforms**: this includes the discrete Fourier transform (DFT), the discrete cosine transform, the discrete sine transform and the Hadamard transform. For instance, the Hadamard matrix $\mathbf{H}$ of dimension a power of two is in $\mathcal{B}$, and the DFT matrix $\mathbf{F}$ is in $(\mathcal{B}\mathcal{B}^*)^2$, because it can be written as $\mathbf{F} = \mathbf{B}\mathbf{P}$ where $\mathbf{B} \in \mathcal{B} \subset \mathcal{B}\mathcal{B}^*$ and $\mathbf{P}$ is the bit-reversal matrix, which is shown to be also in $\mathcal{B}\mathcal{B}^*$.
- **Circulant matrices**, which are associated to convolutions: any circulant matrix $\mathbf{C}$ can be expressed as $\mathbf{C} = \mathbf{F}^{-1}\mathbf{D}\mathbf{F}$ where $\mathbf{D}$ is a diagonal matrix by [51, Theorem 2.6.4], and $\mathbf{C} \in (\mathcal{B}\mathcal{B}^*)^4$, since the DFT matrix $\mathbf{F}$ as well as its inverse $\mathbf{F}^{-1} = \mathbf{F}^*$, and their scaled versions $\mathbf{D}\mathbf{F}$, $\mathbf{F}^{-1}\mathbf{D}$ are all in $(\mathcal{B}\mathcal{B}^*)^2$. In fact a tighter analysis can show that any circulant matrix $\mathbf{C}$ is in $\mathcal{B}\mathcal{B}^*$ [13, Lemma J.5].
- **Toeplitz matrices**: for any Toeplitz matrix $\mathbf{T}$ of size $N \times N$, there exists a circulant matrix of size $2N \times 2N$ such that $\mathbf{T} = \mathbf{R}\mathbf{C}\mathbf{R}^*$ with $\mathbf{R} := [\, \mathbf{I}_N \; \mathbf{0}_N \,] \in \mathbb{R}^{N \times 2N}$ a reduced identity matrix ($\mathbf{I}_N$ denotes the identity matrix of size $N$). Moreover, any matrix $\mathbf{M}$ of size $N \times N$ can be expressed as a product $\mathbf{M} := \mathbf{T}^{(1)}\mathbf{T}^{(2)} \dots \mathbf{T}^{(2N+5)}$ of (at most) $2N + 5$ Toeplitz matrices [61], hence can be written as $\mathbf{R}\mathbf{N}\mathbf{R}^*$ with $\mathbf{N} \in (\mathcal{B}\mathcal{B}^*)^{2N+5}$. Indeed, writing $\mathbf{T}^{(i)} = \mathbf{R}\mathbf{C}^{(i)}\mathbf{R}^*$ with $\mathbf{C}^{(i)} \in \mathcal{B}\mathcal{B}^*$ a circulant matrix of size $2N \times 2N$, we have $\mathbf{M} := \mathbf{R}\mathbf{N}\mathbf{R}^*$ with $\mathbf{N} := \mathbf{C}^{(1)}(\mathbf{R}^*\mathbf{R})\mathbf{C}^{(2)} \dots (\mathbf{R}^*\mathbf{R})\mathbf{C}^{(2N+5)} \in (\mathcal{B}\mathcal{B}^*)^{2N+5}$, as $(\mathbf{R}^*\mathbf{R})\mathbf{C}^{(i)} \in \mathcal{B}\mathcal{B}^*$ for each $i$ due to the fact that $(\mathbf{R}^*\mathbf{R})$ is simply a diagonal matrix.
- **Fastfood transform** [29]: the matrix $\mathbf{V} = \frac{1}{\sigma\sqrt{N}}\mathbf{S}\mathbf{H}\mathbf{G}\mathbf{P}\mathbf{H}\mathbf{B}$, with $\mathbf{S}, \mathbf{G}, \mathbf{B}$ some diagonal matrices, $\mathbf{P}$ a permutation matrix and $\mathbf{H}$ the Hadamard matrix, is used for fast approximation of the Gaussian kernel of scale $\sigma$. Since $\mathbf{H} \in \mathcal{B} \subseteq \mathcal{B}\mathcal{B}^*$ and $\mathbf{P} \in \mathcal{B}\mathcal{B}^*$, we get $\mathbf{V} \in (\mathcal{B}\mathcal{B}^*)^3$, but it is also possible to show that $\mathbf{V} \in (\mathcal{B}\mathcal{B}^*)^2$ [13, Lemma J.7].

The above properties of the butterfly structure motivate the interest in studying problem (1.1) with butterfly constraints, that is, the problem of finding an accurate approximation of any given matrix by a product of butterfly factors. Ideally, one would like to solve this problem efficiently, in order to design a proximal operator [52] that projects any matrix into the class of butterfly matrices $\mathcal{B}$, and use it to promote sparsity in various learning algorithms, like, e.g., on the learned dictionary matrix in dictionary learning [57], or on the learned weight matrices in neural network training. However, since problem (1.1) with the butterfly constraint is nonconvex, methods based on first-order optimization [12] lack theoretical guarantees for finding the optimal solution, and their performance is heavily dependent on initialization and hyperparameter tuning.

In constrast, this paper shows that every tuple of factors $(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(J)})$ satisfying the butterfly constraint can be reconstructed *with guarantee* by a *polynomial* algorithm from $\mathbf{Z} := \mathbf{X}^{(1)} \dots \mathbf{X}^{(J)}$ (see Algorithm 3.1). This algorithm is associated with our *identifiability* results (see Theorem 3.10), which claims that the factorization $\mathbf{Z} := \mathbf{X}^{(1)} \dots \mathbf{X}^{(J)}$ for any $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(J)}$ satisfying the butterfly constraint is *essentially unique* up to natural scaling ambiguities. The algorithm is based on a hierarchical approach [35] in which the target

matrix is iteratively factorized into two factors, until the desired number of sparse factors $J$ is obtained. These successive two-layer factorizations rely on a non-trivial application of the singular value decomposition (SVD) to compute best rank-one approximations of specific submatrices, instead of iterative gradient descent steps [31]. The total complexity of the hierarchical algorithm is only $\mathcal{O}(N^2)$ where $N$ is the size of $\mathbf{Z}$. This is remarkably of the same order of magnitude as the complexity of matrix-vector multiplication, and needs to be performed only once to enable $\mathcal{O}(N \log N)$ matrix-vector multiplications with the resulting factored representation of $\mathbf{Z}$. In other words, our work shows that problem (1.1) with fixed butterfly constraints on the factors can be solved efficiently by our algorithm in the noiseless setting, i.e., under the assumption that the target matrix $\mathbf{Z}$ admits an exact butterfly factorization. This naturally opens perspective for studying approximation guarantees in the noisy setting and in the quest for a proximal operator associated to the butterfly structure.

### 1.1. Related work.

*Butterfly structure.* The butterfly structure as defined in Definition 3.2 below and illustrated in Figure 1 is related to the divide-and-conquer structure of the fast Fourier transform. It was used in earlier works to design fast randomization methods for preconditioning in computational linear algebra [53]. It also offers an efficient parametrization of orthogonal matrices, which allows for fast approximation of Hessian matrices [47] and efficient trainable unitary matrices in recurrent neural networks [25]. In the context of kernel approximation via random features, the butterfly structure is used to generate random orthogonal matrices [21] in order to design quadrature rules for approximating the spherical part of the integral form of a kernel function [49, 7]. Recently, it has been shown that many common discrete operators associated to fast transforms can be approximated via gradient-descent methods by a data-sparse representation based on the butterfly structure [12, 13].

*Data-sparse representation of structured matrices.* Our work proposes an efficient hierarchical algorithm to recover the butterfly factors from their product, by computing successive SVDs on specific submatrices. This approach is similar to existing algorithms for approximating other types of structured matrices by a data-sparse representation that allows for fast matrix-vector multiplication. These structured matrices typically include: low-rank matrices [60, 24], $\mathcal{H}$-matrices [59], $\mathcal{H}^2$-matrices [23], hierarchically semi-separable matrices [46], and complementary low-rank matrices [48, 50, 3, 40]. The complementary low-rank structure, which is satisfied in many applications involving differential equations [3], is the closest to the butterfly structure considered in this paper. Similarly to what we propose it is associated to decomposition algorithms, called *butterfly algorithms* [48, 50, 3, 40], involving a recursive use of SVDs and certain types of binary trees. On the one hand, in our hierarchical factorization approach, the decomposition algorithm (Algorithm 3.1) is parameterized by a single but flexible "factor-bracketing" binary tree somehow describing the bracketing of factors associated to their product. On the other hand, butterfly algorithms [40] are applicable to factorize matrices satisfying the complementary low-rank model, which is parameterized by two "index-partitioning" binary trees. These two trees are associated to certain hierarchical partitioning of row (resp. column) indices. As further discussed in subsection 3.6, the butterfly algorithm [40] (using a standard SVD rather than a randomized one) can be interpreted as a specific instance of our hierarchical factorization approach with a "symmetric" factor-bracketing tree,

while our approach allows a much wider range of factor-bracketing trees. Vice-versa, we show that any matrix admitting a butterfly factorization in the sense of Definition 3.2 satisfies the complementary low-rank model with a particular set of index-partitioning binary trees, so none of the approaches is more general than the other. Making the best of both is an interesting direction for future work with a potential to further accelerate our approach.

*Identifiability in multilinear inverse problems.* Our paper also provides an analysis of identifiability, i.e., essential uniqueness, in butterfly sparse matrix factorization. Many identifiability results in multilinear inverse problems are derived from the *lifting* procedure [8, 9, 44, 39, 45]. This procedure was originally used in the PhaseLift method [36, 5, 4] to address the phase retrieval problem. Other bilinear inverse problems, like blind-deconvolution [1, 8, 2, 38, 37, 26, 39] or self-calibration [42], have also been addressed using the lifting procedure. A general framework to analyze identifiability for any bilinear inverse problem has been given in [8]. Following the work from [33, Chapter 7], our analysis of identifiability in the case with $J = 2$ factors is a specialization of this general lifting procedure to the matrix factorization problem *with support constraints*. Identifiability results for $J \geq 3$ are more challenging as the usual lifting procedure from bilinear inverse problems cannot be directly leveraged. This multi-layer setting requires extending this procedure using a so-called tensorial lifting [44, 45, 43]. However, due to this multi-layer structure, conditions obtained via tensorial lifting might be difficult to verify in practice, although stable recovery conditions for convolutional linear networks have been derived [45, 43]. In contrast, our work proves identifiability results in matrix factorization with $J \geq 3$ factors *without using tensorial lifting*, using a hierarchical approach that reduces the analysis with multiple factors to the case with only two factors.

*Empirical performance of our hierarchical algorithm.* The hierarchical factorization algorithm (Algorithm 3.1) has been studied empirically[2] in [32] where it was shown that the butterfly factors of the Hadamard and the DFT matrix can be recovered using Algorithm 3.1, with reduced computational time and increased accuracy compared to gradient-based methods like [12]. As a contribution, our work on identifiability gives theoretical foundations for Algorithm 3.1, since Theorem 3.10 shows exact recovery guarantees of the algorithm. We also justify the observed speed up of the factorization algorithm compared to gradient-based optimization methods, by showing that Algorithm 3.1 has a controlled time complexity of $\mathcal{O}(N^2)$.

**1.2. Summary.** The main contributions of this paper are the following ones:

1. We show in Theorem 3.10 that enforcing the butterfly structure on the $J$ sparse factors is sufficient to ensure a hierarchical identifiability property, meaning that we can recover (up to natural scaling ambiguities) sparse factors $(\mathbf{X}^{(\ell)})_{\ell=1}^{J}$ from $\mathbf{Z} := \mathbf{X}^{(1)} \dots \mathbf{X}^{(J)}$, using the hierarchical method described in Algorithm 3.1.
2. We show that this algorithm has a time complexity of only $\mathcal{O}(N^2)$ where $N$ is the size of $\mathbf{Z}$, which is of the order of a few dense matrix-vector multiplications, while the obtained factorizations enable fast $\mathcal{O}(N \log N)$ matrix-vector multiplications.
3. We illustrate this complexity by implementing[3] Algorithm 3.1 using *truncated* SVDs.

The paper is organized as follows: section 2 analyzes identifiability in the two-layer setting

[2]These two contributions have been produced in parallel, and the conference publication [32] refers to the current submission/preprint to support its theoretical claims.

[3]Implementation available in the FAµST 3.25 toolbox (https://faust.inria.fr/).

with fixed-support constraint; section 3 describes how the butterfly structure can ensure the hierarchical identifiability of the sparse factors from their product; section 4 presents some numerical experiments about the recovery of the sparse butterfly factors from their product; section 5 discusses perspectives of this work. An annex gathers technical proofs.

*Notations.* The set of integers $\{1, \dots, n\}$ is denoted $[n]$. The *support* of a matrix $\mathbf{M} \in \mathbb{C}^{m\times n}$ of size $m \times n$ is the set of indices $\text{supp}(\mathbf{M}) \subseteq [m] \times [n]$ of its nonzero entries. Depending on the context, such a matrix support can be seen either as a set of indices, or as a binary matrix (belonging to $\mathbb{B}^{m\times n} := \{0,1\}^{m\times n}$) with only nonzero entries for indices in this set. The *column support*, denoted $\text{colsupp}(\mathbf{M})$, is the subset of indices $i \in [n]$ such that the $i$-th column of $\mathbf{M}$, denoted $\mathbf{M}_i$, is nonzero. The entry of $\mathbf{M}$ indexed by $(k,l)$ is $\mathbf{M}_{k,l}$. The notation $\mathbf{M}^{(i)}$ denotes the $i$-th matrix in a collection. Given a support $\mathbf{S} \in \mathbb{B}^{m\times n}$, the restriction of $\mathbf{M}$ to $\mathbf{S}$ is the matrix $\mathbf{M} \odot \mathbf{S} \in \mathbb{C}^{m\times n}$ defined by the entries: $(\mathbf{M} \odot \mathbf{S})_{k,l} = \mathbf{M}_{k,l}$ if $(k,l) \in \mathbf{S}$, and zero otherwise. The identity matrix of size $N$ is denoted $\mathbf{I}_N$[4]. The Kronecker product [58] between $\mathbf{A}$ and $\mathbf{B}$ is written $\mathbf{A} \otimes \mathbf{B}$. We recall:

$$(\mathbf{A} \otimes \mathbf{C})(\mathbf{B} \otimes \mathbf{D}) = \mathbf{AB} \otimes \mathbf{CD}. \tag{1.2}$$

## 2. Identifiability in two-layer sparse matrix factorization.

In order to show hierarchical identifiability results, we first analyze identifiability in two-layer sparse matrix factorization. Given a matrix $\mathbf{Z} \in \mathbb{C}^{m\times n}$, and a subset of pairs of factors $\Sigma \subseteq \mathbb{C}^{m\times r} \times \mathbb{C}^{n\times r}$, the so-called *exact matrix factorization* problem with two factors of $\mathbf{Z}$ in $\Sigma$ is:

$$\text{find if possible } (\mathbf{X}, \mathbf{Y}) \in \Sigma \text{ such that } \mathbf{Z} = \mathbf{X}\mathbf{Y}^\intercal. \tag{2.1}$$

Uniqueness properties are studied in the exact setting, hence for the rest of the paper, we assume that $\mathbf{Z}$ admits such an exact factorization.

We are interested in the particular problem variation where the constraint set $\Sigma$ encodes some chosen sparsity patterns for the factorization. For a given binary matrix $\mathbf{S} \in \mathbb{B}^{m\times r}$ associated to a sparsity pattern, denote

$$\Sigma_{\mathbf{S}} := \{\mathbf{M} \in \mathbb{C}^{m\times r} \mid \text{supp}(\mathbf{M}) \subseteq \text{supp}(\mathbf{S})\}, \tag{2.2}$$

which is the set of matrices with a support included in $\mathbf{S}$. A pair of sparsity patterns is written $\mathrm{S} := (\mathbf{S}^L, \mathbf{S}^R)$, where $\mathbf{S}^L$ and $\mathbf{S}^R$ are the left and right sparsity patterns respectively, also referred to as a left and a right (allowed) support. Given any pair of allowed supports represented by binary matrices $\mathrm{S} := (\mathbf{S}^L, \mathbf{S}^R) \in \mathbb{B}^{m\times r} \times \mathbb{B}^{n\times r}$, the set

$$\Sigma_{\mathrm{S}} := \Sigma_{\mathbf{S}^L} \times \Sigma_{\mathbf{S}^R} \subseteq \mathbb{C}^{m\times r} \times \mathbb{C}^{n\times r} \tag{2.3}$$

is a linear subspace. Since the support of a matrix is unchanged under arbitrary rescaling of its columns, $\Sigma_{\mathrm{S}}$ is invariant by column scaling for any pair of supports S. Uniqueness of a solution to (2.1) with such sparsity constraints will always be considered up to unavoidable scaling ambiguities. Using the terminology from the tensor decomposition literature [28], such a uniqueness property will be referred to as *essential uniqueness*.

[4]It should be clear in the text when the subscript $N$ indicates the size $N$ of the matrix and not its $N$-th column.

Definition 2.1 (Essential uniqueness of a two-layer factorization in $\Sigma$). *Let $\Sigma$ be a set of pairs of factors, and $\mathbf{Z}$ be a matrix admitting a factorization $\mathbf{Z} := \mathbf{X}\mathbf{Y}^\intercal$ such that $(\mathbf{X}, \mathbf{Y}) \in \Sigma$. This factorization is* essentially unique *in $\Sigma$, if any solution $(\bar{\mathbf{X}}, \bar{\mathbf{Y}})$ to (2.1) with $\mathbf{Z}$ and $\Sigma$ is equivalent to $(\mathbf{X}, \mathbf{Y})$, written $(\bar{\mathbf{X}}, \bar{\mathbf{Y}}) \sim (\mathbf{X}, \mathbf{Y})$, in the sense that there is an invertible diagonal matrix $\mathbf{D}$ such that $(\bar{\mathbf{X}}, \bar{\mathbf{Y}}) = \left(\mathbf{X}\mathbf{D}, \mathbf{Y}\mathbf{D}^{-1}\right)$.*

*Remark* 2.2. Another definition of essential uniqueness [28] involves permutation ambiguity in addition to scaling ambiguity. Nevertheless, we only consider scaling ambiguity in our definition of essential uniqueness, because all the fixed-support constraints $\Sigma_{\mathrm{S}}$ considered in this paper remove this permutation ambiguity, as detailed in Remark 2.7 below.

