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 $\mathbf{Z} = \mathbf{X}^{(J)} \dots \mathbf{X}^{(1)}$, 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 contrasts with existing approaches which fit the product of butterfly factors to a given matrix via gradient descent. The proposed method can be applied in particular to retrieve the factorizations 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. This hierarchical identifiability property relies on a simple identifiability condition in the two-layer and fixed-support setting.

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 $\mathbf{X}^{(J)}\mathbf{X}^{(J-1)} \dots \mathbf{X}^{(1)}$. 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 [9, 33, 21, 22].

Sparsity constraints are usually encoded by a family of *allowed* supports, which force the factors to have some prescribed sparsity patterns. For instance, for $J = 2$, in the sparse coding problem [11, 9] with known dictionary $\mathbf{X}^{(2)}$, the factor $\mathbf{X}^{(1)}$ is constrained to be k-sparse by column, i.e., to have at most k nonzero entries per column.

This paper focuses on *identifiability* in the sparse matrix factorization problem, i.e., *essential uniqueness* of the solution up to natural ambiguities. Typical results read as:

Informal Theorem 1.1. *Let $\mathbf{Z}$ be a matrix, and Ω be a* family of sparsity patterns. *Assume that $\mathbf{Z}$ admits two sparse factorizations $\mathbf{Z} = \mathbf{X}^{(J)} \dots \mathbf{X}^{(1)} = \bar{\mathbf{X}}^{(J)} \dots \bar{\mathbf{X}}^{(1)}$ where the factors satisfy the sparsity constraint encoded by the family Ω. If some certain conditions on $\mathbf{Z}$, Ω are satisfied, then the tuples of factors $(\mathbf{X}^{(\ell)})_{\ell=1}^{J}$ and $(\bar{\mathbf{X}}^{(\ell)})_{\ell=1}^{J}$ are equivalent in a certain sense, which takes into account unavoidable ambiguities in the sparse matrix factorization problem.*

Understanding conditions for the sparse matrix factorization problem to be well-posed, in the sense that it admits a unique solution, is still very much open. It is also key to derive methods that are *guaranteed* to compute a good sparse approximation of the target matrix.

Typically, heuristic methods based on iterative first-order optimization [8, 22] do not have this kind of guarantees, and their performance heavily depends on initialization. They

*Preprint. This work is an extension of [19].

Funding: This project was supported in part by the AllegroAssai ANR project ANR-19-CHIA-0009.

[†]Université 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.

empirically address the following formulation of the sparse matrix factorization problem:

$$\min_{\mathbf{X}^{(J)},\dots,\mathbf{X}^{(1)}} \|\mathbf{Z} - \mathbf{X}^{(J)} \dots \mathbf{X}^{(1)}\|_F, \quad \text{where } \|\cdot\|_F \text{ denotes the Frobenius norm,} \tag{1.1}$$

with the constraint that the factors $\mathbf{X}^{(\ell)}$, $1 \leq \ell \leq J$, are sparse, possibly with structured constraints on their supports. For example, k-sparsity can be imposed globally on $\mathbf{X}^{(\ell)}$, or on each of its rows, or each column, or both. Proximal gradient algorithms [22] can for instance be employed to explore the family of sparsity patterns, and find an allowed sparsity pattern that yields the smallest approximation error. To overcome the difficulties of using this technique in the multi-layer setting (i.e. $J \geq 3$), a hierarchical factorization method has been proposed [22], in which the target matrix is iteratively factorized into two factors, until the desired number of sparse factors J is obtained. However, recent works have shown some important pitfalls of this hierarchical paradigm [17]: it is possible that an intermediate factor obtained at a certain level of the hierarchy cannot be further factorized into factors with admissible sparsity patterns, which yields a poor overall performance of the hierarchical method.

This paper shows that when $\mathbf{Z}$ admits an *exact* sparse factorization $\mathbf{Z} = \mathbf{X}^{(J)} \dots \mathbf{X}^{(1)}$, such a hierarchical method can be guaranteed to recover the sparse factors $(\mathbf{X}^{(\ell)})_{\ell=1}^{J}$ from $\mathbf{Z}$ up to unavoidable ambiguities, assuming some appropriate *fixed-support constraints* at each level in the hierarchy. In this case, the pitfall mentioned above can be avoided.

Specifically, our main contribution is to show that the so-called *butterfly* structure [21, 8], which appears naturally in many fast transforms such as the discrete Fourier transform or the Hadamard transform, is a good choice of *fixed* structured support constraint ensuring such a *hierarchical identifiability* (see Theorem 3.10). We derive from this identifiability analysis an algorithm for the recovery of the sparse butterfly factors $\mathbf{X}^{(\ell)}$, $1 \leq \ell \leq J$, from their product $\mathbf{Z} = \mathbf{X}^{(J)} \dots \mathbf{X}^{(1)}$ (see Algorithm 3.1). In contrast to existing approaches for butterfly factorization, which rely on iterative gradient descent [8, 22], the proposed algorithm is based on a non-trivial application of singular value decomposition (SVD) to compute best rank-one approximations of specific submatrices. It has bounded complexity and is endowed with exact recovery guarantees. Indeed, we show that the recovery algorithm has a time complexity of 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 can be performed only once to enable $\mathcal{O}(N \log N)$ matrix-vector multiplications with the resulting factored representation of $\mathbf{Z}$.

The proof of hierarchical identifiability for the butterfly structure relies on an identifiability condition in exact sparse matrix factorization with $J = 2$ factors and *fixed* supports. This condition is based on the *lifting* principle commonly considered in multilinear inverse problems [5, 6, 26, 30, 28, 29], which, in the context of matrix factorization into two factors, proposes to represent a pair of factors $(\mathbf{X}^{(2)}, \mathbf{X}^{(1)})$ by the tuple of rank-one matrices $(\boldsymbol{\mathcal{C}}^i)_i$, which are the outer products between columns of $\mathbf{X}^{(2)}$ and rows of $\mathbf{X}^{(1)}$ of the same index i.

1.1. Related work. Many identifiability results in multilinear inverse problems are derived from the *lifting* procedure [5, 6, 29, 26, 30]. This procedure was originally used in the PhaseLift method [23, 4, 3] to address the phase retrieval problem, which can be seen as a particular instance of the matrix factorization problem with two factors, where the two factors have a specific structure. Other bilinear inverse problems, like blind-deconvolution [1, 5, 2, 25, 24,

14, 26] or self-calibration [27], have also been addressed using the lifting procedure. A general framework to analyze identifiability for any bilinear inverse problem has been given in [5]. It takes into account *scaling ambiguities* inherent to bilinear inverse problems, and transforms a bilinear inverse problem into a low-rank matrix recovery problem. Identifiability is then related to geometric properties of the corresponding lifting operator, and more precisely, to the rank-two null space of the operator. Following the work from [20, 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 [29, 30, 28]. However, due to this multi-layer structure, conditions obtained via tensorial lifting might be difficult to verify in practice, although necessary and sufficient stable recovery conditions for convolutional linear networks have been derived [30, 28]. In contrast, our work proves identifiability results in matrix factorization with $J \geq 3$ factors *without using tensorial lifting*, using a hierarchical approach which reduces the analysis with multiple factors to the case with only two factors.

In our work, the considered sparsity constraint is the butterfly structure, a common sparsity pattern to many fast transforms [8]. As described in subsection 3.1, this butterfly structure enforces the factors to have at most two nonzero entries per row and per column, with a specific block-diagonal structure on each factor. This is a *fixed*-support constraint, i.e., each factor $\mathbf{X}^{(\ell)}$ is constrained to have a support included in a fixed subset of indices $\mathbf{S}^{(\ell)}$. However, as mentioned above, traditional approaches in sparse coding (see [32] for an overview) involve more relaxed sparsity constraints, which can be modeled by a *family* of sparsity patterns (i.e., of allowed supports), like the family of supports with at most k nonzero entries per column. For factorization with such families, permutation ambiguities might be unavoidable (in addition to scaling ambiguities), and should be taken into account in the analysis of identifiability. By focusing on *fixed* butterfly supports we avoid having to handle such technicalities. This of course raises natural research directions for future work.

Finally, the hierarchical factorization algorithm described by Algorithm 3.1 has been studied empirically[1] in [19] where it was empirically 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 optimization methods like [8]. 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 $\mathbf{Z} := \mathbf{X}^{(J)} \dots \mathbf{X}^{(1)}$, using the hierarchical method described in Algorithm 3.1.

[1]These two contributions have been produced in parallel, and the conference publication [19] refers to the current submission/preprint to support its theoretical claims.

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[2] Algorithm 3.1 using *truncated* SVDs.

The paper is organized as follows: section 2 analyzes identifiability in the two-layer setting 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 $\operatorname{supp}(\mathbf{M}) \subseteq [m] \times [n]$ 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 $\operatorname{colsupp}(\mathbf{M})$, is the subset of indices $i \in [n]$ such that the i-th column of $\mathbf{M}$, denoted $\mathbf{M}_i$, is nonzero. The notation $\mathbf{M}^{(i)}$ denotes the i-th matrix in a collection. The entry of $\mathbf{M}$ indexed by (k,l) is $\mathbf{M}_{k,l}$. The identity matrix of size n is denoted $\mathbf{I_n}$. The Kronecker product [34] 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 Σ 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 Σ 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

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

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, Σ_{S} is invariant by column scaling for any pair of supports S. Uniqueness of a

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

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 [16], such a uniqueness property will be referred to as *essential uniqueness*.

