THEORY OF COMPUTING, Volume 18 (16), 2022, pp. 1–54
www.theoryofcomputing.org

Tensor Network Complexity of Multilinear Maps

Per Austrin*　　Petteri Kaski†　　Kaie Kubjas‡

Received November 28, 2018; Revised December 20, 2020; Published June 22, 2022

Abstract. We study *tensor networks* as a model of arithmetic computation for evaluating multilinear maps. These capture any algorithm based on low-rank tensor decompositions, such as $O(n^{\omega+\epsilon})$ time matrix multiplication, and in addition many other algorithms such as $O(n \log n)$ time discrete Fourier transform and $O^*(2^n)$ time for computing the permanent of a matrix. However, tensor networks sometimes yield faster algorithms than those that follow from low-rank decompositions. For instance the fastest known $O(n^{(\omega+\epsilon)t})$ time algorithms for counting $3t$-cliques can be implemented with tensor networks, even though the underlying tensor has rank n^{3t} for all $t \geq 2$. For counting homomorphisms of a general pattern graph P into a host graph on n vertices we obtain an upper bound of $O(n^{(\omega+\epsilon)\,\mathrm{bw}(P)/2})$ where $\mathrm{bw}(P)$ is the branchwidth of P. This essentially matches the bound for counting cliques, and yields small improvements over previous algorithms for many choices of P.

An extended abstract of this paper appeared in the Proceedings of the 10th Innovations in Theoretical Computer Science Conference (ITCS 2019) [9].

*Supported by the Swedish Research Council, Grant 621-2012-4546 and the Approximability and Proof Complexity project funded by the Knut and Alice Wallenberg Foundation.

†Supported by the European Research Council, Grant 338077.

‡Supported by the European Union's Horizon 2020 research and innovation programme: Marie Skłodowska-Curie Grant 748354, research carried out at LIDS, MIT and Team PolSys, LIP6, Sorbonne Université.

ACM Classification: F.1.1, F.2.2, G.2.1, G.2.2

AMS Classification: 15A69, 68Q17, 68Q25, 68W05, 05C85, 14Q20

Key words and phrases: arithmetic complexity, lower bound, multilinear map, tensor network

DOI: 10.4086/toc.2022.v018a016

While powerful, the model still has limitations, and we are able to show a number of unconditional lower bounds for various multilinear maps, including the following.

(a) An $\Omega(n^{\mathrm{bw}(P)})$ time lower bound for counting homomorphisms from P to an n-vertex graph, matching the upper bound if $\omega = 2$. In particular for P a v-clique this yields an $\Omega(n^{\lceil 2v/3 \rceil})$ time lower bound for counting v-cliques, and for P a k-uniform v-hyperclique we obtain an $\Omega(n^v)$ time lower bound for $k \geq 3$, ruling out tensor networks as an approach to obtaining non-trivial algorithms for hyperclique counting and the Max-3-CSP problem.

(b) An $\Omega(2^{0.918n})$ time lower bound for the determinant and the permanent of an $n \times n$ matrix.

1 Introduction

One of the cornerstones of the theory of computation is the study of efficient algorithms:

For a function f, how much time is required to evaluate f on given inputs?

Answering this question for almost any specific f is well beyond reach of contemporary tools. For example, it is theoretically possible that canonical NP-complete problems, such as the Circuit Satisfiability problem, can be solved in linear time whereas they are widely believed to require super-polynomial (or somewhat less widely, exponential) time [47, 51, 52]. The main reason for this barrier to quantitative understanding is that it is very hard to prove lower bounds for explicit functions in general models of computation such as circuits or Turing machines.

This situation withstanding, a more modest program to advance our understanding of computation is to study restricted models of computation that for many f of interest are simultaneously

1. general enough to capture the fastest-known algorithms for f, and
2. restricted enough to admit proofs of strong *unconditional* time lower bounds for f.

There is a substantial body of existing work that fits under this program, ranging from the study of low-depth or otherwise restricted circuits (see, e. g., [8, Ch. 14]) to models of algorithm-design principles such as greedy algorithms, backtracking, or dynamic programming [3, 41], to linear or semidefinite programming relaxations for hard combinatorial optimization problems [71].