For any set $\Sigma$ of pairs of factors, the set of all pairs $(\mathbf{X}, \mathbf{Y}) \in \Sigma$ such that the factorization $\mathbf{Z} := \mathbf{X}\mathbf{Y}^\intercal$ is essentially unique in $\Sigma$ is denoted $\mathcal{U}(\Sigma)$. In other words, we define:

$$\mathcal{U}(\Sigma) := \left\{(\mathbf{X}, \mathbf{Y}) \in \Sigma \mid \forall (\bar{\mathbf{X}}, \bar{\mathbf{Y}}) \in \Sigma,\ \bar{\mathbf{X}}\bar{\mathbf{Y}}^\intercal = \mathbf{X}\mathbf{Y}^\intercal \implies (\bar{\mathbf{X}}, \bar{\mathbf{Y}}) \sim (\mathbf{X}, \mathbf{Y})\right\}. \tag{2.4}$$

To characterize $\mathcal{U}(\Sigma_{\mathrm{S}})$ for $\Sigma_{\mathrm{S}}$ defined as in (2.3) with S any pair of supports, we first establish a non-degeneration property, i.e., a necessary condition for identifiability, involving the so-called *column support*. Define the set of pairs of factors with *identical*, resp. *maximal*, column supports in $\Sigma_{\mathrm{S}}$ as

$$\mathtt{IC}_{\mathrm{S}} := \{(\mathbf{X}, \mathbf{Y}) \in \Sigma_{\mathrm{S}} \mid \operatorname{colsupp}(\mathbf{X}) = \operatorname{colsupp}(\mathbf{Y})\}, \tag{2.5}$$

$$\mathtt{MC}_{\mathrm{S}} := \{(\mathbf{X}, \mathbf{Y}) \in \Sigma_{\mathrm{S}} \mid \operatorname{colsupp}(\mathbf{X}) = \operatorname{colsupp}(\mathbf{S}^L) \text{ and } \operatorname{colsupp}(\mathbf{Y}) = \operatorname{colsupp}(\mathbf{S}^R)\}. \tag{2.6}$$

Lemma 2.3. *For any pair of supports* S*, we have: $\mathcal{U}(\Sigma_{\mathrm{S}}) \subseteq \mathtt{IC}_{\mathrm{S}} \cap \mathtt{MC}_{\mathrm{S}}$.*

*Remark* 2.4. Consequently, if $\operatorname{colsupp}(\mathbf{S}^L) \neq \operatorname{colsupp}(\mathbf{S}^R)$, then the set $\mathcal{U}(\Sigma_{\mathrm{S}})$ is empty.

In other words, if the factorization $\mathbf{Z} := \mathbf{X}\mathbf{Y}^\intercal$ is essentially unique in $\Sigma_{\mathrm{S}}$, then the left and right supports have necessarily the same column support, and $\mathbf{X}$, $\mathbf{Y}$ do not have a zero column inside this column support. The proof is deferred to Appendix A.

As the product $\mathbf{X}\mathbf{Y}^\intercal$ is the sum of rank-one matrices $\sum_{i=1}^r \mathbf{X}_i \mathbf{Y}_i{}^\intercal$, the lifting procedure [8, 44] suggests to represent the pair $(\mathbf{X}, \mathbf{Y})$ by its $r$-tuple of so-called *rank-one contributions*

$$\varphi(\mathbf{X}, \mathbf{Y}) := (\mathbf{X}_i \mathbf{Y}_i{}^\intercal)_{i=1}^r \in (\mathbb{C}^{m \times n})^r. \tag{2.7}$$

Indeed, one can identify, up to scaling ambiguities, the columns $\mathbf{X}_i$, $\mathbf{Y}_i$ from their outer product $\boldsymbol{\mathcal{C}}^i := \mathbf{X}_i \mathbf{Y}_i{}^\intercal$ $(1 \leq i \leq r)$, *as long as* the rank-one contribution $\boldsymbol{\mathcal{C}}^i$ is not zero.

Lemma 2.5 (Reformulation of [33, Chapter 7, Lemma 1]). *Consider $\boldsymbol{\mathcal{C}}$ the outer product of two vectors $\mathbf{a}$, $\mathbf{b}$. If $\boldsymbol{\mathcal{C}} = \mathbf{0}$, then $\mathbf{a} = \mathbf{0}$ or $\mathbf{b} = \mathbf{0}$. If $\boldsymbol{\mathcal{C}} \neq \mathbf{0}$, then $\mathbf{a}$, $\mathbf{b}$ are nonzero, and for any $(\mathbf{a}', \mathbf{b}')$ such that $\mathbf{a}'\mathbf{b}'^\intercal = \boldsymbol{\mathcal{C}}$, there exists a scalar $\lambda \neq 0$ such that $\mathbf{a}' = \lambda \mathbf{a}$ and $\mathbf{b}' = \frac{1}{\lambda}\mathbf{b}$.*

With this lifting approach, each support constraint $\mathrm{S} = (\mathbf{S}^L, \mathbf{S}^R)$ is represented by the $r$-tuple of rank-one support constraints $\mathcal{S} = \varphi(\mathbf{S}^L, \mathbf{S}^R) = (\boldsymbol{\mathcal{S}}^i)_{i=1}^r$. Thus, if $(\mathbf{X}, \mathbf{Y}) \in \Sigma_{\mathrm{S}} \subseteq \mathbb{C}^{m \times r} \times \mathbb{C}^{n \times r}$, then the $r$-tuple of rank-one matrices $\varphi(\mathbf{X}, \mathbf{Y}) \in (\mathbb{C}^{m \times n})^r$ belongs to the set:

$$\Gamma_{\mathcal{S}} := \left\{(\boldsymbol{\mathcal{C}}^i)_{i=1}^r \mid \forall i \in [r],\ \operatorname{rank}(\boldsymbol{\mathcal{C}}^i) \leq 1,\ \operatorname{supp}(\boldsymbol{\mathcal{C}}^i) \subseteq \boldsymbol{\mathcal{S}}^i\right\} \subseteq (\mathbb{C}^{m \times n})^r. \tag{2.8}$$

As explained in the general framework of [8] established to analyze identifiability in bilinear inverse problems, the main advantage of this approach is to remove the inherent scaling

ambiguity, while preserving a one-to-one correspondence between a pair of factors $(\mathbf{X}, \mathbf{Y})$ and its rank-one contributions representation $\varphi(\mathbf{X}, \mathbf{Y})$. Our work specializes this general framework to matrix factorization problem with two factors and support constraints. Details about the lifting procedure for fixed-support two-layer matrix factorization are in Appendix B.

From this lifting procedure, we show that it is possible to derive a simple necessary and sufficient condition for identifiability. Despite its simplicity, this condition is the key to derive identifiability results in butterfly sparse matrix factorization using a hierarchical approach, see subsection 3.1 and Lemma 3.13. The proof is deferred to Appendix C.

Proposition 2.6. *Assuming* $\operatorname{colsupp}(\mathbf{S}^L) = \operatorname{colsupp}(\mathbf{S}^R)$ *(which is natural by Remark 2.4),* $\mathcal{U}(\Sigma_{\mathrm{S}}) = \mathtt{IC}_{\mathrm{S}} \cap \mathtt{MC}_{\mathrm{S}}$ *if, and only if, the tuple* $\varphi(\mathrm{S}) = (\boldsymbol{\mathcal{S}}^i)_{i=1}^r$ *has disjoint rank-one supports, i.e.,* $\operatorname{supp}(\boldsymbol{\mathcal{S}}^i) \cap \operatorname{supp}(\boldsymbol{\mathcal{S}}^j) = \emptyset$ *for every* $i \neq j$.

More conditions on fixed-support identifiability are given in [33, 62].

*Remark* 2.7. Following the discussion of Remark 2.2, the assumption that $\varphi(\mathrm{S})$ has disjoint rank-one supports is sufficient to remove any potential permutation ambiguity in the factorization of $\mathbf{Z} := \mathbf{X}\mathbf{Y}^\intercal$ in $\Sigma$ for any $(\mathbf{X}, \mathbf{Y}) \in \mathtt{IC}_{\mathrm{S}} \cap \mathtt{MC}_{\mathrm{S}}$, in the sense that: for any $(\mathbf{X}, \mathbf{Y}) \in \mathtt{IC}_{\mathrm{S}} \cap \mathtt{MC}_{\mathrm{S}}$, any invertible diagonal matrix $\mathbf{D}$ and any permutation matrix $\mathbf{P}$ such that $(\mathbf{XDP}, \mathbf{YD}^{-1}\mathbf{P}) \in \Sigma_{\mathrm{S}}$, the permutation matrix $\mathbf{P}$ is necessarily the identity matrix. This is true because otherwise, denoting $(\boldsymbol{\mathcal{S}}^i)_{i=1}^r := \varphi(\mathrm{S})$, $(\boldsymbol{\mathcal{C}}^i)_{i=1}^r := \varphi(\mathbf{X}, \mathbf{Y})$ and noticing that $\varphi(\mathbf{XDP}, \mathbf{YD}^{-1}\mathbf{P})$ is equal to $(\boldsymbol{\mathcal{C}}^i)_{i=1}^r$ up to a permutation on the index $i$, there would exist two indices $j \neq k$ such that $\operatorname{supp}(\boldsymbol{\mathcal{C}}^j) \subseteq \operatorname{supp}(\boldsymbol{\mathcal{S}}^j)$ and $\operatorname{supp}(\boldsymbol{\mathcal{C}}^j) \subseteq \operatorname{supp}(\boldsymbol{\mathcal{S}}^k)$ with $\operatorname{supp}(\boldsymbol{\mathcal{C}}^j) \neq \emptyset$ (because $(\mathbf{X}, \mathbf{Y}) \in \mathtt{IC}_{\mathrm{S}} \cap \mathtt{MC}_{\mathrm{S}}$), which would contradict the fact that $\operatorname{supp}(\boldsymbol{\mathcal{S}}^j) \cap \operatorname{supp}(\boldsymbol{\mathcal{S}}^k) = \emptyset$.

## 3. Hierarchical identifiability of butterfly factors.

The main contribution of the paper is to establish some identifiability results in the multi-layer sparse matrix factorization when the sparse factors are constrained to have the so-called *butterfly supports*.

### 3.1. Definition and properties of the butterfly structure.

We now formally introduce the butterfly structure and its important properties used to establish identifiability results.

Definition 3.1 (Butterfly supports). *The* butterfly supports *of size* $N = 2^J$ *are the* $J$*-tuple of supports* $\mathrm{S}_{\mathtt{bf}} := (\mathbf{S}_{\mathtt{bf}}^{(1)}, \ldots, \mathbf{S}_{\mathtt{bf}}^{(J)}) \in (\mathbb{B}^{N \times N})^J$ *defined by:*

$$\mathbf{S}_{\mathtt{bf}}^{(\ell)} := \mathbf{I}_{2^{\ell-1}} \otimes \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \otimes \mathbf{I}_{N/2^\ell}, \ 1 \leq \ell \leq J. \tag{3.1}$$

Figure 1 is an illustration of the butterfly supports. They are 2-regular [30], i.e., they have at most 2 nonzero entries per row and per column, and they are also block-diagonal, including the leftmost factor if we consider a single block that covers the matrix entirely.

Definition 3.2 (Butterfly structure). *We say that a matrix* $\mathbf{Z}$ *of size* $N = 2^J$ *admits a butterfly structure if it can be factorized into* $J$ *factors* $(\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(J)})$ *that have a support included in the butterfly supports* $\mathrm{S}_{\mathtt{bf}} := (\mathbf{S}_{\mathtt{bf}}^{(1)}, \ldots, \mathbf{S}_{\mathtt{bf}}^{(J)})$*, in the sense that:*

$$\mathbf{Z} = \mathbf{X}^{(1)} \ldots \mathbf{X}^{(J)}, \quad \textit{with} \quad (\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(J)}) \in \Sigma_{\mathbf{S}_{\mathtt{bf}}^{(1)}} \times \ldots \times \Sigma_{\mathbf{S}_{\mathtt{bf}}^{(J)}}.$$

*For the rest of the paper we write by slight abuse of notation:*

$$\Sigma_{\mathrm{S_{bf}}} := \Sigma_{\mathbf{S}_{\mathtt{bf}}^{(1)}} \times \ldots \times \Sigma_{\mathbf{S}_{\mathtt{bf}}^{(J)}}. \tag{3.2}$$

With such a structure, matrix-vector multiplication has complexity $\mathcal{O}(N \log N)$ [12]: there are $J = \log_2(N)$ butterfly factors and the complexity of matrix-vector multiplication by each butterfly factor $\mathbf{X}^{(\ell)}$ ($\ell \in [J]$) is $\mathcal{O}(N)$, since $\mathbf{X}^{(\ell)}$ has at most $2N$ nonzero entries. As explained in the introduction, the butterfly structure is involved in many matrices associated to fast linear transforms [12], such as the discrete Fourier transform (DFT) matrix.

*Example* 3.3 (Butterfly factorization of the DFT matrix [12]). Consider the DFT matrix of size $N \times N$ with $N = 2^J$ denoted as $\mathbf{DFT}_N$, defined by:

$$\mathbf{DFT}_N := (\omega_N{}^{(k-1)(l-1)})_{k,l\in[N]}, \quad \text{where } \omega_N := e^{-i\frac{2\pi}{N}}. \tag{3.3}$$

The butterfly factorization of $\mathbf{DFT}_N$ [12] relies on the recursive relation

$$\mathbf{DFT}_N = \mathbf{B}_N \begin{pmatrix} \mathbf{DFT}_{N/2} & 0 \\ 0 & \mathbf{DFT}_{N/2} \end{pmatrix} \mathbf{P}_N, \tag{3.4}$$

where $\mathbf{P}_N \in \mathbb{B}^{N\times N}$ is the permutation matrix that sorts the odd then the even indices[5], and

$$\mathbf{B}_N := \begin{pmatrix} \mathbf{I}_{N/2} & \mathbf{A}_{N/2} \\ \mathbf{I}_{N/2} & -\mathbf{A}_{N/2} \end{pmatrix}, \tag{3.5}$$

with $\mathbf{A}_{N/2}$ the diagonal matrix with diagonal entries $1, \omega_N, \omega_N^2, \ldots, \omega_N{}^{\frac{N}{2}-1}$. Applying recursively (3.4) to the block $\mathbf{DFT}_{N/2}$, we obtain the butterfly factorization of $\mathbf{DFT}_N$:

$$\begin{aligned} \mathbf{DFT}_N = \mathbf{F}^{(1)} \ldots \mathbf{F}^{(J)} \mathbf{R}_N, \quad \text{where } \mathbf{F}^{(\ell)} := \mathbf{I}_{2^{\ell-1}} \otimes \mathbf{B}_{N/2^{\ell-1}},\ \ell \in [J], \\ \text{and } \mathbf{R}_N := \mathbf{Q}^{(J)} \mathbf{Q}^{(J-1)} \ldots \mathbf{Q}^{(1)}, \quad \text{with } \mathbf{Q}^{(\ell)} := \mathbf{I}_{2^{\ell-1}} \otimes \mathbf{P}_{N/2^{\ell-1}}. \end{aligned} \tag{3.6}$$

Matrix $\mathbf{R}_N := \mathbf{Q}^{(J)} \mathbf{Q}^{(J-1)} \ldots \mathbf{Q}^{(1)}$ is the so-called *bit-reversal* permutation matrix.

For any $(\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(J)}) \in \Sigma_{\mathrm{S_{bf}}}$, the partial product of any consecutive factors $\mathbf{X}^{(p)} \ldots \mathbf{X}^{(q)}$ ($1 \le p \le q \le J$) is shown to have a very precise structure encoded by the support $\mathbf{S}_{\mathtt{bf}}^{(p:q)}$ as detailed in the following lemma (proved in Appendix D). Figure 2 illustrates this structure on some examples of size $N = 16$. In particular this structure is involved in the hierarchical matrix factorization method, as explained below in subsection 3.2.

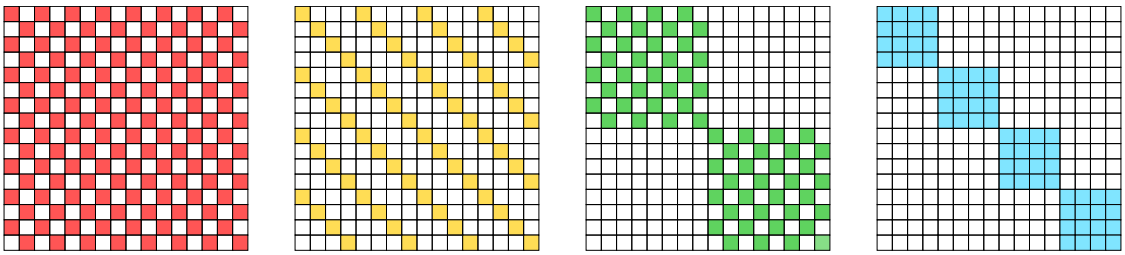
Lemma 3.4. *Let* $\mathrm{S_{bf}} := (\mathbf{S}_{\mathtt{bf}}^{(1)}, \ldots, \mathbf{S}_{\mathtt{bf}}^{(J)})$ *be the butterfly supports of size* $N = 2^J$. *Then, for any tuple* $(\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(J)}) \in \Sigma_{\mathrm{S_{bf}}}$, *for any* $1 \le p \le q \le J$: $\mathrm{supp}(\mathbf{X}^{(p)} \ldots \mathbf{X}^{(q)}) \subseteq \mathbf{S}_{\mathtt{bf}}^{(p:q)}$, *where*

$$\mathbf{S}_{\mathtt{bf}}^{(p:q)} := \mathbf{I}_{2^{p-1}} \otimes \mathbf{V}^{(p,q)} \in \mathbb{B}^{N\times N}, \text{ and} \tag{3.7}$$

$$\mathbf{V}^{(p,q)} := \mathbf{U}_{2^{q-p+1}} \otimes \mathbf{I}_{N/2^q} \in \mathbb{B}^{\frac{N}{2^{p-1}} \times \frac{N}{2^{p-1}}}, \tag{3.8}$$

*denoting, for any* $n$, $\mathbf{U}_n \in \mathbb{B}^{n\times n}$ *as the binary matrix full of ones.*

[5]For instance, for $N = 4$, it permutes $(1, 2, 3, 4)$ to $(1, 3, 2, 4)$.



(a) $\mathbf{S}_{\mathtt{bf}}^{(1:3)}$ (b) $\mathbf{S}_{\mathtt{bf}}^{(1:2)}$ (c) $\mathbf{S}_{\mathtt{bf}}^{(2:3)}$ (d) $\mathbf{S}_{\mathtt{bf}}^{(3:4)}$

Figure 2: Examples of supports $\mathbf{S}_{\mathtt{bf}}^{(p:q)}$ defined by (3.7) ($1 \le p \le q \le 4$) of size $N \times N$ with $N = 16$. Nonzero entries are in color, and zero entries are in white.

*Remark* 3.5. We have $\mathbf{S}_{\mathtt{bf}}^{(\ell:\ell)} = \mathbf{S}_{\mathtt{bf}}^{(\ell)}$ for $1 \le \ell \le J$, and viewing matrix supports as binary matrices, one can verify that $\mathbf{S}_{\mathtt{bf}}^{(p:q)} = \mathbf{S}_{\mathtt{bf}}^{(p)} \dots \mathbf{S}_{\mathtt{bf}}^{(q)}$ for $1 \le p \le q \le J$. Also, by (3.8), $\mathbf{V}^{(p,q)}$ is symmetric, i.e., $\mathbf{V}^{(p,q)} = (\mathbf{V}^{(p,q)})^\intercal$. By (3.7), $\mathbf{S}_{\mathtt{bf}}^{(p:q)}$ is block-diagonal with blocks $\mathbf{V}^{(p,q)}$, so $\mathbf{S}_{\mathtt{bf}}^{(p:q)}$ is also symmetric. Finally, $\mathbf{S}_{\mathtt{bf}}^{(p:q)}$ and $\mathbf{V}^{(p,q)}$ are both $2^{q-p+1}$-sparse by column.