Definition 2.1 (Essential uniqueness of a two-layer factorization in Σ). *Denote $\mathcal{D}_r$ the group of invertible diagonal matrices of size $r \times r$ and let Σ 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 Σ, if any solution $(\bar{\mathbf{X}}, \bar{\mathbf{Y}})$ to (2.1) with $\mathbf{Z}$ and Σ 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 $\mathbf{D} \in \mathcal{D}_r$ such that $(\bar{\mathbf{X}}, \bar{\mathbf{Y}}) = \left(\mathbf{X}\mathbf{D}, \mathbf{Y}\mathbf{D}^{-1}\right)$.*

For any set Σ 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 Σ 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 Σ_{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 Σ_{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 2.2. *For any pair of supports* S*, we have: $\mathcal{U}(\Sigma_{\mathrm{S}}) \subseteq \mathtt{IC}_{\mathrm{S}} \cap \mathtt{MC}_{\mathrm{S}}$.*

In other words, if the factorization $\mathbf{Z} := \mathbf{X}\mathbf{Y}^\intercal$ is essentially unique in Σ_{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 [5, 29] 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 2.3 (Reformulation of [20, 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 [5] 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 of 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 sufficient condition for identifiability. When a pair of supports $\mathrm{S} := (\mathbf{S}^L, \mathbf{S}^R)$ is such that $\varphi(\mathrm{S}) = (\boldsymbol{\mathcal{S}}^i)_{i=1}^r$ has disjoint rank-one supports, *i.e.*, $\mathrm{supp}(\boldsymbol{\mathcal{S}}^i) \cap \mathrm{supp}(\boldsymbol{\mathcal{S}}^j) = \emptyset$ for every $i \neq j$, the factorization $\mathbf{Z} := \mathbf{X}\mathbf{Y}^\intercal$ is essentially unique in Σ_S for every non-degenerate pair of factors $(\mathbf{X}, \mathbf{Y})$.

Proposition 2.4. *If the tuple $\varphi(\mathrm{S})$ has disjoint rank-one supports then: $\mathcal{U}(\Sigma_\mathrm{S}) = \mathtt{IC}_\mathrm{S} \cap \mathtt{MC}_\mathrm{S}$.*

The proof is deferred to Appendix C. Despite its simplicity, this condition of disjoint rank-one supports is verified when considering a so-called butterfly structure on the sparse factors, see subsection 3.1. More conditions on fixed-support identifiability are given in [35].

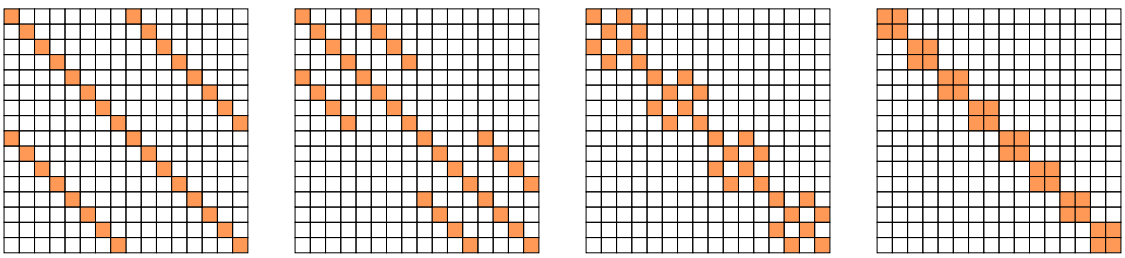
3. Hierarchical identifiability of butterfly factors. The main contribution of the paper is to show the hierarchical identifiability of the butterfly factors, i.e., we establish some identifiability results in the multi-layer sparse matrix factorization in a specific setting where the sparse factors 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 $\mathrm{S}_\mathtt{bf} := (\mathbf{S}_\mathtt{bf}^{(J)}, \ldots, \mathbf{S}_\mathtt{bf}^{(1)}) \in (\mathbb{B}^{N \times N})^J$ defined by:*

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

Figure 1 is an illustration of the butterfly supports. They are 2-regular [17], 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.



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

Figure 1: Butterfly supports of size $N = 16$. Nonzero entries (colored) *vs* zero entries (white).

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 $(\mathbf{X}^{(J)}, \ldots, \mathbf{X}^{(1)})$ which have a support*

included in the butterfly supports $\mathrm{S}_{\mathtt{bf}} := (\mathbf{S}_{\mathtt{bf}}^{(J)}, \dots, \mathbf{S}_{\mathtt{bf}}^{(1)})$*, in the sense that:*

$$\mathbf{Z} = \mathbf{X}^{(J)} \dots \mathbf{X}^{(1)}, \quad \textit{with} \quad (\mathbf{X}^{(J)}, \dots, \mathbf{X}^{(1)}) \in \Sigma_{\mathrm{S}_{\mathtt{bf}}} := \Sigma_{\mathbf{S}_{\mathtt{bf}}^{(J)}} \times \dots \times \Sigma_{\mathbf{S}_{\mathtt{bf}}^{(1)}}. \tag{3.2}$$

Such a structure is involved in the matrices associated to many fast linear transforms [8] such as the Hadamard transform, the discrete sine and discrete cosine transforms (DST, DCT) or, as we detail here, the discrete Fourier transform (DFT).

Example 3.3 (Butterfly factorization of the DFT matrix [8]). 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 $\mathbf{DFT_N}$ [8] 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 which sorts the odd indices, then the even indices, e.g., for $N = 4$, it permutes $(1, 2, 3, 4)$ to $(1, 3, 2, 4)$, 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 $1, \omega_N, \omega_N^2, \dots, \omega_N{}^{\frac{N}{2}-1}$. Applying recursively (3.4) to the block $\mathbf{DFT_{N/2}}$, we obtain the butterfly factorization of $\mathbf{DFT_N}$:

$$\mathbf{DFT_N} = \mathbf{F}^{(J)} \dots \mathbf{F}^{(1)} \mathbf{R_N}, \quad \text{where } \mathbf{F}^{(\ell)} := \mathbf{I_{N/2^\ell}} \otimes \mathbf{B_{2^\ell}},\ 1 \le \ell \le J, \tag{3.6}$$

and $\mathbf{R_N} \in \mathbb{B}^{N \times N}$ is the so-called *bit-reversal* permutation matrix, defined by:

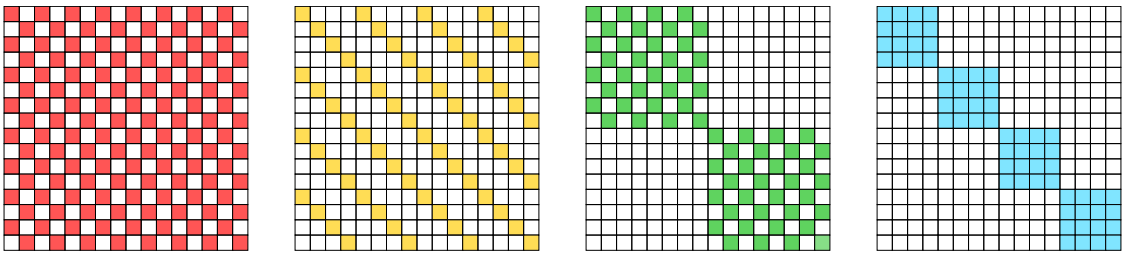
$$\mathbf{R_N} := \mathbf{Q}^{(1)} \mathbf{Q}^{(2)} \dots \mathbf{Q}^{(J)}, \quad \text{where } \mathbf{Q}^{(\ell)} := \mathbf{I_{N/2^\ell}} \otimes \mathbf{P_{2^\ell}},\ 1 \le \ell \le J. \tag{3.7}$$

Given $(\mathbf{X}^{(J)}, \dots, \mathbf{X}^{(1)}) \in \Sigma_{\mathrm{S}_{\mathtt{bf}}}$ where $\mathrm{S}_{\mathtt{bf}}$ is the tuple of butterfly supports of size 2^J, we show that the partial product of any consecutive factors $\mathbf{X}^{(q)} \dots \mathbf{X}^{(p)}$ $(1 \le p \le q \le J)$ has a very precise structure as detailed in the following lemma. Figure 2 illustrates this structure on some examples of partial products for the case of the butterfly supports of size $N = 16$. In particular, this structure, which is encoded by the support $\mathbf{W}^{(q:p)}$ defined in the following lemma (proved in subsection 3.5), is involved in the hierarchical matrix factorization method when enforcing the butterfly structure.

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

$$\mathbf{W}^{(q:p)} := \mathbf{I_{N/2^q}} \otimes \mathbf{V}^{(q,p)} = \begin{pmatrix} \mathbf{V}^{(q,p)} & & 0 \\ & \ddots & \\ 0 & & \mathbf{V}^{(q,p)} \end{pmatrix} \in \mathbb{B}^{N \times N}, \ \textit{and} \tag{3.8}$$

$$\mathbf{V}^{(q,p)} := \mathbf{U_{2^{q-p+1}}} \otimes \mathbf{I_{2^{p-1}}} = \begin{pmatrix} \mathbf{I_{2^{p-1}}} & \dots & \mathbf{I_{2^{p-1}}} \\ \vdots & & \vdots \\ \mathbf{I_{2^{p-1}}} & \dots & \mathbf{I_{2^{p-1}}} \end{pmatrix} \in \mathbb{B}^{2^q \times 2^q}, \tag{3.9}$$



(a) $\mathbf{W}^{(4:2)}$ (b) $\mathbf{W}^{(4:3)}$ (c) $\mathbf{W}^{(3:2)}$ (d) $\mathbf{W}^{(2:1)}$

Figure 2: Examples of supports $\mathbf{W}^{(q:p)}$ defined by (3.8) ($1 \leq p \leq q \leq 4$) of size $N \times N$ with $N = 16$. Nonzero entries are in color, and zero entries are in white.

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

Remark 3.5. We have $\mathbf{W}^{(\ell:\ell)} = \mathbf{S}_{\mathtt{bf}}^{(\ell)}$ for $1 \leq \ell \leq J$, and viewing matrix supports as binary matrices, one can verify that $\mathbf{W}^{(q:p)} = \mathbf{S}_{\mathtt{bf}}^{(q)} \dots \mathbf{S}_{\mathtt{bf}}^{(p)}$ for $1 \leq p \leq q \leq J$. Also, by (3.9), $\mathbf{V}^{(q,p)}$ is symmetric, i.e., $\mathbf{V}^{(q,p)} = (\mathbf{V}^{(q,p)})^\mathsf{T}$. By (3.8), $\mathbf{W}^{(q:p)}$ is block-diagonal with blocks $\mathbf{V}^{(q,p)}$, so $\mathbf{W}^{(q:p)}$ is also symmetric. Finally, $\mathbf{W}^{(q:p)}$ and $\mathbf{V}^{(q,p)}$ 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):

$$\min_{\mathbf{X}^{(J)},\dots,\mathbf{X}^{(1)}} \|\mathbf{Z} - \mathbf{X}^{(J)} \dots \mathbf{X}^{(1)}\|_F, \quad \text{such that } (\mathbf{X}^{(J)},\dots,\mathbf{X}^{(1)}) \in \Sigma_{\mathrm{S}_{\mathtt{bf}}}. \tag{3.10}$$

Remark 3.6. Perhaps surprisingly, as shown in [18, 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 $\mathbf{Z} = \mathbf{X}\mathbf{Y}^\mathsf{T}$ for any $(\mathbf{X}, \mathbf{Y}) \in \Sigma_\mathrm{S}$; (b) $\mathbf{Z}$ can be approximated arbitrarily well by such a product, i.e.

$$0 = \inf_{\mathbf{X},\mathbf{Y}} \|\mathbf{Z} - \mathbf{X}\mathbf{Y}^\mathsf{T}\|_F \quad \text{such that } (\mathbf{X}, \mathbf{Y}) \in \Sigma_\mathrm{S}.$$

This corresponds to a lack of closure of the set $\{\mathbf{X}\mathbf{Y}^\mathsf{T} : (\mathbf{X}, \mathbf{Y}) \in \Sigma_\mathrm{S}\}$. 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 specific case of butterfly supports, problem (3.10) always admits a minimizer even when $\mathbf{Z}$ does not admit an exact butterfly factorization.