1.1 Multilinear maps

One class of functions f that are of substantial importance is the family of *ℓ-linear maps* (*multilinear maps*) from ℓ input vector spaces to an output vector space.[1] Examples range from maps of known near-linear-time complexity in the input size, such as the discrete Fourier transform [35, 97], to maps without known polynomial-time-complexity algorithms, such as

[1]Multilinear maps with $\ell = 1$ are called *linear*, $\ell = 2$ *bilinear*, $\ell = 3$ *trilinear*, and so forth.

the permanent of a matrix [89, 95]. Beyond motivation in numerical multilinear algebra and its applications, recent advances in the study of fine-grained algorithm design and complexity have highlighted the fundamental role of algebraic methods in the fastest-known algorithm designs across a broad range of tasks from graph problems, such as all-pairs shortest paths and k-clique, to parsing and constraint satisfaction problems, such as maximum satisfiability and graph coloring [2, 17, 19, 44, 53, 75, 102, 103].

In this paper, we study the *arithmetic complexity* of evaluating a multilinear map, that is, the number of operations (scalar additions, subtractions, and multiplications) needed to evaluate the map. To set up a baseline, a generic ℓ-linear map from ℓ vector spaces of dimension n to a scalar requires $\Omega(n^\ell)$ scalars to represent the map directly using combinations of basis vectors. Given this complexity of a direct explicit representation, it is a fundamental problem to seek less costly representations, along with associated efficient algorithms that work on the chosen representation.

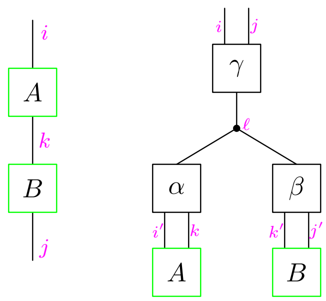
We propose the systematic study of *tensor networks* on hypergraphs as a framework for fast evaluation of multilinear maps, and show a number of upper and lower bounds on the computational power of tensor networks in the spirit of (i) and (ii) above.

1.2 Tensor networks

Tensor networks have a long and rich history which can be traced as far back as 19th-century studies in invariant theory due to Cayley [27, 28], Clebsch [31], Clifford [32], Sylvester [94], and Kempe [57, 58]. Tensor networks are extensively deployed in applications from pure mathematics and theoretical physics [56, 64, 66, 67, 80, 81, 85, 90] to computational physics and chemistry [78, 83, 92]. In theoretical computer science, they appear in various guises, including, for example, in the Holant framework [96, 25, 24], in the study of probabilistic graphical models [62, 87], in the study of parallel programming [91], in the study of quantum computing [7], and in the study of arithmetic complexity [11, 26, 40]. Tensor contraction is also emerging as a basic computational primitive in computer hardware [54, 73]. (See Section 1.5 for a detailed discussion.) As the precise definitions are somewhat technical, let us start with a few simple motivating examples and then state our results, with the understanding that precise definitions appear in Section 4.

In our setting, a tensor network is a hypergraph in which the vertices are tensors and the hyperedges are called *modes*. Each mode that is incident to a tensor defines a "dimension" for indexing the entries of the tensor—for example, a matrix is a tensor that is incident to two modes, one mode for the rows of the matrix, and the other mode for the columns of the matrix. A network may be simplified by a sequence of *contractions*, where each contraction takes a subset of tensors and replaces it with a single tensor whose entries are obtained as generalized inner products of the entries of the tensors being contracted.