**3.2. Hierarchical matrix factorization method.** The so-called *butterfly sparse matrix factorization* problem is the following special instance of (1.1):

$$\min_{\mathbf{X}^{(1)},\dots,\mathbf{X}^{(J)}} \|\mathbf{Z} - \mathbf{X}^{(1)} \dots \mathbf{X}^{(J)}\|_F, \quad \text{such that } (\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(J)}) \in \Sigma_{\mathrm{S}_{\mathtt{bf}}}. \tag{3.9}$$

*Remark* 3.6. As shown in [31, Remark A.1], there exists a support constraint $\mathrm{S} = (\mathbf{S}^L, \mathbf{S}^R)$ and a matrix $\mathbf{Z}$ such that: (a) $\mathbf{Z}$ *cannot be written exactly* as $\mathbf{Z} = \mathbf{X}\mathbf{Y}^\intercal$ for any $(\mathbf{X}, \mathbf{Y}) \in \Sigma_{\mathrm{S}}$; (b) $\mathbf{Z}$ can be approximated arbitrarily well by such a product, i.e., $0 = \inf_{(\mathbf{X},\mathbf{Y})\in\Sigma_{\mathrm{S}}} \|\mathbf{Z} - \mathbf{X}\mathbf{Y}^\intercal\|_F$. This corresponds to a lack of closure of the set $\{\mathbf{X}\mathbf{Y}^\intercal : (\mathbf{X}, \mathbf{Y}) \in \Sigma_{\mathrm{S}}\}$. Fortunately this pathological behavior does not happen here since $\mathbf{Z}$ is assumed to admit an exact factorization. It is left for future work whether in the case of butterfly supports, problem (3.9) always admits a minimizer even when $\mathbf{Z}$ does not admit an exact butterfly factorization.

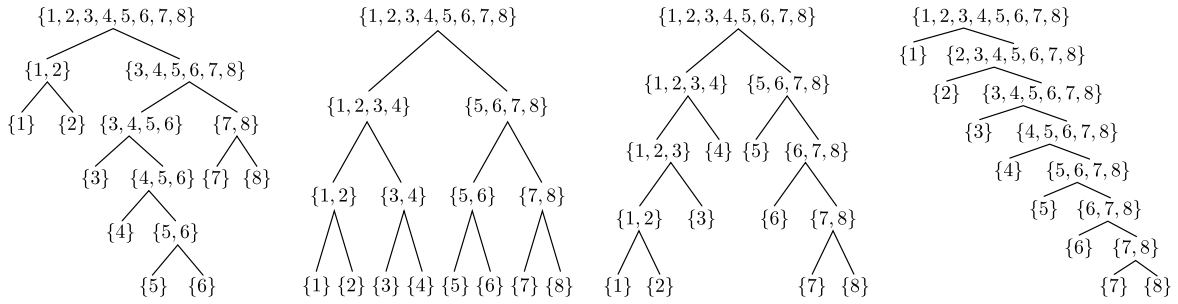
As proposed in [35], instead of directly optimizing over the $J$ factors, the hierarchical matrix factorization method is a heuristic approach that performs successive two-layer matrix factorizations, until $J$ sparse factors are obtained. Let us illustrate this method in the scenario where we want to recover a tuple of sparse factors $(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(J)}) \in \Sigma_{\mathrm{S}_{\mathtt{bf}}}$ from $\mathbf{Z} := \mathbf{X}^{(1)} \dots \mathbf{X}^{(J)}$. At the first level, the factors $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2:J)} := \mathbf{X}^{(2)} \dots \mathbf{X}^{(J)}$ are recovered from their known product $\mathbf{X}^{(1)}\mathbf{X}^{(2:J)} = \mathbf{Z}$. At the second level, the factors $\mathbf{X}^{(2)}$ and $\mathbf{X}^{(3:J)} := \mathbf{X}^{(3)} \dots \mathbf{X}^{(J)}$ are in turn recovered from their known product $\mathbf{X}^{(2)}\mathbf{X}^{(3:J)} = \mathbf{X}^{(2:J)}$. The process is repeated recursively, until all the sparse factors are recovered. At each level, adequate support constraints are enforced on the factors: they correspond to sparsity patterns obtained from the partial products of several sparse factors. In our case with the butterfly constraint, these adequate support constraints correspond to $\mathbf{S}_{\mathtt{bf}}^{(p:q)}$ defined at (3.7).

It was shown [32] that Algorithm 3.1, which is based on the hierarchical approach, performs empirically better than gradient-based optimization methods [12] to address problem

(3.9). Before discussing the details of Algorithm 3.1, note that in the specific case of the butterfly support constraint, the hierarchical factorization method can be generalized to other factorization orders: for instance, in the case of four factors $\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(4)}$ of size $16 \times 16$, one can instead factorize $\mathbf{Z} := \mathbf{X}^{(1)}\mathbf{X}^{(2)}\mathbf{X}^{(3)}\mathbf{X}^{(4)}$ into $\mathbf{X}^{(1)}\mathbf{X}^{(2)}$ and $\mathbf{X}^{(3)}\mathbf{X}^{(4)}$ at the first level, then $\mathbf{X}^{(1)}\mathbf{X}^{(2)}$ into $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}$, and finally $\mathbf{X}^{(3)}\mathbf{X}^{(4)}$ into $\mathbf{X}^{(3)}, \mathbf{X}^{(4)}$. Let us formally introduce a tree structure that describes the factorization order in the hierarchical method.

Definition 3.7 (Factor-bracketing binary tree). *A* factor-bracketing binary tree *of a set of consecutive integers $\{p, \dots, q\}$, with $1 \leq p \leq q$, is a binary tree, where nodes are non-empty subsets of $\{p, \dots, q\}$, that satisfies the axioms: (a) each node is a subset of consecutive indices in $\{p, \dots, q\}$; (b) the root is the set $\{p, \dots, q\}$; (c) a node is a singleton if, and only if, it is a leaf; (d) for each non-leaf node, the left and right children form a partition of their parent, in such a way that the indices of the left child are* smaller *than those in the right child.*

Examples of factor-bracketing binary trees are illustrated in Figure 3.




(a) Arbitrary (b) Balanced (c) Symmetric (d) Unbalanced

Figure 3: Four possible factor-bracketing binary trees of $[J]$, $J = 8$, that can serve as input to Algorithm 3.1. NB: the algorithm is applicable to any number of factors $J$ (not only $J = 8$).

Given as inputs a factor-bracketing binary tree $\mathcal{T}$ of $[J]$ and any target matrix $\mathbf{Z}$, Algorithm 3.1 visits the nodes of $\mathcal{T}$ in a breadth-first search order, starting by the root node. At each non-leaf node $n := \{p, \dots, q\} \subseteq [J]$ characterized its "splitting index" $\ell$, which is the maximum value of its left child, Algorithm 3.2 approximates an intermediate matrix $\mathbf{M}^{(p:q)}$ by a product $\mathbf{M}^{(p:\ell)}\mathbf{M}^{(\ell+1:q)}$, where the left and right factor satisfy the support constraints encoded by $\mathbf{S}_{\mathtt{bf}}^{(p:\ell)}$ and $\mathbf{S}_{\mathtt{bf}}^{(\ell+1:q)}$ (line 7 of Algorithm 3.2). The procedure used to compute these two factors is described in Algorithm 3.3, and is proved to be optimal in [31] in the sense that $\|\mathbf{M}^{(p:q)} - \mathbf{X}\mathbf{Y}^\mathsf{T}\|_F^2$ is minimized among all $\mathbf{X}, \mathbf{Y}$ satisfying the support constraints. In essence, denoting $(\boldsymbol{\mathcal{S}}^i)_{i=1}^N := \varphi(\mathbf{S}_{\mathtt{bf}}^{(p:\ell)}, \mathbf{S}_{\mathtt{bf}}^{(\ell+1:q)\mathsf{T}})$, the procedure consists of successive best rank-one approximations (in the Frobenius norm) of submatrices $\mathbf{M}^{(p:q)} \odot \boldsymbol{\mathcal{S}}^i$ for each $i \in [N]$ (line 4 of Algorithm 3.3), which can be computed for instance via a truncated SVD. The hierarchical procedure is then repeated recursively on $\mathbf{M}^{(p:\ell)}$ and $\mathbf{M}^{(\ell+1:q)}$, with their respective trees.

*Remark* 3.8. One can exploit Algorithm 3.1 *beyond the exact setting* to approximate *any* matrix $\mathbf{Z}$ of size $2^J$ by a matrix having the butterfly structure, since the procedure in Algo-

rithm 3.3 for two-layer fixed-support matrix factorization is optimal [31]. But because this procedure is used in a recursive greedy fashion in Algorithm 3.1, global optimality of the resulting multi-layer factorization is not necessarily guaranteed. Understanding the stability of the algorithm beyond exact recovery is an interesting challenge left to future work.

**Algorithm 3.1** Hierarchical butterfly factorization method, size $N = 2^J$.

**procedure** BUTTERFLY($\mathbf{Z} \in \mathbb{C}^{N\times N}$, $\mathcal{T}$ factor-bracketing binary tree of $[J]$)
  **return** HIERARCHICAL($\mathbf{Z}, \mathcal{T}$) from Algorithm 3.2
**end procedure**

**Algorithm 3.2** Hierarchical step for indices $1 \le p \le q \le J$ in the butterfly factorization method of size $N = 2^J$.

**procedure** HIERARCHICAL($\mathbf{M}^{(p:q)} \in \mathbb{C}^{N\times N}$, factor-bracketing tree $\mathcal{T}^{(p:q)}$ of $\{p, \dots, q\}$)
  **if** $p = q$ **then return** $\mathbf{M}^{(p:q)}$
  **end if**
  $\ell \leftarrow$ maximum value in the left child of the root of $\mathcal{T}^{(p:q)}$
  $\left(\mathcal{T}^{(p:\ell)}, \mathcal{T}^{(\ell+1:q)}\right) \leftarrow$ left and right subtrees of the root of $\mathcal{T}^{(q:p)}$
  $(\mathbf{S}_{\mathtt{bf}}^{(p:\ell)}, \mathbf{S}_{\mathtt{bf}}^{(\ell+1:q)}) \leftarrow$ supports defined by (3.7)
  $(\mathbf{M}^{(p:\ell)}, \mathbf{M}^{(\ell+1:q)\mathsf{T}}) \leftarrow \mathrm{FSMF}(\mathbf{M}^{(p:q)}, \mathbf{S}_{\mathtt{bf}}^{(p:\ell)}, \mathbf{S}_{\mathtt{bf}}^{(\ell+1:q)\mathsf{T}})$ from Algorithm 3.3
  $(\bar{\mathbf{X}}^{(p)}, \dots, \bar{\mathbf{X}}^{(\ell)}) \leftarrow$ HIERARCHICAL($\mathbf{M}^{(p:\ell)}, \mathcal{T}^{(p:\ell)}$)
  $(\bar{\mathbf{X}}^{(\ell+1)}, \dots, \bar{\mathbf{X}}^{(q)}) \leftarrow$ HIERARCHICAL($\mathbf{M}^{(\ell+1:q)}, \mathcal{T}^{(\ell+1:q)}$)
  **return** $(\bar{\mathbf{X}}^{(p)}, \dots, \bar{\mathbf{X}}^{(\ell)}, \bar{\mathbf{X}}^{(\ell+1)}, \dots, \bar{\mathbf{X}}^{(q)})$
**end procedure**

**Algorithm 3.3** Fixed-support matrix factorization under assumptions of Proposition 2.6 [31, Algorithm 3.1].

**procedure** FSMF($\mathbf{Z} \in \mathbb{C}^{m\times n}, \mathbf{S}^L \in \mathbb{B}^{m\times r}, \mathbf{S}^R \in \mathbb{B}^{n\times r}$)
  $(\boldsymbol{\mathcal{S}}^i)_{i=1}^N \leftarrow \varphi\left(\mathbf{S}^L, \mathbf{S}^R\right)$ as defined in (2.7)
  **for** $i \in [r]$ **do**
    $(\mathbf{x}_i, \mathbf{y}_i) \leftarrow \arg\min \left\{ \|\mathbf{Z} \odot \boldsymbol{\mathcal{S}}^i - \mathbf{x}_i \mathbf{y}_i{}^\mathsf{T}\|_F, \text{ such that } \mathrm{supp}(\mathbf{x}_i \mathbf{y}_i{}^\mathsf{T}) \subseteq \boldsymbol{\mathcal{S}}^i \right\}$
  **end for**
  $\mathbf{X} \leftarrow (\mathbf{x}_1, \dots, \mathbf{x}_r) \in \mathbb{C}^{m\times r}$
  $\mathbf{Y} \leftarrow (\mathbf{y}_1, \dots, \mathbf{y}_r) \in \mathbb{C}^{n\times r}$
  **return** $(\mathbf{X}, \mathbf{Y})$
**end procedure**

### 3.3. Uniqueness of the butterfly factorization.

As the main contribution of the paper, we now show that the exact factorization $\mathbf{Z} = \mathbf{X}^{(1)} \dots \mathbf{X}^{(J)}$ into $J$ factors constrained to the butterfly supports is essentially unique, and that these factors can be recovered by Algorithm 3.1. Let us first generalize Definition 2.1 to the multi-layer case with $J \ge 3$.

Definition 3.9 (Essential uniqueness of a multi-layer factorization in $\Sigma$). *Consider integers $N_0, \dots, N_J$, a set $\Sigma \subseteq \mathbb{C}^{N_0 \times N_1} \times \dots \times \mathbb{C}^{N_{J-1} \times N_J}$ of $J$-tuples of factors, and a matrix $\mathbf{Z}$ admitting a factorization $\mathbf{Z} := \mathbf{X}^{(1)} \dots \mathbf{X}^{(J)}$ such that $(\mathbf{X}^{(\ell)})_{\ell=1}^J \in \Sigma$. We say that this factorization is* essentially unique *in $\Sigma$, if any $(\bar{\mathbf{X}}^{(1)}, \dots, \bar{\mathbf{X}}^{(J)}) \in \Sigma$ such that $\bar{\mathbf{X}}^{(1)} \dots \bar{\mathbf{X}}^{(J)} = \mathbf{Z}$ is equivalent to $(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(J)})$, written $(\mathbf{X}^{(\ell)})_{\ell=1}^J \sim (\bar{\mathbf{X}}^{(\ell)})_{\ell=1}^J$, in the sense that there exist invertible diagonal matrices $\mathbf{D}^{(1)}, \dots, \mathbf{D}^{(J-1)}$ such that $\bar{\mathbf{X}}^{(\ell)} = {\mathbf{D}^{(\ell-1)}}^{-1} \mathbf{X}^{(\ell)} \mathbf{D}^{(\ell)}$ for all $\ell \in [J]$, with the convention that $\mathbf{D}^{(0)}$ and $\mathbf{D}^{(J)}$ are identity matrices.*

Theorem 3.10. *Consider $\mathrm{S}_{\mathtt{bf}}$ the butterfly supports of size $N = 2^J$ and $(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(J)}) \in \Sigma_{\mathrm{S}_{\mathtt{bf}}}$. Assume that $\mathbf{X}^{(\ell)}$ does not have a zero column for $1 \leq \ell \leq J-1$, and not a zero row for $2 \leq \ell \leq J$. Then, the factorization $\mathbf{Z} := \mathbf{X}^{(1)} \dots \mathbf{X}^{(J)}$ is essentially unique in $\Sigma_{\mathrm{S}_{\mathtt{bf}}}$. These factors can be recovered from $\mathbf{Z}$, up to scaling ambiguities only, using the procedure* Butterfly$(\mathbf{Z}, \mathcal{T})$ *detailed in Algorithm 3.1, where $\mathcal{T}$ is* any *factor-bracketing tree of $[J]$.*

In other words, Algorithm 3.1 is endowed with *exact recovery guarantees*. In particular, Theorem 3.10 can be applied to show identifiability of the butterfly factorization of the DFT matrix as suggested in [33, Chapter 7], but also the one of the Hadamard matrix. In both cases, the butterfly factors of the DFT or the Hadamard matrix can be recovered up to scaling ambiguities via Algorithm 3.1 with *any* factor-bracketing binary tree of $[J]$ as input. Before proving Theorem 3.10, we show that Algorithm 3.1 has controlled complexity bounds.

**3.4. Complexity bounds.** Existing algorithms for butterfly factorization [12, 35] are based on gradient descent, and as such they require to tune several criteria such as learning rate or stopping criteria. In contrast, Algorithm 3.1 has a bounded complexity as it essentially consists in a controlled number of truncated SVDs to compute rank-one approximations of submatrices. While the full SVD of a matrix of size $m \times n$ would require $\mathcal{O}(mn \min(m,n))$ flops, truncated SVD with numerical rank $k$ requires only $\mathcal{O}(kmn)$ flops (see e.g. [24] and references therein). Hence, in our complexity analysis of Algorithm 3.1, the theoretical complexity of computing the best rank-one approximation of a matrix of size $m \times n$ will be $\mathcal{O}(mn)$. Below we estimate and compare the complexity for two types of factor-bracketing binary trees.

*Unbalanced tree.* First we consider running Algorithm 3.1 with a matrix $\mathbf{Z}$ of size $N \times N$, $N = 2^J$ ($J \geq 2$), and the *unbalanced* factor-bracketing binary tree $\mathcal{T}$ of $[J]$ (defined as the factor-bracketing binary tree where the left child of each non-leaf node is a singleton, see Figure 3d) as inputs. There are in total $J-1$ non-leaf nodes in this tree. At the non-leaf node of depth $j \in \{0, \dots, J-2\}$, the algorithm computes the best rank-one approximation of $N$ submatrices of size $2 \times N/2^{j+1}$, which yields a cost of the order of $N \times (2 \times N/2^{j+1}) = N^2/2^j$. Hence, the total cost of Algorithm 3.1 with the unbalanced factor-bracketing binary tree is of the order of: $\sum_{j=0}^{J-2} \frac{N^2}{2^j} = 2(1-2^{-J+1})N^2 = 2\left(1 - \frac{2}{N}\right) N^2 = \mathcal{O}(N^2)$. Similarly, the complexity is $\mathcal{O}(N^2)$ when best rank-one approximations are computed with full SVDs (see Appendix E).

*Balanced tree.* Consider now Algorithm 3.1 with an $N \times N$ matrix $\mathbf{Z}$, $N = 2^J$ where $J$ is also a power of 2, and the *balanced* factor-bracketing binary tree $\mathcal{T}$ of $[J]$ (i.e., all children of each non-leaf node have the same cardinality, see Figure 3b). At each non-leaf node of depth $k \in \{0, \dots, \log_2(J) - 1\}$, the best rank-one approximation of $N$ square submatrices of size $\sqrt{N^{1/2^k}}$ is computed, at a cost of the order of $N \times (\sqrt{N^{1/2^k}} \times \sqrt{N^{1/2^k}}) = N \times N^{1/2^k}$. At each depth $k \in \{0, \dots, \log_2(J) - 1\}$, there are $2^k$ nodes. As $2^k \leq J/2 = \log_2(N)/2$ the total cost

of Algorithm 3.1 is of the order of: $\sum_{k=0}^{\log_2(J)-1} 2^k N \times N^{1/2^k} = N^2 + \sum_{k=1}^{\log_2(J)-1} 2^k N^{1+1/2^k} \leq N^2 + \sum_{k=1}^{\log_2(\log_2(N))-1} \frac{\log_2(N)}{2} N^{3/2} = \mathcal{O}(N^2)$. This contrasts with the complexity $\mathcal{O}(N^{5/2})$ when using full SVDs (see Appendix E).

