Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Equivalent polyadic decompositions of matrix multiplication tensors[☆]

Guillaume O. Berger[a,1], Pierre-Antoine Absil[a], Lieven De Lathauwer[b,c], Raphaël M. Jungers[a,2], Marc Van Barel[d,*]

[a] ICTEAM Institute, UCLouvain, 1348 Louvain-la-Neuve, Belgium
[b] Group of Science, Engineering and Technology, KU Leuven Kulak, 8500 Kortrijk, Belgium
[c] Department of Electrical Engineering (ESAT), KU Leuven, 3001 Leuven, Belgium
[d] Department of Computer Science, KU Leuven, 3001 Leuven, Belgium

ARTICLE INFO

Article history:
Received 8 July 2020
Received in revised form 8 November 2021

MSC:
15A69
14Q20
68W30

Keywords:
Fast matrix multiplication
Polyadic tensor decompositions
Eigenvalue decomposition

ABSTRACT

Invariance transformations of polyadic decompositions of matrix multiplication tensors define an equivalence relation on the set of such decompositions. In this paper, we present an algorithm to efficiently decide whether two polyadic decompositions of a given matrix multiplication tensor are equivalent. With this algorithm, we analyze the equivalence classes of decompositions of several matrix multiplication tensors. This analysis is relevant for the study of fast matrix multiplication as it relates to the question of how many essentially different fast matrix multiplication algorithms there exist. This question has been first studied by de Groote, who showed that for the multiplication of 2×2 matrices with 7 active multiplications, all algorithms are essentially equivalent to Strassen's algorithm. In contrast, the results of our analysis show that for the multiplication of larger matrices (e.g., 2×3 by 3×2 or 3×3 by 3×3 matrices), two decompositions are very likely to be essentially different. We further provide a necessary criterion for a polyadic decomposition to be equivalent to a polyadic decomposition with integer entries. Decompositions with specific integer entries, e.g., powers of two, provide fast matrix multiplication algorithms with better efficiency and stability properties. This condition can be tested algorithmically and we present the conclusions obtained for the decompositions of small/medium matrix multiplication tensors.

1. Introduction

The straightforward way to multiply two $N\times N$ matrices costs $\mathcal{O}(N^3)$ operations. In particular, multiplying two 2×2 matrices requires 8 scalar multiplications. However, as first remarked by V. Strassen [1] in 1969, the arithmetic operations

☆ The work of G. Berger was supported by (i) the Fonds de la Recherche Scientifique – FNRS, Belgium and the Fonds Wetenschappelijk Onderzoek – Vlaanderen under EOS Project no 30468160, (ii) "Communauté française de Belgique, Belgium – Actions de Recherche Concertées" (contract ARC 14/19-060). L. De Lathauwer and M. Van Barel's research is funded by (i) the Research Council KU Leuven, Belgium, C1-project c16/15/059-nD (Numerical Linear Algebra and Polynomial Computations), and by (ii) the Fund for Scientific Research–Flanders (Belgium), EOS Project no 30468160 (SeLMA). R. Jungers is supported by the Walloon Region and the Innoviris, Belgium Foundation.

* Corresponding author.
E-mail addresses: guillaume.berger@uclouvain.be (G.O. Berger), pa.absil@uclouvain.be (P.-A. Absil), lieven.delathauwer@kuleuven.be (L. De Lathauwer), raphael.jungers@uclouvain.be (R.M. Jungers), marc.vanbarel@cs.kuleuven.be (M. Van Barel).

1 G. Berger is an FNRS/FRIA Fellow.
2 R. Jungers is an FNRS Research Associate.

https://doi.org/10.1016/j.cam.2021.113941
0377-0427/© 2021 Elsevier B.V. All rights reserved.

can be grouped cleverly to reduce the work to 7 multiplications only. By doing this recursively, we can reduce the cost for the multiplication of $N \times N$ matrices to $\mathcal{O}(N^{2.81})$ operations. Strassen's discovery opened the door to a considerable amount of research on the algorithmic complexity of matrix multiplication (see paragraphs below for a survey). The reduction of the complexity may actually become so significant that a new architecture for large matrix multiplication is emerging. Essential is first that we find inexpensive schemes for the multiplication of relatively small matrices.

The multiplication of $m \times p$ matrices by $p \times n$ matrices can be represented by a third-order tensor. Finding inexpensive schemes for the multiplication of such matrices can be approached by decomposing the associated tensor as a sum of rank-1 terms (such a decomposition is called *polyadic decomposition*). The minimal number of rank-1 terms necessary to decompose a tensor is its rank. In the case of matrix multiplication, the rank of the associated tensor is equal to the smallest number of active multiplications needed to compute the matrix product. (By *active multiplication*, we mean a multiplication of two scalars that both depend on the matrices to be multiplied.) As a consequence, determining the rank of the associated tensor allows us to find an exponent α such that the complexity for the multiplication of $N \times N$ matrices is at most of $\mathcal{O}(N^{\alpha})$ arithmetic operations [2].

Although the problem of matrix multiplication complexity is quite old, only partial results are known so far. Even for the multiplication of small matrices, determining the rank of the associated tensor is still an open problem. The largest case that is completely understood is the multiplication of 2×2 matrices by 2×2 matrices. The rank of the associated tensor is 7 [3] (so that Strassen's algorithm is optimal), and it was proved by de Groote [4] that the decomposition induced by Strassen's algorithm is essentially unique (with respect to a class of transformations acting on polyadic decompositions of matrix multiplication; see the paragraph hereunder). For the multiplication of 3×3 matrices, an algorithm using 23 active multiplications was proposed by Laderman [5] in 1976, and by Makarov in 1986 [6]; see also [7]. This means that the rank of the associated tensor is at most 23. On the other hand, Bläser proved [8] in 2003 that the rank for the multiplication of 3×3 matrices should be at least 19. The gap 19–23 has not been reduced since then. Other algorithms for the multiplication of matrices with small and medium sizes are proposed in [9–12]; also the paper [13] of Ballard et al. gives a good overview of the practical algorithms that are available in 2016. Further reductions of the exponent of matrix multiplication complexity have been achieved, e.g., by Pan [14,15], Bini et al. [16], Schönhage [17], and Coppersmith and Winograd [18] by means of more advanced techniques (including namely the study of the "border rank" of the associated tensor; see also [19,20]). Currently, the best known upper bound for the complexity of matrix multiplication is $\mathcal{O}(N^{2.3729})$ by Le Gall [21].

The aim of this paper is to study the connections between polyadic decompositions for matrix multiplication tensors. In particular, we consider three types of transformations, called invariance transformations, acting on the set of polyadic decompositions of a given matrix multiplication tensor, and we study the equivalence relation induced by these transformations. These transformations have been studied by de Groote [4,22] who has shown that Strassen's algorithm is essentially unique in the sense that every other decomposition with 7 rank-1 terms is equivalent to it. In contrast, for the multiplication of 3×3 matrices, Johnson and McLoughlin [23] showed that Laderman's algorithm [5] is not essentially unique. They provided two parametrized families of decompositions (with 23 rank-1 terms) of the 3×3 matrix multiplication tensor that are mutually inequivalent and also inequivalent to Laderman's. Later, Oh, Kim and Moon [24] discovered other decompositions of the 3×3 matrix multiplication tensor inequivalent to the previous ones, and Sedoglavic [7] showed that Laderman's algorithm can be constructed using Strassen's algorithm and related tensor's transformations.

The techniques used by de Groote, Johnson and McLoughlin, and Oh et al. to prove the equivalence/inequivalence of decompositions are either too specific [4,23] or too conservative [24] (some inequivalent decompositions are not recognized as such) to be applied to general decompositions of matrix multiplication tensors of arbitrary size. In this paper, we present an algorithm for deciding whether two given decompositions are equivalent through invariance transformations. Thanks to this algorithm, we were able to study the equivalence classes of large sample sets of matrix multiplication tensor decompositions (computed with numerical methods, see Section 6). This allows us to get a better understanding of the equivalence relation of decompositions: for instance, the numerical experiments (Section 6.3) suggest that for tensors larger than the 2×2 by 2×2 case, two "generic" decompositions are inequivalent.

In addition, we describe a necessary criterion for a matrix multiplication tensor decomposition to be *discretizable*, that is, to be equivalent to a decompositions whose rank-1 terms can be factorized into vectors or matrices whose entries only take a few distinct values (for instance, we may want that all entries of the factor vectors/matrices of the rank-1 terms belong to the set $\{-1, 0, +1\}$). Such decompositions are called discrete decompositions [25]. Our interest in discretizable decompositions originates from the observation that for small/medium matrix multiplication, the decompositions proposed in the literature are generally discrete: Strassen's and Laderman's algorithms are discrete with factor matrices coefficients belonging to $\{-1, 0, +1\}$, other (inequivalent) decompositions with coefficients in $\{-1, 0, +1\}$ are proposed, e.g., in [13,24,26] for the multiplication of 2×3 by 3×2, 2×3 by 3×3, and 3×3 by 3×3 matrices. In particular, all the decompositions listed in [13] are discrete.

Discrete decompositions provide matrix multiplication algorithms with better efficiency and stability properties. However, the classical iterative processes for computing tensor decompositions do not lead in general to solutions of this kind. A reasonable approach to compute discrete decompositions is then to (i) compute a general decomposition, and (ii) use invariance transformations to obtain an equivalent discrete decomposition. Closely related methods are used, e.g., in [23,26]. The necessary criterion for discretizability allows us to identify some decompositions that cannot be transformed via invariance transformations into a discrete decomposition with a given "target set" for the coefficients. By applying the necessary criterion to the sample sets of decompositions, we observed that, contrary to what the decompositions available in the literature suggest, most of the decompositions for tensors larger than the 2×2 by 2×2 case are not discretizable with respect to the commonly-used target sets (e.g., $\{0, \pm 1\}$ or $\{0, \pm 1/2, \pm 1\}$).

The paper is organized as follows. In Section 2, we introduce the notation and recall the definitions of matrix multiplication tensors and polyadic decompositions. Invariance transformations and the induced equivalence relations are introduced in Section 3. In Section 4, we describe the algorithm for deciding whether two decompositions are equivalent and if so, computing the invariance transformations involved in their equivalence. The necessary criterion for discretizability is discussed in Section 5. Numerical experiments are presented in Section 6.

2. Preliminaries

2.1. Matrix multiplication tensors and polyadic decompositions

Let U, V and W be vector spaces over a field $\mathbb{F}$. We denote by $\mathrm{Bil}(U, V; W)$ the set of $\mathbb{F}$-bilinear maps from $U \times V$ to W. For positive integers m, p, n, the multiplication of $m \times p$ matrices by $p \times n$ matrices can be represented by the bilinear map $\Phi_{m,p,n} \in \mathrm{Bil}(\mathbb{F}^{m\times p}, \mathbb{F}^{p\times n}; \mathbb{F}^{m\times n})$ defined by

$$\Phi_{m,p,n}(\mathbf{A}, \mathbf{B}) = \mathbf{AB}.$$

From the identification between multilinear maps and tensors, $\Phi_{m,p,n}$ is sometimes referred to as the (m, p, n) *matrix multiplication tensor.*

The concept of rank of a bilinear map $\Phi \in \mathrm{Bil}(U, V; W)$ is central in the analysis of the asymptotic complexity of matrix multiplication. We say that $\Phi \neq 0$ has rank 1 if $\Phi(u, v) = f(u)g(v)w$ for some $f \in U^*$, $g \in V^*$ and $w \in W$, where U^* and V^* are the dual spaces of U and V respectively. For a general $\Phi \in \mathrm{Bil}(U, V; W)$, an *$F$-term polyadic decomposition* (in short F-PD) of Φ is a decomposition of Φ as the sum of F rank-1 terms [27,28]:

$$\Phi(u, v) = \sum_{r=1}^{F} f_r(u) g_r(v) w_r \tag{1}$$

for some $f_r \in U^*$, $g_r \in V^*$ and $w_r \in W$. The *rank* of Φ is the smallest F such that Φ admits an F-term polyadic decomposition (1).

For a matrix multiplication tensor $\Phi_{m,p,n}$, a polyadic decomposition like (1) requires $f_r \in (\mathbb{F}^{m\times p})^*$, $g_r \in (\mathbb{F}^{p\times n})^*$ and $w_r = \mathbf{W}_r \in \mathbb{F}^{m\times n}$. We may identify f_r with the unique matrix $\mathbf{U}_r \in \mathbb{F}^{p\times m}$ such that

$$f_r(\mathbf{A}) = \mathrm{trace}(\mathbf{U}_r\mathbf{A}) = \sum_{i=1}^{p}\sum_{j=1}^{m} \mathbf{U}_r^{(i,j)}\mathbf{A}^{(j,i)} \tag{2}$$

for every $\mathbf{A} \in \mathbb{R}^{m\times p}$, where $\mathbf{M}^{(i,j)}$ denotes the (i, j)th entry of a matrix $\mathbf{M}$. In the same way, g_r can be identified with a unique matrix $\mathbf{V}_r \in \mathbb{F}^{n\times p}$. If $\mathbf{U}_1, \ldots, \mathbf{U}_F$, $\mathbf{V}_1, \ldots, \mathbf{V}_F$ and $\mathbf{W}_1, \ldots, \mathbf{W}_F$ give rise to a decomposition (1) of $\Phi_{m,p,n}$, we will say with slight abuse of notation that the triple $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$[3] is an F-term polyadic decomposition (F-PD) of $\Phi_{m,p,n}$.