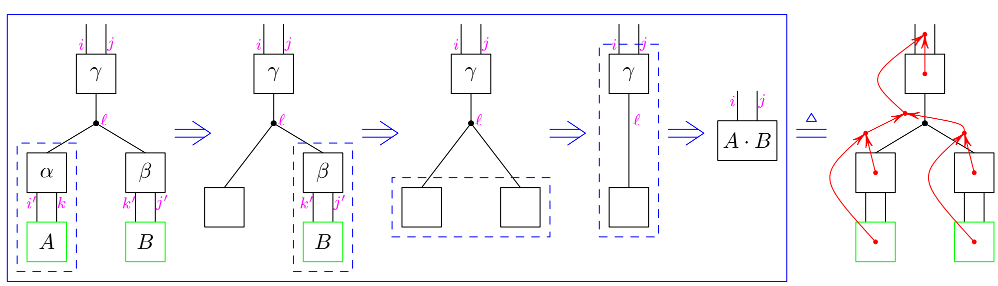
As a concrete first example of these concepts, let us consider the task of multiplying two matrices, A and B. More specifically, let A be a matrix with rows indexed by mode i and columns indexed by mode k, and let B be a matrix with rows indexed by mode k and columns indexed by mode j. We may represent the multiplication task as the tensor network depicted on the left in (1.1). The result of contracting A and B is a new matrix with rows indexed by i and columns indexed by j, where the entry at each position (i, j) is $\sum_k A_{ik}B_{kj}$. If the three index sets all have size n, then computing $A \cdot B$ by contracting them in such a direct manner uses $\Theta(n^3)$ operations. To obtain faster matrix multiplication, we can rewrite the bare-bones network on the left in (1.1) using a low-rank decomposition of the matrix multiplication tensor. For example, Strassen's decomposition [93] of 2×2 matrix multiplication can be represented using the second network in (1.1). Note that the index i used by A and the result has been separated into two distinct indices i and i', and similarly for j and k.



$$\gamma = \underset{\ell}{\begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}_j \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}_j \end{bmatrix}_i}$$

$$\alpha = \underset{\ell}{\begin{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}_k \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & -1 \end{bmatrix}_k \end{bmatrix}_{i'}} \qquad \beta = \underset{\ell}{\begin{bmatrix} \begin{bmatrix} 1 & 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}_{j'} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & 1 \end{bmatrix}_{j'} \end{bmatrix}_{k'}} \tag{1.1}$$

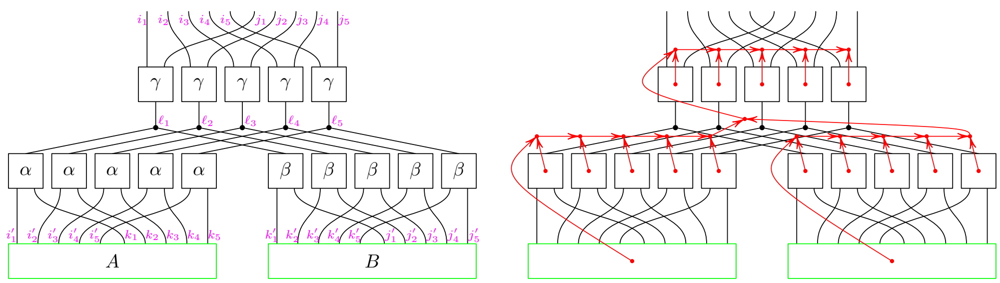
We can *execute* the network by successively contracting groups of vertices. In (1.2) we see the process of successively contracting pairs of tensors in a carefully chosen order, until only a single tensor—the result of the computation—remains. Such an execution can be naturally represented by a rooted binary tree, as shown on the right in (1.2), where the tensors of the network form the leaves, and each internal node represents the result of contracting its two children. To summarize, a tensor-network algorithm is specified by providing (a) a tensor network that when contracted yields the desired result, *and* (b) an execution tree indicating the order in which to contract tensors in the network.



(1.2)

The *cost* of performing one of the contractions in an execution is the product of the lengths of the modes used by any tensor involved in the contraction. This simply measures (up to a constant factor) the number of arithmetic operations (additions/multiplications) used to compute the result by a direct, naïve computation that does not depend on the values of the tensors. For example, the contraction of α and A in the first step of (1.2) has cost 28 because it involves the three modes i' (length 2), k (length 2) and ℓ (length 7).