*Discussion.* The complexity of Algorithm 3.1 is of the same order of magnitude as that of matrix-vector multiplications of size $N \times N$, which is $\mathcal{O}(N^2)$. Assume that we want to compute the product $\mathbf{AB}$, where $\mathbf{A}, \mathbf{B}$ are of size $N \times N$, and $\mathbf{A}$ admits the butterfly structure. The naive computation requires $\mathcal{O}(N^3)$. But the data-sparse representation of $\mathbf{A}$ as a product of butterfly factors can be recovered by Algorithm 3.1 in $\mathcal{O}(N^2)$, in order to enable fast $\mathcal{O}(N \log N)$ matrix-vector multiplication [12]. In other words, this method allows for a computation of $\mathbf{AB}$ in $\mathcal{O}(N^2(\log N + 1))$ flops only, instead of $\mathcal{O}(N^3)$.

Finally, it is possible to implement Algorithm 3.1 in a distributed fashion. Indeed, the computation of the best rank-one approximation of each submatrix at line Algorithm 3.3 can be performed in parallel: in the setting with $T$ threads, each thread computes the best rank-one approximation of $\lfloor N/T \rfloor$ submatrices. Moreover, when running Algorithm 3.1 with a balanced factor-bracketing binary tree, the implementation can be further parallelized, since the factorization at each node of the same depth can be performed by independent threads.

**3.5. Proof of uniqueness (Theorem 3.10).** This subsection is dedicated to the proof of Theorem 3.10. The reader more interested in the numerical aspects of the proposed method can skip this part and jump to section 4. In order to prove Theorem 3.10, consider $(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(J)}) \in \Sigma_{\mathbf{S}_{\text{bf}}}$ satisfying the butterfly constraint. These factors are assumed to verify the assumption of Theorem 3.10. For any $1 \leq p \leq q \leq J$, denote $\mathbf{X}^{(p:q)} := \mathbf{X}^{(p)} \dots \mathbf{X}^{(q)}$, and $\mathbf{Z} := \mathbf{X}^{(1:J)} = \mathbf{X}^{(1)} \dots \mathbf{X}^{(J)}$. Fix any factor-bracketing binary tree $\mathcal{T}$ of $[J]$. Given $\mathbf{Z}$ and $\mathcal{T}$ as the inputs of Algorithm 3.1, we write $\mathbf{M}^{(p:q)}$ the intermediate matrix obtained at each node $n := \{p, \dots, q\}$ of $\mathcal{T}$ from the hierarchical factorization procedure. The proof is now separated into two steps. Firstly, we prove that Algorithm 3.1 recovers the butterfly factors $(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(J)})$ from $\mathbf{Z}$, up to scaling ambiguities. Secondly, we prove that the butterfly factorization $\mathbf{Z} = \mathbf{X}^{(1)} \dots \mathbf{X}^{(J)}$ is indeed essentially unique in the sense of Definition 3.9.

**3.5.1. The algorithm recovers the butterfly factors.** The proof of the first part consists in conducting an induction over the non-leaf nodes $n_1, \dots, n_{J-1}$ of the tree, ordered in a breadth-first-search order (there are indeed in total $J-1$ non-leaf nodes in $\mathcal{T}$). The general idea is to show that Algorithm 3.1 reconstructs recursively from $\mathbf{Z}$ the partial products $\mathbf{X}^{(p:q)}$ up to scaling ambiguities at each node $n = \{p, \dots, q\}$ of the tree $\mathcal{T}$. To that end, we need to prove a crucial lemma (Lemma 3.12) that essentially reduces the analysis of the multi-layer factorization to the case with only two factors. Then, the proof of Lemma 3.12 itself relies on two key ingredients about optimality (Theorem 3.14) and essential uniqueness (Proposition 3.17) of the considered two-layer factorization problem.

For each $v \in [J-1]$, denote $p_v$, $q_v$ the minimum and maximum index of node $n_v = \{p_v, \dots, q_v\}$, and $\ell_v$ as the "splitting" index of node $n_v$, which is the maximum value of its *left* child. In the proof we will use the following consequence of the breadth-first-search order.

Fact 3.11. *For any $u \in \{2, \dots, J-1\}$, the node $n_u = \{p_u, \dots, q_u\}$ is a child of some node $n_v = \{p_v, \dots \ell_v, \ell_v + 1, \dots, q_v\}$ with $v \in \{1, \dots, u-1\}$. If $n_u$ is the left child of $n_v$, then $(p_u, q_u) = (p_v, \ell_v)$. If $n_u$ is the right child of $n_v$, then $(p_u, q_u) = (\ell_v + 1, q_v)$.*

Define for any $V \in [J-1]$, the assertion $P_V$: "there exist invertible diagonal matrices $\mathbf{D}^{(\ell_1)}, \dots, \mathbf{D}^{(\ell_V)}$ such that, for each $v \in [V]$, we have: $\mathbf{M}^{(p_v:\ell_v)} = {\mathbf{D}^{(p_v-1)}}^{-1}\mathbf{X}^{(p_v:\ell_v)}\mathbf{D}^{(\ell_v)}$ and $\mathbf{M}^{(\ell_v+1:q_v)} = {\mathbf{D}^{(\ell_v)}}^{-1}\mathbf{X}^{(\ell_v+1:q_v)}\mathbf{D}^{(q_v)}$", with the convention $\mathbf{D}^{(0)} = \mathbf{D}^{(J)} = \mathbf{I}_N$[6]. The principle of the proof is to show $P_V$ by induction for all $V \in [J-1]$. In particular, proving $P_{J-1}$ yields our claim, as we now explain. Indeed, any leaf node $n = \{\ell\}$ is either a left child or a right child of a non-leaf node $n_v = \{p_v, \dots, q_v\}$ with $v \in [J-1]$. In the case of a left child, $\{\ell\} = \{p_v, \dots, \ell_v\}$ hence $\ell = p_v = \ell_v$, and in the other case $\{\ell\} = \{\ell_v + 1, \dots, q_v\}$ hence $\ell = \ell_v + 1 = q_v$. In both cases assertion $P_{J-1}$ implies that $\mathbf{M}^{(\ell:\ell)} = {\mathbf{D}^{(\ell-1)}}^{-1}\mathbf{X}^{(\ell:\ell)}\mathbf{D}^{(\ell)} = {\mathbf{D}^{(\ell-1)}}^{-1}\mathbf{X}^{(\ell)}\mathbf{D}^{(\ell)}$, hence $(\mathbf{M}^{(\ell:\ell)})_{\ell=1}^{J} \sim (\mathbf{X}^{(\ell)})_{\ell=1}^{J}$, meaning that the algorithm recovers the butterfly factors up to scaling ambiguities only. The crux of the proof is the following lemma.

Lemma 3.12. *Under the assumptions of Theorem 3.10, consider $V \in [J-1]$ and assume that there are invertible diagonal matrices $\mathbf{D}^{(p_V-1)}$ and $\mathbf{D}^{(q_V)}$ such that $\mathbf{M}^{(p_V:q_V)} = {\mathbf{D}^{(p_V-1)}}^{-1}\mathbf{X}^{(p_V:q_V)}\mathbf{D}^{(q_V)}$. Then the pair $(\mathbf{M}^{(p_V:\ell_V)}, \mathbf{M}^{(\ell_V+1:q_V)\intercal})$ computed at line 7 of Algorithm 3.2 such that the product $\mathbf{M}^{(p_V:\ell_V)}\mathbf{M}^{(\ell_V+1:q_V)}$ approximates $\mathbf{M}^{(p_V:q_V)}$ is equal (up to scaling ambiguities) to the pair $(\bar{\mathbf{X}}, \bar{\mathbf{Y}})$ where $\bar{\mathbf{X}} := {\mathbf{D}^{(p_V-1)}}^{-1}\mathbf{X}^{(p_V:\ell_V)}$, $\bar{\mathbf{Y}} := (\mathbf{X}^{(\ell_V+1:q_V)}\mathbf{D}^{(q_V)})^{\intercal}$.*

Indeed, since $\mathbf{M}^{(p_1:q_1)} = \mathbf{Z} = \mathbf{X}^{(p_1:q_1)}$ and since $\mathbf{D}^{(p_1-1)} = \mathbf{D}^{(q_1)} = \mathbf{I}_N$ (recall that $p_1 = 1$ and $q_1 = J$), this lemma applied to $V = 1$ shows that $P_1$ is true. This starts the induction, and we now show that the lemma can similarly be used to proceed to the induction. Assume that $P_{V-1}$ is true where $V \in \{2, \dots, J-1\}$ and consider $\mathbf{D}^{(\ell_1)}, \dots, \mathbf{D}^{(\ell_{V-1})}$ the corresponding invertible diagonal matrices. By Fact 3.11, the parent of node $n_V$ is necessarily some node $n_v$ with $v \in [V-1]$ and, depending on whether $n_V$ is a left or right child of $n_v$, we have either $n_V = \{p_v, \dots, \ell_v\}$ or $n_V = \{\ell_v+1, \dots, q_v\}$. Without loss of generality assume the former (the proof is similar if we suppose the latter) so that $(p_V, q_V) = (p_v, \ell_v)$. Since $P_{V-1}$ is true we have that $\mathbf{M}^{(p_V:q_V)} = \mathbf{M}^{(p_v:\ell_v)} = {\mathbf{D}^{(p_v-1)}}^{-1}\mathbf{X}^{(p_v:\ell_v)}\mathbf{D}^{(\ell_v)} = {\mathbf{D}^{(p_V-1)}}^{-1}\mathbf{X}^{(p_V:q_V)}\mathbf{D}^{(q_V)}$. By Lemma 3.12, there exists an invertible diagonal matrix $\mathbf{D}^{(\ell_V)}$ such that $\mathbf{M}^{(p_V:\ell_V)} = {\mathbf{D}^{(p_V-1)}}^{-1}\mathbf{X}^{(p_V:\ell_V)}\mathbf{D}^{(\ell_V)}$ and $\mathbf{M}^{(\ell_V+1:q_V)} = {\mathbf{D}^{(\ell_V)}}^{-1}\mathbf{X}^{(\ell_V+1:q_V)}\mathbf{D}^{(q_V)}$, which proves $P_V$.

We now focus on the proof of Lemma 3.12. It relies on two key ingredients formulated in Theorem 3.14 and Proposition 3.17 below. They are both derived from the following property of the butterfly supports, proved in Appendix F.

Lemma 3.13. *Given $1 \le p \le \ell < q \le J$, $\mathbf{S}^L := \mathbf{S}_{\mathtt{bf}}^{(p:\ell)}$ and $\mathbf{S}^R := \mathbf{S}_{\mathtt{bf}}^{(\ell+1:q)\intercal}$ where we recall (3.7), the tuple of rank-one supports $\varphi\left(\mathbf{S}^L, \mathbf{S}^R\right)$ has disjoint rank-one supports.*

The first consequence of this property is the following optimality theorem.

Theorem 3.14 ([31, Application of Theorem 3.3]). *Denote $\mathrm{S} := (\mathbf{S}_{\mathtt{bf}}^{(p:\ell)}, \mathbf{S}_{\mathtt{bf}}^{(\ell+1:q)\intercal})$. For any matrix $\mathbf{M}$, the procedure $\mathrm{FSMF}(\mathbf{M}, \mathbf{S}_{\mathtt{bf}}^{(p:\ell)}, \mathbf{S}_{\mathtt{bf}}^{(\ell+1:q)\intercal})$ described in Algorithm 3.3 computes in polynomial time a pair of factors solving the problem: $\min_{(\mathbf{X},\mathbf{Y})\in\Sigma_{\mathrm{S}}} \|\mathbf{M} - \mathbf{X}\mathbf{Y}^{\intercal}\|_F$.*

Lemma 3.13 also allows us to characterize the set $\mathcal{U}(\Sigma_{\mathrm{S}})$ when $\mathrm{S} := (\mathbf{S}_{\mathtt{bf}}^{(p:\ell)}, \mathbf{S}_{\mathtt{bf}}^{(\ell+1:q)\intercal})$.

[6]Remark that for any $v \in [V]$ the node $n_v = \{p_v, \dots, q_v\}$ is either the root node (when $v = 1$) or a child of a node $n_w$ with $w < v$. In the latter case, by Fact 3.11, either $(p_v, q_v) = (p_w, \ell_w)$, or $(p_v, q_v) = (\ell_w + 1, q_w)$, with $w \in [v-1] \subseteq [V]$. In other words, $p_v - 1, \ell_v, q_v \in \{0, \ell_1, \dots, \ell_V, J\}$ for all $v \in [V]$, meaning that the diagonal matrices $\mathbf{D}^{(p_v-1)}$, $\mathbf{D}^{(\ell_v)}$ and $\mathbf{D}^{(q_v)}$ used in the definition of $P_V$ above are well defined.

Lemma 3.15. *Denote* $\mathrm{S} = (\mathbf{S}^L, \mathbf{S}^R) := (\mathbf{S}_{\mathtt{bf}}^{(p:\ell)}, \mathbf{S}_{\mathtt{bf}}^{(\ell+1:q)\mathsf{T}})$. *We have*

$$\mathcal{U}(\Sigma_{\mathrm{S}}) = \{(\mathbf{X}, \mathbf{Y}) \in \Sigma_{\mathrm{S}} \mid \operatorname{colsupp}(\mathbf{X}) = \operatorname{colsupp}(\mathbf{Y}) = [N]\}.$$

*Proof.* By Proposition 2.6, $\mathcal{U}(\Sigma_{\mathrm{S}}) = \mathtt{IC}_{\mathrm{S}} \cap \mathtt{MC}_{\mathrm{S}}$. Since $\operatorname{colsupp}(\mathbf{S}^L) = \operatorname{colsupp}(\mathbf{S}^R) = [N]$, by definition of $\mathtt{IC}_{\mathrm{S}}$ and $\mathtt{MC}_{\mathrm{S}}$ (cf (2.5)-(2.6)) we have $(\mathbf{X}, \mathbf{Y}) \in \mathtt{IC}_{\mathrm{S}} \cap \mathtt{MC}_{\mathrm{S}}$ if, and only if, $\mathbf{X}$ and $\mathbf{Y}$ do not have a zero column, i.e., $\operatorname{colsupp}(\mathbf{X}) = \operatorname{colsupp}(\mathbf{Y}) = [N]$. ∎

Recall that the factors $(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(J)})$ verify the assumption of Theorem 3.10, i.e., $\mathbf{X}^{(\ell)}$ does not have a zero column for $1 \leq \ell \leq J-1$, and not a zero row for $2 \leq \ell \leq J$. As claimed in the following lemma proved in Appendix G, this assumption is in fact a necessary and sufficient condition to ensure that each pair of partial products $(\mathbf{X}^{(p:\ell)}, \mathbf{X}^{(\ell+1:q)\mathsf{T}})$ with $1 \leq p \leq \ell < q \leq J$ is non-degenerate (the left and right factor do not have a zero column).

Lemma 3.16. *Let* $\mathrm{S}_{\mathtt{bf}}$ *be the butterfly supports of size* $N = 2^J$, *and* $(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(J)}) \in \Sigma_{\mathrm{S}_{\mathtt{bf}}}$. *The following are equivalent:*

*(i) for each* $1 \leq p \leq \ell < q \leq J$, $\mathbf{X}^{(p:\ell)} := \mathbf{X}^{(p)} \dots \mathbf{X}^{(\ell)}$ *does not have a zero column, and* $\mathbf{X}^{(\ell+1:q)} := \mathbf{X}^{(\ell+1)} \dots \mathbf{X}^{(q)}$ *does not have a zero row;*

*(ii)* $\mathbf{X}^{(\ell)}$ *does not have a zero column for* $1 \leq \ell \leq J-1$, *and not a zero row for* $2 \leq \ell \leq J$.

Consequently, we obtain the second key ingredient for the proof of Lemma 3.12.

Proposition 3.17. *Assume that* $(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(J)}) \in \Sigma_{\mathrm{S}_{\mathtt{bf}}}$ *verifies the hypothesis of Theorem 3.10. Let* $1 \leq p \leq \ell < q \leq J$. *Denote* $\mathrm{S} := (\mathbf{S}_{\mathtt{bf}}^{(p:\ell)}, \mathbf{S}_{\mathtt{bf}}^{(\ell+1:q)\mathsf{T}})$. *Then, for any invertible diagonal matrices* $\mathbf{D}, \bar{\mathbf{D}}$, *denoting* $\bar{\mathbf{X}} := \mathbf{D}^{-1}\mathbf{X}^{(p:\ell)}$ *and* $\bar{\mathbf{Y}} = (\mathbf{X}^{(\ell+1:q)}\bar{\mathbf{D}})^{\mathsf{T}}$, *the factorization* $\mathbf{D}^{-1}\mathbf{X}^{(p:q)}\bar{\mathbf{D}} = \bar{\mathbf{X}}\bar{\mathbf{Y}}^{\mathsf{T}}$ *into two factors* $(\bar{\mathbf{X}}, \bar{\mathbf{Y}})$ *is essentially unique in* $\Sigma_{\mathrm{S}}$.

*Proof.* By Lemma 3.4, $(\mathbf{X}^{(p:\ell)}, \mathbf{X}^{(\ell+1:q)\mathsf{T}}) \in \Sigma_{\mathrm{S}}$. By Lemma 3.16 and the assumption on the factors $\mathbf{X}^{(\ell)}$, $1 \leq \ell \leq J$, the matrices $\mathbf{X}^{(p:\ell)}$ and $\mathbf{X}^{(\ell+1:q)\mathsf{T}}$ do not have a zero column. The same is true for $\mathbf{D}^{-1}\mathbf{X}^{(p:\ell)}$ and $\bar{\mathbf{D}}\mathbf{X}^{(\ell+1:q)\mathsf{T}}$, as the multiplication of a matrix by $\mathbf{D}^{-1}$ or $\bar{\mathbf{D}}$ does not change its support. By Lemma 3.15, $(\mathbf{D}^{-1}\mathbf{X}^{(p:\ell)}, \bar{\mathbf{D}}\mathbf{X}^{(\ell+1:q)\mathsf{T}}) \in \mathcal{U}(\Sigma_{\mathrm{S}})$. ∎

We now have all the ingredients to prove the crucial Lemma 3.12.