The link between matrix multiplication complexity and the rank of matrix multiplication tensors is nicely explained in [2, Chapter 15]. Especially, it is shown how to build from an F-term polyadic decomposition (F-PD) of $\Phi_{m,p,n}$ a recursive algorithm for the multiplication of $N \times N$ matrices over $\mathbb{F}$ with complexity in $\mathcal{O}(N^{\omega+\varepsilon})$ arithmetic operations $\{+, -, \times\}$, with $\omega = 3\log_{mpn}(F)$ and for any $\varepsilon > 0$. For instance, Strassen's algorithm can be obtained from a decomposition of $\Phi_{2,2,2}$ with 7 terms. This directly gives the well-known upper bound $\omega = 3\log_8(7) \approx 2.81$ for the exponent of matrix multiplication complexity [1].

2.2. Discrete decompositions

In this paper, we focus on algorithms for matrix multiplication over the field of real numbers, i.e., on the case $\mathbb{F} = \mathbb{R}$ and $\mathbf{U}_r$, $\mathbf{V}_r$ and $\mathbf{W}_r$ are real matrices.

For the problem of matrix multiplication over $\mathbb{R}$ (or $\mathbb{C}$), two decompositions might be not equally useful even if they have the same number of rank-1 terms. For instance, decompositions with "structured" values in the rank-1 terms are more useful in practice. This leads us to the following definition:

Definition 2.1. A decomposition $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ of $\Phi_{m,p,n}$ is said to be *discrete* if the entries of $\mathbf{U}_r$, $\mathbf{V}_r$ and $\mathbf{W}_r$ belong to $q\mathbb{Z}$ for some $q \in \mathbb{R}$.

[3] The rational behind this notation is that if $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_F$ are F mathematical objects, then $\mathbf{X}_{[F]}$ denotes the ordered set, or F-uple, $(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_F)$.

Discrete decompositions are favorable for two reasons. The first reason concerns the exactness of the decomposition: if $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ is computed with numerical methods, then this will give a decomposition of $\Phi_{m,p,n}$ only up to some *finite* accuracy (and also up to machine precision, due to floating-point arithmetic computations). Hence, the matrix multiplication algorithm obtained from this decomposition will compute the product of $\mathbf{A} \in \mathbb{R}^{m\times p}$ and $\mathbf{B} \in \mathbb{R}^{p\times n}$ with a small error, even in exact arithmetic. This is not advisable because this error will accumulate when we will apply the algorithm in a recursive way to compute the product of general $N \times N$ matrices [2, Chapter 15]. These limited-accuracy issues can be overcome if we know a priori that $\mathbf{U}_r$, $\mathbf{V}_r$ and $\mathbf{W}_r$ have their entries in a known discrete set.

The second reason to favor discrete decompositions is that the obtained algorithm for matrix multiplication will have better stability and computational cost properties. Indeed, if the entries of $\mathbf{U}_r$, $\mathbf{V}_r$ and $\mathbf{W}_r$ belong to $q\mathbb{Z}$, then it is not hard to show (we do not go into the details) that, modulo some pre- and post multiplication of $\mathbf{A} \in \mathbb{R}^{m\times p}$ and $\mathbf{B} \in \mathbb{R}^{p\times n}$ by q, the product $\mathbf{AB}$ can be computed using only additions and multiplications of the entries of $\mathbf{A}$ and $\mathbf{B}$ by integers. (For example, in Strassen's algorithm, $f_r(\mathbf{A})$ [resp. $g_r(\mathbf{B})$] can be obtained using only additions and subtractions of the entries of $\mathbf{A}$ [resp. $\mathbf{B}$].) Multiplication by integers is more rapid and stable than multiplication by arbitrary floating-point numbers (for instance, multiplication by a power of 2 is equivalent to changing the exponent in the floating point representation). For more detailed information on the forward normwise error induced by a fast matrix multiplication algorithm, we refer the interested reader to [13].

3. Invariance transformations

The main goal of this paper is to study relations between decompositions of a given matrix multiplication tensor. We describe three types of operations that transform an F-PD of a matrix multiplication tensor into another F-PD of the same tensor. These transformations will be referred to as *invariance transformations*.

Proposition 3.1 (*Invariance Transformations*). *Let $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ be an F-PD of $\Phi_{m,p,n}$. The following transformations produce matrices $\mathbf{U}'_r$, $\mathbf{V}'_r$ and $\mathbf{W}'_r$ ($1 \le r \le F$) such that $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ is also an F-PD of $\Phi_{m,p,n}$:*

- *Permutation transformations: let $\sigma \in \mathfrak{S}_F$ (where $\mathfrak{S}_F$ is the set of permutations of $\{1, \ldots, F\}$) and define*
$$\mathbf{U}'_r = \mathbf{U}_{\sigma(r)}, \quad \mathbf{V}'_r = \mathbf{V}_{\sigma(r)}, \quad \mathbf{W}'_r = \mathbf{W}_{\sigma(r)}.$$
- *Scaling transformations: choose coefficients $\lambda_r, \mu_r, \nu_r \in \mathbb{R}$ such that $\lambda_r\mu_r\nu_r = 1$ for each $1 \le r \le F$ and define*
$$\mathbf{U}'_r = \lambda_r\mathbf{U}_r, \quad \mathbf{V}'_r = \mu_r\mathbf{V}_r, \quad \mathbf{W}'_r = \nu_r\mathbf{W}_r.$$
- *Trace transformations: let $\mathbf{P} \in \mathrm{GL}(m)$, $\mathbf{Q} \in \mathrm{GL}(p)$ and $\mathbf{R} \in \mathrm{GL}(n)$ (where $\mathrm{GL}(h)$ denotes the set of invertible $h \times h$ matrices), and define*
$$\mathbf{U}'_r = \mathbf{Q}^{-1}\mathbf{U}_r\mathbf{P}, \quad \mathbf{V}'_r = \mathbf{R}^{-1}\mathbf{V}_r\mathbf{Q}, \quad \mathbf{W}'_r = \mathbf{P}^{-1}\mathbf{W}_r\mathbf{R}.$$

The first two classes of invariance transformations (permutations and scaling) provide invariance transformations for the decompositions of any tensors. However, the third class (trace transformations) is somehow specific to matrix multiplication tensors, as it originates from the invariance of the trace operator (see proof below); hence the name "trace transformations".

Proof of Proposition 3.1. (See, e.g., [22]). The invariance of the permutation and scaling transformations is straightforward. For the trace transformations, let $f'_r \in (\mathbb{R}^{m\times p})^*$, $g'_r \in (\mathbb{R}^{p\times n})^*$ and $\Phi' \in \mathrm{Bil}(\mathbb{R}^{m\times p}, \mathbb{R}^{p\times n}; \mathbb{R}^{m\times n})$ be given by (2) and (1) with $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$. Then
$$f'_r(\mathbf{A}) = \mathrm{trace}(\mathbf{U}'_r\mathbf{A}) = \mathrm{trace}(\mathbf{U}_r[\mathbf{PAQ}^{-1}]) = f_r(\mathbf{PAQ}^{-1})$$
where we have used the invariance property of the trace operator with respect to cyclic permutations. Similarly, $g'_r(\mathbf{B}) = g_r(\mathbf{QBR}^{-1})$. It follows that
$$\begin{aligned}\Phi'(\mathbf{A}, \mathbf{B}) &= \mathbf{P}^{-1}\left[\sum_{r=1}^{F} f_r(\mathbf{PAQ}^{-1})g_r(\mathbf{QBR}^{-1})\mathbf{W}_r\right]\mathbf{R} \\ &= \mathbf{P}^{-1}\Phi_{m,p,n}(\mathbf{PAQ}^{-1}, \mathbf{QBR}^{-1})\mathbf{R} = \mathbf{P}^{-1}(\mathbf{PAQ}^{-1})(\mathbf{QBR}^{-1})\mathbf{R} = \mathbf{AB}.\end{aligned}$$
Thus $\Phi' = \Phi_{m,p,n}$ showing that $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ is an F-PD of $\Phi_{m,p,n}$. $\square$

Invariance transformations define an equivalence relation on the set of F-PDs of a given matrix multiplication tensor. For fixed m, p, n and F, two polyadic decompositions $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ of $\Phi_{m,p,n}$ are *equivalent* if there exist permutation, scaling and/or trace transformations that allow one to transform $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ into $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$. We will also say that $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ are *permutation-equivalent* if there exists a permutation transformation allowing us to transform $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ into $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$. Similarly, we define the notion of *(scaling+trace)-equivalence*.

We have just seen that invariance transformations can be used to produce many different decompositions (i.e., many fast matrix multiplication algorithms) from a given one. This raises the following questions that we will address in this paper:

How many inequivalent polyadic decompositions does $\Phi_{m,p,n}$ admit? In other words, how many essentially different fast matrix multiplication algorithms are there for the multiplication of $m \times p$ matrices by $p \times n$ matrices? We will tackle this question in Sections 4 and 6.3.

Starting from a given algorithm for the multiplication of $m \times p$ matrices by $p \times n$ matrices, can we obtain with invariance transformations another algorithm with better performance (e.g., in terms of stability and efficiency)? We will tackle this question in Sections 5 and 6.2.

Remark 3.1. At first sight, it might look like the scaling transformations act as a particular case of the trace transformations with

$$\mathbf{P} = \left(\frac{\lambda}{\nu}\right)^{1/3} \mathbf{I}_m, \quad \mathbf{Q} = \left(\frac{\mu}{\lambda}\right)^{1/3} \mathbf{I}_p, \quad \mathbf{R} = \left(\frac{\nu}{\mu}\right)^{1/3} \mathbf{I}_n$$

for example. In fact, this is not the case since the above $\mathbf{P}, \mathbf{Q}, \mathbf{R}$ will rescale all the matrices $\mathbf{U}_r, \mathbf{V}_r, \mathbf{W}_r$ with the same coefficients λ, μ, ν (provided $\lambda\mu\nu = 1$) while the scaling transformations admit different coefficients $\lambda_1, \ldots, \lambda_F$, $\mu_1, \ldots, \mu_F$ and $\nu_1, \ldots, \nu_F$. $\triangleleft$

4. An algorithm for checking equivalence

In this section, we present an algorithm for deciding whether two F-PDs of a matrix multiplication tensor are equivalent. Under mild assumptions on the input F-PDs, the algorithm will either return the permutation, scaling and trace transformations that allow one to connect both F-PDs or conclude that the two F-PDs are not equivalent to each other. The working assumptions were satisfied for 100% of the samples on which we performed numerical experiments (see Section 6.3), motivating the qualifier "mild" assumptions.

We start this section by introducing the concept of clustering number of a matrix. This number can be computed efficiently and is used in the assumption to guarantee proper working of the algorithm.

4.1. The clustering number of a matrix

Let $\mathbf{A}$ be an $m \times n$ matrix. Let $\{U_1, \ldots, U_S\}$ be a family of linearly independent subspaces of $\mathbb{R}^m$ (i.e., $u_1 + \cdots + u_S = 0$ with $u_i \in U_i$ for each $1 \leq i \leq S$ implies $u_i = 0$ for every $1 \leq i \leq S$). If each column of $\mathbf{A}$ belongs to some U_i (in fact, except the case where the column contains only zeros, it may belong to at most one U_i since they are linearly independent), then we say that $\{U_1, \ldots, U_S\}$ is a *cover* of $\mathbf{A}$. The largest integer S^* such that there exists a cover of $\mathbf{A}$ with S^* linearly independent subspaces is called the *clustering number* of $\mathbf{A}$ and is denoted by $\mathrm{cl}_\oplus(A) = S^*$ (the choice of the symbol $\oplus$ comes from the fact that if $\{U_1, \ldots, U_S\}$ is a cover of A with $S = S^*$ then $U_1 \oplus \cdots \oplus U_S = \mathbb{R}^m$).