We observe that cost is data-oblivious—the tensor α is fixed with many zero-entries but these entries still contribute to the cost. Indeed, in many cases there may be faster ways of evaluating a contraction than to evaluate it naively, and just like we saw above, this can often be dealt with by rewriting the network appropriately. For instance, consider now the multiplication of two $2^k \times 2^k$ matrices. Because the family of matrix multiplication tensors is closed under Kronecker products, this operation may be computed by a tensor network like the one shown in (1.3) (depicting the case $k = 5$), where α, β and γ are as in (1.2). The rows/columns of the matrices are now indexed by k-tuples of bits. A cost-efficient execution of this network successively contracts tensors in the order shown to the right in (1.3). In this execution, the first contraction of A with the first α block has a cost of $2^k \cdot 2^k \cdot 7$, and results in a tensor of size $2^{k-1} \times 2^{k-1} \times 7$. Then the contraction with the next α block has a cost of $2^{k-1} \cdot 2^{k-1} \cdot 7^2$ and produces a result of size $2^{k-2} \times 2^{k-2} \times 7 \times 7$, and so on, until the contraction with the last α block which has a cost of $2 \cdot 2 \cdot 7^k = O(7^k)$. The contractions with the β and γ blocks behave similarly. Thus all the $\Theta(k)$ contractions in the execution have cost bounded by $O(7^k)$, meaning that we get a total running time of $O(k7^k) = O(N^{\log_2 7} \log N)$ for $N \times N$ matrices.[2]



(1.3)

This type of argument can capture any algorithm based on a low-rank decomposition of the underlying tensor of the multilinear map, and indeed, this enables $O(n^\omega)$-time[3] matrix multiplication using tensor networks. Beyond simple low-rank decompositions, which always give rise to "star-like" networks as in (1.3), there are many interesting algorithms that can be captured using networks with a more complicated topology. For instance, many maps of substantial importance have a layered structure that decomposes the map to a sequence of elementary maps. A canonical example is the discrete Fourier transform (DFT), which for a smooth composite order such as 2^k, can be decomposed into a fast Fourier transform (FFT) that consists of a sequence of k transforms of order 2 interleaved with diagonal-matrix multiplications of twiddle factors [35, 97].

1.3 Our results

Starting with motivation (i) and seeking to express existing fastest-known algorithms as executions of tensor networks by a sequence of contractions, we show upper bounds for a

[2] In fact, a more careful analysis gives running time $O(N^{\log_2 7})$.

[3] Throughout the paper, we write $\omega(h) = \omega_{\mathbb{F}(h)}$ for the infimum over all t such that the arithmetic complexity of multiplying an $n \times \lfloor n^h \rfloor$ matrix with an $\lfloor n^h \rfloor \times n$ matrix is $O(n^t)$ where $h > 0$ is a constant. While the value of $\omega(h)$ may depend on the underlying field $\mathbb{F}$, we tacitly ignore this, since the field is fixed throughout the paper. Also, we write simply $\omega = \omega(1)$ for the exponent of square matrix multiplication. For all fields it is known that $2 \leq \omega < 2.37286$ [4, 69, 98]; for the state of the art on $\omega(h)$, see [70].

number of natural problems. Beyond standard linear settings such as the FFT, not only do tensor networks realize classical bilinear settings such as Abelian group algebra products and fast matrix multiplication algorithms based on low tensor rank, they are powerful enough to capture a substantial number of higher-linearity applications, including Ryser's algorithm for matrix permanent [89], and the *Kruskal operator* [60, 63] (see Section 3.5), which underlies realization of rank-decompositions for tensor rank [61] and current fastest algorithms for detecting outlier correlations [55].