As proposed in [22], instead of directly optimizing over the J factors, the hierarchical matrix factorization method is a heuristic approach, which performs successive two-layer matrix factorizations, starting from the factorization of the target matrix $\mathbf{Z}$ into two factors, until J sparse factors are obtained. For instance, in the scenario where we want to recover a tuple of sparse factors $(\mathbf{X}^{(J)},\dots,\mathbf{X}^{(1)}) \in \Sigma_{\mathrm{S}_{\mathtt{bf}}}$ from the product $\mathbf{Z} := \mathbf{X}^{(J)} \dots \mathbf{X}^{(1)}$, the hierarchical method proceeds as follows. At the first level, the factors $\mathbf{X}^{(J)}$ and $\mathbf{H}^{(J-1)} := \mathbf{X}^{(J-1)} \dots \mathbf{X}^{(1)}$ are recovered from their known product $\mathbf{X}^{(J)}\mathbf{H}^{(J-1)} = \mathbf{Z}$. At the second level, the factors $\mathbf{X}^{(J-1)}$ and $\mathbf{H}^{(J-2)} := \mathbf{X}^{(J-2)} \dots \mathbf{X}^{(1)}$ are in turn recovered from their known product $\mathbf{X}^{(J-1)}\mathbf{H}^{(J-2)} = \mathbf{H}^{(J-1)}$. The process is repeated recursively, until all the sparse factors are

recovered. At each step of this hierarchical factorization, 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 $\mathbf{W}^{(q:p)}$ defined at (3.8).

It was shown [19] that Algorithm 3.1, which is based on the hierarchical approach, performs empirically better than gradient-based optimization methods [8] to address problem (3.10). Before discussing the details of Algorithm 3.1 let us mention that in the specific case of the butterfly support constraint, by Lemma 3.4, the hierarchical factorization method previously described can be generalized to *any* kind of binary tree structure. For instance, in the case of four factors $\mathbf{X}^{(4)}, \dots, \mathbf{X}^{(1)}$ of size 16×16, one can perform hierarchical factorization in an alternative fashion, by factorizing $\mathbf{Z} := \mathbf{X}^{(4)}\mathbf{X}^{(3)}\mathbf{X}^{(2)}\mathbf{X}^{(1)}$ into $\mathbf{X}^{(4)}\mathbf{X}^{(3)}$ and $\mathbf{X}^{(2)}\mathbf{X}^{(1)}$ at the first layer, then $\mathbf{X}^{(4)}\mathbf{X}^{(3)}$ into $\mathbf{X}^{(4)}, \mathbf{X}^{(3)}$, and finally $\mathbf{X}^{(2)}\mathbf{X}^{(1)}$ into $\mathbf{X}^{(2)}, \mathbf{X}^{(1)}$. Let us formally define such a tree structure that describes the factorization order in the hierarchical method.

Definition 3.7 (Partitioning binary tree). *A* partitioning 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 $\{p, \dots, q\}$, which satisfies the axioms:*

(i) Each node is a subset of consecutive indices in $\{p, \dots, q\}$.
(ii) The root is the set $\{p, \dots, q\}$.
(iii) A node is a singleton if, and only if, it is a leaf.
(iv) 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 larger *than those in the right child.*

Examples of partitioning binary trees are illustrated in Figure 3.

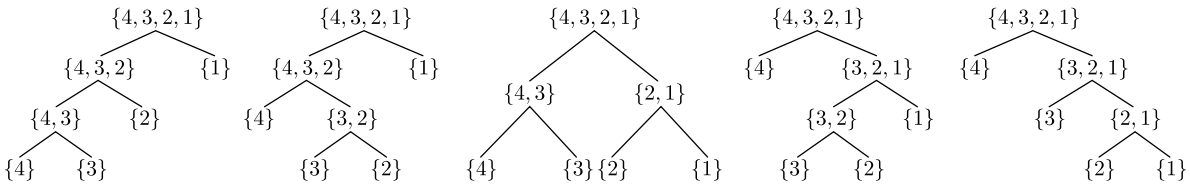


Figure 3: All possible partitioning binary trees of $\{1, 2, 3, 4\}$, which can serve as input to Algorithm 3.1. Note that the algorithm is applicable to any J (not only $J = 4$).

We can now describe Algorithm 3.1. Given as inputs a partitioning binary tree $\mathcal{T}$ of $[J]$ and any target matrix $\mathbf{Z}$, the algorithm visits the nodes of $\mathcal{T}$ in a breadth-first search order, starting by the root node. At each non-leaf node $n \subseteq [J]$ characterized by its maximum value q, its minimum value p, and its "splitting index" ℓ, which is the maximum value of its right child, the algorithm performs an (approximate) factorization of an intermediate matrix $\mathbf{H}^{(q:p)}$ by a sum of rank-one matrices $(\boldsymbol{\mathcal{C}}^i)_{i=1}^N$, whose supports are constrained by the rank-one supports $(\boldsymbol{\mathcal{S}}^i)_{i=1}^N = \varphi(\mathbf{W}^{(q:\ell+1)}, \mathbf{W}^{(\ell:p)\mathsf{T}})$. This is done by defining $\boldsymbol{\mathcal{C}}^i$ $(1 \leq i \leq N)$ as the best rank-one approximation (written `BestRankOneApprox` in Algorithm 3.1) of the submatrix $(\mathbf{H}^{(q:p)}{}_{j,k})_{(j,k)\in\boldsymbol{\mathcal{S}}^i}$, which can be computed via a truncated SVD. This yields an approximation of $\mathbf{H}^{(q:p)}$ by a product of two factors $\mathbf{H}^{(q:\ell+1)}\mathbf{H}^{(\ell:p)}$, where the left factor $\mathbf{H}^{(q:\ell+1)}$ and the right factor $\mathbf{H}^{(\ell:p)\mathsf{T}}$ have a support respectively included in $\mathbf{W}^{(q:\ell+1)}$ and

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

Require: Integers $1 \leq p \leq q \leq J$, matrix $\mathbf{H}^{(q:p)} \in \mathbb{C}^{N \times N}$, partitioning binary tree $\mathcal{T}^{(q:p)}$ of $\{p, \dots, q\}$

1: **if** $q = p$ **then**
2: **return** $\mathbf{H}^{(q:p)}$
3: **end if**
4: $\ell \leftarrow$ maximum value in the right child of the root of $\mathcal{T}^{(q:p)}$
5: $\left(\mathcal{T}^{(q:\ell+1)}, \mathcal{T}^{(\ell:p)}\right) \leftarrow$ left and right subtrees of the root of $\mathcal{T}^{(q:p)}$
6: $(\boldsymbol{\mathcal{S}}^i)_{i=1}^N \leftarrow \varphi\left(\mathbf{W}^{(q:\ell+1)}, \mathbf{W}^{(\ell:p)\mathsf{T}}\right)$
7: **for** $i = 1$ **to** N **do**
8: $\boldsymbol{\mathcal{C}}^i \leftarrow \texttt{BestRankOneApprox}\left((\mathbf{H}^{(q:p)}{}_{j,k})_{(j,k)\in\boldsymbol{\mathcal{S}}^i}\right)$, where $\boldsymbol{\mathcal{S}}^i$ is viewed as an index set
9: **end for**
10: Set $(\mathbf{H}^{(q:\ell+1)}, \mathbf{H}^{(\ell:p)\mathsf{T}})$ as a pair of factors in $\varphi^{-1}(\{\mathcal{C}\}) \cap \mathtt{IC}_{(\mathbf{W}^{(q:\ell+1)}, \mathbf{W}^{(\ell:p)\mathsf{T}})}$
11: $(\bar{\mathbf{X}}^{(q)}, \dots, \bar{\mathbf{X}}^{(\ell+1)}) \leftarrow$ result of Algorithm 3.1 with inputs $\mathbf{H}^{(q:\ell+1)}$ and $\mathcal{T}^{(q:\ell+1)}$
12: $(\bar{\mathbf{X}}^{(\ell)}, \dots, \bar{\mathbf{X}}^{(p)}) \leftarrow$ result of Algorithm 3.1 with inputs $\mathbf{H}^{(\ell:p)}$ and $\mathcal{T}^{(\ell:p)}$
13: **return** $(\bar{\mathbf{X}}^{(q)}, \dots, \bar{\mathbf{X}}^{(\ell+1)}, \bar{\mathbf{X}}^{(\ell)}, \dots, \bar{\mathbf{X}}^{(p)})$

$\mathbf{W}^{(\ell:p)\mathsf{T}}$. This approximation is optimal as proved in [18] in the sense that $\|\mathbf{H}^{(q:p)} - \mathbf{X}\mathbf{Y}^\mathsf{T}\|_F^2$ is minimized among all $\mathbf{X}, \mathbf{Y}$ satisfying the same support constraints. The process is then repeated recursively on the two factors $\mathbf{H}^{(q:\ell+1)}$ and $\mathbf{H}^{(\ell:p)}$.

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. Indeed, as proved in [18], the procedure at Line 8 yields an optimal approximation (in the sense of the Frobenius norm) of each submatrix $(\mathbf{Z}_{k,l})_{(k,l)\in\boldsymbol{\mathcal{S}}^i}$ as a product of two factors with the prescribed supports. However as this is used in a recursive greedy fashion in the algorithm, 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.