Example 4.1. Let $U_1 = \mathrm{colspan}(\mathbf{A})$ and suppose that the rank of $\mathbf{A}$ is r. Then it is easy to build a cover $\{U_1, U_1', \ldots, U_{m-r}'\}$ where the U_i''s are one-dimensional subspaces. Hence, we conclude that $\mathrm{cl}_\oplus(\mathbf{A}) \geq m + 1 - \mathrm{rank}(\mathbf{A})$. $\triangleleft$

Suppose that $\mathbf{A}$ has full row-rank, i.e., $\mathrm{rank}(\mathbf{A}) = m$. We give a characterization of the clustering number of $\mathbf{A}$ in terms of the connected components of a graph. Denote by $\mathbf{a}_1, \ldots, \mathbf{a}_n$ the columns of $\mathbf{A}$. Without loss of generality, we may assume that the first m columns of $\mathbf{A}$ span $\mathbb{R}^m$. Let $\mathbf{A}' = [\mathbf{a}_1, \ldots, \mathbf{a}_m]$ and $\mathbf{A}'' = [\mathbf{a}_{m+1}, \ldots, \mathbf{a}_n]$.

Define the undirected graph G as follows. The integers $\{1, \ldots, m\}$ are the nodes of G and for each column $\mathbf{a}_j$ of $\mathbf{A}''$, let $\mathbf{q}_j = (q_{j,1}, \ldots, q_{j,m})^\top$ be its coordinates in the basis defined by $\mathbf{A}'$, i.e., $\mathbf{q}_j = (\mathbf{A}')^{-1}\mathbf{a}_j$. For each $i_1, i_2 \in \{1, \ldots, m\}$, draw an edge between the nodes i_1 and i_2 if and only if there is a column $\mathbf{a}_j$ of $\mathbf{A}''$ such that its coordinate vector has a nonzero component in both $\mathbf{a}_{i_1}$ and $\mathbf{a}_{i_2}$, i.e., if $q_{j,i_1} \neq 0$ and $q_{j,i_2} \neq 0$ (clearly, there might be multiple edges and also loops). Moreover, each edge receives a *label*: this label is simply j, the index of the column of $\mathbf{A}''$ that led to this edge. An example is represented in Fig. 1. This construction allows us to state the following lemma:

Proposition 4.1. *Let $\mathbf{A}$ and G be defined as above. Then the clustering number of $\mathbf{A}$ is equal to the number of connected components of G.*

Proof. Let $\{\mathsf{G}_1, \ldots, \mathsf{G}_T\}$ be the connected components of G. First, we show that $\mathrm{cl}_\oplus(\mathbf{A}) \geq T$. For each $1 \leq t \leq T$, let I_t be the set of all column indices involved in the nodes of G_t. (For example, considering the graph in Fig. 1, we would have $I_1 = \{1, 2\}$ and $I_2 = \{3\}$.) For each t, define the subspace U_t as the subspace spanned by the columns $\mathbf{a}_i$ with $i \in I_t$. By hypothesis on $\mathbf{A}'$ having full rank, the subspaces U_t satisfy $\mathbb{R}^m = U_1 \oplus \cdots \oplus U_T$. We have to show that each column $\mathbf{a}_j$ of $\mathbf{A}''$ belongs to some U_t.

Therefore, we show that the label j appears in the edges of at most one component G_t. Indeed, if j appears in G_{t_1} and G_{t_2}, then $\mathbf{a}_j$ has a nonzero component in at least one node i_1 of G_{t_1} and one node i_2 of G_{t_2}. Hence, there must be an edge between i_1 and i_2 and thus G_{t_1} and G_{t_2} are connected, a contradiction. Thus $\mathrm{cl}_\oplus(\mathbf{A}) \geq T$.

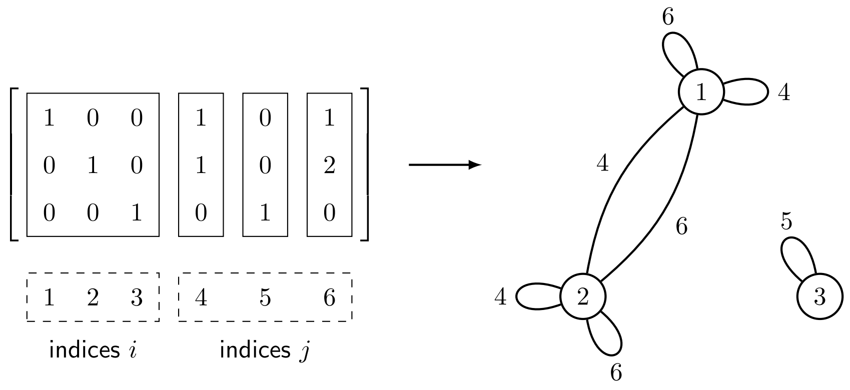

Fig. 1. Example of matrix **A** and the associated graph G.

To show that $\mathrm{cl}_\oplus(\mathbf{A}) \le T$, let $S = \mathrm{cl}_\oplus(\mathbf{A})$ and let $\{U_1, \dots, U_S\}$ be a cover of $\mathbb{R}^m$. For each $1 \le s \le S$, let I_s be the set of the indices of the columns of $\mathbf{A}'$ belonging to U_s. Since $\mathbf{A}'$ has full rank, it is clear that $U_s = \mathrm{span}(\{\mathbf{a}_i\}_{i\in I_s})$. We show that $\{I_1, \dots, I_S\}$ defines connected components of G. Indeed, if there is an edge, say with label j, between nodes $i_1 \in I_{s_1}$ and $i_2 \in I_{s_2}$, then $\mathbf{a}_j$ has nonzero components in $\mathbf{a}_{i_1}$ and in $\mathbf{a}_{i_2}$ and thus it belongs to the subspaces U_{s_1} and U_{s_2}. However, by definition, the subspaces U_s are linearly independent. Hence, we must have $s_1 = s_2$. We conclude that there are at least S connected components in G and thus $T \ge \mathrm{cl}_\oplus(\mathbf{A})$. □

Proposition 4.1 above gives an efficient way to compute the clustering number of a matrix. We describe below another way to efficiently compute the clustering number of a matrix using linear algebra only (and that will be useful in the proof of Lemma 4.5). To do this, let $\mathbf{A}$ be an $m \times n$ matrix and consider the following linear system:

$$\mathbf{M}\mathbf{A} = \mathbf{A}\,\mathrm{diag}(\xi_1, \dots, \xi_n) \tag{3}$$

with variables $\mathbf{M} \in \mathbb{R}^{m\times m}$ and $\xi_{[n]} = (\xi_1, \dots, \xi_n) \in \mathbb{R}^n$. Clearly the system (3) is homogeneous. Let us denote by $\mathscr{S}$ the vector space of solutions $(\mathbf{M}, \xi_{[n]})$ to (3).

Lemma 4.2. *Let $\mathbf{A}$ have full row-rank and no zero columns and let $\mathscr{S}$ be defined as above. Then* $\mathrm{cl}_\oplus(\mathbf{A}) = \dim(\mathscr{S})$.

Proof. First note that if $\mathbf{A} \in \mathbb{R}^{m\times n}$ has full row-rank then $m \le n$. Let $\mathbf{A}'$ and $\mathbf{A}''$ be defined as previously. Without loss of generality, we may again assume that $\mathbf{A}'$ has full rank. Let $\xi_{[n]} = (\xi_1, \dots, \xi_n)$ be fixed. Then we have

$$\mathbf{M} = \mathbf{A}'\,\mathrm{diag}(\xi_1, \dots, \xi_m)\,\mathbf{A}'^{-1}$$

whence $\mathbf{M}$ is completely determined by $\xi_{[n]}$. Reversely, if $\mathbf{M}$ is known, then every $\xi_1, \dots, \xi_n$ are also determined since $\mathbf{A}$ has no zero columns. Hence, we only have to compute the number of degrees of freedom in $\xi_{[n]}$ to compute the dimension of $\mathscr{S}$.

Let G be the graph associated to $\mathbf{A}$. Suppose that i_1 and i_2 are two nodes that are adjacent to each other with an edge labeled by j. Then, we have

$$\mathbf{M}\mathbf{a}_j = \left[\mathbf{A}'\,\mathrm{diag}(\xi_1, \dots, \xi_m)\,\mathbf{A}'^{-1}\right]\mathbf{a}_j = \sum_{i=1}^m \mathbf{a}_i \xi_i q_{j,i} = \mathbf{a}_j \xi_j = \xi_j \sum_{i=1}^m \mathbf{a}_i q_{j,i}.$$

Since $\mathbf{a}_1, \dots, \mathbf{a}_m$ are linearly independent and $q_{j,i_1} \ne 0$ and $q_{j,i_2} \ne 0$, the only solution provides that $\xi_{i_1} = \xi_{i_2} = \xi_j$. We conclude that if two nodes i_1, i_2 belong to the same connected component G_t, then $\xi_{i_1} = \xi_{i_2}$. Hence, using Proposition 4.1, the dimension of $\mathscr{S}$ is lower than or equal to $\mathrm{cl}_\oplus(\mathbf{A})$.

On the other hand, let $S = \mathrm{cl}_\oplus(\mathbf{A})$ and let $\{U_1, \dots, U_S\}$ be a cover of $\mathbf{A}$. Then for each $1 \le s \le S$, let $\mathbf{M}_s$ be the projection on U_s, i.e.,

$$\mathbf{M}_s(\mathbf{x} + \mathbf{y}) = \mathbf{x} \qquad \text{for all } \mathbf{x} \in U_s \text{ and } \mathbf{y} \in \bigoplus_{s'\ne s} U_{s'}.$$

Let $(\eta_1, \dots, \eta_S)$ be a fixed vector and define $\mathbf{M} = \sum_{s=1}^S \eta_s \mathbf{M}_s$. For each $1 \le i \le n$, let $\xi_i = \eta_{s_i}$ where s_i is the unique index $1 \le s_i \le S$ such that $\mathbf{a}_i \in U_{s_i}$. Then $(\mathbf{M}, \xi_{[n]})$ is a solution of (3). Hence, the dimension of $\mathscr{S}$ is at least $\mathrm{cl}_\oplus(\mathbf{A})$. □

We are now able to state and prove the main theorem for this subsection:

Theorem 4.3. *Let **A** be an $m \times n$ matrix with rank r and let Z be the number of zero columns in **A**. Then the clustering number of **A** and the dimension of the solution space $\mathscr{S}$ of (3) satisfy*

$$\dim(\mathscr{S}) = \mathrm{cl}_{\oplus}(\mathbf{A}) + (m-1)(m-r) + Z.$$

Proof. First we suppose that there are no zero columns in **A**. Let $\mathbf{X} \in \mathrm{GL}(m)$ and observe that $\mathrm{cl}_{\oplus}(\mathbf{XA}) = \mathrm{cl}_{\oplus}(\mathbf{A})$. Consider the linear system

$$\mathbf{MXA} = \mathbf{XA}\,\mathrm{diag}(\xi_1, \dots, \xi_n) \tag{4}$$

and note that $(\mathbf{M}, \xi_{[n]})$ is a solution of (4) if and only if $(\mathbf{X}^{-1}\mathbf{MX}, \xi_{[n]})$ is a solution of (3). Hence, taking an appropriate matrix **X**, we may assume without loss of generality that the last $m-r$ rows of **A** are zero.

Let $\tilde{\mathbf{A}}$ be the $r \times n$ matrix consisting of the first r rows of **A**. Then $\tilde{\mathbf{A}}$ has full row-rank and it is easy to check that $\mathrm{cl}_{\oplus}(\mathbf{A}) = \mathrm{cl}_{\oplus}(\tilde{\mathbf{A}}) + m - r$. The matrix **M** may be partitioned into the following blocks:

$$\mathbf{M} = \left[\begin{array}{c|c} \boxed{\mathbf{M}_1} & \\ & \mathbf{M}_3 \\ \boxed{\mathbf{M}_2} & \end{array}\right] \quad \text{with} \quad \left\{\begin{array}{l} \mathbf{M}_1 \in \mathbb{R}^{r\times r} \\ \mathbf{M}_2 \in \mathbb{R}^{(m-r)\times r} \\ \mathbf{M}_3 \in \mathbb{R}^{m\times(m-r)} \end{array}\right. .$$

It is clear that $(\mathbf{M}, \xi_{[n]})$ is a solution of (3) if and only if

$$\mathbf{M}_1\, \mathbf{A}t = \tilde{\mathbf{A}}\,\mathrm{diag}(\xi_1, \dots, \xi_n) \quad \text{and} \quad \mathbf{M}_2\, \mathbf{A}t = 0. \tag{5}$$

From Lemma 4.2 and the fact that $\tilde{\mathbf{A}}$ has full row-rank, the solutions $(\mathbf{M}_1, \xi_{[n]}, \mathbf{M}_2)$ of (5) form a vector space with dimension $\mathrm{cl}_{\oplus}(\tilde{\mathbf{A}})$. On the other hand, there are no constraints on $\mathbf{M}_3 \in \mathbb{R}^{m\times(m-r)}$. This proves the assertion when $Z = 0$.