One problem for which tensor networks turn out to be particularly useful is counting homomorphisms from a fixed pattern graph P to a large host graph G on n vertices. The most well-studied such problem is when P is a k-clique. For this problem, the currently fastest algorithms run in time roughly $O(n^{\omega k/3})$ [75, 44]. For general P, it is known that the problem can be solved in $O(n^{\mathrm{tw}(P)+1})$ time [42], where $\mathrm{tw}(P)$ is the treewidth of P. We show that tensor networks can solve the problem in $O(n^{(\omega+\epsilon)\,\mathrm{bw}(P)/2})$ time, where $\mathrm{bw}(P)$ is the *branchwidth* of P. When P is a k-clique, we have $\mathrm{bw}(P) = \lceil 2k/3 \rceil$; if $\omega = 2$, the running time coincides with that of the currently fastest algorithms. On the other hand, if $\omega > 2$, we can slightly improve upon this to recover the currently fastest known running time of Eisenbrand and Grandoni [44] relying on fast rectangular matrix multiplication. In the case of general P, the bound $O(n^{(\omega+\epsilon)\,\mathrm{bw}(P)/2})$ improves on the treewidth-based bound for graphs with $\mathrm{bw}(P) \le 2(\mathrm{tw}(P) + 1)/\omega$ (and in particular if $\omega = 2$ it is always as fast as the treewidth-based bound, ignoring the ϵ). By recent results of Curticapean, Dell, and Marx [37], fast algorithms for homomorphism-counting can be used to obtain fast algorithms for counting subgraphs of G isomorphic to P, and in some cases our new branchwidth-based bound leads to an improvement; for example, for counting paths of length 7, 8 or 9, we get a running time of $O(n^{3\omega/2+\epsilon}) < O(n^{3.56})$ compared to $O(n^4)$ using the treewidth-based bound, whereas for very long paths it is not clear whether we would need $\omega = 2$ in order for this bound to improve on the treewidth-based bound. Previous work that combines branch decompositions and fast matrix multiplication includes Dorn [43] and Bodlaender *et al.* [21].

Further applications captured by tensor networks are the set covering and set partitioning frameworks via fast zeta and Möbius transforms that underlie the current fastest algorithms for graph coloring [19] and its generalizations such as computing the Tutte polynomial [16, 17]. To summarize, we have the following compendium of upper bound results. For the detailed definitions of the relevant multilinear maps, see Sections 3 and 5.

Theorem 1.1. *We have the following upper bounds on arithmetic complexity via tensor networks.*

1. $O(n^{\omega(h)+\epsilon})$ *for the multiplication of matrices of shape* $n \times \lfloor n^h \rfloor$ *and* $\lfloor n^h \rfloor \times n$ *where* $h > 0$ *is a constant.*

2. $O(n^{(\omega+\epsilon)\,\mathrm{bw}(P)/2})$ *for counting homomorphisms of a fixed pattern hypergraph* P *into a hypergraph on* n *vertices.*

3. $O(n^{\beta+\epsilon})$ *for counting* v*-cliques in an* n*-vertex graph, where* β *is the exponent of the complexity of multiplying matrices of shape* $n^{\lfloor v/3 \rfloor} \times n^{\lfloor (v+1)/3 \rfloor}$ *and* $n^{\lfloor (v+1)/3 \rfloor} \times n^{\lfloor (v+2)/3 \rfloor}$.

4. $O(\max(n^{\lceil \ell/2 \rceil (\omega+\epsilon-1)} r,\ n^{2\lceil \ell/2 \rceil} r^{\omega+\epsilon-2}))$ *for the Kruskal operator of* ℓ *matrices of shape* $n \times r$.

5. *$O(2^k k)$ for the discrete Fourier transforms for the Abelian groups $\mathbb{Z}_{2^k}$ and $\mathbb{Z}_2^k$.*

6. *$O(2^k k)$ for group algebra products on $\mathbb{F}[\mathbb{Z}_{2^k}]$ and $\mathbb{F}[\mathbb{Z}_2^k]$ when 2 is unit in $\mathbb{F}$.*

7. *$O(2^k k)$ for the semigroup algebra product on $\mathbb{F}[(\{0,1\}^k, \subseteq, \cap, \cup)]$.*

8. *$O(2^n n)$ for the permanent of an $n \times n$ matrix.*

Above $\epsilon > 0$ is an arbitrary constant.