3.3. Uniqueness of the butterfly factorization.

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

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

We now claim the hierarchical identifiability in multi-layer sparse matrix factorization

when the factors are constrained to the butterfly supports introduced in subsection 3.2.

Theorem 3.10. *Consider* $\mathrm{S}_{\mathtt{bf}}$ *the butterfly supports of size* $N = 2^J$ *and* $(\mathbf{X}^{(J)}, \dots, \mathbf{X}^{(1)}) \in \Sigma_{\mathrm{S}_{\mathtt{bf}}}$. *Assume that* $\mathbf{X}^{(\ell)}$ *does not have a zero column for* $2 \leq \ell \leq J$, *and not a zero row for* $1 \leq \ell \leq J-1$. *Then, the factorization* $\mathbf{Z} := \mathbf{X}^{(J)} \dots \mathbf{X}^{(1)}$ *is essentially unique in* $\Sigma_{\mathrm{S}_{\mathtt{bf}}}$. *Moreover, these factors can be recovered from* $\mathbf{Z}$, *up to scaling ambiguities only, through the hierarchical factorization method detailed in Algorithm 3.1, where* $\mathbf{Z}$ *and* any *partitioning binary tree* $\mathcal{T}$ *of* $[J]$ *(see Definition 3.7) are given as the inputs of the algorithm.*

This theorem can be applied to show identifiability of the butterfly factorization of the DFT matrix as suggested in [20, Chapter 7], but also the one of the Hadamard matrix.

Corollary 3.11. *The Hadamard matrix and the permuted DFT matrix* $\mathbf{DFT_N R_N}^\intercal$ *of size* $N = 2^J$, *where* $\mathbf{R_N}$ *is the bit-reversal permutation matrix defined by (3.7), admit an essentially unique factorization in* $\Sigma_{\mathrm{S}_{\mathtt{bf}}}$. *In both cases, the butterfly factors can be recovered up to scaling ambiguities via Algorithm 3.1 with* any *partitioning binary tree of* $[J]$ *as input.*

Before proving this, we show that Algorithm 3.1 has controlled complexity bounds.

3.4. Complexity bounds. Existing algorithms for butterfly factorization [8, 22] 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 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. [13] 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 partitioning 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* partitioning binary tree $\mathcal{T}$ of $[J]$ (defined as the partitioning binary tree where the left child of each non-leaf node is a singleton) 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 partitioning 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 SVD (see Appendix D).

Balanced tree. Consider now running Algorithm 3.1 with a matrix $\mathbf{Z}$ of size $N \times N$, $N = 2^J$ where J itself is also a power of 2, and the *balanced* partitioning binary tree $\mathcal{T}$ of $[J]$ (defined as the partitioning binary tree where the children of each non-leaf node have the same cardinality) as inputs. At each non-leaf node of depth $k \in \{0, \dots, \log_2(J)-1\}$, the algorithm computes the best rank-one approximation of N submatrices of size $\sqrt{N^{1/2^k}} \times \sqrt{N^{1/2^k}}$, 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. Moreover, for $1 \leq k \leq \log_2(J)-1$ we have $2^k \leq J/2 = \log_2(N)/2$, hence the total cost of Algorithm 3.1 with the balanced tree 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 D).

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 $\mathcal{O}(N^2)$. Suppose that we want to compute the product $\mathbf{AB}$, where $\mathbf{A}, \mathbf{B}$ are of size $N \times N$. The naive computation requires $\mathcal{O}(N^3)$ flops. Assume now that $\mathbf{A}$ admits a butterfly structure, i.e., there exists $(\mathbf{X}^{(J)}, \dots, \mathbf{X}^{(1)}) \in \Sigma_{\mathbf{S}_{\mathtt{bf}}}$ such that $\mathbf{A} = \mathbf{X}^{(J)} \dots \mathbf{X}^{(1)}$. Then, one can compute such a butterfly factorization with Algorithm 3.1, which requires only $\mathcal{O}(N^2)$ flops. This allows to compute $\mathbf{X}^{(J)} \dots \mathbf{X}^{(1)}\mathbf{B}$ with $\mathcal{O}(N^2 \log(N))$ flops, since the linear operator $\mathbf{b} \mapsto \mathbf{X}^{(J)} \dots \mathbf{X}^{(1)}\mathbf{b}$ defined on $\mathbb{C}^N$ is a fast transform which can be evaluated with only $\mathcal{O}(N \log N)$ flops [8]. In total, 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 8 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 partitioning 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). The remaining of the section is dedicated to the proof of Theorem 3.10. The reader more interested in the numerical aspects of the proposed method can skip this section and jump to section 4. We first start by proving Lemma 3.4. The proof of this lemma follows from Remark 3.5 and the following lemma.

Lemma 3.12. *Given two matrix suports* $\mathbf{S}^{(2)} \in \mathbb{B}^{m \times r}$ *and* $\mathbf{S}^{(1)} \in \mathbb{B}^{r \times n}$*, for any pair* $(\mathbf{X}^{(2)}, \mathbf{X}^{(1)}) \in \Sigma_{\mathbf{S}^{(2)}} \times \Sigma_{\mathbf{S}^{(1)}}$*, we have* $\operatorname{supp}(\mathbf{X}^{(2)}\mathbf{X}^{(1)}) \subseteq \operatorname{supp}(\mathbf{S}^{(2)}\mathbf{S}^{(1)})$.

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

Proof of Lemma 3.4. We start by the case $q = J$ and show by backward induction that $\operatorname{supp}(\mathbf{X}^{(J)} \dots \mathbf{X}^{(\ell)}) \subseteq \mathbf{V}^{(J,\ell)}$ for $1 \leq \ell \leq J$. This is true for $\ell = J$, because by (3.1) and (3.9), $\operatorname{supp}(\mathbf{X}^{(J)}) \subseteq \mathbf{S}_{\mathtt{bf}}^{(J)} = \left[\begin{smallmatrix} 1 & 1 \\ 1 & 1 \end{smallmatrix}\right] \otimes \mathbf{I}_{\mathbf{2^{J-1}}} = \mathbf{V}^{(J,J)}$. Suppose now that $2 \leq \ell \leq J$ and $\operatorname{supp}(\mathbf{X}^{(J)} \dots \mathbf{X}^{(\ell)}) \subseteq \mathbf{V}^{(J,\ell)}$, which is the matrix full of blocks $\mathbf{I}_{\mathbf{2^{\ell-1}}}$. Since $\operatorname{supp}(\mathbf{X}^{(\ell-1)}) \subseteq \mathbf{S}_{\mathtt{bf}}^{(\ell-1)}$, by Lemma 3.12, we have $\operatorname{supp}(\mathbf{X}^{(J)} \dots \mathbf{X}^{(\ell)}\mathbf{X}^{(\ell-1)}) \subseteq \operatorname{supp}(\mathbf{V}^{(J,\ell)}\mathbf{S}_{\mathtt{bf}}^{(\ell-1)})$. As $\mathbf{S}_{\mathtt{bf}}^{(\ell-1)}$ is block diagonal with blocks of size $2^{\ell-1} \times 2^{\ell-1}$ equal to $\mathbf{U_2} \otimes \mathbf{I}_{\mathbf{2^{\ell-2}}} = \left[\begin{smallmatrix} \mathbf{I}_{\mathbf{2^{\ell-2}}} & \mathbf{I}_{\mathbf{2^{\ell-2}}} \\ \mathbf{I}_{\mathbf{2^{\ell-2}}} & \mathbf{I}_{\mathbf{2^{\ell-2}}} \end{smallmatrix}\right]$, we get:

$$\begin{aligned}
\mathbf{V}^{(J,\ell)}\mathbf{S}_{\mathtt{bf}}^{(\ell-1)} &= \left(\mathbf{U}_{\mathbf{2^{J-\ell+1}}} \otimes \mathbf{I}_{\mathbf{2^{\ell-1}}}\right)\left(\mathbf{I}_{\mathbf{2^{J-\ell+1}}} \otimes \mathbf{U_2} \otimes \mathbf{I}_{\mathbf{2^{\ell-2}}}\right) \\
&\overset{(1.2)}{=} \left(\mathbf{U}_{\mathbf{2^{J-\ell+1}}}\mathbf{I}_{\mathbf{2^{J-\ell+1}}}\right) \otimes \left(\mathbf{I}_{\mathbf{2^{\ell-1}}}\left(\mathbf{U_2} \otimes \mathbf{I}_{\mathbf{2^{\ell-2}}}\right)\right) \\
&\overset{(1.2)}{=} \mathbf{U}_{\mathbf{2^{J-\ell+1}}} \otimes \mathbf{U_2} \otimes \mathbf{I}_{\mathbf{2^{\ell-2}}} = \mathbf{U}_{\mathbf{2^{J-\ell}}} \otimes \mathbf{I}_{\mathbf{2^{\ell-2}}} = \mathbf{V}^{(J,\ell-1)}.
\end{aligned}$$

Consequently, $\operatorname{supp}(\mathbf{X}^{(J)} \dots \mathbf{X}^{(\ell)}\mathbf{X}^{(\ell-1)}) \subseteq \operatorname{supp}(\mathbf{V}^{(J,\ell)}\mathbf{S}_{\mathtt{bf}}^{(\ell-1)}) = \operatorname{supp}(\mathbf{V}^{(J,\ell-1)}) = \mathbf{V}^{(J,\ell-1)}$. This ends the induction. Now, in the case $q < J$, the butterfly support $\mathbf{S}_{\mathtt{bf}}^{(q)}$ of size $N \times N$

is block diagonal, where each block is simply the leftmost butterfly support of size $2^q \times 2^q$. Applying the previous case to each of these blocks of size $2^q \times 2^q$ yields the claimed result. ∎

In order to prove Theorem 3.10, let us introduce $(\mathbf{X}^{(J)}, \dots, \mathbf{X}^{(1)}) \in \Sigma_{\mathrm{S_{bf}}}$ satisfying the butterfly constraint. These factors are assumed to verify the assumption of Theorem 3.10. For any $1 \leq p \leq q \leq J$, we denote $\mathbf{X}^{(q:p)} := \mathbf{X}^{(q)} \dots \mathbf{X}^{(p)}$, and $\mathbf{Z} := \mathbf{X}^{(J:1)} = \mathbf{X}^{(J)} \dots \mathbf{X}^{(1)}$. Fix any partitioning binary tree $\mathcal{T}$ of $[J]$. Given $\mathbf{Z}$ and $\mathcal{T}$ as the inputs of Algorithm 3.1, we write $\mathbf{H}^{(q:p)}$ 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. In a first part, we prove that Algorithm 3.1 recovers the butterfly factors $(\mathbf{X}^{(J)}, \dots, \mathbf{X}^{(1)})$ from $\mathbf{Z}$, up to scaling ambiguities. In a second part, we prove that the butterfly factorization $\mathbf{Z} = \mathbf{X}^{(J)} \dots \mathbf{X}^{(1)}$ 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 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 $\mathcal{T}$). For each $v \in [J-1]$, denote p_v and q_v respectively the minimum and maximum index of node $n_v = \{p_v, \dots, q_v\}$ and ℓ_v as the "splitting" index of node n_v, which is the maximum value of its *right* child (recall from Definition 3.7 that the right child corresponds to the *smallest* indices in the parent node, as they index the *rightmost* factors in the corresponding subproduct). In the proof we will use the following fact, which is a direct consequence of the breadth-first-search order.