Finally, observe that appending a zero column to **A** does not change its clustering number and also does not change the space of solutions **M** of (3). The only thing that changes is that the coefficient ξ_{n+1} affected to this zero column might take any value. Hence, the dimension of $\mathscr{S}$ is increased by one. This concludes the proof of the theorem. □

Remark 4.1. For the interested reader, let us mention that the clustering number has an interpretation in terms of matroids: considering **A** as a *linear matroid* with ground set given by the columns of **A** [29, Chapter 39], then we can show that the clustering number of **A** is in fact equal to the number of *connected components* (in the matroid sense [29, Chapter 39]) of the matroid **A**. Dedicated softwares exist to compute the connected components of a matroid. See, e.g., [30]. In fact, in the case of a linear matroid, the implementation in [30] is equivalent to dynamically computing the connected components of the graph G in Proposition 4.1. ◁

4.2. Computation of the scaling and trace transformations

Let $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ be two F-PDs of a matrix multiplication tensor $\Phi_{m,p,n}$. We would like to know whether they are equivalent (see Section 3) and, if they are, to compute the invariance transformations connecting $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ to $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$. At first, we assume that the permutation transformation is given and we focus on the computation of the scaling and trace transformations. (We will see in the next subsection how we can compute this permutation transformation without trying all permutations of $\{1, \dots, F\}$.) Under mild assumptions on $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$, we will see how to do this computation using linear algebra only.

First, we make two important comments. In the sequel, we will always assume that the rank-1 terms[4] $(f_r \otimes g_r)w_r$, $1 \le r \le F$, in the F-PDs (1) are *linearly independent*. Indeed, if one of the rank-1 terms $(f_r \otimes g_r)w_r$ can be decomposed as a linear combination of the other rank-1 terms, then it is easy to build a polyadic decomposition (1) of $\Phi_{m,p,n}$ with $F-1$ terms (which would be extremely lucky and never happened in the numerical experiments we performed; see Section 6).[5]

The second comment is summarized in the following theorem:

Theorem 4.4. *Let $\Phi_{m,p,n}$ be a matrix multiplication tensor and let $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ be an F-PD of $\Phi_{m,p,n}$. Let $I \subseteq \{1, \dots, F\}$ be a subset of indices with $|I| + n \ge F + 1$. Then the family $\{\mathbf{U}_r\}_{r\in I}$ fully spans $\mathbb{R}^{p\times m}$.*

[4] $f_r \otimes g_r$ denotes the *tensor product* of functions f_r and g_r, i.e., $(f_r \otimes g_r)(u, v) = f_r(u)g_r(v)$.

[5] Even in this eventuality, if the decompositions $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ are equivalent and if the rank-1 terms of $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ are linearly dependent, then the rank-1 terms of $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ are also linearly dependent. Moreover, the F_*-term polyadic decompositions $(\mathbf{U}_{[F^*]}, \mathbf{V}_{[F^*]}, \mathbf{W}_{[F^*]})$ and $(\mathbf{U}'_{[F^*]}, \mathbf{V}'_{[F^*]}, \mathbf{W}'_{[F^*]})$, $1 \le F_* < F$, obtained by removing in $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ the linearly dependent rank-1 terms and the *corresponding terms* in $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$, are equivalent and contain both linearly independent rank-1 terms. Hence, modulo a little extra work (due to the non-uniqueness in the choice of linearly dependent terms to remove), we may always reduce to the case where $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ contains only linearly independent rank-1 terms.

Proof. Suppose, on the contrary, that there exists a subset $I \subseteq \{1, \dots, F\}$ with size ℓ such that $\ell + n \geq F + 1$ and $\operatorname{span}(\{\mathbf{U}_r\}_{r\in I}) \neq \mathbb{R}^{p\times m}$. Then there exists $\mathbf{A}_* \in \mathbb{R}^{m\times p} \setminus \{0\}$ such that $\operatorname{trace}(\mathbf{U}_r\mathbf{A}_*) = 0$ for each $r \in I$. Denote by Y the vector space of $p \times n$ matrices $\mathbf{B}$ such that $\operatorname{trace}(\mathbf{V}_r\mathbf{B}) = 0$ for each $r \in F \setminus I$. Since $|F \setminus I| = F - \ell \leq n - 1$, we have that $\dim(Y) \geq pn - n + 1$.

Now define Z as the vector space of all matrices $\mathbf{Z} = (\mathbf{C}\mathbf{A}_*)^\top \in \mathbb{R}^{p\times n}$ for some $\mathbf{C} \in \mathbb{R}^{n\times m}$. Since $\mathbf{A}_* \neq 0$, the dimension of Z is at least n. Now let $\mathbf{B} \in Y$ and $\mathbf{Z} = (\mathbf{C}\mathbf{A}_*)^\top \in Z$ and observe that

$$\begin{aligned}\operatorname{trace}(\mathbf{B}\mathbf{Z}^\top) &= \operatorname{trace}(\mathbf{A}_*\mathbf{B}\mathbf{C}) = \operatorname{trace}\big(\Phi_{m,p,n}(\mathbf{A}_*, \mathbf{B})\mathbf{C}\big) \\ &= \sum_{r=1}^{F} \operatorname{trace}(\mathbf{U}_r\mathbf{A}_*)\operatorname{trace}(\mathbf{V}_r\mathbf{B})\operatorname{trace}(\mathbf{W}_r\mathbf{C}).\end{aligned}$$

From the definitions of $\mathbf{A}_*$ and Y, we conclude that $\operatorname{trace}(\mathbf{B}\mathbf{Z}^\top) = 0$. Thus $Y \cap Z = \{0\}$, so $\dim(Y) + \dim(Z) = pn - n + 1 + n > pn$. This contradicts $Y, Z \subseteq \mathbb{R}^{p\times n}$. $\square$

Remark 4.2. Note that Theorem 4.4 applies, *mutatis mutandis*, to $\mathbf{V}_{[F]}$ and $\mathbf{W}_{[F]}$. To see this, it suffices to observe that if $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ is an F-PD of $\Phi_{m,p,n}$ then $(\mathbf{W}_{[F]}, \mathbf{U}_{[F]}, \mathbf{V}_{[F]})$ and $(\mathbf{V}_{[F]}, \mathbf{W}_{[F]}, \mathbf{U}_{[F]})$ provide F-PDs of $\Phi_{n,m,p}$ and $\Phi_{p,n,m}$ respectively. $\triangleleft$

Among other conclusions of this theorem, we get that $\operatorname{span}(\{\mathbf{U}_r\}_{1\leq r\leq F}) = \mathbb{R}^{p\times m}$. Indeed, it suffices to apply Theorem 4.4 with $I = \{1, \dots, F\}$. Similar conclusions hold for $\mathbf{V}_{[F]}$ and $\mathbf{W}_{[F]}$.

We now present the algorithm to compute the scaling and trace transformations between $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ or conclude that no such transformations exist. To simplify the notation, it will be useful to consider $\mathbf{U}_r$, $\mathbf{V}_r$ and $\mathbf{W}_r$ as column vectors and gather them into matrices. Therefore, we define

$$\tilde{\mathbf{U}} = [\operatorname{vec}(\mathbf{U}_1), \dots, \operatorname{vec}(\mathbf{U}_F)] \in \mathbb{R}^{pm\times F} \tag{6}$$

where $\operatorname{vec}(\cdot)$ is the vectorization (column stacking) operator. Similarly, we define $\tilde{\mathbf{V}} \in \mathbb{R}^{np\times F}$ and $\tilde{\mathbf{W}} \in \mathbb{R}^{mn\times F}$, and also $\tilde{\mathbf{U}}' \in \mathbb{R}^{pm\times F}$, $\tilde{\mathbf{V}}' \in \mathbb{R}^{np\times F}$ and $\tilde{\mathbf{W}}' \in \mathbb{R}^{mn\times F}$. The algorithm is guaranteed to work if we make the following assumption on $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$:

Assumption 4.1. Let $\tilde{\mathbf{U}}$, $\tilde{\mathbf{V}}$ and $\tilde{\mathbf{W}}$ be defined as above. We assume that either $\tilde{\mathbf{U}}$, $\tilde{\mathbf{V}}$ or $\tilde{\mathbf{W}}$ has clustering number equal to one.

Remark 4.3. It is not difficult to see that if $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ are equivalent, then $\operatorname{cl}_\oplus(\tilde{\mathbf{U}}) = \operatorname{cl}_\oplus(\tilde{\mathbf{U}}')$, $\operatorname{cl}_\oplus(\tilde{\mathbf{V}}) = \operatorname{cl}_\oplus(\tilde{\mathbf{V}}')$ and $\operatorname{cl}_\oplus(\tilde{\mathbf{W}}) = \operatorname{cl}_\oplus(\tilde{\mathbf{W}}')$. Thus the clustering numbers (which can be efficiently computed) already offer us a way to eliminate F-PDs that are not equivalent. Therefrom, Assumption 4.1 can be rephrased (without loss of generality) as follows: either $\operatorname{cl}_\oplus(\tilde{\mathbf{U}}) = \operatorname{cl}_\oplus(\tilde{\mathbf{U}}') = 1$, or $\operatorname{cl}_\oplus(\tilde{\mathbf{V}}) = \operatorname{cl}_\oplus(\tilde{\mathbf{V}}') = 1$, or $\operatorname{cl}_\oplus(\tilde{\mathbf{W}}) = \operatorname{cl}_\oplus(\tilde{\mathbf{W}}') = 1$. $\triangleleft$

We will see in Section 6.3 that Assumption 4.1 is satisfied for 100% of the randomly computed samples on which we have performed numerical experiments. The goal of the algorithm presented in this subsection is to compute matrices $\mathbf{P} \in GL(m)$, $\mathbf{Q} \in GL(p)$ and $\mathbf{R} \in GL(n)$, and scaling coefficients $\lambda_1, \dots, \lambda_F$, $\mu_1, \dots, \mu_F$ and $\nu_1, \dots, \nu_F$ such that $\lambda_r\mu_r\nu_r = 1$ and

$$\lambda_r\mathbf{U}'_r = \mathbf{Q}^{-1}\mathbf{U}_r\mathbf{P}, \;\; \mu_r\mathbf{V}'_r = \mathbf{R}^{-1}\mathbf{V}_r\mathbf{Q}, \;\; \nu_r\mathbf{W}'_r = \mathbf{P}^{-1}\mathbf{W}_r\mathbf{R} \tag{7}$$

for every $1 \leq r \leq F$. The above conditions are nonlinear in $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{R}$, λ_r, μ_r and ν_r. However, relying on the assumptions on $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$, these conditions can be reduced to linear matrix equations.

First of all, we show that the requirement $\lambda_r\mu_r\nu_r = 1$ can be dropped. Indeed, suppose that $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ satisfy (7), and let f'_r, g'_r and w'_r be given by (2) with $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$. Also let Φ'' be given by (1) and (2) with $\mathbf{U}''_r = \mathbf{Q}^{-1}\mathbf{U}_r\mathbf{P}$, $\mathbf{V}''_r = \mathbf{R}^{-1}\mathbf{V}_r\mathbf{Q}$, $\mathbf{W}''_r = \mathbf{P}^{-1}\mathbf{W}_r\mathbf{R}$. Then $\Phi'' = \Phi_{m,p,n}$ (trace transformations) and

$$\Phi''(u, v) = \sum_{r=1}^{F} \lambda_r\mu_r\nu_r f'_r(u)g'_r(v)w'_r.$$

From the linear independence assumption on the rank-1 terms $(f'_r \otimes g'_r)w'_r$, $1 \leq r \leq F$, we conclude that $\lambda_r\mu_r\nu_r = 1$ is trivially satisfied if (7) holds.

According to Assumption 4.1, we assume for the rest of this subsection that $\operatorname{cl}_\oplus(\tilde{\mathbf{U}}) = \operatorname{cl}_\oplus(\tilde{\mathbf{U}}') = 1$. We denote by $\mathbf{A}\otimes\mathbf{B}$ the *Kronecker product* of two matrices $\mathbf{A}$ and $\mathbf{B}$, and we will use the following property of the vectorization operator:

$$\operatorname{vec}(\mathbf{A}\mathbf{X}\mathbf{B}) = (\mathbf{B}^\top \otimes \mathbf{A})\operatorname{vec}(\mathbf{X}).$$

Then the first equation of (7) is equivalent to

$$(\mathbf{P}^\top \otimes \mathbf{Q}^{-1})\tilde{\mathbf{U}} = \tilde{\mathbf{U}}'\operatorname{diag}(\lambda_1, \dots, \lambda_F). \tag{8}$$

Considering $\mathbf{P}^\top \otimes \mathbf{Q}^{-1}$ as a single matrix $\mathbf{M} \in \mathbb{R}^{pm\times pm}$, (8) becomes

$$\mathbf{M}\tilde{\mathbf{U}} = \tilde{\mathbf{U}}' \operatorname{diag}(\lambda_1, \ldots, \lambda_F) \tag{9}$$

which is linear in $\mathbf{M}$ and $\lambda_{[F]} = (\lambda_1, \ldots, \lambda_F)$. The fact that no unwanted solutions are created by this linearization is shown in the following developments.