*Proof of Lemma 3.12.* Since $\mathbf{X}^{(p_V:q_V)} = \mathbf{X}^{(p_V:\ell_V)}\mathbf{X}^{(\ell_V+1:q_V)}$, we can factorize the matrix $\mathbf{M}^{(p_V:q_V)} = \mathbf{D}^{(p_V-1)^{-1}}\mathbf{X}^{(p_V:q_V)}\mathbf{D}^{(q_V)}$ as $\mathbf{M}^{(p_V:q_V)} = \bar{\mathbf{X}}\bar{\mathbf{Y}}^{\mathsf{T}}$, with $\bar{\mathbf{X}} := \mathbf{D}^{(p_V-1)^{-1}}\mathbf{X}^{(p_V:\ell_V)}$, $\bar{\mathbf{Y}} := (\mathbf{X}^{(\ell_V+1:q_V)}\mathbf{D}^{(q_V)})^{\mathsf{T}}$. By Lemma 3.4, $(\bar{\mathbf{X}}, \bar{\mathbf{Y}}) \in \Sigma_{\mathrm{S}}$ where $\mathrm{S} := (\mathbf{S}_{\mathtt{bf}}^{(p_V:\ell_V)}, \mathbf{S}_{\mathtt{bf}}^{(\ell_V+1:q_V)\mathsf{T}})$. By Theorem 3.14, the factors $(\mathbf{M}^{(p_V:\ell_V)}, \mathbf{M}^{(\ell_V+1:q_V)\mathsf{T}}) \in \Sigma_{\mathrm{S}}$ computed at line 7 of Algorithm 3.2 minimimize the optimization problem: $\min_{(\mathbf{X},\mathbf{Y})\in\Sigma_{\mathrm{S}}} \|\mathbf{M}^{(p_V:q_V)} - \mathbf{X}\mathbf{Y}^{\mathsf{T}}\|_F$. Since $\mathbf{M}^{(p_V:q_V)} = \bar{\mathbf{X}}\bar{\mathbf{Y}}^{\mathsf{T}}$ with $(\bar{\mathbf{X}}, \bar{\mathbf{Y}}) \in \Sigma_{\mathrm{S}}$, this minimum is zero hence the computed pair satisfies $\mathbf{M}^{(p_V:q_V)} = \mathbf{M}^{(p_V:\ell_V)}\mathbf{M}^{(\ell_V+1:q_V)}$. By Proposition 3.17, the factorization $\mathbf{M}^{(p_V:q_V)} = \bar{\mathbf{X}}\bar{\mathbf{Y}}^{\mathsf{T}}$ into two factors is essentially unique in $\Sigma_{\mathrm{S}}$, so $(\mathbf{M}^{(p_V:\ell_V)}, \mathbf{M}^{(\ell_V+1:q_V)\mathsf{T}}) \sim (\bar{\mathbf{X}}, \bar{\mathbf{Y}})$. ∎

This ends the inductive proof of the first part of Theorem 3.10 showing that Algorithm 3.1 recovers the butterfly factors $(\mathbf{X}^{(\ell)})_{\ell=1}^J$ from $\mathbf{Z} = \mathbf{X}^{(1)} \dots \mathbf{X}^{(J)}$, up to scaling ambiguities.

**3.5.2. The factorization is essentially unique.** We now show that the factorization $\mathbf{Z} = \mathbf{X}^{(1)} \dots \mathbf{X}^{(J)}$ is essentially unique in $\Sigma_{\mathrm{S}_{\mathtt{bf}}}$ in the sense of Definition 3.9. To that end, consider the factors $(\mathbf{M}^{(\ell:\ell)})_{\ell=1}^J$ computed using Algorithm 3.1 with $\mathbf{Z}$ as input, as well as arbitrary

factors $(\bar{\mathbf{X}}^{(\ell)})_{\ell=1}^{J} \in \Sigma_{\mathrm{S}_{\mathtt{bf}}}$ such that $\bar{\mathbf{X}}^{(1)} \dots \bar{\mathbf{X}}^{(J)} = \mathbf{Z}$. We will show that $(\bar{\mathbf{X}}^{(\ell)})_{\ell=1}^{J}$ verifies the assumptions of Theorem 3.10: this will imply that $(\mathbf{M}^{(\ell:\ell)})_{\ell=1}^{J}$ is rescaling-equivalent *both* to $(\mathbf{X}^{(\ell)})_{\ell=1}^{J}$ and to $(\bar{\mathbf{X}}^{(\ell)})_{\ell=1}^{J}$, hence, by transitivity, $(\mathbf{X}^{(\ell)})_{\ell=1}^{J} \sim (\bar{\mathbf{X}}^{(\ell)})_{\ell=1}^{J}$ as claimed.

Denote $\bar{\mathbf{X}}^{(p:q)} := \bar{\mathbf{X}}^{(p)} \dots \bar{\mathbf{X}}^{(q)}$ for any $1 \leq p \leq q \leq J$. For any $\ell \in [J-1]$, we have $\mathbf{X}^{(1:\ell)}\mathbf{X}^{(\ell+1:J)} = \mathbf{Z} = \bar{\mathbf{X}}^{(1:\ell)}\bar{\mathbf{X}}^{(\ell+1:J)}$. By Lemma 3.4, $(\bar{\mathbf{X}}^{(1:\ell)}, \bar{\mathbf{X}}^{(\ell+1:J)\mathsf{T}}) \in \Sigma_{\mathrm{S}}$ where $\mathrm{S} := (\mathbf{S}_{\mathtt{bf}}^{(1:\ell)}, \mathbf{S}_{\mathtt{bf}}^{(\ell+1:J)\mathsf{T}})$. Besides, by assumption and by Lemma 3.16, $\mathbf{X}^{(1:\ell)}$ and $\mathbf{X}^{(\ell+1:J)\mathsf{T}}$ do not have a zero column, so by Lemma 3.15, $(\mathbf{X}^{(1:\ell)}, \mathbf{X}^{(\ell+1:J)\mathsf{T}}) \in \mathcal{U}(\Sigma_{\mathrm{S}})$. By definition of the set $\mathcal{U}(\Sigma_{\mathrm{S}})$ of essentially unique factors, $(\bar{\mathbf{X}}^{(1:\ell)}, \bar{\mathbf{X}}^{(\ell+1:J)\mathsf{T}}) \sim (\mathbf{X}^{(1:\ell)}, \mathbf{X}^{(\ell+1:J)\mathsf{T}})$. Since $\mathbf{X}^{(1:\ell)}$ and $\mathbf{X}^{(\ell+1:J)\mathsf{T}}$ do not have a zero column, this implies that $\bar{\mathbf{X}}^{(1:\ell)}$ and $\bar{\mathbf{X}}^{(\ell+1:J)\mathsf{T}}$ also do not have a zero column. Consequently, $\bar{\mathbf{X}}^{(\ell)}$ does not have a zero column (otherwise $\bar{\mathbf{X}}^{(1:\ell)}$ would have a zero column) and similarly $\bar{\mathbf{X}}^{(\ell+1)}$ does not have a zero row. As this holds for any $\ell \in [J-1]$, $(\bar{\mathbf{X}}^{(\ell)})_{\ell=1}^{J}$ verifies the assumption of Theorem 3.10. This ends the proof of Theorem 3.10.

**3.6. Relation with the complementary low-rank property.** We now relate the butterfly structure (Definition 3.2) to the complementary low-rank property [3, 40], formally introduced here for a square matrix of size $N \times N$ for $N = 2^J$ using the notations from [40].

Definition 3.18 (Complementary low-rank property [40]). *Consider two binary trees $T_X$ and $T_\Omega$ of maximum depth $J$, called* index-partitioning *binary trees, such that each node is a non-empty subset of indices of $[N]$, $N := 2^J$, with the root being $[N]$ and the children of each non-leaf node forming a partition of their parent. A matrix $\mathbf{Z}$ of size $N \times N$ satisfies the* complementary low-rank property *for the trees $T_X$ and $T_\Omega$ if, for any level $\ell \in \{0, \dots, J\}$, any node $A$ in $T_X$ at level $J - \ell$, and any node $B$ in $T_\Omega$ at level $\ell$, the submatrix $\mathbf{Z}_{A,B}$ obtained by restricting $\mathbf{Z}$ to the rows indexed by $A$ and to the columns indexed by $B$ is of low rank.*

We claim that a matrix having the butterfly structure satisfies the complementary low-rank property for specific choices of index-partitioning binary trees $T_X$ and $T_\Omega$.

Proposition 3.19. *Construct the index-partitioning binary trees $T_X$ and $T_\Omega$ in the following way: (a) each node $A := \{a_1, a_2, \dots, a_{N/2^\ell}\}$ of $T_X$ at level $\ell \in \{0, \dots, J-1\}$, where $a_1 < \dots < a_{N/2^\ell}$, has $\{a_1, a_3, \dots, a_{N/2^\ell - 1}\}$ as its left child and $\{a_2, a_4, \dots, a_{N/2^\ell}\}$ as its right child; (b) each node $B := \{b_1, b_2, \dots, b_{N/2^\ell}\}$ of $T_\Omega$ at level $\ell \in \{0, \dots, J-1\}$, where $b_1 < \dots < b_{N/2^\ell}$, has $\{b_1, b_2, \dots, b_{N/2^{\ell+1}}\}$ as its left child and $\{b_{N/2^{\ell+1}+1}, b_{N/2^{\ell+1}+2}, \dots, b_{N/2^\ell}\}$ as its right child. Consider $\mathrm{S}_{\mathtt{bf}}$ the butterfly supports of size $N = 2^J$ and $(\mathbf{X}^{(1)}, \dots, \mathbf{X}^{(J)}) \in \Sigma_{\mathrm{S}_{\mathtt{bf}}}$. Then, $\mathbf{Z} := \mathbf{X}^{(1)} \dots \mathbf{X}^{(J)}$ satisfies the complementary low-rank property for the trees $T_X$ and $T_\Omega$.*

*Remark* 3.20. Each node $A$ in $T_X$ at level $J - \ell$ for $\ell \in \{0, \dots, J\}$ is of the form $\{c + kN/2^\ell,\ k = 0, \dots, 2^\ell - 1\}$ for an index $c \in [N/2^\ell]$. Each node $B$ in $T_\Omega$ at level $\ell \in \{0, \dots, J\}$ is of the form $\{(c-1)N/2^\ell + k,\ k = 1, \dots, N/2^\ell\}$ for an index $c \in [2^\ell]$.

*Proof.* Denoting $\mathbf{X}^{(p:q)} := \mathbf{X}^{(p)} \dots \mathbf{X}^{(q)}$ for any $1 \leq p \leq q \leq J$, we have $\mathbf{Z} = \mathbf{X}^{(1:\ell)}\mathbf{X}^{(\ell+1:J)}$ for any $\ell \in [J-1]$. Fix $\ell \in [J-1]$, and denote $\mathrm{S} := (\mathbf{S}_{\mathtt{bf}}^{(1:\ell)}, \mathbf{S}_{\mathtt{bf}}^{(\ell+1:J)\mathsf{T}})$, where we recall the notation (3.7). By Lemma 3.4, $(\mathbf{X}^{(1:\ell)}, \mathbf{X}^{(\ell+1:J)\mathsf{T}}) \in \Sigma_{\mathrm{S}}$. This means that $(\boldsymbol{\mathcal{C}}^i)_{i=1}^N := \varphi(\mathbf{X}^{(1:\ell)}, \mathbf{X}^{(\ell+1:J)\mathsf{T}}) \in \Gamma_{\mathcal{S}}$ where $\mathcal{S} = (\boldsymbol{\mathcal{S}}^i)_{i=1}^N := \varphi(\mathbf{S}_{\mathtt{bf}}^{(1:\ell)}, \mathbf{S}_{\mathtt{bf}}^{(\ell+1:J)\mathsf{T}})$. Consider any node $A$ in $T_X$ at level $J - \ell$ and any node $B$ in $T_\Omega$ at level $\ell$. By Remark 3.20 and (3.7), one can verify that there exists $i \in [N]$ such that the support of the $i$-th column of $\mathbf{S}_{\mathtt{bf}}^{(1:\ell)}$ and the support of the $i$-th row of $\mathbf{S}_{\mathtt{bf}}^{(\ell+1:J)}$ are respectively the node $A$ and $B$, which means that $\mathbf{Z} \odot \boldsymbol{\mathcal{S}}^i$ is the

submatrix $\mathbf{Z}_{A,B}$. But by Lemma 3.13, the rank-one supports $(\boldsymbol{\mathcal{S}}^i)_{i=1}^N$ are pairwise disjoint, so $\mathbf{Z} \odot \boldsymbol{\mathcal{S}}^i = (\sum_{i'=1}^N \boldsymbol{\mathcal{C}}^{i'}) \odot \boldsymbol{\mathcal{S}}^i = \boldsymbol{\mathcal{C}}^i$, which by definition is of rank one. This is true for any level $\ell$, any node $A$ in $T_X$ at level $J-\ell$, and any node $B$ in $T_\Omega$ at level $\ell$. By Definition 3.18, $\mathbf{Z}$ satisfies the complementary low-rank property for the trees $T_X$ and $T_\Omega$. ∎

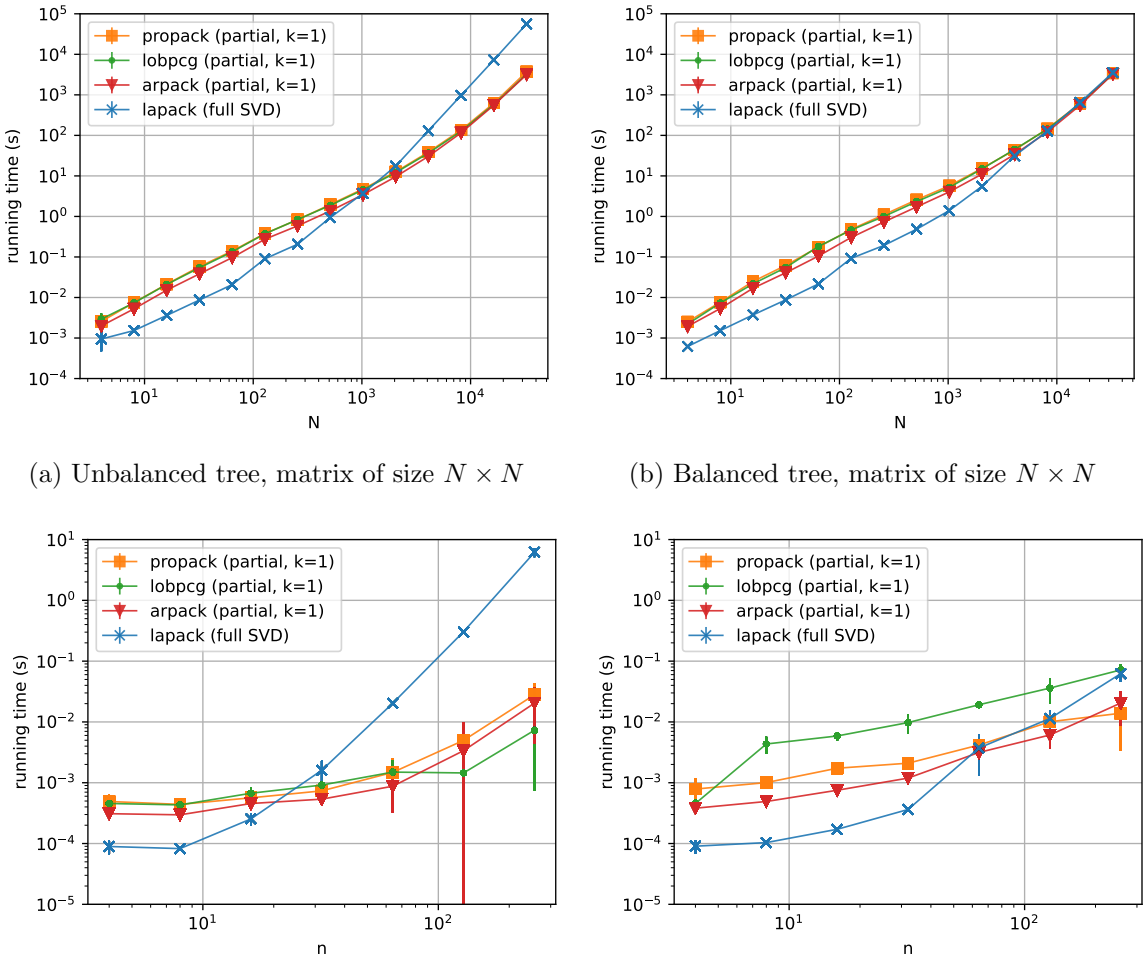
With the index-partitioning binary trees $T_X$ and $T_\Omega$ of Proposition 3.19, the butterfly algorithm from [40] (with non-randomized SVDs) is Algorithm 3.1 with the *symmetric* factor-bracketing binary tree $\mathcal{T}$ illustrated in Figure 3c and defined for $N = 2^J$ with an even integer $J$ as: (a) the children of the root node of $\mathcal{T}$ are $\{1, \dots, J/2\}$ and $\{J/2+1, \dots, J\}$; (b) in the left (resp. right) subtree of the root of $\mathcal{T}$, every right (resp. left) child of a non-leaf node is a singleton. Applying the procedure of line 7 in Algorithm 3.2 at the root node of $\mathcal{T}$ corresponds to the "middle-level factorization" in the butterfly algorithm [40], and applying successively this procedure at the other nodes of $\mathcal{T}$ corresponds to the "recursive factorization" in [40].

**4. Numerical experiments.** The empirical behavior of Algorithm 3.1 was first studied in [32], showing this superiority of the algorithm for solving (3.9) compared to gradient-based optimization [12], in terms of running time and approximation error, both in the exact and noisy setting. As an original contribution of this paper, the analysis in subsection 3.4 explains this improved running time, as we show that the total complexity of Algorithm 3.1 is $\mathcal{O}(N^2)$ using truncated SVDs. However, as empirically shown below, it is important to use an appropriate implementation of SVD in order to achieve this complexity bound.

*Protocol.* We implement Algorithm 3.1 in Python using the `Scipy 1.8.0` package to compute the SVD for the best rank-one approximation. We measure the running time of our implementation of Algorithm 3.1 to approximate $\mathbf{Z} = \mathbf{H} + \sigma\mathbf{W}$ as a product of $J$ butterfly factors, where $\mathbf{H}$ is the Hadamard matrix of size $N = 2^J$ with $J \in \{2, \dots, 15\}$, $\mathbf{W}$ is a random matrix with i.i.d. standard Gaussian entries (zero mean and variance equal to 1), and $\sigma = 0.01$. We use different SVD solvers in our experiments, among: LAPACK, ARPACK, PROPACK, LOBPCG. The first one performs a full SVD before truncating it, while the last ones perform partial SVD at a given order $k$ ($k = 1$ in our case). Experiments are run on an Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz. In the interest of reproducible research, our implementation is available in open source [63]. An optimized C++ implementation with Python and Matlab wrappers is also provided in the FAµST 3.25 toolbox at https://faust.inria.fr/.

*Comparing different solvers.* We compare the running time of the hierarchical factorization implemented with different SVD solvers. In Figure 4a, the unbalanced hierarchical factorization is faster with the LAPACK solver for $N \leq 1024$, while the other solvers are faster for $N \geq 1024$. In Figure 4b, the balanced hierarchical factorization is faster with the LAPACK solver for $N \leq 4096$, and all the solvers have a similar running time for $N \geq 4096$. The difference in running time can be empirically explained by our benchmark of these SVD solvers on a rectangular matrix (size $2 \times n^2/2$) in Figure 4c, and on a square matrix (size $n \times n$) in Figure 4d. These matrix sizes correspond to the size of submatrices on which an SVD is performed in the hierarchical algorithm. We indeed observe that the full SVD with LAPACK is faster than partial SVDs with ARPACK, PROPACK, LOBPCG for lower matrix size.