Fact 3.13. *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 right child of n_v, then $p_u = p_v$ and $q_u = \ell_v$. If n_u is the left child of n_v, then $p_u = \ell_v + 1$ and $q_u = q_v$.*

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: $\mathbf{H}^{(q_v:\ell_v+1)} = \mathbf{D}^{(q_v)^{-1}} \mathbf{X}^{(q_v:\ell_v+1)} \mathbf{D}^{(\ell_v)}$ and $\mathbf{H}^{(\ell_v:p_v)} = \mathbf{D}^{(\ell_v)^{-1}} \mathbf{X}^{(\ell_v:p_v)} \mathbf{D}^{(p_v-1)}$", with the convention[3] $\mathbf{D}^{(J)} = \mathbf{D}^{(0)} = \mathbf{I_N}$. The principle of the proof is to show P_V by induction for all $V \in [J-1]$: in other words, Algorithm 3.1 reconstructs recursively the partial products $\mathbf{X}^{(q:p)}$ up to scaling ambiguities at each node $n = \{p, \dots, q\}$ of the tree $\mathcal{T}$. 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 right 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 $\mathbf{H}^{(\ell:\ell)} = \mathbf{D}^{(\ell)^{-1}} \mathbf{X}^{(\ell:\ell)} \mathbf{D}^{(\ell-1)} = \mathbf{D}^{(\ell)^{-1}} \mathbf{X}^{(\ell)} \mathbf{D}^{(\ell-1)}$, hence $(\mathbf{H}^{(\ell:\ell)})_{\ell=1}^{J} \sim (\mathbf{X}^{(\ell)})_{\ell=1}^{J}$.

The crux of the proof is the following lemma.

Lemma 3.14. *Under the assumptions of Theorem 3.10, consider $V \in [J-1]$ and assume that there are invertible diagonal matrices $\mathbf{D}^{(q_V)}$ and $\mathbf{D}^{(p_V-1)}$ such that*

$$\mathbf{H}^{(q_V:p_V)} = \mathbf{D}^{(q_V)^{-1}} \mathbf{X}^{(q_V:p_V)} \mathbf{D}^{(p_V-1)}.$$

[3] 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 3.13, either $p_v = \ell_w + 1$ and $q_v = q_w$, or $p_v = p_w$ and $q_v = \ell_w$, with $w \in [v-1] \subseteq [V]$. In other words, $q_v, \ell_v, p_v - 1 \in \{0, \ell_1, \dots, \ell_V, J\}$ for all $v \in [V]$, meaning that the diagonal matrices $\mathbf{D}^{(q_v)}$, $\mathbf{D}^{(\ell_v)}$ and $\mathbf{D}^{(p_v-1)}$ used in the definition of P_V above are well defined.

Then the pair $(\mathbf{H}^{(q_V:\ell_V+1)}, \mathbf{H}^{(\ell_V:p_V)\mathsf{T}})$ *computed at Line 10 of Algorithm 3.1 such that the product* $\mathbf{H}^{(q_V:\ell_V+1)}\mathbf{H}^{(\ell_V:p_V)}$ *approximates* $\mathbf{H}^{(q_V:p_V)}$*, is equal (up to scaling ambiguities) to the pair* $(\bar{\mathbf{X}}, \bar{\mathbf{Y}})$ *where* $\bar{\mathbf{X}} := {\mathbf{D}^{(q_V)}}^{-1}\mathbf{X}^{(q_V:\ell_V+1)}$, $\bar{\mathbf{Y}} := (\mathbf{X}^{(\ell_V:p_V)}\mathbf{D}^{(p_V-1)})^{\mathsf{T}}$.

Indeed, since $\mathbf{H}^{(q_1:p_1)} = \mathbf{Z} = \mathbf{X}^{(q_1:p_1)}$ and since $\mathbf{D}^{(q_1)} = \mathbf{D}^{(p_1-1)} = \mathbf{I_N}$ (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 3.13, 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 $n_V = \{\ell_v + 1, \dots, q_v\}$ or $n_V = \{p_v, \dots, \ell_v\}$. Without loss of generality assume the former (the proof is similar if we suppose the latter) so that $p_V = \ell_v + 1$ and $q_V = q_v$. Since P_{V-1} is true we have that $\mathbf{H}^{(q_V:p_V)} = \mathbf{H}^{(q_v:\ell_v+1)} = {\mathbf{D}^{(q_v)}}^{-1}\mathbf{X}^{(q_v:\ell_v+1)}\mathbf{D}^{(\ell_v)} = {\mathbf{D}^{(q_V)}}^{-1}\mathbf{X}^{(q_V:p_V)}\mathbf{D}^{(p_V-1)}$. By Lemma 3.14, there exists an invertible diagonal matrix $\mathbf{D}^{(\ell_V)}$ such that $\mathbf{H}^{(q_V:\ell_V+1)} = {\mathbf{D}^{(q_V)}}^{-1}\mathbf{X}^{(q_V:\ell_V+1)}\mathbf{D}^{(\ell_V)}$ and $\mathbf{H}^{(\ell_V:p_V)} = {\mathbf{D}^{(\ell_V)}}^{-1}\mathbf{X}^{(\ell_V:p_V)}\mathbf{D}^{(p_V-1)}$, which proves P_V.

We now focus on the proof of Lemma 3.14. It relies on two key ingredients formulated in Theorem 3.16 and Proposition 3.19 below. They are both derived from the following property.

Lemma 3.15. *Given* $1 \le p \le \ell < q \le J$, $\mathbf{S}^L := \mathbf{W}^{(q:\ell+1)}$ *and* $\mathbf{S}^R := \mathbf{W}^{(\ell:p)\mathsf{T}}$ *(cf notations of Lemma 3.4), the tuple of rank-one supports* $\varphi\left(\mathbf{S}^L, \mathbf{S}^R\right)$ *has disjoint rank-one supports.*

Proof. Denote $\mathcal{S} := \varphi(\mathbf{W}^{(q:\ell+1)}, \mathbf{W}^{(\ell:p)\mathsf{T}})$. We also write $I_k := \{(k-1)2^\ell + 1, \dots, k2^\ell - 1\}$ for $k \in [\frac{N}{2^\ell}]$. One the one hand, the right support $\mathbf{W}^{(\ell:p)\mathsf{T}}$, which is equal to $\mathbf{W}^{(\ell:p)}$ by Remark 3.5, is block diagonal, with blocks $\mathbf{V}^{(\ell,p)}$ of size $2^\ell \times 2^\ell$. Hence, for $i \in I_k$ and $j \in I_{k'}$ with $k, k' \in [\frac{N}{2^\ell}]$, $k \neq k'$, the rank-one supports $\boldsymbol{\mathcal{S}}^i$ and $\boldsymbol{\mathcal{S}}^j$ are disjoint. On the other hand, viewing column supports as subsets of indices, the columns of the left support $\mathbf{W}^{(q:\ell+1)}$ restricted to I_k are pairwise disjoint for each $k \in [\frac{N}{2^\ell}]$, by definition of $\mathbf{V}^{(q,\ell+1)}$. This means that for $i, j \in I_k$ ($k \in [\frac{N}{2^\ell}]$), the rank-one supports $\boldsymbol{\mathcal{S}}^i$ and $\boldsymbol{\mathcal{S}}^j$ are also disjoint, when $i \neq j$. ∎

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

Theorem 3.16 ([18, Application of Theorem 3.3]). *Denote* $\mathrm{S} := (\mathbf{W}^{(q:\ell+1)}, \mathbf{W}^{(\ell:p)\mathsf{T}})$ *(cf notations of Lemma 3.4). For any matrix* $\mathbf{H}$*, the algorithm of [18, Algorithm 3.1] computes in polynomial time a pair of factors solving the following constrained optimization problem:*

$$\min_{\mathbf{X},\mathbf{Y}} \|\mathbf{H} - \mathbf{X}\mathbf{Y}^{\mathsf{T}}\|_F \quad \textit{such that } (\mathbf{X}, \mathbf{Y}) \in \Sigma_{\mathrm{S}}. \tag{3.11}$$

Lemma 3.15 also allows us to characterize the set $\mathcal{U}(\Sigma_{\mathrm{S}})$ when $\mathrm{S} := (\mathbf{W}^{(q:\ell+1)}, \mathbf{W}^{(\ell:p)\mathsf{T}})$.

Lemma 3.17. *Denote* $\mathrm{S} = (\mathbf{S}^L, \mathbf{S}^R) := (\mathbf{W}^{(q:\ell+1)}, \mathbf{W}^{(\ell:p)\mathsf{T}})$. *We have*

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

Proof. By Proposition 2.4, $\mathcal{U}\left(\Sigma_{\mathrm{S}}\right) = \mathtt{IC}_{\mathrm{S}} \cap \mathtt{MC}_{\mathrm{S}}$. 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 $(\mathbf{X}^{(J)}, \dots, \mathbf{X}^{(1)})$ verify the assumption of Theorem 3.10, i.e., $\mathbf{X}^{(\ell)}$ does not have a zero column for $2 \leq \ell \leq J$, and not a zero row for $1 \leq \ell \leq J-1$. As claimed in the following lemma, this assumption is in fact a necessary and sufficient condition to ensure that each pair of partial products $(\mathbf{X}^{(q:\ell+1)}, \mathbf{X}^{(\ell:p)\mathsf{T}})$ with $1 \leq p \leq \ell < q \leq J$ is non-degenerate (the left and right factor do not have a zero column).