Let $\mathbf{A}$ and $\mathbf{A}'$ be two $m \times n$ matrices with full row-rank, containing no zero columns and with $\mathrm{cl}_\oplus(\mathbf{A}) = \mathrm{cl}_\oplus(\mathbf{A}') = 1$. Then consider the linear system

$$\mathbf{M}\mathbf{A} = \mathbf{A}' \operatorname{diag}(\xi_1, \ldots, \xi_n) \tag{10}$$

with variables $\mathbf{M} \in \mathbb{R}^{m\times m}$ and $\xi_{[n]} = (\xi_1, \ldots, \xi_n) \in \mathbb{R}^n$. This problem is close to problem (3) except that we allow $\mathbf{A} \neq \mathbf{A}'$. Let $\mathscr{S}$ be the vector space of $(\mathbf{M}, \xi_{[n]})$ that are solutions of (10).

Lemma 4.5. *Let $\mathscr{S}$ be defined as above. If $\mathscr{S}$ contains a solution $(\mathbf{M}, \xi_{[n]})$ such that $\xi_i \neq 0$ for every $1 \le i \le n$, then* $\dim(\mathscr{S}) = 1$.

Proof. Let $(\mathbf{M}, \xi_{[n]})$ be a solution of (10) with $\xi_i \neq 0$ for every $1 \le i \le n$. We have assumed that $\mathbf{A}'$ has full row-rank and thus $\mathbf{A}' \operatorname{diag}(\xi_1, \ldots, \xi_n)$ has full row-rank as well. Hence, $\mathbf{M}$ must be invertible. In a similar way as in the proof of Lemma 4.2, we may assume without loss of generality that the first m columns of $\mathbf{A}$ span $\mathbb{R}^m$. Hence, the first m columns of $\mathbf{A}'$ span $\mathbb{R}^m$ too. We conclude the proof with a similar reasoning as for the first part of the proof of Lemma 4.2. □

Hence, two cases can happen when solving (8): (i) either the linearized system (9) admits no solutions with $\lambda_r \neq 0$ for every $1 \le r \le F$; in this case, we conclude that the two F-PDs are not (scaling+trace)-equivalent; or (ii) the solution space $\mathscr{S}$ of (9) is one-dimensional and thus taking an arbitrary nonzero $(\mathbf{M}, \lambda_{[F]}) \in \mathscr{S}$, it is easy to check whether $\mathbf{M}$ has the form $\mathbf{M} = \mathbf{P}^\top \otimes \mathbf{Q}^{-1}$ for some $\mathbf{P} \in \mathrm{GL}(m)$ and $\mathbf{Q} \in \mathrm{GL}(p)$. If the latter does not hold, then the two F-PDs are not (scaling+trace)-equivalent. Otherwise, $\mathbf{P}$ and $\mathbf{Q}$ are the *unique* (up to a scalar multiplication) matrices involved in the invariance transformations (7).

Now that we have determined $\mathbf{P}$ and $\mathbf{Q}$, we consider the following linear system:

$$\begin{cases} \mathbf{R}\mathbf{V}'_r = \tilde{\mu}_r \mathbf{V}_r \mathbf{Q} & \text{for all } 1 \le r \le F, \\ \mathbf{W}'_r \mathbf{R} = \tilde{\nu}_r \mathbf{P}\mathbf{W}_r & \text{for all } 1 \le r \le F, \end{cases} \tag{11}$$

where the unknowns are $\mathbf{R} \in \mathbb{R}^{n\times n}$ and $\tilde{\mu}_{[F]}, \tilde{\nu}_{[F]} \in \mathbb{R}^F$ for $1 \le r \le F$. If (11) admits no solutions $(\mathbf{R}, \tilde{\mu}_{[F]}, \tilde{\nu}_{[F]})$ with $\tilde{\mu}_r \neq 0$ and $\tilde{\nu}_r \neq 0$ for every $1 \le r \le F$, then we conclude that $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ are not (scaling+trace)-equivalent. On the other hand, if $\tilde{\mu}_r \neq 0$ and $\tilde{\nu}_r \neq 0$ for every $1 \le r \le F$, then $\mathbf{R}$ is invertible because $\mathrm{span}(\{\mathbf{V}_r\mathbf{Q}\}_{1\le r\le F}) = \mathbb{R}^{n\times p}$ (see Remark 4.2). The 6-tuple $(\mathbf{P}, \mathbf{Q}, \mathbf{R}, \lambda_{[F]}, \mu_{[F]}, \nu_{[F]})$ with $\mu_r = \tilde{\mu}_r^{-1}$ and $\nu_r = \tilde{\nu}_r^{-1}$ then provides a solution to the (scaling+trace)-equivalence problem (7).

4.3. Computation of the permutation transformation

In the previous subsection, we have described a procedure to compute the scaling and trace transformations connecting two F-PDs $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ or conclude that no such transformations exist. The equivalence of $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ can then be decided in finite time by trying every permutation $\sigma \in \mathfrak{S}_F$ and testing the (scaling+trace)-equivalence of $\sigma((\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]}))$[6] and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$. Due to the combinatorial growth of $|\mathfrak{S}_F|$, an exhaustive exploration of $\mathfrak{S}_F$ is generally not feasible in practice. In this section, we explain how to efficiently decide whether the two F-PDs are equivalent without trying all permutations $\sigma \in \mathfrak{S}_F$.

Definition 4.1. Let $\mathbf{A}_{[m]} = (\mathbf{A}_1, \ldots, \mathbf{A}_m)$ and $\mathbf{B}_{[m]} = (\mathbf{B}_1, \ldots, \mathbf{B}_m)$ be two ordered sets of $n \times n$ matrices. We say that $\mathbf{A}_{[m]}$ and $\mathbf{B}_{[m]}$ are *simultaneously similar* if there exists $\mathbf{X} \in \mathrm{GL}(n)$ such that $\mathbf{A}_i = \mathbf{X}^{-1}\mathbf{B}_i\mathbf{X}$ for every $1 \le i \le m$.

Let $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ be two F-PDs of the matrix multiplication tensor $\Phi_{m,p,n}$. For each $1 \le r \le F$, define the matrices $\mathbf{M}_r = \mathbf{W}_r\mathbf{V}_r\mathbf{U}_r$ and $\mathbf{M}'_r = \mathbf{W}'_r\mathbf{V}'_r\mathbf{U}'_r$. If $\sigma((\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]}))$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ are (scaling+trace)-equivalent for some $\sigma \in \mathfrak{S}_F$, then from (7) we have

$$\begin{aligned} \mathbf{M}'_r &= \lambda_r \mu_r \nu_r \mathbf{M}'_r \\ &= (\mathbf{P}^{-1}\mathbf{W}_{\sigma(r)}\mathbf{R})(\mathbf{R}^{-1}\mathbf{V}_{\sigma(r)}\mathbf{Q})(\mathbf{Q}^{-1}\mathbf{U}_{\sigma(r)}\mathbf{P}) = \mathbf{P}^{-1}\mathbf{M}_{\sigma(r)}\mathbf{P}. \end{aligned} \tag{12}$$

In other words, $\sigma(\mathbf{M}_{[F]})$ and $\mathbf{M}'_{[F]}$ are simultaneously similar.

We define a *partial permutation* of $\{1, \ldots, F\}$ as any injective function π from $I \subseteq \{1, \ldots, F\}$ into $\{1, \ldots, F\}$. We say that π *coincides* with the (total) permutation $\sigma \in \mathfrak{S}_F$ if $\pi(r) = \sigma(r)$ for every $r \in I$. If σ is as in (12) and π coincides with σ, then it is clear that

$$\pi\left((\mathbf{M}_r)_{r\in I}\right) \quad \text{and} \quad (\mathbf{M}'_r)_{r\in I} \quad \text{are simultaneously similar.} \tag{13}$$

[6] Where $\sigma((\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})) = (\sigma(\mathbf{U}_{[F]}), \sigma(\mathbf{V}_{[F]}), \sigma(\mathbf{W}_{[F]}))$ and $\sigma(\mathbf{X}_{[F]})$ is the permuted F-uple $(\mathbf{X}_{\sigma(1)}, \ldots, \mathbf{X}_{\sigma(F)})$.

The following notation will be useful for the description of the algorithm for computing σ. For $F' \in \{0, \ldots, F\}$, we denote by $\mathsf{Inj}(F', F)$ the set of injective functions from $\{1, \ldots, F'\}$ into $\{1, \ldots, F\}$. Each function of $\mathsf{Inj}(F', F)$ is seen as a subset of $\{1, \ldots, F'\} \times \{1, \ldots, F\}$. The length of $\pi \in \mathsf{Inj}(F', F)$ is simply $|\pi| = F'$, and the range of $\pi \in \mathsf{Inj}(F', F)$ is defined as $\mathrm{Range}(\pi) = \{\pi(r) : 1 \le r \le F'\}$.

The idea behind the algorithm to compute σ is the following. First, we start from a partial permutation $\pi \in \mathsf{Inj}(F', F)$ with F' small. We check whether π is susceptible to coincide with σ by checking whether (13) is satisfied or not (see also Remark 4.4). If (13) is satisfied, then we try to extend π to a larger partial permutation $\pi^+ = \pi \cup \{(F'+1, \ell)\}$ with $\ell \in \{1, \ldots, F\} \setminus \mathrm{Range}(\pi)$. We check again whether π^+ is susceptible to coincide with σ according to (13). If this is the case, we repeat the process with π^+. Otherwise, we try other extensions $\pi \cup \{(F'+1, \ell')\}$. If all possible extensions $\pi \cup \{(F'+1, \ell)\}$, $\ell \in \{1, \ldots, F\} \setminus \mathrm{Range}(\pi)$, have been tried and none of them coincides with σ, then we restart the process with the restriction $\pi^- = \pi|_{\{1,\ldots,F'-1\}} \in \mathsf{Inj}(F'-1, F)$ and try to extend π^- to $\pi^- \cup \{(F', \ell)\}$ with $\ell \in \{1, \ldots, F\} \setminus \mathrm{Range}(\pi)$.

When we reach a full permutation $\pi \in \mathsf{Inj}(F, F) = \mathfrak{S}_F$, then we can decide whether the permuted decomposition $\pi(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and the decomposition $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ are (scaling+trace)-equivalent using the procedure of the previous subsection. If they are, then we have found the correct permutation transformation between $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$. Otherwise, we continue to search for another permutation π.

When the algorithm terminates, if the two F-PDs are equivalent, the algorithm is guaranteed to give the corresponding scaling, trace and permutation transformations. If they are not equivalent, the algorithm will also detect it because all permutations $\pi \in \mathfrak{S}_F$ will be rejected: either because the partial permutation $\pi|_{\{1,\ldots,F'-1\}}$ has been rejected previously in the algorithm, or because π does not lead to (scaling+trace)-equivalent decompositions. Clearly, the computational savings (compared to trying all permutations) are interesting if most of the "incorrect" permutations π are rejected in a early stage, i.e., $\pi|_{\{1,\ldots,F'-1\}}$ is rejected for $F' \ll F$. The computational aspects are discussed in the paragraphs below.

We have implemented the algorithm as the recursive function described in Algorithm 1. The recursive function must be called with $(\pi, b) = $ `FnRecursive(∅, false)`. If the output b is true, then the two decompositions are equivalent and the permutation transformation is given by π. On the other hand, if b is false, then the two F-PDs are not equivalent.