*Comparing balanced and unbalanced tree.* We now compare the running time of the hierarchical factorization algorithm with the unbalanced and balanced tree. In Figure 5a, the comparison is performed in a setting with small matrix size with $N \leq 1024$, using the LA-




(a) Unbalanced tree, matrix of size $N \times N$

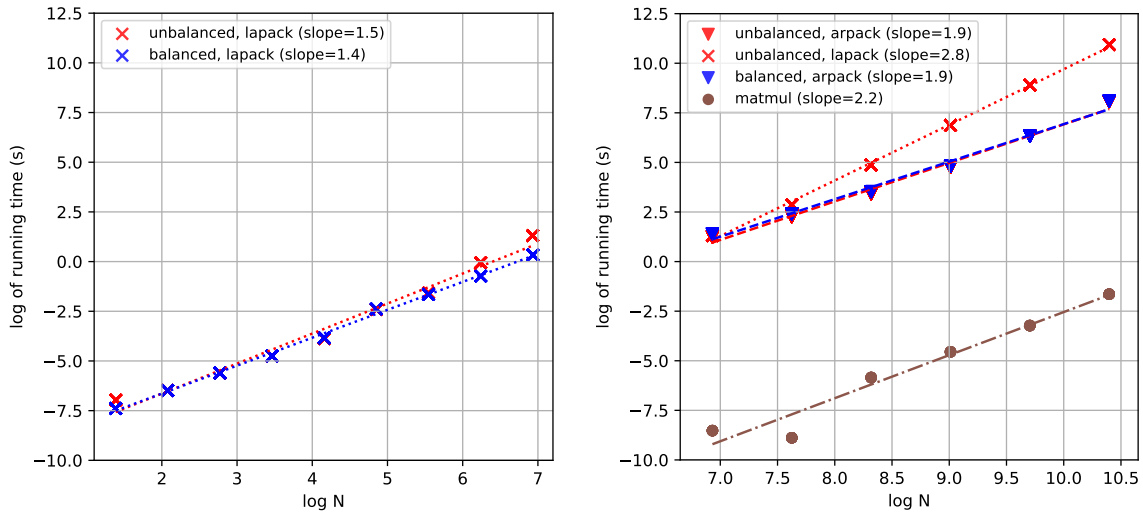
(b) Balanced tree, matrix of size $N \times N$

(c) SVD for matrices of size $2 \times n^2/2$

(d) SVD for matrices of size $n \times n$

Figure 4: Comparing different solvers - running time of our implementation of Algorithm 3.1. For a given matrix size, each factorization (resp. SVD) is repeated 3 (resp. 10) times in order to plot the mean running time with a vertical error bar showing the standard deviation.

PACK solver to compute full SVDs. Hierarchical factorization is slightly faster with a balanced tree compared to an unbalanced tree. In Figure 5b, the comparison is performed in a setting with large matrix size with $N \geq 1024$. When using the ARPACK solver, the running time for hierarchical factorization is the same for both trees. This is coherent with our analysis of complexity bounds for Algorithm 3.1, which is $\mathcal{O}(N^2)$ for both trees. For completeness Figure 5b also compares the running time of hierarchical factorization with the one of matrix-vector multiplication for a matrix of size $N \times N$, whose complexity is also $\mathcal{O}(N^2)$. Finally we see in Figure 5b that using an unbalanced tree with the LAPACK solver for large matrix size, as done in experiments of [32], is not optimal. This explains why it was observed in




(a) Small matrix size $N \times N$

(b) Large matrix size $N \times N$

Figure 5: Comparing balanced and unbalanced trees - running time of our implementation of Algorithm 3.1, in logarithmic scale. For better visualization a least-square regression is performed for each set of measurements, and we report the estimated slope $a$ of the regression line $Y = aX + b$. For slope comparison, we measure the running time for matrix-vector multiplication, computed with `numpy.matmul` method from `Numpy 1.22.3`.

[32] that a balanced tree is preferable to an unbalanced one, while the finer theoretical and computational analysis performed here leads to a different conclusion.

**5. Conclusion.** We established hierarchical identifiability in the butterfly sparse matrix factorization. We proved that the butterfly factors $(\mathbf{X}^{(\ell)})_{\ell=1}^{J}$ of $\mathbf{Z} := \mathbf{X}^{(1)} \dots \mathbf{X}^{(J)}$ can be recovered up to scaling ambiguities from $\mathbf{Z}$ with a hierarchical factorization method, described by Algorithm 3.1, which is endowed with exact recovery guarantees, and has controlled time complexity of $\mathcal{O}(N^2)$. We now provide some interesting future research directions.

*Stability results.* As discussed in Remark 3.8, a challenge is to explore stability properties of the matrix factorization problem with a butterfly structure, to ensure robustness of Algorithm 3.1 with respect to noise. Such guarantees would justify why Algorithm 3.1 still performs well in some noisy settings, as shown empirically in [32]. As the balanced and unbalanced variants of the proposed hierarchical method have similar computation times, it would be interesting to determine if their stability to noise is also similar.

*Implementation of the hierarchical algorithm.* Our benchmark of different solvers of SVD in Figures 4c and 4d suggests that our implementation of the hierarchical algorithm can be further improved by choosing the optimal SVD solver depending on the block's dimension. One could also envision randomized SVDs in the spirit of the butterfly algorithm [40].

*Recovering latent sparse butterfly factors.* When a data matrix is generated from the product of some latent butterfly factors, our identifiability results for butterfly sparse matrix

factorization (Theorem 3.10) could be useful to recover with minimal ambiguity these latent factors. One example of such a setting is neural network distillation, where a dense pre-trained teacher network, assumed to involve latent butterfly factors, is distilled into a sparse student network with butterfly structure. In this scenario, identifying the student butterfly factors can be based on sampling of the teacher network's derivatives, in the spirit of [19, 18].

*Identifiability results for $J = 2$ factors.* Our analysis of identifiability using the hierarchical approach relies on identifiability results in the case with $J = 2$ factors. This motivates further explorations of identifiability conditions in this setting. Our work focuses on fixed-support constraints, and we considered in Proposition 2.6 the case of disjoint rank-one supports. More relaxed conditions can be envisioned, such as supports satisfying the so-called *complete equivalence class* condition introduced in [31]. Beyond the fixed-support setting, one can also explore essential uniqueness by considering a *family* of sparsity patterns like in [62]. One can then use results from matrix completion literature [17, 27, 10] to establish more elaborate conditions for identifiability in the case with two factors.

**Acknowledgments.** The authors thank Hakim Hadj-Djilani for his valuable help in our numerical experiments and the implementation of Algorithm 3.1 in the FAµST 3.25 toolbox. The authors gratefully acknowledge the support of the Centre Blaise Pascal's IT test platform at ENS de Lyon for Machine Learning facilities. The platform operates the SIDUS solution [54] developed by Emmanuel Quemener. The authors also thank the anonymous reviewers for the insightful comments and suggestions during the revision of the manuscript.

## REFERENCES

[1] A. Ahmed, B. Recht, and J. Romberg, *Blind deconvolution using convex programming*, IEEE Transactions on Information Theory, 60 (2013), pp. 1711–1732, https://doi.org/10.1109/TIT.2013.2294644.

[2] S. Bahmani and J. Romberg, *Lifting for blind deconvolution in random mask imaging: Identifiability and convex relaxation*, SIAM Journal on Imaging Sciences, 8 (2015), pp. 2203–2238, https://doi.org/10.1137/141002165.

[3] E. Candes, L. Demanet, and L. Ying, *A fast butterfly algorithm for the computation of Fourier integral operators*, Multiscale Modeling & Simulation, 7 (2009), pp. 1727–1750, https://doi.org/10.1137/080734339.

[4] E. J. Candes, Y. C. Eldar, T. Strohmer, and V. Voroninski, *Phase retrieval via matrix completion*, SIAM review, 57 (2015), pp. 225–251, https://doi.org/10.1137/151005099.

[5] E. J. Candes, T. Strohmer, and V. Voroninski, *Phaselift: Exact and stable signal recovery from magnitude measurements via convex programming*, Communications on Pure and Applied Mathematics, 66 (2013), pp. 1241–1274, https://doi.org/10.1002/cpa.21432.

[6] B. Chen, T. Dao, K. Liang, J. Yang, Z. Song, A. Rudra, and C. Re, *Pixelated butterfly: Simple and efficient sparse training for neural network models*, in International Conference on Learning Representations, 2022, https://openreview.net/forum?id=Nfl-iXa-y7R.

[7] K. Choromanski, M. Rowland, W. Chen, and A. Weller, *Unifying orthogonal Monte Carlo methods*, in International Conference on Machine Learning, PMLR, 2019, pp. 1203–1212, https://proceedings.mlr.press/v97/choromanski19a/choromanski19a.pdf.

[8] S. Choudhary and U. Mitra, *Identifiability scaling laws in bilinear inverse problems*, arXiv preprint arXiv:1402.2637, (2014), https://arxiv.org/abs/1402.2637.

[9] S. Choudhary and U. Mitra, *Sparse blind deconvolution: What cannot be done*, in 2014 IEEE International Symposium on Information Theory, IEEE, 2014, pp. 3002–3006, https://doi.org/10.1109/ISIT.2014.6875385.

[10] A. Cosse and L. Demanet, *Stable rank-one matrix completion is solved by the level 2 Lasserre re-*

*laxation*, Foundations of Computational Mathematics, (2020), pp. 1–50, https://doi.org/10.1007/s10208-020-09471-y.

[11] T. Dao, B. Chen, N. S. Sohoni, A. D. Desai, M. Poli, J. Grogan, A. Liu, A. Rao, A. Rudra, and C. Ré, *Monarch: Expressive structured matrices for efficient and accurate training*, in International Conference on Machine Learning, PMLR, 2022, pp. 4690–4721, https://proceedings.mlr.press/v162/dao22a.html.

[12] T. Dao, A. Gu, M. Eichhorn, A. Rudra, and C. Ré, *Learning fast algorithms for linear transforms using butterfly factorizations*, in International Conference on Machine Learning, PMLR, 2019, pp. 1517–1527, http://proceedings.mlr.press/v97/dao19a/dao19a.pdf.

[13] T. Dao, N. Sohoni, A. Gu, M. Eichhorn, A. Blonder, M. Leszczynski, A. Rudra, and C. Ré, *Kaleidoscope: An efficient, learnable representation for all structured linear maps*, in International Conference on Learning Representations, 2020, https://openreview.net/forum?id=BkgrBgSYDS.

[14] J. Devlin, M. Chang, K. Lee, and K. Toutanova, *BERT: pre-training of deep bidirectional transformers for language understanding*, in North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, 2019, pp. 4171–4186, https://doi.org/10.18653/v1/n19-1423, https://doi.org/10.18653/v1/n19-1423.

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, *An image is worth 16x16 words: Transformers for image recognition at scale*, in International Conference on Learning Representations, 2021, https://openreview.net/forum?id=YicbFdNTTy.

[16] M. Elad, *Sparse and redundant representations: from theory to applications in signal and image processing*, Springer Science & Business Media, 2010, https://doi.org/10.1007/978-1-4419-7011-4.

[17] Y. C. Eldar, D. Needell, and Y. Plan, *Uniqueness conditions for low-rank matrix recovery*, Applied and Computational Harmonic Analysis, 33 (2012), pp. 309–314, https://doi.org/10.1016/j.acha.2012.04.002.

[18] M. Fornasier, T. Klock, and M. Rauchensteiner, *Robust and resource-efficient identification of two hidden layer neural networks*, Constructive Approximation, 55 (2022), pp. 475–536, https://doi.org/10.1007/s00365-021-09550-5.

[19] M. Fornasier, J. Vybíral, and I. Daubechies, *Robust and resource efficient identification of shallow neural networks by fewest samples*, Information and Inference: A Journal of the IMA, 10 (2021), pp. 625–695, https://doi.org/10.1093/imaiai/iaaa036.

[20] S. Foucart and H. Rauhut, *A mathematical introduction to compressive sensing*, Applied and Numerical Harmonic Analysis, Birkhäuser Basel, 2013, https://doi.org/10.1007/978-0-8176-4948-7.

[21] A. Genz, *Methods for generating random orthogonal matrices*, in Monte-Carlo and Quasi-Monte Carlo Methods 1998, Springer, 2000, pp. 199–213, https://doi.org/10.1007/978-3-642-59657-5_13.

[22] N. Gillis, *Nonnegative matrix factorization*, SIAM, 2020, https://doi.org/10.1137/1.9781611976410.

[23] W. Hackbusch and S. Börm, *Data-sparse approximation by adaptive H2-matrices*, Computing, 69 (2002), pp. 1–35, https://doi.org/10.1007/s00607-002-1450-4.

[24] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM review, 53 (2011), pp. 217–288, https://doi.org/10.1137/090771806.

[25] L. Jing, Y. Shen, T. Dubcek, J. Peurifoy, S. Skirlo, Y. LeCun, M. Tegmark, and M. Soljačić, *Tunable efficient unitary neural networks and their application to RNNs*, in International Conference on Machine Learning, PMLR, 2017, pp. 1733–1741, https://proceedings.mlr.press/v70/jing17a/jing17a.pdf.

[26] M. Kech and F. Krahmer, *Optimal injectivity conditions for bilinear inverse problems with applications to identifiability of deconvolution problems*, SIAM Journal on Applied Algebra and Geometry, 1 (2017), pp. 20–37, https://doi.org/10.1137/16M1067469.

[27] F. J. Király, L. Theran, and R. Tomioka, *The algebraic combinatorial approach for low-rank matrix completion.*, J. Mach. Learn. Res., 16 (2015), pp. 1391–1436, https://dl.acm.org/doi/10.5555/2789272.2886794.

[28] J. B. Kruskal, *Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics*, Linear algebra and its applications, 18 (1977), pp. 95–138, https://doi.org/10.1016/0024-3795(77)90069-6.

[29] Q. Le, T. Sarlos, and A. Smola, *Fastfood - computing hilbert space expansions in loglinear time*, in International Conference on Machine Learning, PMLR, 2013, pp. 244–252, https://proceedings.mlr.press/v28/le13.html.

[30] Q.-T. Le and R. Gribonval, *Structured support exploration for multilayer sparse matrix factorization*, in ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021, pp. 3245–3249, https://doi.org/10.1109/ICASSP39728.2021.9414238.

[31] Q.-T. Le, E. Riccietti, and R. Gribonval, *Spurious Valleys, Spurious Minima and NP-hardness of Sparse Matrix Factorization With Fixed Support*. preprint, 2022, https://hal.inria.fr/hal-03364668.

[32] Q.-T. Le, L. Zheng, E. Riccietti, and R. Gribonval, *Fast learning of fast transforms, with guarantees*, in ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing, Singapore, Singapore, May 2022, https://hal.inria.fr/hal-03438881.

[33] L. Le Magoarou, *Matrices efficientes pour le traitement du signal et l'apprentissage automatique*, PhD thesis, INSA de Rennes, 2016, https://hal.inria.fr/tel-01412558. Written in French.

[34] L. Le Magoarou and R. Gribonval, *Chasing butterflies: In search of efficient dictionaries*, in 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 3287–3291, https://doi.org/10.1109/ICASSP.2015.7178579.

[35] L. Le Magoarou and R. Gribonval, *Flexible multilayer sparse approximations of matrices and applications*, IEEE Journal of Selected Topics in Signal Processing, 10 (2016), pp. 688–700, https://doi.org/10.1109/JSTSP.2016.2543461.

[36] X. Li and V. Voroninski, *Sparse signal recovery from quadratic measurements via convex programming*, SIAM Journal on Mathematical Analysis, 45 (2013), pp. 3019–3033, https://doi.org/10.1137/120893707.

[37] Y. Li, K. Lee, and Y. Bresler, *Identifiability in bilinear inverse problems with applications to subspace or sparsity-constrained blind gain and phase calibration*, IEEE Transactions on Information Theory, 63 (2016), pp. 822–842, https://doi.org/10.1109/TIT.2016.2637933.

[38] Y. Li, K. Lee, and Y. Bresler, *Identifiability in blind deconvolution with subspace or sparsity constraints*, IEEE Transactions on information Theory, 62 (2016), pp. 4266–4275, https://doi.org/10.1109/TIT.2016.2569578.

[39] Y. Li, K. Lee, and Y. Bresler, *Identifiability and stability in blind deconvolution under minimal assumptions*, IEEE Transactions on Information Theory, 63 (2017), pp. 4619–4633, https://doi.org/10.1109/TIT.2017.2689779.

[40] Y. Li, H. Yang, E. R. Martin, K. L. Ho, and L. Ying, *Butterfly factorization*, Multiscale Modeling & Simulation, 13 (2015), pp. 714–732, https://doi.org/10.1137/15M1007173.

[41] R. Lin, J. Ran, K. H. Chiu, G. Chesi, and N. Wong, *Deformable butterfly: A highly structured and sparse linear transform*, Advances in Neural Information Processing Systems, 34 (2021), pp. 16145–16157, https://proceedings.neurips.cc/paper/2021/file/86b122d4358357d834a87ce618a55de0-Paper.pdf.

[42] S. Ling and T. Strohmer, *Self-calibration and biconvex compressive sensing*, Inverse Problems, 31 (2015), p. 115002, https://doi.org/10.1088/0266-5611/31/11/115002.

[43] F. Malgouyres, *On the stable recovery of deep structured linear networks under sparsity constraints*, in Mathematical and Scientific Machine Learning, PMLR, 2020, pp. 107–127, http://proceedings.mlr.press/v107/malgouyres20a/malgouyres20a.pdf.

[44] F. Malgouyres and J. Landsberg, *On the identifiability and stable recovery of deep/multi-layer structured matrix factorization*, in 2016 IEEE Information Theory Workshop (ITW), IEEE, 2016, pp. 315–319, https://doi.org/10.1109/ITW.2016.7606847.

[45] F. Malgouyres and J. Landsberg, *Multilinear compressive sensing and an application to convolutional linear networks*, SIAM Journal on Mathematics of Data Science, 1 (2019), pp. 446–475, https://doi.org/10.1137/18M119834X.

[46] P.-G. Martinsson, *A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix*, SIAM Journal on Matrix Analysis and Applications, 32 (2011), pp. 1251–1274, https://doi.org/10.1137/100786617.

[47] M. Mathieu and Y. LeCun, *Fast approximation of rotations and Hessians matrices*, arXiv preprint arXiv:1404.7195, (2014), https://arxiv.org/abs/1404.7195.

[48] E. Michielssen and A. Boag, *A multilevel matrix decomposition algorithm for analyzing scattering*

*from large structures*, IEEE Transactions on Antennas and Propagation, 44 (1996), pp. 1086–1093, https://doi.org/10.1109/8.511816.

[49] M. Munkhoeva, Y. Kapushev, E. Burnaev, and I. Oseledets, *Quadrature-based features for kernel approximation*, Advances in neural information processing systems, 31 (2018), https://proceedings.neurips.cc/paper/2018/file/6e923226e43cd6fac7cfe1e13ad000ac-Paper.pdf.