Lemma 3.18. *Let* $\mathrm{S}_{\mathtt{bf}}$ *be the butterfly supports of size* $N = 2^J$*, and* $(\mathbf{X}^{(J)}, \dots, \mathbf{X}^{(1)}) \in \Sigma_{\mathrm{S}_{\mathtt{bf}}}$*. The following are equivalent:*

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

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

Proof. The implication (i) $\Rightarrow$ (ii) is direct by considering values of p, ℓ, q such that the partial products are made of a single factor. We now prove (ii) $\Rightarrow$ (i). Suppose (ii). Fix $\ell \in \{2, \dots, J\}$, and let us show by induction that $\mathbf{X}^{(q:\ell)}$ does not have a zero column for each $q \in \{\ell, \dots, J\}$. By assumption, $\mathbf{X}^{(\ell:\ell)} = \mathbf{X}^{(\ell)}$ does not have a zero column. Let $q \in \{\ell, \dots, J-1\}$, and suppose that $\mathbf{X}^{(q:\ell)}$ does not have a zero column. Let $i \in [N]$. Then, the i-th column of $\mathbf{X}^{(q+1:\ell)} = \mathbf{X}^{(q+1)}\mathbf{X}^{(q:\ell)}$ is a linear combination of columns in $\mathbf{X}^{(q+1)}$ indexed by $k \in \mathrm{supp}((\mathbf{X}^{(q:\ell)})_i)$. By Lemma 3.17, the supports of the rank-one contributions in $\varphi(\mathbf{X}^{(q+1)}, \mathbf{X}^{(q:\ell)\mathsf{T}})$ are pairwise disjoint. In particular, the supports of the columns in $\mathbf{X}^{(q+1)}$ indexed by $k \in \mathrm{supp}((\mathbf{X}^{(q:\ell)})_i)$ are pairwise disjoint. But $\mathrm{supp}((\mathbf{X}^{(q:\ell)})_i)$ is not empty because by assumption $\mathbf{X}^{(q:\ell)}$ does not have a zero column, and $\mathbf{X}^{(q+1)}$ also by assumption. Thus, the i-th column of $\mathbf{X}^{(q+1:\ell)}$ is not a zero column, and this is true for each $i \in [N]$, which ends the induction. A similar induction shows that $\mathbf{X}^{(\ell:p)}$ does not have a zero row for each $1 \leq p \leq \ell \leq J-1$. ∎

Consequently, we obtain the second key ingredient for the proof of Lemma 3.14.

Proposition 3.19. *Assume that* $(\mathbf{X}^{(J)}, \dots, \mathbf{X}^{(1)}) \in \Sigma_{\mathrm{S}_{\mathtt{bf}}}$ *verifies the hypothesis of Theorem 3.10. Let* $1 \leq p \leq \ell < q \leq J$*. Denote* $\mathrm{S} := (\mathbf{W}^{(q:\ell+1)}, \mathbf{W}^{(\ell:p)\mathsf{T}})$ *(recall (3.8)). Then, for any invertible diagonal matrices* $\mathbf{D}, \bar{\mathbf{D}}$*, denoting* $\bar{\mathbf{X}} := \mathbf{D}^{-1}\mathbf{X}^{(q:\ell+1)}$ *and* $\bar{\mathbf{Y}} = (\mathbf{X}^{(\ell:p)}\bar{\mathbf{D}})^{\mathsf{T}}$*, the factorization* $\mathbf{D}^{-1}\mathbf{X}^{(q:p)}\bar{\mathbf{D}} = \bar{\mathbf{X}}\bar{\mathbf{Y}}^{\mathsf{T}}$ *into two factors* $(\bar{\mathbf{X}}, \bar{\mathbf{Y}}) \in \Sigma_{\mathrm{S}}$ *is essentially unique.*

Proof. By Lemma 3.4, $(\mathbf{X}^{(q:\ell+1)}, \mathbf{X}^{(\ell:p)\mathsf{T}}) \in \Sigma_{\mathrm{S}}$. By Lemma 3.18 and the assumption on the factors $\mathbf{X}^{(\ell)}$, $1 \leq \ell \leq J$, the matrices $\mathbf{X}^{(q:\ell+1)}$ and $\mathbf{X}^{(\ell:p)\mathsf{T}}$ do not have a zero column. The same is true for $\mathbf{D}^{-1}\mathbf{X}^{(q:\ell+1)}$ and $\bar{\mathbf{D}}\mathbf{X}^{(\ell:p)\mathsf{T}}$, as the multiplication of a matrix by $\mathbf{D}^{-1}$ or $\bar{\mathbf{D}}$ does not change its support. By Lemma 3.17, we obtain $(\mathbf{D}^{-1}\mathbf{X}^{(q:\ell+1)}, \bar{\mathbf{D}}\mathbf{X}^{(\ell:p)\mathsf{T}}) \in \mathcal{U}(\Sigma_{\mathrm{S}})$. ∎

Proof of Lemma 3.14. Since $\mathbf{X}^{(q_V:p_V)} = \mathbf{X}^{(q_V:\ell_V+1)}\mathbf{X}^{(\ell_V:p_V)}$, we can factorize the matrix $\mathbf{H}^{(q_V:p_V)} = {\mathbf{D}^{(q_V)}}^{-1}\mathbf{X}^{(q_V:p_V)}\mathbf{D}^{(p_V-1)}$ as $\mathbf{H}^{(q_V:p_V)} = \bar{\mathbf{X}}\bar{\mathbf{Y}}^{\mathsf{T}}$, with $\bar{\mathbf{X}} := {\mathbf{D}^{(q_V)}}^{-1}\mathbf{X}^{(q_V:\ell_V+1)}$, $\bar{\mathbf{Y}} := (\mathbf{X}^{(\ell_V:p_V)}\mathbf{D}^{(p_V-1)})^{\mathsf{T}}$. By Lemma 3.4, $(\bar{\mathbf{X}}, \bar{\mathbf{Y}}) \in \Sigma_{\mathrm{S}}$ where $\mathrm{S} := (\mathbf{W}^{(q_V:\ell_V+1)}, \mathbf{W}^{(\ell_V:p_V)\mathsf{T}})$. By Theorem 3.16, the factors $(\mathbf{H}^{(q_V:\ell_V+1)}, \mathbf{H}^{(\ell_V:p_V)\mathsf{T}}) \in \Sigma_{\mathrm{S}}$ computed at Line 10 of Algorithm 3.1 (which matches [18, Algorithm 3.1]) minimimize the optimization problem:

$$\min_{\mathbf{X},\mathbf{Y}} \|\mathbf{H}^{(q_V:p_V)} - \mathbf{X}\mathbf{Y}^{\mathsf{T}}\|_F \quad \text{such that } (\mathbf{X}, \mathbf{Y}) \in \Sigma_{\mathrm{S}}.$$

Since $\mathbf{H}^{(q_V:p_V)} = \bar{\mathbf{X}}\bar{\mathbf{Y}}^{\mathsf{T}}$ with $(\bar{\mathbf{X}}, \bar{\mathbf{Y}}) \in \Sigma_{\mathrm{S}}$, this minimum is zero hence the computed pair

satisfies $\mathbf{H}^{(q_V:p_V)} = \mathbf{H}^{(q_V:\ell_V+1)}\mathbf{H}^{(\ell_V:p_V)}$. By Proposition 3.19, the factorization $\mathbf{H}^{(q_V:p_V)} = \bar{\mathbf{X}}\bar{\mathbf{Y}}^\mathsf{T}$ into two factors is essentially unique in Σ_S, so $(\mathbf{H}^{(q_V:\ell_V+1)}, \mathbf{H}^{(\ell_V:p_V)\mathsf{T}}) \sim (\bar{\mathbf{X}}, \bar{\mathbf{Y}})$. ∎

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 $\mathbf{Z} = \mathbf{X}^{(J)} \dots \mathbf{X}^{(1)}$, up to scaling ambiguities.