Algorithm 1: Recursive function to decide whether two F-PDs are equivalent.

```
Data: $\pi \in \bigcup_{n=0}^{F} \mathsf{Inj}(n, F)$ and $b$ is a boolean.
Function FnRecursive($\pi$, $b$)
    if $b$ = true then
        Return ($\pi$, $b$);
    else if $|\pi| = F$ then                                                        /* [$\star$] */
        if $\pi(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ are (scaling+trace)-equivalent then   /* [$\clubsuit$] */
            Return ($\pi$, true);
        else
            Return ($\varnothing$, false);
        end if
    else
        foreach $\ell \in \{1, \ldots, F\} \setminus \mathrm{Range}(\pi)$ do
            Let $\pi^+ = \pi \cup \{(|\pi| + 1, \ell)\}$;
            if (13) holds with $\pi^+$ then
                Let $(\pi', b') = $ FnRecursive($\pi^+$, $b$);
                if $b'$ = true then
                    Return ($\pi'$, true);
                end if
            end if
        end foreach
        Return ($\varnothing$, false);                                               /* [$\spadesuit$] */
    end if
end function
```

Remark 4.4. Checking the simultaneous similarity of $\mathbf{A}_{[m]} = (\mathbf{A}_1, \ldots, \mathbf{A}_m)$ and $\mathbf{B}_{[m]} = (\mathbf{B}_1, \ldots, \mathbf{B}_m)$ can be approached by solving a linear system

$$\mathbf{X}\mathbf{A}_i - \mathbf{B}_i\mathbf{X} = 0 \qquad \text{for all} \quad 1 \le i \le m,$$

with unknown $\mathbf{X} \in \mathbb{R}^{n \times n}$, and check whether there exists a solution $\mathbf{X}$ that is invertible. However, this approach is not efficient and not robust to rounding errors. Therefore, we have used a different approach. Consider scalar coefficients $\alpha_1, \ldots, \alpha_m \in \mathbb{R}$. A necessary condition for $\mathbf{A}_{[m]}$ and $\mathbf{B}_{[m]}$ to be simultaneously similar is that $\sum_{i=1}^{m} \alpha_i \mathbf{A}_i$ and $\sum_{i=1}^{m} \alpha_i \mathbf{B}_i$ have the same eigenvalues counted with multiplicity. By doing this for randomly generated sets of coefficients $\alpha_1, \ldots, \alpha_m \in \mathbb{R}$, this gives a very efficient way to check the simultaneous similarity of $\mathbf{A}_{[m]}$ and $\mathbf{B}_{[m]}$ with high probability. $\triangleleft$

Remark 4.5. Strictly speaking, the use of Algorithm 1 supposes that Assumption 4.1 is satisfied. One could wonder whether we can still obtain some information from Algorithm 1 even if the assumption is not satisfied. The answer is yes. We modify the algorithm as follows. If condition [$\star$] is satisfied, then instead of testing whether the F-PDs are (scaling+trace)-equivalent, we directly output (π, true) and exit the function. With this modified algorithm, if the call of the function $(\pi, b) =$ `FnRecursive`($\varnothing$, false) returns the value $b =$ true, then we cannot say anything about the equivalence of the two F-PDs. However, if $b =$ false, then we are sure that the two F-PDs are not equivalent. $\triangleleft$

Numerical experiments for the algorithm described in this section are presented in Section 6.3. Regarding the complexity of the algorithm, the computation of the scaling and trace transformations relies only on solving linear systems of equations. The system (9) consists of mpF equations with $(mp)^2 + F$ variables. Because $F \geq mp$ (consequence of Theorem 4.4), the complexity of solving (9) is at most $\mathcal{O}([Fmp]^3)$. Similarly, solving (11) requires at most $\mathcal{O}([F(pn+nm)]^3)$. Therefore, the complexity of the (scaling+trace)-equivalence part of the algorithm is bounded by $\mathcal{O}([F \max\{mp, pn, nm\}]^3)$.

The complexity of the permutation computation part is more difficult to evaluate. It is obviously bounded by $F!$. Hence, an upper bound for the global complexity of the algorithm is $\mathcal{O}(F![F \max\{mp, pn, nm\}]^3)$. However, in all the numerical experiments we have performed (see Section 6.3), it appears that Algorithm 1 never reaches step [$\star$] more than once. In fact, all the partial permutations π for which the algorithm reaches step [$\spadesuit$] satisfy $|\pi| \leq 9$ (see Table 2–Depth). In other words, whenever $\pi \in \mathsf{Inj}(F', F)$ could not lead to a correct permutation, then the algorithm detected it rapidly. Hence, in practice (for our numerical experiments), the computational complexity of the complete algorithm is $\mathcal{O}([F \max\{mp, pn, nm\}]^3)$.

5. Characteristic polynomials and discretizable decompositions

Drawing upon the simultaneous similarity property (12) of equivalent decompositions, we introduce a simple necessary criterion for a decomposition of a matrix multiplication tensor to be equivalent to a discrete decomposition.

Definition 5.1. A decomposition $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ is *discretizable* if it is equivalent to a discrete decomposition $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$. [Clearly, it is necessary and sufficient to require that $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ is only (scaling+trace)-equivalent to $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$.]

We refer the reader to Section 2 for the definition and relevance of discrete decompositions in the context of fast matrix multiplication. Numerical algorithms for computing polyadic decompositions of matrix multiplication tensors do not lead in general to solutions of this kind. The possibility to transform a general decomposition into a discrete one using invariance transformation opens the door to a new generation of algorithms to compute discrete solutions relying on a two-step approach (first compute a general decomposition and then discretize it). However, it is not clear when a decomposition can be discretized with invariance transformations so that the two-step approach may be inapplicable in some cases. The aim of this section is *not* to describe algorithms for transforming general decompositions into discrete decompositions but we propose a necessary criterion for a decomposition to be discretizable.

The criterion draws upon the observations made in Section 4.3: if $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ are (scaling+trace)-equivalent, then the families $\mathbf{M}_{[F]}$ and $\mathbf{M}'_{[F]}$, defined by $\mathbf{M}_r = \mathbf{W}_r\mathbf{V}_r\mathbf{U}_r$ and $\mathbf{M}'_r = \mathbf{W}'_r\mathbf{V}'_r\mathbf{U}'_r$, are simultaneously similar (Definition 4.1). In particular, $\sum_{r=1}^F \beta_r\mathbf{M}_r$ and $\sum_{r=1}^F \beta_r\mathbf{M}'_r$ are also similar for every coefficients $\beta_r \in \mathbb{R}$ (cf. Remark 4.4) and thus they have the same characteristic polynomial.

Assume that $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ is a discrete F-PD. Then $\mathbf{U}'_r \in (q\mathbb{Z})^{p\times m}$, $\mathbf{V}'_r \in (q\mathbb{Z})^{n\times p}$ and $\mathbf{W}'_r \in (q\mathbb{Z})^{m\times n}$ for some $q \in \mathbb{R}$. Hence, $\mathbf{M}'_r \in (q^3\mathbb{Z})^{m\times m}$ for every $1 \leq r \leq F$. Let the coefficients β_r in the paragraph above be integers. If we denote the characteristic polynomial of $\frac{1}{q^3}\sum_{r=1}^F \beta_r\mathbf{M}_r$ by

$$\begin{aligned} p(t) &= p(t; \beta_1, \dots, \beta_F) \\ &= \det\left(tI - \frac{1}{q^3}\sum_{r=1}^F \beta_r\mathbf{M}_r\right) = t^m + \alpha_{m-1}t^{m-1} + \cdots + \alpha_0, \end{aligned} \tag{14}$$

it is not hard to see that the coefficients $\alpha_i \in \mathbb{Z}$ for every $0 \leq i < m$.

Definition 5.2. Let the matrices $\mathbf{M}_r$ be defined as above. We say that the decomposition $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ satisfies the *discretizability criterion with parameter* q if for every integer coefficients β_r, $1 \leq r \leq F$, the coefficients of the characteristic polynomial (14) satisfy $\alpha_i \in \mathbb{Z}$ for every $0 \leq i < m$.

From the developments above, it is clear that satisfying the discretizability criterion with some parameter $q \in \mathbb{R}$ is a necessary condition for being discretizable. In the following section, we will see that most of the sample decompositions on which we have performed numerical experiments do not satisfy the discretizability criterion with $q = 1$ or $q = 1/2$ for tensors larger than the 2×2 by 2×2 case, contrasting with the abundance in the literature of discrete decompositions with $q = 1$ or $q = 1/2$ for these tensors (see also Section 1).

Table 1
First and second columns: different cases considered in the numerical experiments. Fourth column: total time required to compute the $N_s = 10\,000$ decompositions with Tichavský et al.'s method [26]. Third column: number of randomly generated initial guesses (trials) $(\mathbf{U}_{[F],0}, \mathbf{V}_{[F],0}, \mathbf{W}_{[F],0})$ we had to use to compute the $N_s = 10\,000$ samples.

(m, p, n)	F	# trials	Elapsed time [h]
(1, 2, 1)	2	10 000	0.017
(2, 1, 2)	4	10 000	0.05
(2, 2, 2)	7	10 762	0.32
(2, 3, 2)	11	10 133	0.51
(3, 2, 3)	15	18 003	40
(3, 3, 3)	23	15 829	16.8

6. Numerical experiments

We have applied the results of Sections 4 and 5 on large sample sets of decompositions for matrix multiplication tensors up to the $m = p = n = 3$ case. The goal is to get for the first time a view on the distributions of essentially unique decompositions and the distributions of discretizable decompositions: how many essentially unique decompositions do there exist? If two different decompositions are computed with a numerical algorithm, are they likely to be equivalent? Likely to be discretizable for some given q?

The way to obtain these samples is described in the next subsection. The reason we restrict to cases smaller than or equal to the $m = p = n = 3$ case is explained in the next subsection as well. All computations were performed in Matlab. The computation-intensive part, namely, the generation of the samples, was executed on a Linux machine with 28 cores and 128 GBytes of RAM. The other computations were done on a laptop having 4 cores and 16 GBytes of RAM running Linux.

6.1. Computing polyadic decompositions

In the numerical experiments, we considered the six different cases $(m, p, n; F)$ summarized in Table 1 (first two columns), where (m, p, n) is the size of the matrix multiplication tensor and F is the number of rank-1 terms, i.e., we considered F-PDs of $\Phi_{m,p,n}$. For each (m, p, n), the associated F is the smallest F for which we know there exists in the literature a decomposition of $\Phi_{m,p,n}$ with F terms (see, e.g., [13,25]).[7] For each case, we want to obtain large sets of decompositions on which to apply the results of Sections 4 and 5.

Computing polyadic decompositions of matrix multiplication tensors is notoriously difficult (see, e.g., [25,26] and references therein). Quite a few papers in the literature about tensor decompositions are devoted to this specific problem. For the numerical experiments of this paper, we have used the method proposed by Tichavský et al. [26] to compute $N_s = 10\,000$ samples (decompositions) for the six cases listed in Table 1. For an alternative method, we refer the reader to [25]. See also [32]. We have used $\mathbf{U}_{r,0} \in \mathbb{R}^{p\times m}$, $\mathbf{V}_{r,0} \in \mathbb{R}^{n\times p}$ and $\mathbf{W}_{r,0} \in \mathbb{R}^{m\times n}$ with entries chosen uniformly at random in $[-1, 1]$ as initial iterates for Tichavský et al.'s method. The method does not always converge to a global minimum; hence we sometimes had to try more than one initial iterate to converge to an exact solution (the third and fourth columns give an idea of the effort required to compute the N_s decompositions). In the end, we have at our disposal for each case N_s samples of F-term polyadic decompositions of $\Phi_{m,p,n}$. We denote them by $(\mathbf{U}^{\kappa}{}_{[F]}, \mathbf{V}^{\kappa}{}_{[F]}, \mathbf{W}^{\kappa}{}_{[F]})$ with $\kappa \in \{1, \ldots, N_s\}$.

In the numerical computations, the tensors $\Phi \in \mathrm{Bil}(\mathbb{R}^{m\times p}, \mathbb{R}^{p\times n}; \mathbb{R}^{m\times n})$ are represented by the three-dimensional arrays $\tilde{\Phi} \in \mathbb{R}^{mp\times pn\times nm}$ obtained from the canonical identifications $\mathbb{R}^{m\times p} \cong \mathbb{R}^{mp}$, etc. Regarding floating-point arithmetic limitations, a sample $(\mathbf{U}^{\kappa}{}_{[F]}, \mathbf{V}^{\kappa}{}_{[F]}, \mathbf{W}^{\kappa}{}_{[F]})$ is considered as an F-PD of $\Phi_{m,p,n}$ if

$$|\tilde{\Phi}^{\kappa}(i, j, k) - \tilde{\Phi}_{m,p,n}(i, j, k)| < 10^{-9} \qquad \forall i, j, k$$

where Φ^{κ} is the tensor defined by (1) and (2) with $(\mathbf{U}^{\kappa}{}_{[F]}, \mathbf{V}^{\kappa}{}_{[F]}, \mathbf{W}^{\kappa}{}_{[F]})$.

Remark 6.1. For matrix multiplication tensors larger than the (3, 3, 3) case, it becomes very difficult to compute polyadic decompositions of these tensors: the global convergence of the algorithm decreases significantly while the cost for a single iteration of Tichavský et al.'s method grows as $\mathcal{O}([F(mp + pn + nm)]^3)$. It becomes thus unrealistic to compute large sets of decompositions for these tensors. $\triangleleft$

[7] Note that for the first four cases in Table 1, F is equal to the rank of the associated tensor and thus cannot be decreased; see, e.g., [2, Chapter 15] for the (1, 2, 1) and (2, 1, 2) cases; for (2, 2, 2), see [3, Theorem 11]; and for (2, 3, 2), see [31]. For the (3, 2, 3) case, the best known lower bound on the rank of $\Phi_{3,2,3}$ is $\mathrm{rank}(\Phi_{3,2,3}) \geq 14$ (see, e.g., [3, Theorem 11]), and for (3, 3, 3) the best known lower bound is $\mathrm{rank}(\Phi_{3,3,3}) \geq 19$, shown by Bläser [8]. However, no F-term polyadic decompositions of $\Phi_{3,2,3}$ and $\Phi_{3,3,3}$ with $F < 15$ and $F < 23$ respectively are known for the moment.