[50] M. O'Neil, F. Woolfe, and V. Rokhlin, *An algorithm for the rapid evaluation of special function transforms*, Applied and Computational Harmonic Analysis, 28 (2010), pp. 203–226, https://doi.org/10.1016/j.acha.2009.08.005.

[51] V. Pan, *Structured matrices and polynomials: unified superfast algorithms*, Springer Science & Business Media, 2001, https://doi.org/10.1007/978-1-4612-0129-8.

[52] N. Parikh, S. Boyd, et al., *Proximal algorithms*, Foundations and trends® in Optimization, 1 (2014), pp. 127–239, https://doi.org/10.1561/2400000003.

[53] D. S. Parker, *Random butterfly transformations with applications in computational linear algebra*, tech. report, 1995, https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.7932&rep=rep1&type=pdf.

[54] E. Quemener and M. Corvellec, *Sidus—the solution for extreme deduplication of an operating system*, Linux J., 2013 (2013), https://dl.acm.org/doi/abs/10.5555/2555789.2555792.

[55] R. Rubinstein, M. Zibulevsky, and M. Elad, *Double sparsity: Learning sparse dictionaries for sparse signal approximation*, IEEE Transactions on Signal Processing, 58 (2010), pp. 1553–1564, https://doi.org/10.1109/TSP.2009.2036477.

[56] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, et al., *MLP-mixer: An all-MLP architecture for vision*, Advances in Neural Information Processing Systems, 34 (2021), pp. 24261–24272, https://proceedings.neurips.cc/paper/2021/file/cba0a4ee5ccd02fda0fe3f9a3e7b89fe-Paper.pdf.

[57] I. Tošić and P. Frossard, *Dictionary learning*, IEEE Signal Processing Magazine, 28 (2011), pp. 27–38, https://doi.org/10.1109/MSP.2010.939537.

[58] C. F. Van Loan, *The ubiquitous Kronecker product*, Journal of Computational and Applied Mathematics, 123 (2000), pp. 85–100, https://doi.org/10.1016/s0377-0427(00)00393-9.

[59] H. Wolfgang, *A sparse matrix arithmetic based on H-matrices. Part I: Introduction to h-matrices*, Computing, 62 (1999), pp. 89–108, https://doi.org/10.1007/s006070050015.

[60] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert, *A fast randomized algorithm for the approximation of matrices*, Applied and Computational Harmonic Analysis, 25 (2008), pp. 335–366, https://doi.org/10.1016/j.acha.2007.12.002.

[61] K. Ye and L.-H. Lim, *Every matrix is a product of Toeplitz matrices*, Foundations of Computational Mathematics, 16 (2016), pp. 577–598, https://doi.org/10.1007/s10208-015-9254-z.

[62] L. Zheng, E. Riccietti, and R. Gribonval, *Identifiability in Two-Layer Sparse Matrix Factorization*, arXiv preprint arXiv:2110.01235, (2021), https://arxiv.org/abs/2110.01235.

[63] L. Zheng, E. Riccietti, and R. Gribonval, *Code for reproducible research - Efficient Identification of Butterfly Sparse Matrix Factorizations*, Mar. 2022, https://hal.inria.fr/hal-03620052. Code repository, updates at https://github.com/leonzheng2/efficient-butterfly.

## Appendix A. Proof of Lemma 2.3.

We begin with a simple lemma [22, Chapter 3].

Lemma A.1. *Let $\Sigma$ be any set of pairs of factors, and $(\mathbf{X}, \mathbf{Y}) \in \Sigma$. We have*

$$(\mathbf{X}, \mathbf{Y}) \in \mathcal{U}(\Sigma) \iff (\mathbf{X}, \mathbf{Y}) \in \bigcap_{\substack{\Sigma' \subseteq \Sigma \\ (\mathbf{X}, \mathbf{Y}) \in \Sigma'}} \mathcal{U}(\Sigma').$$

*Proof.* Let $(\mathbf{X}, \mathbf{Y}) \in \mathcal{U}(\Sigma)$, and consider $\Sigma' \subseteq \Sigma$ such that $(\mathbf{X}, \mathbf{Y}) \in \Sigma'$, as well as $(\bar{\mathbf{X}}, \bar{\mathbf{Y}}) \in \Sigma'$ such that $\bar{\mathbf{X}}\bar{\mathbf{Y}}^\intercal = \mathbf{X}\mathbf{Y}^\intercal$. Since $\Sigma' \subseteq \Sigma$, we have $(\bar{\mathbf{X}}, \bar{\mathbf{Y}}) \in \Sigma$, and because $(\mathbf{X}, \mathbf{Y}) \in \mathcal{U}(\Sigma)$, $(\bar{\mathbf{X}}, \bar{\mathbf{Y}}) \sim (\mathbf{X}, \mathbf{Y})$. Moreover, $(\mathbf{X}, \mathbf{Y}) \in \Sigma'$, hence $(\mathbf{X}, \mathbf{Y}) \in \mathcal{U}(\Sigma')$. This is true for every $\Sigma' \subseteq \Sigma$, proving one implication. The converse is true considering the case $\Sigma' := \Sigma$. ∎

Lemma 2.3 is then derived from the following trivial but crucial observation.

Lemma A.2. *Let $\Sigma$ be any set of pairs of factors, and $\mathbf{X}$, $\mathbf{Y}, \bar{\mathbf{Y}}$ such that $(\mathbf{X}, \mathbf{Y}), (\mathbf{X}, \bar{\mathbf{Y}}) \in \Sigma$. If $\mathbf{X}\mathbf{Y}^\intercal = \mathbf{X}\bar{\mathbf{Y}}^\intercal$ and $\mathrm{colsupp}(\bar{\mathbf{Y}})$, $\mathrm{colsupp}(\mathbf{Y})$ do not have the same cardinality, then $(\mathbf{X}, \mathbf{Y}) \not\sim (\mathbf{X}, \bar{\mathbf{Y}})$ and hence $(\mathbf{X}, \mathbf{Y}) \notin \mathcal{U}(\Sigma)$ and $(\mathbf{X}, \bar{\mathbf{Y}}) \notin \mathcal{U}(\Sigma)$. This is true in particular if there exists an index $i \in [r]$ for which $\mathbf{X}_i = \mathbf{0}$, $\mathbf{Y}_i = \mathbf{0}$, $\bar{\mathbf{Y}}_i \neq \mathbf{0}$, and $\mathbf{Y}_j = \bar{\mathbf{Y}}_j$ for all $j \neq i$.*

*Proof of Lemma 2.3.* We first show $\mathcal{U}(\Sigma_{\mathrm{S}}) \subseteq \mathtt{IC}_{\mathrm{S}}$ by contraposition. Let $(\mathbf{X}, \mathbf{Y}) \in \Sigma_{\mathrm{S}}$, and suppose that $\mathrm{colsupp}(\mathbf{X}) \neq \mathrm{colsupp}(\mathbf{Y})$. Up to matrix transposition, we can suppose without loss of generality that $\mathrm{colsupp}(\mathbf{Y})$ is not a subset of $\mathrm{colsupp}(\mathbf{X})$, so there is $i \in [r]$ such that $\mathbf{X}_i = \mathbf{0}$ and $\mathbf{Y}_i \neq \mathbf{0}$. Define $\bar{\mathbf{Y}}$ a right factor such that $\bar{\mathbf{Y}}_i = \mathbf{0}$ and $\bar{\mathbf{Y}}_{[r]\setminus\{i\}} = \mathbf{Y}_{[r]\setminus\{i\}}$ [7]. By construction, $\mathrm{supp}(\bar{\mathbf{Y}}) \subseteq \mathrm{supp}(\mathbf{Y})$, so $(\mathbf{X}, \bar{\mathbf{Y}}) \in \Sigma_{\mathrm{S}}$. Applying Lemma A.2 to $\Sigma = \Sigma_{\mathrm{S}}$, we obtain $(\mathbf{X}, \mathbf{Y}) \notin \mathcal{U}(\Sigma_{\mathrm{S}})$. We now show $\mathcal{U}(\Sigma_{\mathrm{S}}) \subseteq \mathtt{MC}_{\mathrm{S}}$, also by contraposition. Let $(\mathbf{X}, \mathbf{Y}) \notin \mathtt{MC}_{\mathrm{S}}$, and assume $\mathrm{colsupp}(\mathbf{X}) \neq \mathrm{colsupp}(\mathbf{S}^L)$. The reasoning would be symmetric if we supposed $\mathrm{colsupp}(\mathbf{Y}) \neq \mathrm{colsupp}(\mathbf{S}^R)$. By the previously shown inclusion $\mathcal{U}(\Sigma_{\mathrm{S}}) \subseteq \mathtt{IC}_{\mathrm{S}}$, we also assume $\mathrm{colsupp}(\mathbf{X}) = \mathrm{colsupp}(\mathbf{Y})$ without loss of generality. To conclude we treat three cases:

- If $\mathrm{colsupp}(\mathbf{S}^R) \not\subseteq \mathrm{colsupp}(\mathbf{S}^L)$ then we can fix $i \in [r]$ such that $\mathbf{S}_i^L = \mathbf{0}$ and $\mathbf{S}_i^R \neq \mathbf{0}$. This means $\mathbf{X}_i = \mathbf{Y}_i = \mathbf{0}$. Setting $\bar{\mathbf{Y}} \in \Sigma_{\mathbf{S}^R}$ such that $\bar{\mathbf{Y}}_i = \mathbf{S}_i^R$ and $\bar{\mathbf{Y}}_{[r]\setminus\{i\}} = \mathbf{Y}_{[r]\setminus\{i\}}$, we build an instance as in Lemma A.2 with $\Sigma = \Sigma_{\mathrm{S}}$, to show $(\mathbf{X}, \mathbf{Y}) \notin \mathcal{U}(\Sigma_{\mathrm{S}})$.
- If $\mathrm{colsupp}(\mathbf{S}^L) \not\subseteq \mathrm{colsupp}(\mathbf{S}^R)$, then the same arguments yield $(\mathbf{X}, \mathbf{Y}) \notin \mathcal{U}(\Sigma_{\mathrm{S}})$.
- There remains the case $\mathrm{colsupp}(\mathbf{S}^L) = \mathrm{colsupp}(\mathbf{S}^R)$. Since $\mathrm{colsupp}(\mathbf{X}) \subsetneq \mathrm{colsupp}(\mathbf{S}^L)$, let us fix $i \in \mathrm{colsupp}(\mathbf{S}^L)$ such that $\mathbf{X}_i = \mathbf{0}$. Then, $\mathbf{S}_i^L \neq 0$, $\mathbf{S}_i^R \neq 0$, and $\mathbf{X}_i = \mathbf{Y}_i = \mathbf{0}$. Again, we construct $\bar{\mathbf{Y}} \in \Sigma_{\mathbf{S}^R}$ with $\bar{\mathbf{Y}}_i = \mathbf{S}_i^R$, $\bar{\mathbf{Y}}_{[r]\setminus\{i\}} = \mathbf{Y}_{[r]\setminus\{i\}}$, and we obtain an instance as in Lemma A.2 with $\Sigma = \Sigma_{\mathrm{S}}$, showing that $(\mathbf{X}, \mathbf{Y}) \notin \mathcal{U}(\Sigma_{\mathrm{S}})$. ∎

The following key proposition will be useful in the lifting approach presented below.

Proposition A.3. *For any pair of supports $\mathrm{S}$, we have: $\mathcal{U}(\Sigma_{\mathrm{S}}) = \mathcal{U}(\mathtt{IC}_{\mathrm{S}}) \cap \mathtt{MC}_{\mathrm{S}}$.*

*Proof.* The direct inclusion is immediate by Lemmas 2.3 and A.1. For the inverse inclusion, let $(\mathbf{X}^*, \mathbf{Y}^*) \in \mathcal{U}(\mathtt{IC}_{\mathrm{S}}) \cap \mathtt{MC}_{\mathrm{S}}$, and $(\mathbf{X}, \mathbf{Y}) \in \Sigma_{\mathrm{S}}$ such that $\mathbf{X}\mathbf{Y}^\intercal = \mathbf{X}^*\mathbf{Y}^{*\intercal}$. The goal is to show $(\mathbf{X}, \mathbf{Y}) \sim (\mathbf{X}^*, \mathbf{Y}^*)$. Denote $J = \mathrm{colsupp}(\mathbf{X}) \cap \mathrm{colsupp}(\mathbf{Y})$. Define $(\bar{\mathbf{X}}, \bar{\mathbf{Y}}) \in \mathtt{IC}_{\mathrm{S}}$ such that $(\bar{\mathbf{X}}_J, \bar{\mathbf{Y}}_J) = (\mathbf{X}_J, \mathbf{Y}_J)$ and $(\bar{\mathbf{X}}_{[r]\setminus J}, \bar{\mathbf{Y}}_{[r]\setminus J}) = (\mathbf{0}, \mathbf{0})$. Since $\bar{\mathbf{X}}\bar{\mathbf{Y}}^\intercal = \mathbf{X}\mathbf{Y}^\intercal = \mathbf{X}^*\mathbf{Y}^{*\intercal}$, and $(\mathbf{X}^*, \mathbf{Y}^*) \in \mathcal{U}(\mathtt{IC}_{\mathrm{S}})$, we have $(\bar{\mathbf{X}}, \bar{\mathbf{Y}}) \sim (\mathbf{X}^*, \mathbf{Y}^*)$. But $\mathrm{colsupp}(\mathbf{X}^*) = \mathrm{colsupp}(\mathbf{S}^L)$ and $\mathrm{colsupp}(\mathbf{Y}^*) = \mathrm{colsupp}(\mathbf{S}^R)$, because $(\mathbf{X}^*, \mathbf{Y}^*) \in \mathtt{MC}_{\mathrm{S}}$. Hence, $J = \mathrm{colsupp}(\bar{\mathbf{X}}) = \mathrm{colsupp}(\mathbf{X}^*) = \mathrm{colsupp}(\mathbf{S}^L)$ and similarly $J = \mathrm{colsupp}(\mathbf{S}^R)$. This necessarily yields $\bar{\mathbf{X}} = \mathbf{X}$ and $\bar{\mathbf{Y}} = \mathbf{Y}$. In conclusion, $(\mathbf{X}, \mathbf{Y}) = (\bar{\mathbf{X}}, \bar{\mathbf{Y}}) \sim (\mathbf{X}^*, \mathbf{Y}^*)$. ∎

**Appendix B. Lifting procedure.** We start by claiming two technical lemmas, whose proof is left to the reader. Recall that $\mathtt{IC}_{\mathrm{S}}$, $\mathtt{MC}_{\mathrm{S}}$ and $\Gamma_{\mathcal{S}}$ are defined in (2.5), (2.6), and (2.8).

Lemma B.1. *Let $\mathrm{S}$ be any pair of supports satisfying $\mathrm{colsupp}(\mathbf{S}^L) = \mathrm{colsupp}(\mathbf{S}^R)$, and denote $\mathcal{S} := \varphi(\mathrm{S})$. Then: $(\mathbf{X}, \mathbf{Y}) \in \mathtt{IC}_{\mathrm{S}} \cap \mathtt{MC}_{\mathrm{S}} \iff \varphi(\mathbf{X}, \mathbf{Y}) \in \mathtt{MI}_{\mathcal{S}}$, where $\mathtt{MI}_{\mathcal{S}}$ denotes the set of tuples $\mathcal{C} \in \Gamma_{\mathcal{S}}$ with "maximal" index support [8]:*

$$\mathtt{MI}_{\mathcal{S}} := \{\mathcal{C} \in \Gamma_{\mathcal{S}} \mid \forall i \in [r], \boldsymbol{\mathcal{C}}^i = \mathbf{0} \implies \boldsymbol{\mathcal{S}}^i = \mathbf{0}\}. \tag{B.1}$$

[7] By abuse of notations, $\mathbf{M}_I$ is the submatrix of $\mathbf{M} \in \mathbb{C}^{m \times n}$ composed of columns of $\mathbf{M}$ indexed by $I \subset [n]$.

[8] By analogy with column support, the index support of $\mathcal{C} = (\boldsymbol{\mathcal{C}}^i)_{i=1}^r$ is the subset of indices $i \in [r]$ such that $\boldsymbol{\mathcal{C}}^i \neq \mathbf{0}$.

Lemma B.2. *The application $\varphi$ is invariant to column rescaling: $\varphi(\mathbf{X},\mathbf{Y}) = \varphi(\bar{\mathbf{X}},\bar{\mathbf{Y}})$ for any equivalent pair of factors $(\mathbf{X},\mathbf{Y}) \sim (\bar{\mathbf{X}},\bar{\mathbf{Y}})$. Moreover, the application $\varphi$ restricted to $\mathtt{IC}_\mathrm{S}$, denoted $\varphi_\mathrm{S}\colon \mathtt{IC}_\mathrm{S} \to \Gamma_\mathcal{S}$ is surjective, and injective up to equivalences, in the sense that $(\mathbf{X},\mathbf{Y}) \sim (\bar{\mathbf{X}},\bar{\mathbf{Y}})$ for any $(\mathbf{X},\mathbf{Y}),(\bar{\mathbf{X}},\bar{\mathbf{Y}}) \in \mathtt{IC}_\mathrm{S}$ such that $\varphi_\mathrm{S}(\mathbf{X},\mathbf{Y}) = \varphi_\mathrm{S}(\bar{\mathbf{X}},\bar{\mathbf{Y}})$.*

We now define the operator that sums the $r$ matrices of a tuple $\mathcal{C}$ as: $\mathcal{A}:\mathcal{C} = (\boldsymbol{\mathcal{C}}^i)_{i=1}^r \mapsto \sum_{i=1}^r \boldsymbol{\mathcal{C}}^i$. This operator corresponds to a lifting operator in the terminology of [8]. For any set $\Gamma \subseteq (\mathbb{C}^{m\times n})^r$ of $r$-tuples of rank-one matrices, denote (by slight abuse of notation) $\mathcal{U}(\Gamma) := \{\mathcal{C} \in \Gamma \mid \forall \mathcal{C}' \in \Gamma,\ \mathcal{A}(\mathcal{C}) = \mathcal{A}(\mathcal{C}') \implies \mathcal{C} = \mathcal{C}'\}$. The previous lemmas lead to the following theorem characterizing essential uniqueness of the factorization $\mathbf{Z} := \mathbf{X}\mathbf{Y}^\intercal$ in $\Sigma_\mathrm{S}$ as the identifiability of the rank-one contributions $\varphi(\mathbf{X},\mathbf{Y})$ from $\mathbf{Z}$ with the constraint set $\Gamma_\mathcal{S}$.