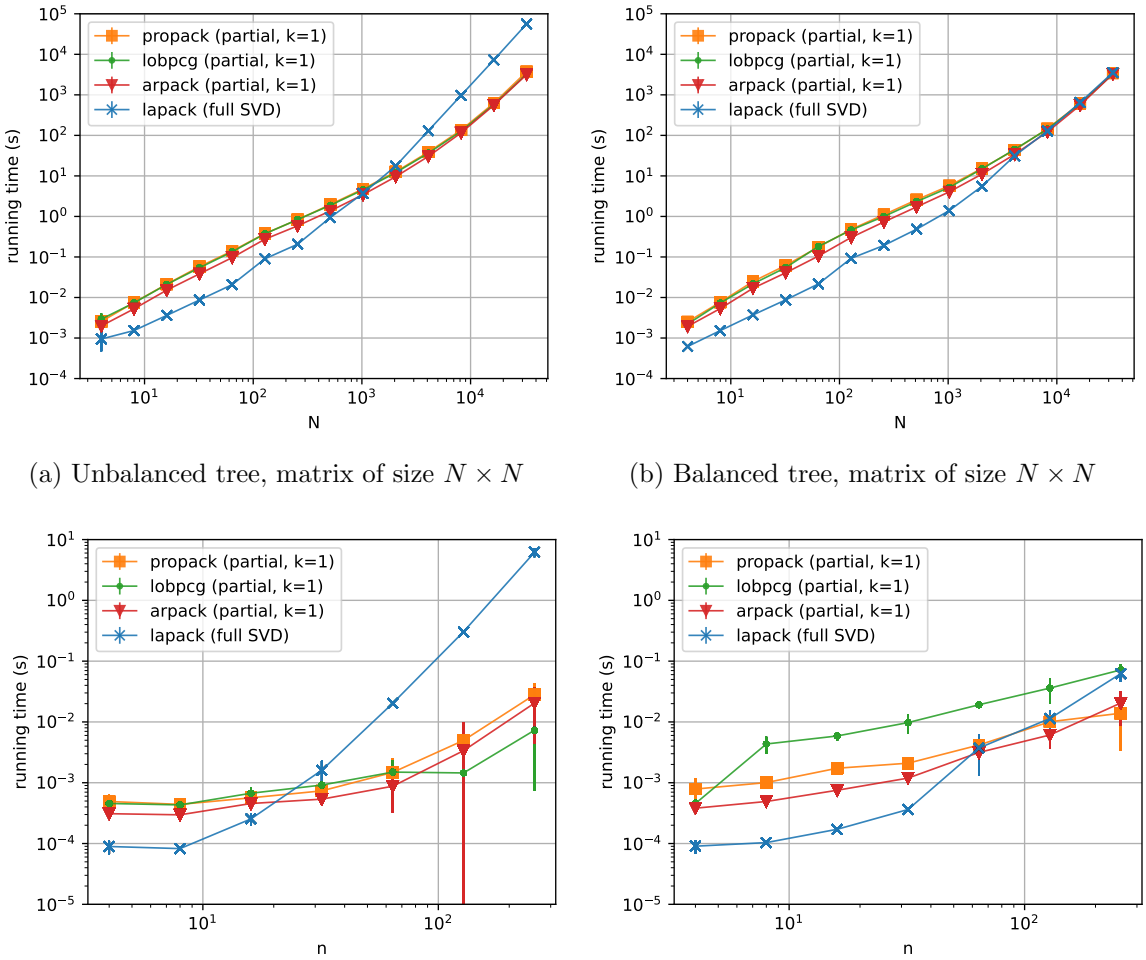
3.5.2. The factorization is essentially unique. We now show that the factorization $\mathbf{Z} = \mathbf{X}^{(J)} \dots \mathbf{X}^{(1)}$ is essentially unique in $\Sigma_{\mathrm{S}_\mathtt{bf}}$ in the sense of Definition 3.9. To that end, consider the factors $(\mathbf{H}^{(\ell:\ell)})_{\ell=1}^J$ computed using Algorithm 3.1 with $\mathbf{Z}$ as input, as well as arbitrary factors $(\bar{\mathbf{X}}^{(J)}, \dots, \bar{\mathbf{X}}^{(1)}) \in \Sigma_{\mathrm{S}_\mathtt{bf}}$ such that $\bar{\mathbf{X}}^{(J)} \dots \bar{\mathbf{X}}^{(1)} = \mathbf{Z}$. We will show that $(\bar{\mathbf{X}}^{(J)}, \dots, \bar{\mathbf{X}}^{(1)})$ verifies the assumptions of Theorem 3.10: this will imply that $(\mathbf{H}^{(\ell:\ell)})_{\ell=1}^J$ 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 $\bar{\mathbf{X}}^{(q:p)} := \bar{\mathbf{X}}^{(q)} \dots \bar{\mathbf{X}}^{(p)}$ for any $1 \leq p \leq q \leq J$. For any $\ell \in [J]$, we have $\mathbf{X}^{(J:\ell+1)}\mathbf{X}^{(\ell:1)} = \mathbf{Z} = \bar{\mathbf{X}}^{(J:\ell+1)}\bar{\mathbf{X}}^{(\ell:1)}$. By Lemma 3.4, $(\bar{\mathbf{X}}^{(J:\ell+1)}, \bar{\mathbf{X}}^{(\ell:1)\mathsf{T}}) \in \Sigma_\mathrm{S}$ where $\mathrm{S} := (\mathbf{W}^{(J:\ell+1)}, \mathbf{W}^{(\ell:1)\mathsf{T}})$. Besides, by assumption and by Lemma 3.18, $\mathbf{X}^{(J:\ell+1)}$ and $\mathbf{X}^{(\ell:1)\mathsf{T}}$ do not have a zero column, so by Lemma 3.17, $(\mathbf{X}^{(J:\ell+1)}, \mathbf{X}^{(\ell:1)\mathsf{T}}) \in \mathcal{U}(\Sigma_\mathrm{S})$. By definition of the set $\mathcal{U}(\Sigma_\mathrm{S})$ of essentially unique factors, $(\bar{\mathbf{X}}^{(J:\ell+1)}, \bar{\mathbf{X}}^{(\ell:1)\mathsf{T}}) \sim (\mathbf{X}^{(J:\ell+1)}, \mathbf{X}^{(\ell:1)\mathsf{T}})$. Since $\mathbf{X}^{(J:\ell+1)}$ and $\mathbf{X}^{(\ell:1)\mathsf{T}}$ do not have a zero column, this implies that $\bar{\mathbf{X}}^{(J:\ell+1)}$ and $\bar{\mathbf{X}}^{(\ell:1)\mathsf{T}}$ do not have zero column either. Consequently, $\bar{\mathbf{X}}^{(\ell+1)}$ does not have a zero column (otherwise $\bar{\mathbf{X}}^{(J:\ell+1)}$ would have a zero column) and similarly $\bar{\mathbf{X}}^{(\ell)}$ does not have a zero row. As this holds for any $\ell \in [J]$, $(\bar{\mathbf{X}}^{(\ell)})_{\ell=1}^J$ verifies the assumption of Theorem 3.10. This ends the proof of Theorem 3.10.

4. Numerical experiments. The empirical behavior of Algorithm 3.1 was first studied in [19], showing the superiority of the hierarchical algorithm for solving (3.10) compared to gradient-based optimization [8], 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, 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 [36]. 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 dif-



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

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

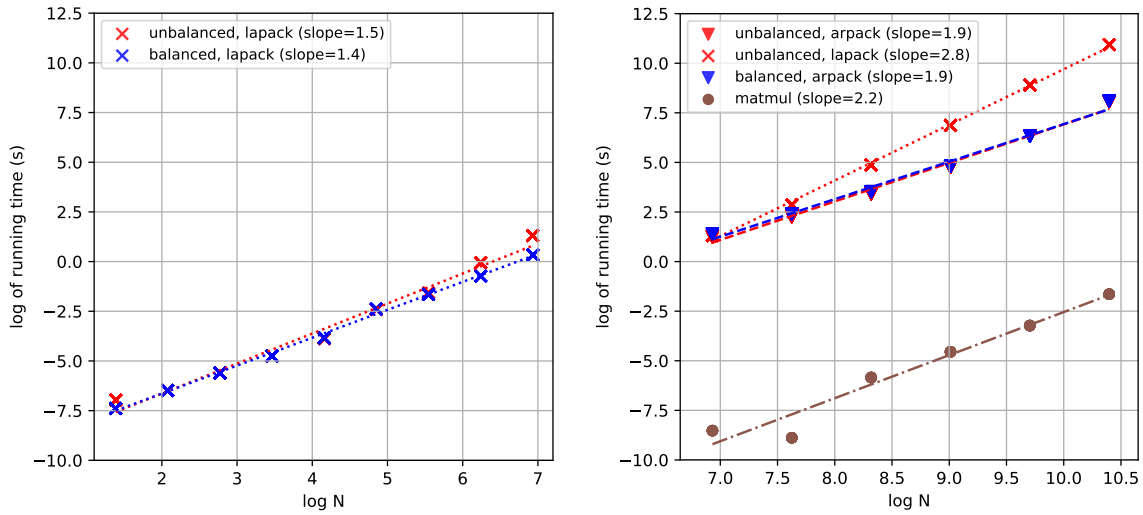
(c) SVD for matrices of size $2 \times n^2/2$

(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.

ference 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 LAPACK solver to compute full SVDs. Hierarchical factorization is slightly faster with a balanced



(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`.

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 [19], is not optimal. This explains why it was observed in [19] 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 multi-layer sparse matrix factorization, when enforcing the butterfly structure as the sparsity constraint. We proved that the butterfly factors $(\mathbf{X}^{(\ell)})_{\ell=1}^{J}$ of $\mathbf{Z} := \mathbf{X}^{(J)} \dots \mathbf{X}^{(1)}$ 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 $\mathcal{O}(N^2)$.

Uniqueness results for $J = 2$ factors. As illustrated in this paper, the 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 2.4 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 [18]. Beyond the fixed-support setting, one can also explore essential uniqueness by considering a *family* of sparsity patterns like in [35]. The main advantage of reducing the analysis to the case with only two factors is the fact that we can use the rank-one contributions representation from (2.7) instead of tensorial lifting. One can then use results from matrix completion literature [10, 15, 7] to establish more elaborate conditions for identifiability in the case with two factors.

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 [19]. 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.

Sparse ReLU neural networks, learning fast transforms. Finally, because of its efficiency and its complexity bound comparable to matrix-vector multiplication, Algorithm 3.1 can be used in any large-scale learning algorithm whenever an approximation of a dense matrix by a product of sparse butterfly factors is needed to save computational resources, e.g., in dictionary learning or neural network training. This also opens the way to study identifiability in deep neural networks with weight matrices satisfying the butterfly structure.

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 (Lyon, France) for Machine Learning facilities. The platform operates the SIDUS solution [31] developed by Emmanuel Quemener.

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. 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Q.-T. Le, E. Riccietti, and R. Gribonval, *Spurious Valleys, Spurious Minima and NP-hardness of Sparse Matrix Factorization With Fixed Support.* preprint, https://hal.inria.fr/hal-03364668, May 2021, https://hal.inria.fr/hal-03364668.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.
[28] 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.
[29] 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.
[30] 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.
[31] 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.
[32] R. Rubinstein, A. M. Bruckstein, and M. Elad, *Dictionaries for sparse representation modeling*, Proceedings of the IEEE, 98 (2010), pp. 1045–1057, https://doi.org/10.1109/JPROC.2010.2040551.
[33] 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.
[34] 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.
[35] 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.
[36] 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 2.2.

We begin with a simple lemma [12, Chapter 3].

Lemma A.1. *Let Σ 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 2.2 is then derived from the following trivial but crucial observation.