6.2. Discretizable decompositions

We start with the analysis of the discretizability property of the sample decompositions $(\mathbf{U}^{\kappa}{}_{[F]}, \mathbf{V}^{\kappa}{}_{[F]}, \mathbf{W}^{\kappa}{}_{[F]})$. For instance, we would like to find the decompositions that are not equivalent to a discrete decomposition with $q = 1/2$ (Definition 2.1). To do this, we will apply the necessary criterion for discretizability with parameter $1/2$.

For every $\kappa \in \{1, \dots, N_s\}$, let $p(t) = p(t; \beta_1, \dots, \beta_F)$ be as in (14) where the β_r's are randomly chosen integer coefficients. In our experiments, we use 16 sets of coefficients sampled uniformly at random in $\{-5, -4, \dots, 5\}^F$, providing thus 16 polynomials

$$p_j^{\kappa}(t) = t^m + \alpha^{\kappa}_{j,m-1} t^{m-1} + \cdots + \alpha^{\kappa}_{j,0}, \qquad j = 1, \dots, 16.$$

For each $\kappa \in \{1, \dots, N_s\}$, we let

$$\mathrm{ND}_{\kappa} = \max_{1 \le j \le 16} \max_{0 \le i < m} |\alpha^{\kappa}_{j,i} - \mathrm{round}(\alpha^{\kappa}_{j,i})|$$

where $\mathrm{round}(\alpha)$ is the closest integer to α. The value of ND_{κ} is thus a measure of how close are the polynomials $p_j^{\kappa}(t)$ to polynomials with integer coefficients. From the results of Section 5, a decomposition $(\mathbf{U}^{\kappa}{}_{[F]}, \mathbf{V}^{\kappa}{}_{[F]}, \mathbf{W}^{\kappa}{}_{[F]})$ for which ND_{κ} is (significantly) nonzero is not equivalent to a discrete decomposition with $q = 1/2$.

The histograms in Fig. 2 show the distribution of decompositions based on the value of ND_{κ}. More precisely, each bar of the histograms represents the number of decompositions $(\mathbf{U}^{\kappa}{}_{[F]}, \mathbf{V}^{\kappa}{}_{[F]}, \mathbf{W}^{\kappa}{}_{[F]})$ with ND_{κ} in the corresponding range. For the $(1, 2, 1)$, and $(2, 2, 2)$ cases, we observe that, for all the decompositions, the polynomials $p_j^{\kappa}(t)$ have *integer* coefficients (within a very small tolerance). Hence, 100% of the decompositions satisfy the discretizability criterion with $q = 1/2$. This is not surprising since all the 2-PDs (resp. 7-PDs) of $\Phi_{1,2,1}$ (resp. $\Phi_{2,2,2}$) are equivalent (see [4] and Remark 6.2), and $\Phi_{1,2,1}$ (resp. $\Phi_{2,2,2}$) admits a discrete decomposition with $q = 1$.[8]

In contrast, for the $(2, 1, 2)$, $(2, 3, 2)$, $(3, 2, 3)$ and $(3, 3, 3)$ cases, we observe that most of the decompositions do not satisfy the necessary criterion for discretizability with $q = 1/2$. This implies that most of the decompositions are not equivalent to a discrete decompositions with parameter $q = 1/2$. This last observation has to be put in contrast with the abundance of decompositions $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ for which the entries of $\mathbf{U}_r$, $\mathbf{V}_r$ and $\mathbf{W}_r$ belong to $\{0, \pm 1/2, \pm 1\}$ in the literature [5,13,24,26].

Further experiments can be conducted to investigate the discretizability of the decompositions with respect to other parameters q. However, due to space limitations, we do not present them in this paper.

6.3. Equivalence classes of decompositions

In the previous subsection, we have seen that, except for the $(1, 2, 1)$ and $(2, 2, 2)$ cases, most of the decompositions are not equivalent to a discrete decomposition with coefficients in $\{0, \pm 1/2, \pm 1\}$. In this section, we will analyze the pairwise equivalence of the decompositions. This will reveal the distributions of the equivalence classes among the sample sets of decompositions. Therefore, we use the algorithm developed in Section 4.

First, in order to apply Algorithm 1, we need to ensure that Assumption 4.1 is satisfied for every decomposition $(\mathbf{U}^{\kappa}{}_{[F]}, \mathbf{V}^{\kappa}{}_{[F]}, \mathbf{W}^{\kappa}{}_{[F]})$, $\kappa \in \{1, \dots, N_s\}$. Therefore, for each decomposition $(\mathbf{U}^{\kappa}{}_{[F]}, \mathbf{V}^{\kappa}{}_{[F]}, \mathbf{W}^{\kappa}{}_{[F]})$, we have computed (using Theorem 4.3) the *clustering vector* of the decomposition defined as the vector $[\mathrm{cl}_{\oplus}(\tilde{\mathbf{U}}^{\kappa}), \mathrm{cl}_{\oplus}(\tilde{\mathbf{V}}^{\kappa}), \mathrm{cl}_{\oplus}(\tilde{\mathbf{W}}^{\kappa})]$. The results are summarized in Fig. 3. As we can see, for each case, 100% of the decompositions have at least one matrix $\tilde{\mathbf{U}}^{\kappa}$, $\tilde{\mathbf{V}}^{\kappa}$ or $\tilde{\mathbf{W}}^{\kappa}$ with clustering number equal to one and thus satisfy Assumption 4.1.

We can thus apply Algorithm 1 to check the equivalence between pairs of decompositions for the different cases. The results are gathered in Table 2. We observe that for the $(1, 2, 1)$ and $(2, 2, 2)$ cases, every decompositions are pairwise equivalent (see also Remark 6.2). As a consequence, it is not surprising that all the decompositions are discretizable. This situation is more surprising for $(2, 1, 2)$, $(2, 3, 2)$, $(3, 2, 3)$ and $(3, 3, 3)$ cases. For these cases, the decompositions seem to be pairwise equivalent with probability zero.

The third column of Table 2 gives the average computation time to check the equivalence between two decompositions. We observe that the algorithm takes no more than 30 ms. In comparison, for the $(3, 3, 3)$ case for example, the naive method (testing all possible permutations) would have required to test condition [♣] in Algorithm 1, which has a complexity of $\mathcal{O}([F \max\{mp, pn, nm\}]^3)$, $23! = 2.59 \cdot 10^{22}$ times.

Remark 6.2. It can be shown that all the decompositions of $\Phi_{1,2,1}$ and $\Phi_{2,2,2}$ (respectively) are pairwise equivalent, which corroborates the results of the numerical experiments using Algorithm 1. For the $(2, 2, 2)$ case, we refer the reader

[8] For $\Phi_{1,2,1}$, take, e.g.,

$$\mathbf{U}_1 = [1, 0], \quad \mathbf{V}_1 = [1, 0]^{\top}, \quad \mathbf{W}_1 = 1,$$
$$\mathbf{U}_2 = [0, 1], \quad \mathbf{V}_2 = [0, 1]^{\top}, \quad \mathbf{W}_2 = 1,$$

(see also Remark 6.2). For $\Phi_{2,2,2}$, take, e.g., Strassen's algorithm.

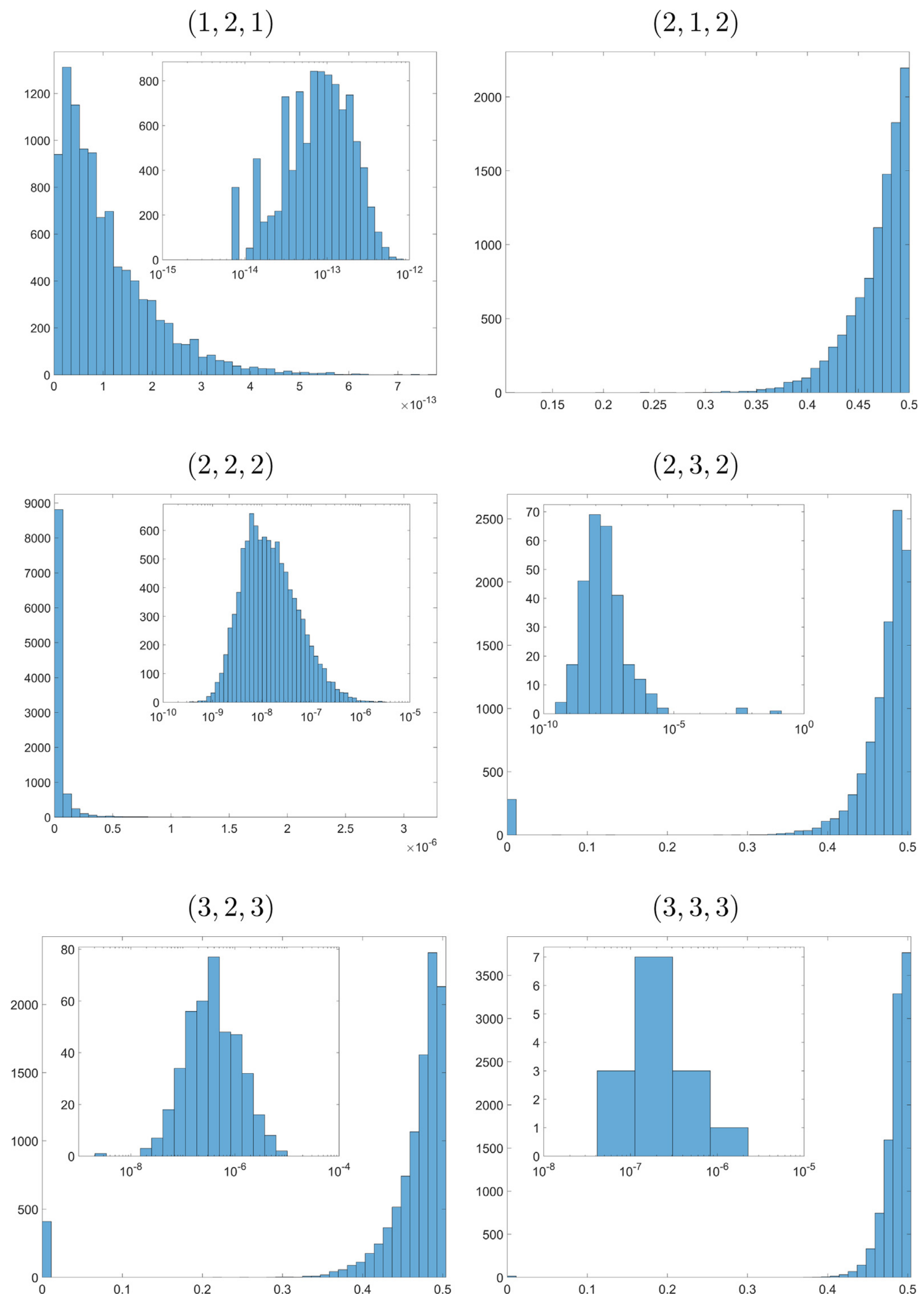

Fig. 2. Distribution of decompositions based on how close the polynomials $p_j^\kappa(t)$ are to characteristic polynomials with integer coefficients. Horizontal axis: ND_κ. Vertical axis: # decompositions $(\mathbf{U}^\kappa{}_{[F]}, \mathbf{V}^\kappa{}_{[F]}, \mathbf{W}^\kappa{}_{[F]})$ with ND_κ in the corresponding range. Remember that the total number of decompositions is equal to $N_s = 10\,000$. The insets provide a zoom on the decompositions $(\mathbf{U}^\kappa{}_{[F]}, \mathbf{V}^\kappa{}_{[F]}, \mathbf{W}^\kappa{}_{[F]})$ with $\mathrm{ND}_\kappa < 0.1$. Note the logarithmic scale of the horizontal axis in the inset.