Perhaps the most interesting application above is the v-clique problem, which suggests that one should seek to pursue generalizations to v-vertex hypercliques of $\binom{v}{k}$ hyperedges with $v > k \geq 3$. Indeed, subgraph counting is a problem that has received substantial interest over the years (e. g., [53, 75, 6, 5, 44, 18, 20, 99, 100, 46, 45, 77, 59, 37]), but progress in the particular case of v-clique has been stuck to the extent that the problem has attracted notoriety as a hardness assumption in fine-grained complexity [1, 2]. Beyond the study of cliques, hypercliques, and subgraph counting, nontrivial algorithms for such forms would have immediate applicability, for example, in the study of maximum constraint satisfaction problems (Max-CSP) for constraints of width $k \geq 3$; see Williams [102] for the case $k = 2$. One of the main goals of our subsequent lower bounds is to rule out tensor networks as candidates to yield improved algorithms in this setting.

Turning from upper bounds to lower bounds and motivation (ii), tensor networks are restricted enough to enable nontrivial lower bounds for many multilinear maps. To begin with, an immediate limitation of tensor networks is that all the intermediate results during the execution of a network are multilinear, and the execution of a network can be simulated by a multilinear circuit. Raz [84] shows that multilinear formulas cannot compute the determinant of an $n \times n$ matrix in a polynomial number of operations in n, even though polynomial-size general circuits are known for the determinant (see [13, 22, 36, 88]).

It turns out that considerably stronger lower bounds can be shown for tensor networks. In particular, we establish lower bounds for arithmetic complexity via tensor networks of P-homomorphism counting and the Kruskal operator. These lower bounds are tight under the assumption $\omega = 2$. Furthermore, we rule out the possibility of any nontrivial algorithm designs via tensor networks f or counting cliques in hypergraphs. The following theorem collects our main lower-bound results, and should be contrasted with the upper bounds in Theorem 1.1.

Theorem 1.2. *We have the following lower bounds on arithmetic complexity via tensor networks.*

1. *$\Omega(n^{\mathrm{bw}(P)})$ for the multilinear form corresponding to P-homomorphism counting. In particular, this yields a lower bound of $\Omega(n^{\lceil 2v/3 \rceil})$ for counting cliques of size v, and a lower bound of $\Omega(n^v)$ for counting hypercliques of size v.*

2. *$\Omega(\max(n^\ell, n^{\lceil \ell/2 \rceil} r))$ for the Kruskal operator of ℓ matrices of shape $n \times r$.*

3. *$\Omega(\binom{n}{n/3})$ for the determinant and the permanent of an $n \times n$ matrix.*

We remark that [72] independently showed that the rank of the v-hyperclique tensor is $\Omega(n^v)$; our $\Omega(n^v)$ lower bound for tensor networks strengthens that. One may wonder about the gap between the bounds of $\Omega(\binom{n}{n/3})$ and $O(2^n n)$ for the permanent. As we explain below, our lower bound methods are inherently rank-based and cannot go beyond $\binom{n}{n/3}$. A curious point is that it is not immediately clear whether tensor networks can even achieve $O^*(2^n)$ time for the determinant, and we do not know whether this is possible.

1.4 Overview of proof ideas

As a running example in this overview, we consider the 6-linear form $A : \mathbb{F}^{[n]\times[n]} \times \mathbb{F}^{[n]\times[n]} \times \ldots \times \mathbb{F}^{[n]\times[n]} \to \mathbb{F}$ that takes as input 6 matrices of size $n \times n$ and is defined by the equation

$$A(X^{(1)}, X^{(2)}, X^{(3)}, X^{(4)}, X^{(5)}, X^{(6)}) = \sum_{i,j,k,\ell \in [n]} X^{(1)}_{ij} X^{(2)}_{ik} X^{(3)}_{i\ell} X^{(4)}_{jk} X^{(5)}_{j\ell} X^{(6)}_{k\ell} \,. \tag{1.4}$$