Theorem B.3. *For any pair of supports* S *such that* $\mathrm{colsupp}(\mathbf{S}^L) = \mathrm{colsupp}(\mathbf{S}^R)$*, denoting $\mathcal{S} := \varphi(\mathrm{S})$, we have: $(\mathbf{X},\mathbf{Y}) \in \mathcal{U}(\Sigma_\mathrm{S}) \iff \varphi(\mathbf{X},\mathbf{Y}) \in \mathcal{U}(\Gamma_\mathcal{S}) \cap \mathtt{MI}_\mathcal{S}$.*

*Proof.* By Lemma B.2, one verifies that $(\mathbf{X},\mathbf{Y}) \in \mathcal{U}(\mathtt{IC}_\mathrm{S}) \iff \varphi(\mathbf{X},\mathbf{Y}) \in \mathcal{U}(\Gamma_\mathcal{S})$ hence

$$\begin{aligned}(\mathbf{X},\mathbf{Y}) \in \mathcal{U}(\Sigma_\mathrm{S}) &\overset{\textit{Proposition A.3}}{\iff} (\mathbf{X},\mathbf{Y}) \in \mathcal{U}(\mathtt{IC}_\mathrm{S}) \cap \mathtt{MC}_\mathrm{S} \\ &\overset{\textit{Lemma B.2}}{\iff} \varphi(\mathbf{X},\mathbf{Y}) \in \mathcal{U}(\Gamma_\mathcal{S}) \text{ and } (\mathbf{X},\mathbf{Y}) \in \mathtt{IC}_\mathrm{S} \cap \mathtt{MC}_\mathrm{S} \\ &\overset{\textit{Lemma B.1}}{\iff} \varphi(\mathbf{X},\mathbf{Y}) \in \mathcal{U}(\Gamma_\mathcal{S}) \cap \mathtt{MI}_\mathcal{S}.\end{aligned}$$

∎

Corollary B.4. *For any pair of supports* S *such that* $\mathrm{colsupp}(\mathbf{S}^L) = \mathrm{colsupp}(\mathbf{S}^R)$*, denoting $\mathcal{S} := \varphi(\mathrm{S})$, we have: $\mathcal{U}(\Sigma_\mathrm{S}) = \mathtt{IC}_\mathrm{S} \cap \mathtt{MC}_\mathrm{S} \iff \mathtt{MI}_\mathcal{S} \subseteq \mathcal{U}(\Gamma_\mathcal{S})$.*

*Proof.* Suppose $\mathtt{MI}_\mathcal{S} \subseteq \mathcal{U}(\Gamma_\mathcal{S})$. Let $(\mathbf{X},\mathbf{Y}) \in \mathtt{IC}_\mathrm{S} \cap \mathtt{MC}_\mathrm{S}$. By Lemma B.1, $\varphi(\mathbf{X},\mathbf{Y}) \in \mathtt{MI}_\mathcal{S}$, which implies by assumption that $\varphi(\mathbf{X},\mathbf{Y}) \in \mathcal{U}(\Gamma_\mathcal{S}) \cap \mathtt{MI}_\mathcal{S}$. By Theorem B.3, $(\mathbf{X},\mathbf{Y}) \in \mathcal{U}(\Sigma_\mathrm{S})$. This shows $\mathtt{IC}_\mathrm{S} \cap \mathtt{MC}_\mathrm{S} \subseteq \mathcal{U}(\Sigma_\mathrm{S})$ and the converse inclusion holds by Lemma 2.3.

Now suppose $\mathcal{U}(\Sigma_\mathrm{S}) = \mathtt{IC}_\mathrm{S} \cap \mathtt{MC}_\mathrm{S}$. Let $\mathcal{C} \in \mathtt{MI}_\mathcal{S}$. By Lemma B.2, there is $(\mathbf{X},\mathbf{Y}) \in \mathtt{IC}_\mathrm{S}$ such that $\varphi(\mathbf{X},\mathbf{Y}) = \mathcal{C}$. Let $i \in \mathrm{colsupp}(\mathbf{S}^L)$. By assumption, $i \in \mathrm{colsupp}(\mathbf{S}^R)$ hence $\boldsymbol{\mathcal{S}}^i \neq \mathbf{0}$. Since $\mathcal{C} \in \mathtt{MI}_\mathcal{S}$, we have $\mathbf{X}_i\mathbf{Y}_i{}^\intercal = \boldsymbol{\mathcal{C}}^i \neq 0$, which means that $i \in \mathrm{colsupp}(\mathbf{X})$. This is true for any $i \in \mathrm{colsupp}(\mathbf{S}^L)$, so $\mathrm{colsupp}(\mathbf{X}) = \mathrm{colsupp}(\mathbf{S}^L)$. By definition of $\mathtt{IC}_\mathrm{S}$, $\mathrm{colsupp}(\mathbf{X}) = \mathrm{colsupp}(\mathbf{Y})$, and by assumption, $\mathrm{colsupp}(\mathbf{S}^L) = \mathrm{colsupp}(\mathbf{S}^R)$, hence $(\mathbf{X},\mathbf{Y}) \in \mathtt{MC}_\mathrm{S}$. Since we supposed $\mathcal{U}(\Sigma_\mathrm{S}) = \mathtt{IC}_\mathrm{S} \cap \mathtt{MC}_\mathrm{S}$, we get $(\mathbf{X},\mathbf{Y}) \in \mathcal{U}(\Sigma_\mathrm{S})$. By Theorem B.3, $\mathcal{C} = \varphi(\mathbf{X},\mathbf{Y}) \in \mathcal{U}(\Gamma_\mathcal{S})$. ∎

## Appendix C. Proof of Proposition 2.6.

*Proof.* By Corollary B.4, it is enough to show that $\mathtt{MI}_\mathcal{S} \subseteq \mathcal{U}(\Gamma_\mathcal{S})$ if, and only if, $\mathcal{S} := \varphi(\mathrm{S})$ has disjoint rank-one supports. Sufficiency is immediate: given $\mathcal{C} \in \mathtt{MI}_\mathcal{S}$ and $\mathbf{Z} := \mathcal{A}(\mathcal{C})$, the entries of $\boldsymbol{\mathcal{C}}^i$ $(1 \le i \le r)$ can be directly identified from the submatrix $\mathbf{Z} \odot \boldsymbol{\mathcal{S}}^i$, because the rank-one supports in the tuple $\mathcal{S}$ are pairwise disjoint. For necessity, suppose there are $i,j \in [r]$, $i \neq j$ such that $\mathrm{supp}(\boldsymbol{\mathcal{S}}^i) \cap \mathrm{supp}(\boldsymbol{\mathcal{S}}^j) \neq \emptyset$. Let $(k,l)$ be an index in this intersection. Denote $\mathbf{E}^{(k,l)} \in \mathbb{B}^{n\times m}$ the canonical binary matrix full of zeros except a one at index $(k,l)$. Observe that $\mathcal{A}(\mathcal{C}) = \sum_{t\in[r]\setminus\{i,j\}} \boldsymbol{\mathcal{S}}^t = \mathcal{A}(\bar{\mathcal{C}})$, but $\mathcal{C} \neq \bar{\mathcal{C}}$ with $\mathcal{C},\bar{\mathcal{C}} \in \mathtt{MI}_\mathcal{S} \subseteq \Gamma_\mathcal{S}$ defined as

$$\forall t \in [r], \quad \mathcal{C}^t = \begin{cases} \mathbf{0} & \text{if } \mathcal{S}^t = \mathbf{0} \\ \mathbf{E}^{(k,l)} & \text{if } t = i \\ -\mathbf{E}^{(k,l)} & \text{if } t = j \\ \mathcal{S}^t & \text{otherwise} \end{cases}, \quad \bar{\mathcal{C}}^t = \begin{cases} \mathbf{0} & \text{if } \mathcal{S}^t = \mathbf{0} \\ 2\mathbf{E}^{(k,l)} & \text{if } t = i \\ -2\mathbf{E}^{(k,l)} & \text{if } t = j \\ \mathcal{S}^t & \text{otherwise} \end{cases}.$$

This shows $\mathcal{C}, \bar{\mathcal{C}} \neq \mathcal{U}(\Gamma_{\mathcal{S}})$, hence, $\mathtt{MI}_{\mathcal{S}} \not\subseteq \mathcal{U}(\Gamma_{\mathcal{S}})$, which ends the proof by contraposition. ∎

## Appendix D. Proof of Lemma 3.4.

The proof of Lemma 3.4 follows from Remark 3.5 and the following lemma.

Lemma D.1. *Given two matrix suports* $\mathbf{S}^{(1)} \in \mathbb{B}^{m \times r}$ *and* $\mathbf{S}^{(2)} \in \mathbb{B}^{r \times n}$*, for any pair of factors* $(\mathbf{X}^{(1)}, \mathbf{X}^{(2)}) \in \Sigma_{\mathbf{S}^{(1)}} \times \Sigma_{\mathbf{S}^{(2)}}$*, we have* $\operatorname{supp}(\mathbf{X}^{(1)}\mathbf{X}^{(2)}) \subseteq \operatorname{supp}(\mathbf{S}^{(1)}\mathbf{S}^{(2)})$.

*Proof.* If $(i,j) \notin \operatorname{supp}(\mathbf{S}^{(1)}\mathbf{S}^{(2)})$ then $0 = (\mathbf{S}^{(1)}\mathbf{S}^{(2)})_{i,j} = \sum_{k=1}^{r} {\mathbf{S}^{(1)}}_{i,k}{\mathbf{S}^{(2)}}_{k,j}$. As $\mathbf{S}^{(1)}$, $\mathbf{S}^{(2)}$ are binary, this implies that for each $k \in [r]$, ${\mathbf{S}^{(1)}}_{i,k} = 0$ or ${\mathbf{S}^{(2)}}_{k,j} = 0$. Since $\operatorname{supp}(\mathbf{X}^{(\ell)}) \subseteq \mathbf{S}^{(\ell)}$ for $\ell \in \{1,2\}$, we obtain ${\mathbf{X}^{(1)}}_{i,k} = 0$ or ${\mathbf{X}^{(2)}}_{k,j} = 0$ for each $k \in [r]$. This yields $\left(\mathbf{X}^{(1)}\mathbf{X}^{(2)}\right)_{i,j} = \sum_{k=1}^{r} {\mathbf{X}^{(1)}}_{i,k}{\mathbf{X}^{(2)}}_{k,j} = 0$ hence $(i,j) \notin \operatorname{supp}(\mathbf{X}^{(1)}\mathbf{X}^{(2)})$. We conclude by contraposition. ∎

*Proof of Lemma 3.4.* We start by the case $p = 1$ and show that $\operatorname{supp}(\mathbf{X}^{(1)} \dots \mathbf{X}^{(\ell)}) \subseteq \mathbf{V}^{(1,\ell)}$ for all $1 \leq \ell \leq J$ by induction. This is true for $\ell = 1$, because by (3.1) and (3.8), $\operatorname{supp}(\mathbf{X}^{(1)}) \subseteq \mathbf{S}^{(1)}_{\mathtt{bf}} = \left[\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}\right] \otimes \mathbf{I}_{2^{J-1}} = \mathbf{V}^{(1,1)}$. Suppose now that $1 \leq \ell \leq J-1$ and $\operatorname{supp}(\mathbf{X}^{(1)} \dots \mathbf{X}^{(\ell)}) \subseteq \mathbf{V}^{(1,\ell)}$, which is the matrix full of blocks $\mathbf{I}_{N/2^\ell}$. Since $\operatorname{supp}(\mathbf{X}^{(\ell+1)}) \subseteq \mathbf{S}^{(\ell+1)}_{\mathtt{bf}}$, by Lemma D.1, we have $\operatorname{supp}(\mathbf{X}^{(1)} \dots \mathbf{X}^{(\ell)}\mathbf{X}^{(\ell+1)}) \subseteq \operatorname{supp}(\mathbf{V}^{(1,\ell)}\mathbf{S}^{(\ell+1)}_{\mathtt{bf}})$. As $\mathbf{S}^{(\ell+1)}_{\mathtt{bf}}$ is block diagonal with blocks of size $\frac{N}{2^\ell} \times \frac{N}{2^\ell}$ equal to $\mathbf{U}_2 \otimes \mathbf{I}_{N/2^{\ell+1}} = \begin{bmatrix} \mathbf{I}_{N/2^{\ell+1}} & \mathbf{I}_{N/2^{\ell+1}} \\ \mathbf{I}_{N/2^{\ell+1}} & \mathbf{I}_{N/2^{\ell+1}} \end{bmatrix}$, we get:

$$\begin{aligned} \mathbf{V}^{(1,\ell)}\mathbf{S}^{(\ell+1)}_{\mathtt{bf}} &= \left(\mathbf{U}_{2^\ell} \otimes \mathbf{I}_{N/2^\ell}\right)\left(\mathbf{I}_{2^\ell} \otimes \mathbf{U}_2 \otimes \mathbf{I}_{N/2^{\ell+1}}\right) \\ &\overset{(1.2)}{=} (\mathbf{U}_{2^\ell}\mathbf{I}_{2^\ell}) \otimes \left(\mathbf{I}_{N/2^\ell}\left(\mathbf{U}_2 \otimes \mathbf{I}_{N/2^{\ell+1}}\right)\right) \\ &\overset{(1.2)}{=} \mathbf{U}_{2^\ell} \otimes \mathbf{U}_2 \otimes \mathbf{I}_{N/2^{\ell+1}} = \mathbf{U}_{2^{\ell+1}} \otimes \mathbf{I}_{N/2^{\ell+1}} = \mathbf{V}^{(1,\ell+1)}. \end{aligned}$$

Consequently, $\operatorname{supp}(\mathbf{X}^{(1)} \dots \mathbf{X}^{(\ell)}\mathbf{X}^{(\ell+1)}) \subseteq \operatorname{supp}(\mathbf{V}^{(1,\ell)}\mathbf{S}^{(\ell+1)}_{\mathtt{bf}}) = \operatorname{supp}(\mathbf{V}^{(1,\ell+1)}) = \mathbf{V}^{(1,\ell+1)}$. This ends the induction. Now, in the case $p > 1$, the support $\mathbf{S}^{(p)}_{\mathtt{bf}}$ of size $N \times N$ is block diagonal, where each block is the leftmost butterfly support of size $N/2^{p-1} \times N/2^{p-1}$. Applying the previous case to each of these blocks of size $N/2^{p-1} \times N/2^{p-1}$ yields the claimed result. ∎

## Appendix E. Complexity bounds of Algorithm 3.1 with full SVDs.

*Unbalanced tree.* At the (unique) non-leaf node of depth $j \in \{0, \dots, J-2\}$, the algorithm computes the best rank-one approximation of $N$ submatrices of size $2 \times N/2^{j+1}$. Using a full SVD costs of the order of $2 \times N/2^{j+1} \times 2 = 2 \times N/2^j$, hence the total cost (with unbalanced tree and full SVD) is $\sum_{j=0}^{J-2} N \times 2 \times \frac{N}{2^j} = 4(1 - 2^{-J+1})N^2 = 4\left(1 - \frac{2}{N}\right)N^2 = \mathcal{O}(N^2)$.

*Balanced tree.* At each of the $2^k$ non-leaf nodes of depth $k \in \{0, \dots, \log_2(J) - 1\}$, the best rank-one approximation of $N$ square submatrices of size $\sqrt{N^{1/2^k}}$ is computed: using a full SVD

on each submatrix costs of the order of $(\sqrt{N^{1/2^k}})^3 = N^{3/2^{k+1}}$. Since $2^k \leq J/2 = \log_2(N)/2$ and $N^{1+3/2^{k+1}} \leq N^{1+3/4} = N^{7/4}$ for $k \geq 1$, the total cost with balanced tree and full SVD is

$$\begin{aligned}\sum_{k=0}^{\log_2(J)-1} 2^k \times N \times N^{3/2^{k+1}} &= N^{5/2} + \sum_{k=1}^{\log_2(J)-1} 2^k N^{1+3/2^{k+1}} \\ &\leq N^{5/2} + \sum_{k=1}^{\log_2(\log_2(N))-1} \frac{\log_2(N)}{2} N^{7/4} \\ &= \mathcal{O}(N^{5/2}).\end{aligned}$$

## Appendix F. Proof of Lemma 3.13.

*Proof.* Denote $\mathcal{S} := \varphi(\mathbf{S}_{\mathtt{bf}}^{(p:\ell)}, \mathbf{S}_{\mathtt{bf}}^{(\ell+1:q)\mathsf{T}})$, and $I_k := \{(k-1)N/2^\ell+1, \ldots, kN/2^\ell\}$ for $k \in [2^\ell]$. The right support $\mathbf{S}_{\mathtt{bf}}^{(\ell+1:q)\mathsf{T}}$, which is equal to $\mathbf{S}_{\mathtt{bf}}^{(\ell+1:q)}$ by Remark 3.5, is block diagonal, with blocks $\mathbf{V}^{(\ell+1,q)}$ of size $N/2^\ell \times N/2^\ell$. Hence, for $i \in I_k$ and $j \in I_{k'}$ with $k, k' \in [2^\ell]$, $k \neq k'$, the rank-one supports $\boldsymbol{\mathcal{S}}^i$ and $\boldsymbol{\mathcal{S}}^j$ are disjoint. The columns supports $\{\mathrm{supp}((\mathbf{S}_{\mathtt{bf}}^{(p:\ell)})_i),\ i \in I_k\}$ are pairwise disjoint for each $k \in [2^\ell]$, by definition of $\mathbf{S}_{\mathtt{bf}}^{(p:\ell)}$ and $\mathbf{V}^{(p,\ell)}$. This means that for $i, j \in I_k$ ($k \in [2^\ell]$), the rank-one supports $\boldsymbol{\mathcal{S}}^i$ and $\boldsymbol{\mathcal{S}}^j$ are also disjoint, when $i \neq j$. ∎

## Appendix G. Proof of Lemma 3.16.

*Proof.* (i) ⇒ (ii): consider $p, \ell, q$ such that the partial products are made of a single factor. (ii) ⇒ (i): Given $\ell \in \{1, \ldots, J-1\}$, we show by backward induction that $\mathbf{X}^{(p:\ell)}$ has no zero column for each $p \in \{1, \ldots, \ell\}$. By assumption, $\mathbf{X}^{(\ell:\ell)} = \mathbf{X}^{(\ell)}$ has no zero column. Let $p \in \{2, \ldots, \ell\}$, and suppose that $\mathbf{X}^{(p:\ell)}$ has no zero column. For $i \in [N]$, the $i$-th column of $\mathbf{X}^{(p-1:\ell)} = \mathbf{X}^{(p-1)}\mathbf{X}^{(p:\ell)}$ is a linear combination of columns $\{(\mathbf{X}^{(p-1)})_k \mid k \in \mathrm{supp}((\mathbf{X}^{(p:\ell)})_i)\}$. By Lemma 3.15, the supports of the rank-one contributions in $\varphi(\mathbf{X}^{(p-1)}, \mathbf{X}^{(p:\ell)\mathsf{T}})$ are pairwise disjoint, hence the column supports $\{\mathrm{supp}((\mathbf{X}^{(p-1)})_k) \mid k \in \mathrm{supp}((\mathbf{X}^{(p:\ell)})_i)\}$ are pairwise disjoint. But $\mathrm{supp}((\mathbf{X}^{(p:\ell)})_i)$ is not empty as $\mathbf{X}^{(p:\ell)}$ has no zero column. As $\mathbf{X}^{(p-1)}$ is also not empty, the $i$-th column of $\mathbf{X}^{(p-1:\ell)}$ is non-zero, and this is true for each $i \in [N]$, which ends the induction. A similar induction shows that $\mathbf{X}^{(\ell+1:q)}$ has no zero row for each $2 \leq \ell+1 \leq q \leq J$. ∎