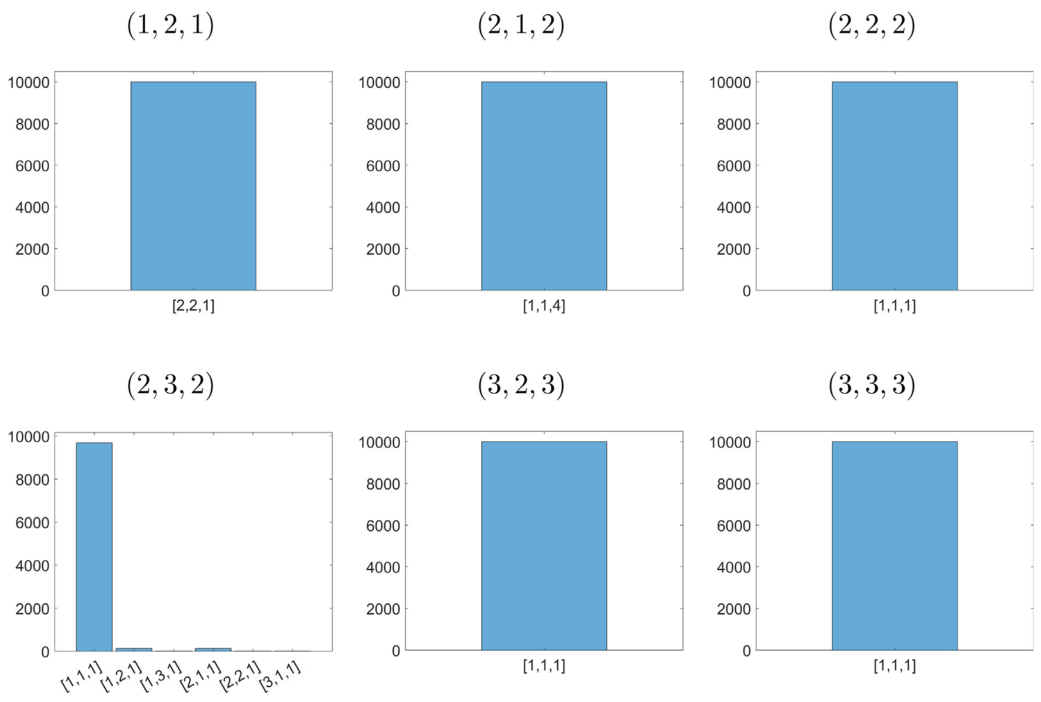

Fig. 3. Clustering vectors $[\mathrm{cl}_\oplus(\tilde{\mathbf{U}}^\kappa), \mathrm{cl}_\oplus(\tilde{\mathbf{V}}^\kappa), \mathrm{cl}_\oplus(\tilde{\mathbf{W}}^\kappa)]$ of the decompositions.

Table 2
Equivalence of decompositions. For each case, we have used Algorithm 1 to check the equivalence between 10 000 randomly chosen pairs of decompositions inside the cluster. The second column gives the percentage of pairs of equivalent decompositions. The third column gives the average time required to check the equivalence of the decompositions with Algorithm 1.

	Percentage of equivalent pairs	Mean elapsed time [s]	Depth[a]	
			Max.	Mean
$(1, 2, 1)$	100%	$4.74 \cdot 10^{-4}$	1	1
$(2, 1, 2)$	0%	$1.02 \cdot 10^{-3}$	1	1
$(2, 2, 2)$	100%	$2.68 \cdot 10^{-3}$	4	0.55
$(2, 3, 2)$	0%	$4.94 \cdot 10^{-3}$	6	1.08
$(3, 2, 3)$	0%	$2.66 \cdot 10^{-2}$	9	2.91
$(3, 3, 3)$	0%	$2.82 \cdot 10^{-2}$	5	1.51

[a]The depth of the algorithm is the maximal length of a partial permutation $\pi \in \mathsf{Inj}(n, F)$ that is rejected (see Algorithm 1).

to [4]. For the $(1, 2, 1)$ case, let $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ and $(\mathbf{U}'_{[F]}, \mathbf{V}'_{[F]}, \mathbf{W}'_{[F]})$ be two 2-PDs of $\Phi_{1,2,1}$. Observe that $\Phi_{1,2,1}$ maps 2-dimensional vectors to their scalar product and thus can be represented with the identity matrix:

$$\Phi_{1,2,1}(u, v) = u^\top \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} v.$$

Since $(\mathbf{U}_{[F]}, \mathbf{V}_{[F]}, \mathbf{W}_{[F]})$ is a decomposition, it is not hard to see that

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \tilde{\mathbf{U}}\,\mathrm{diag}(\mathbf{W}_1, \mathbf{W}_2)\,\tilde{\mathbf{V}}^\top$$

where we remind that $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$ are defined as (6). Using a scaling transformation, we may assume that $\mathbf{W}_1 = \mathbf{W}_2 = 1$. Hence $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}^\top$ are inverses of each other, and so are $\tilde{\mathbf{U}}'$ and $\tilde{\mathbf{V}}'^\top$. Then let $\mathbf{P} = \tilde{\mathbf{U}}\tilde{\mathbf{V}}'^\top$ and observe that

$$\begin{aligned} \mathbf{P}^{-1}\tilde{\mathbf{U}} &= \left[\tilde{\mathbf{U}}'\tilde{\mathbf{U}}^{-1}\right]\tilde{\mathbf{U}} = \tilde{\mathbf{U}}', \\ \mathbf{P}^\top\tilde{\mathbf{V}} &= \left[\tilde{\mathbf{V}}'\tilde{\mathbf{V}}^{-1}\right]\tilde{\mathbf{V}} = \tilde{\mathbf{V}}'. \end{aligned}$$

Hence, we have found a trace transformation with $\mathbf{Q} = \mathbf{R} = 1 \in \mathrm{GL}(1)$ between the two decompositions. $\triangleleft$

7. Conclusions

In this paper, we have described an algorithm for efficiently deciding whether two decompositions of a given matrix multiplication tensor are equivalent through invariance transformations. We have introduced the notion of clustering number of a matrix and we have demonstrated the correctness of the algorithm provided some conditions on the clustering number of the factor matrices of the decompositions are satisfied. This condition was satisfied for 100% of the numerical samples on which we have applied our algorithm.

The analysis of the equivalence classes of decompositions is relevant in the context of fast matrix multiplication as it sheds light on the diversity of essentially unique fast matrix multiplication algorithms. In the numerical experiments we have performed, it appears that two decompositions are equivalent with probability zero (except for the multiplication of 1×2 by 2×2 matrices and the multiplication of 2×2 matrices for which we can prove the essential uniqueness of their decompositions) indicating that there are many essentially different algorithms for the fast multiplication of 3×3 matrices for example.

Drawing upon the observation that decompositions with coefficients in a discrete set provide fast matrix multiplication with better performance, we have also provided a necessary criterion for a decomposition to be equivalent to a decomposition with these properties. We have applied the criterion on numerical samples and observed that the majority of the decompositions do not satisfy the criterion for being equivalent to a decomposition with coefficients in, e.g., $\{0, \pm 1\}$ or $\{0, \pm 1/2, \pm 1\}$.

Acknowledgments

The authors would like to thank Samuel Fiorini and Gwenaël Joret for insightful discussions on the link between the clustering number and matroids.

References

[1] V. Strassen, Gaussian elimination is not optimal, Numer. Math. 13 (4) (1969) 354–356, http://dx.doi.org/10.1007/BF02165411.
[2] P. Bürgisser, M. Clausen, M.A. Shokrollahi, Algebraic Complexity Theory, Vol. 315, Springer Science & Business Media, 2013, http://dx.doi.org/10.1007/978-3-662-03338-8.
[3] R. Brockett, D. Dobkin, On the optimal evaluation of a set of bilinear forms, Linear Algebra Appl. 19 (3) (1978) 207–235, http://dx.doi.org/10.1016/0024-3795(78)90012-5.
[4] H.F. de Groote, On varieties of optimal algorithms for the computation of bilinear mappings II. optimal algorithms for 2×2-matrix multiplication, Theoret. Comput. Sci. 7 (2) (1978) 127–148, http://dx.doi.org/10.1016/0304-3975(78)90045-2.
[5] J.D. Laderman, A noncommutative algorithm for multiplying 3×3 matrices using 23 multiplications, Bull. Amer. Math. Soc. 82 (1) (1976) 126–128.
[6] O.M. Makarov, An algorithm for multiplying 3×3 matrices, USSR Comput. Math. Math. Phys. 26 (1) (1986) 179–180.
[7] A. Sedoglavic, Laderman matrix multiplication algorithm can be constructed using strassen algorithm and related tensor's isotropies, 2017, arXiv preprint arXiv:1703.08298.
[8] M. Bläser, On the complexity of the multiplication of matrices of small formats, J. Complexity 19 (1) (2003) 43–60, http://dx.doi.org/10.1016/S0885-064X(02)00007-9.
[9] O.M. Makarov, A non-commutative algorithm for multiplying 5×5 matrices using one hundred multiplications, USSR Comput. Math. Math. Phys. 27 (1) (1987) 205–207.
[10] A. Sedoglavic, A non-commutative algorithm for multiplying 5×5 matrices using 99 multiplications, 2017, arXiv preprint arXiv:1707.06860.
[11] A. Sedoglavic, A non-commutative algorithm for multiplying (7×7) matrices using 250 multiplications, 2017, Hal-01572046v3.
[12] A. Rosowski, Fast commutative matrix algorithm, 2019, arXiv preprint arXiv:1904.07683.
[13] G. Ballard, A.R. Benson, A. Druinsky, B. Lipshitz, O. Schwartz, Improving the numerical stability of fast matrix multiplication, SIAM J. Matrix Anal. Appl. 37 (4) (2016) 1382–1418, http://dx.doi.org/10.1137/15M1032168.
[14] V.Y. Pan, Strassen's algorithm is not optimal. trilinear technique of aggregating, uniting and canceling for constructing fast algorithms for matrix operations, in: Foundations of Computer Science, 1978., 19th Annual Symposium on, IEEE, 1978, pp. 166–176, http://dx.doi.org/10.1109/SFCS.1978.34.
[15] V.Y. Pan, New fast algorithms for matrix operations, SIAM J. Comput. 9 (2) (1980) 321–342, http://dx.doi.org/10.1137/0209027.
[16] D.A. Bini, M. Capovani, F. Romani, G. Lotti, $O(n^{2.7799})$ complexity for $n \times n$ approximate matrix multiplication, Inform. Process. Lett. 8 (5) (1979) 234–235, http://dx.doi.org/10.1016/0020-0190(79)90113-3.
[17] A. Schönhage, Partial and total matrix multiplication, SIAM J. Comput. 10 (3) (1981) 434–455, http://dx.doi.org/10.1137/0210032.
[18] D. Coppersmith, S. Winograd, On the asymptotic complexity of matrix multiplication, SIAM J. Comput. 11 (3) (1982) 472–492, http://dx.doi.org/10.1109/SFCS.1981.27.
[19] A.J. Stothers, On the Complexity of Matrix Multiplication, The University of Edinburgh, 2010.
[20] J.M. Landsberg, M. Michał, On the geometry of border rank decompositions for matrix multiplication and other tensors with symmetry, SIAM J. Appl. Algebra Geom. 1 (1) (2017) 2–19, http://dx.doi.org/10.1137/16M1067457.
[21] F. Le Gall, Powers of tensors and fast matrix multiplication, in: Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation, ACM, 2014, pp. 296–303, http://dx.doi.org/10.1145/2608628.2608664.
[22] H.F. de Groote, On varieties of optimal algorithms for the computation of bilinear mappings I. The isotropy group of a bilinear mapping, Theoret. Comput. Sci. 7 (1) (1978) 1–24, http://dx.doi.org/10.1016/0304-3975(78)90038-5.
[23] R.W. Johnson, A.M. McLoughlin, Noncommutative bilinear algorithms for 3×3 matrix multiplication, SIAM J. Comput. 15 (2) (1986) 595–603, http://dx.doi.org/10.1137/0215043.
[24] J. Oh, J. Kim, B.-R. Moon, On the inequivalence of bilinear algorithms for 3×3 matrix multiplication, Inform. Process. Lett. 113 (17) (2013) 640–645, http://dx.doi.org/10.1016/j.ipl.2013.05.011.
[25] A.V. Smirnov, The bilinear complexity and practical algorithms for matrix multiplication, Comput. Math. Math. Phys. 53 (12) (2013) 1781–1795, http://dx.doi.org/10.1134/S0965542513120129.

[26] P. Tichavskỳ, A.-H. Phan, A. Cichocki, Numerical CP decomposition of some difficult tensors, J. Comput. Appl. Math. 317 (2017) 362–370, http://dx.doi.org/10.1016/j.cam.2016.12.007.
[27] F.L. Hitchcock, The expression of a tensor or a polyadic as a sum of products, J. Math. Phys. 6 (1–4) (1927) 164–189, http://dx.doi.org/10.1002/sapm192761164.
[28] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, SIAM Rev. 51 (3) (2009) 455–500, http://dx.doi.org/10.1137/07070111X.
[29] A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency, Vol. 24, Springer-Verlag Berlin Heidelberg, 2003.
[30] Collective, The sage notebook leiden university, 2018, https://sage.math.leidenuniv.nl/src/matroids/matroid.pyx, Accessed: 29 November 2018.
[31] V.B. Alekseev, On bilinear complexity of multiplication of $m \times 2$ and 2×2 matrices, Chebyshevskii Sbornik 16 (4) (2015) 11–27.
[32] A.R. Benson, G. Ballard, A framework for practical parallel fast matrix multiplication, in: ACM SIGPLAN Notices, Vol. 50, (8) ACM, 2015, pp. 42–53, http://dx.doi.org/10.1145/2688500.2688513.