If χ is the adjacency matrix of a loopless graph G, then $A(\chi, \chi, \chi, \chi, \chi, \chi)$ counts the 4-cliques in the graph. Associated with A is the 6-tensor $T(A)$ of size $n^2 \times n^2 \times \cdots \times n^2$, where each of the 6 modes is indexed by a pair $(i, j) \in [n] \times [n]$, and the value at a given position is the coefficient of the corresponding term in A. Concretely,

$$T(A)_{i_1 j_1, i_2 k_2, i_3 \ell_3, j_4 k_4, j_5 \ell_5, k_6 \ell_6} = \begin{cases} 1 & \text{if } i_1 = i_2 = i_3 \text{ and } j_1 = j_4 = j_5 \text{ and} \\ & \quad k_2 = k_4 = k_6 \text{ and } \ell_3 = \ell_5 = \ell_6, \\ 0 & \text{otherwise.} \end{cases}$$

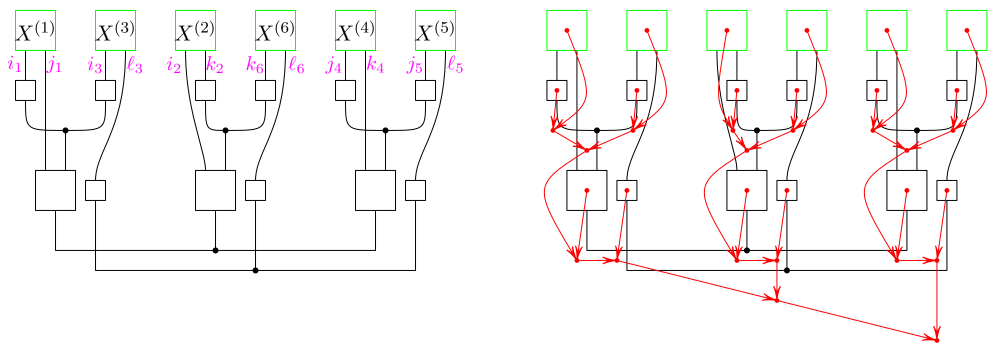
Upper bounds Most, but not all, of the families of multilinear maps we consider are closed under taking Kronecker products. For instance, consider the 4-clique counting form (1.4) for an n-vertex graph and its associated tensor $T(A)$. Then for any $k \geq 1$, the tensor associated with the 4-clique counting form in n^k-vertex graphs is $T(A)^{\otimes k}$, the k-fold Kronecker product of $T(A)$ with itself. We write $A^{\otimes k}$ for the associated map. With this in mind, it is natural to seek general constructions that, given an efficient evaluation of some map A, yields an efficient evaluation of $A^{\otimes k}$.

We give such a construction, and show that the cost of the best tensor network execution for $A^{\otimes k}$ is essentially submultiplicative in a quantity that we call the *amortized cost* of an execution. For tensors of order at most 3, the notion of amortized cost essentially captures the rank of $T(A)$, but for higher-order tensors, the amortized cost may be significantly smaller than the rank. Roughly speaking, the amortized cost of a step in an execution of a map A is: (i) equal to the normal cost if the operation involves the contraction of two tensors that both depend on some input variables of A, but (ii) equal to the size of the result if only one of the tensors involved in the contraction depends on the input variables of A. A precise definition appears in Section 5.1. Our general upper bound for the cost of $A^{\otimes k}$ can, somewhat informally, be stated as follows.

Theorem 1.3 (Submultiplicativity of cost, informal statement). *If a multilinear map A has a tensor network execution consisting of s steps, each with cost at most c and amortized cost at most a, then $A^{\otimes k}$ has a tensor network execution consisting of at most $k \cdot s$ steps, each with cost at most $a^{k-1} \cdot c$.*

An immediate corollary of this is that we can capture any algorithm for $A^{\otimes k}$ based on a low-rank decomposition of $T(A)$ (Corollary 5.2). For example, this implies that tensor networks can multiply $n \times n$ matrices in $O(n^{\omega+\epsilon})$ time (Section 5.2).