Lemma A.2. *Let Σ 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* colsupp$(\bar{\mathbf{Y}})$, 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 2.2. 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 colsupp$(\mathbf{X}) \neq$ colsupp$(\mathbf{Y})$. Up to matrix transposition, we can suppose without loss of generality that colsupp$(\mathbf{Y})$ is not a subset of 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 $\bar{\mathbf{Y}}_{[r]\setminus\{i\}} = \mathbf{Y}_{[r]\setminus\{i\}}$[4]. By construction, supp$(\bar{\mathbf{Y}}) \subseteq$ supp$(\mathbf{Y})$, so $(\mathbf{X},\bar{\mathbf{Y}}) \in \Sigma_{\mathrm{S}}$. Applying Lemma A.2 to $\Sigma = \Sigma_{\mathrm{S}}$, we

[4]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 $I \subseteq [N]$.

obtain $(\mathbf{X}, \mathbf{Y}) \notin \mathcal{U}(\Sigma_S)$. We now show $\mathcal{U}(\Sigma_S) \subseteq \mathtt{MC}_S$, also by contraposition. Let $(\mathbf{X}, \mathbf{Y}) \notin \mathtt{MC}_S$, and assume $\operatorname{colsupp}(\mathbf{X}) \neq \operatorname{colsupp}(\mathbf{S}^L)$. The reasoning would be symmetric if we supposed $\operatorname{colsupp}(\mathbf{Y}) \neq \operatorname{colsupp}(\mathbf{S}^R)$. By the previously shown inclusion $\mathcal{U}(\Sigma_S) \subseteq \mathtt{IC}_S$, we also assume $\operatorname{colsupp}(\mathbf{X}) = \operatorname{colsupp}(\mathbf{Y})$ without loss of generality. To conclude we treat three cases:

- If $\operatorname{colsupp}(\mathbf{S}^R) \not\subseteq \operatorname{colsupp}(\mathbf{S}^L)$ 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 $\Sigma = \Sigma_S$, to show $(\mathbf{X}, \mathbf{Y}) \notin \mathcal{U}(\Sigma_S)$.
- If $\operatorname{colsupp}(\mathbf{S}^L) \not\subseteq \operatorname{colsupp}(\mathbf{S}^R)$, then the same arguments yield $(\mathbf{X}, \mathbf{Y}) \notin \mathcal{U}(\Sigma_S)$.
- There remains the case $\operatorname{colsupp}(\mathbf{S}^L) = \operatorname{colsupp}(\mathbf{S}^R)$. Since $\operatorname{colsupp}(\mathbf{X}) \subsetneq \operatorname{colsupp}(\mathbf{S}^L)$, let us fix $i \in \operatorname{colsupp}(\mathbf{S}^L)$ 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 $\Sigma = \Sigma_S$, showing that $(\mathbf{X}, \mathbf{Y}) \notin \mathcal{U}(\Sigma_S)$. ∎

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

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

Proof. The direct inclusion is immediate by Lemmas 2.2 and A.1. For the inverse inclusion, let $(\mathbf{X}^*, \mathbf{Y}^*) \in \mathcal{U}(\mathtt{IC}_S) \cap \mathtt{MC}_S$, and $(\mathbf{X}, \mathbf{Y}) \in \Sigma_S$ 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 $J = \operatorname{colsupp}(\mathbf{X}) \cap \operatorname{colsupp}(\mathbf{Y})$. Define $(\bar{\mathbf{X}}, \bar{\mathbf{Y}}) \in \mathtt{IC}_S$ 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 $(\mathbf{X}^*, \mathbf{Y}^*) \in \mathcal{U}(\mathtt{IC}_S)$, we have $(\bar{\mathbf{X}}, \bar{\mathbf{Y}}) \sim (\mathbf{X}^*, \mathbf{Y}^*)$. But $\operatorname{colsupp}(\mathbf{X}^*) = \operatorname{colsupp}(\mathbf{S}^L)$ and $\operatorname{colsupp}(\mathbf{Y}^*) = \operatorname{colsupp}(\mathbf{S}^R)$, because $(\mathbf{X}^*, \mathbf{Y}^*) \in \mathtt{MC}_S$. Hence, $J = \operatorname{colsupp}(\bar{\mathbf{X}}) = \operatorname{colsupp}(\mathbf{X}^*) = \operatorname{colsupp}(\mathbf{S}^L)$ and similarly $J = \operatorname{colsupp}(\mathbf{S}^R)$. 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 exploit the correspondence between pairs of factors $(\mathbf{X}, \mathbf{Y})$ and their rank-one contributions representation $\varphi(\mathbf{X}, \mathbf{Y})$.

Lemma B.1. *The application* φ *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* φ *restricted to* $\mathtt{IC}_S$*, denoted* $\varphi_S : \mathtt{IC}_S \to \Gamma_{\mathcal{S}}$ *is surjective, and injective up to equivalences, in the sense that* $(\mathbf{X}, \mathbf{Y}) \sim (\bar{\mathbf{X}}, \bar{\mathbf{Y}})$ *for any* $(\mathbf{X}, \mathbf{Y}), (\bar{\mathbf{X}}, \bar{\mathbf{Y}}) \in \mathtt{IC}_S$ *such that* $\varphi_S(\mathbf{X}, \mathbf{Y}) = \varphi_S(\bar{\mathbf{X}}, \bar{\mathbf{Y}})$.

The proof of Lemma B.1 is straightforward and left to the reader.

We now define the operator which 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 [5]. 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 lemma leads to the following theorem characterizing essential uniqueness of the factorization $\mathbf{Z} := \mathbf{X}\mathbf{Y}^\intercal$ in Σ_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.2. *For any pair of supports* S*, denoting* $\mathcal{S} := \varphi(\mathrm{S})$*, we have:*

$$(\mathbf{X}, \mathbf{Y}) \in \mathcal{U}(\Sigma_S) \iff \varphi(\mathbf{X}, \mathbf{Y}) \in \mathcal{U}(\Gamma_{\mathcal{S}}) \text{ and } (\mathbf{X}, \mathbf{Y}) \in \mathtt{IC}_S \cap \mathtt{MC}_S.$$

Proof. By Lemma B.1, one verifies that $(\mathbf{X}, \mathbf{Y}) \in \mathcal{U}(\mathtt{IC}_S) \iff \varphi(\mathbf{X}, \mathbf{Y}) \in \mathcal{U}(\Gamma_{\mathcal{S}})$. As

$\mathcal{U}(\Sigma_{\mathrm{S}}) = \mathcal{U}(\mathtt{IC}_{\mathrm{S}}) \cap \mathtt{MC}_{\mathrm{S}}$ by Proposition A.3, it follows that

$$\begin{aligned}(\mathbf{X},\mathbf{Y}) \in \mathcal{U}(\Sigma_{\mathrm{S}}) &\iff (\mathbf{X},\mathbf{Y}) \in \mathcal{U}(\mathtt{IC}_{\mathrm{S}}) \cap \mathtt{MC}_{\mathrm{S}} \\ &\iff \varphi(\mathbf{X},\mathbf{Y}) \in \mathcal{U}(\Gamma_{\mathcal{S}}) \text{ and } (\mathbf{X},\mathbf{Y}) \in \mathtt{IC}_{\mathrm{S}} \cap \mathtt{MC}_{\mathrm{S}}.\end{aligned}$$

■

Appendix C. Deriving a simple condition of identifiability. We now use the previous theorem to prove the simple sufficient condition in Proposition 2.4 for essential uniqueness of the factorization $\mathbf{Z} := \mathbf{X}\mathbf{Y}^\intercal$ in Σ_{S}, which has been suggested by [22, Chapter 7]. We start by:

Lemma C.1. *Let $\mathcal{S}$ be an r-tuple of rank-one supports. Then, the linear application $\mathcal{A} : \mathcal{C} \mapsto \sum_{i=1}^r \boldsymbol{\mathcal{C}}^i$ restricted to $\Gamma_{\mathcal{S}}$ is injective, i.e., $\mathcal{U}(\Gamma_{\mathcal{S}}) = \Gamma_{\mathcal{S}}$, if, and only if, the rank-one supports $\{\boldsymbol{\mathcal{S}}^i\}_{i=1}^r$ are pairwise disjoint.*

Proof. Sufficiency is immediate: given $\mathcal{C} \in \Gamma_{\mathcal{S}}$ 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 $(\mathbf{Z}_{k,l})_{(k,l)\in\boldsymbol{\mathcal{S}}^i}$, because the rank-one supports in the tuple $\mathcal{S}$ are pairwise disjoint. For necessity, suppose there exists $i, j \in [r]$ such that $\boldsymbol{\mathcal{S}}^i \cap \boldsymbol{\mathcal{S}}^j \neq \emptyset$. 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). Define $\mathcal{C}, \bar{\mathcal{C}} \in \Gamma_{\mathcal{S}}$ as

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

Then, $\mathcal{A}(\mathcal{C}) = 0 = \mathcal{A}(\bar{\mathcal{C}})$, but $\mathcal{C} \neq \bar{\mathcal{C}}$, which shows $\mathcal{C}, \bar{\mathcal{C}} \neq \mathcal{U}(\Gamma_{\mathcal{S}})$, hence, $\mathcal{U}(\Gamma_{\mathcal{S}}) \neq \Gamma_{\mathcal{S}}$. ■

Proposition 2.4 is then a corollary of Theorem B.2 combined with Lemma C.1.

Appendix D. Complexity bounds of Algorithm 3.1 with full SVD.

In complement to subsection 3.4 we bound the complexity of Algorithm 3.1 computed with *full* SVDs. The complexity of full SVD for a matrix of size $m \times n$ is $\mathcal{O}(mn\min(m,n))$.

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 this costs of the order of $2 \times N/2^{j+1} \times 2 = 2 \times N/2^j$, hence a total cost (with unbalanced tree and full SVD) of $\sum_{j=0}^{J-2} N \times 2 \times \frac{N}{2^j} = 4(1-2^{-J+1})N^2 = 4\left(1-\frac{2}{N}\right)N^2 = \mathcal{O}(4N^2)$.

Balanced tree. At each depth $k \in \{0,\dots,\log_2(J)-1\}$, there are 2^k nodes in this tree. At each non-leaf node of depth $k \in \{0,\dots,\log_2(J)-1\}$, the algorithm computes the best rank-one approximation of N submatrices of size $\sqrt{N^{1/2^k}} \times \sqrt{N^{1/2^k}}$. Using a full SVD, the complexity of each such best rank-one approximation is of the order of $\sqrt{N^{1/2^k}} \times \sqrt{N^{1/2^k}} \times \sqrt{N^{1/2^k}} = N^{3/2^{k+1}}$. Using that for $1 \le k \le \log_2(J)-1$ we have $2^k \le J/2 = \log_2(N)/2$ and $N^{1+3/2^{k+1}} \le N^{1+3/4} = N^{7/4}$, the cost of the algorithm with balanced tree and full SVD is

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