However, returning to our running example form (1.4), as we explain below the tensor $T(A)$ has rank n^4, meaning that Corollary 5.2 only yields a trivial upper bound. This is where the full generality of Theorem 1.3 comes in. Consider the form (1.4) for graphs on some constant number n_0 of vertices. As it turns out, we can design a network and an associated execution for this form, depicted in (1.5), with an execution of cost n_0^{2e+3} and amortized cost n_0^{e+1}, where n_0^e is the rank of the tensor associated with $n_0 \times n_0$ matrix multiplication. Picking n_0 to be a large enough *constant* so that e is approximately ω, and letting k be such that n is approximately n_0^k, we obtain via Theorem 1.3 an $O(n^{\omega+\epsilon+1})$ time upper bound for (1.4).[4]



(1.5)

Lower bounds Just like many other arithmetic complexity lower bounds, our lower bounds boil down to establishing lower bounds on the rank of certain matrices.

In order to establish a lower bound on the *rank* of $T(A)$, we *flatten* it to a matrix and analyze the rank of that matrix. There are 2^5 possible bipartitions of the 6 modes of $T(A)$ into two non-empty subsets, and the lower bound on the rank of $T(A)$ that we obtain is the maximum of the ranks of the resulting matrices. Using this method, it is easy to establish that for our example from (1.4), the rank of $T(A) = n^4$. That this is an upper bound follows from (1.4), and that it is a lower bound follows by considering the bipartition taking variables $X^{(1)}$ and $X^{(6)}$ as row indices, and the other four variables as column indices. The resulting $n^4 \times n^8$ matrix has full rank.

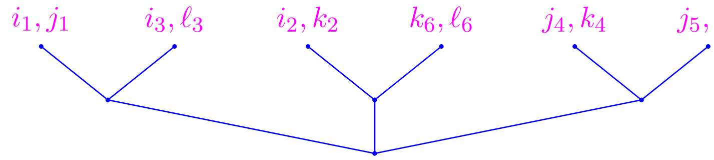
Tensor networks are more versatile and can be more efficient than low-rank decompositions of $T(A)$. Nevertheless, we show limitations on this versatility. In particular we show that every tensor network execution for A induces a tree in which the leaves are the inputs of A and all internal vertices have degree 3. We call this a *socket tree*. Each edge in a socket tree induces a bipartition of the variables and our key technical lemma is to show that for each such bipartition, the rank of the corresponding flattening of $T(A)$ is a lower bound on the cost of the execution that gave rise to the tree. Thus, to obtain a lower bound for the cost of a specific execution, we consider the maximum rank obtained over all edges of the corresponding socket tree, and to lower bound the cost of every tensor network execution, we minimize this quantity over

[4] This bound is the running time of the algorithm of Nešetřil and Poljak [75] for counting 4-cliques, which if $\omega > 2$ is slightly worse than the running time of Theorem 1.1.

all possible socket trees. We refer to the resulting quantity as the *socket width* of A, denoted $w(A)$ (formal definition appears in Section 6). Our general lower bound can thus be phrased as follows, where $c(A)$ denotes the minimum cost of a tensor network for evaluating A (formal definition appears in Section 4.5).

Theorem 1.4. *For every multilinear map A, it holds that $c(A) \geq w(A)$.*

Indeed, for our running example (1.4), there are low-width socket trees establishing $w(A) \leq n^3$, see (1.6). However, that bound is tight: our $\Omega(n^{\lceil 2\cdot 4/3 \rceil}) = \Omega(n^3)$ lower bound for the $\binom{4}{2}$-linear form (Theorem 1.2) is obtained by proving $w(A) \geq n^3$ (Lemma 7.4) and appealing to Theorem 1.4.



(1.6)

1.5 Earlier and related work

We now proceed to a more detailed discussion of earlier work.

Tensor networks The history of tensor networks (or, alternatively, *tensor diagrams* or *diagrams*) as an analytical and computational tool goes back to the 19th-century to the work of Cayley [27, 28], Clebsch [31], Clifford [32], Sylvester [94], and Kempe [57, 58]. The diagrammatic form used here can be traced back to Penrose [81]. Some early appearances of tensor diagrams are by Cvitanovič [38], Cvitanovič and Kennedy [39], Kuperberg [64], and many others. Surveys of tensor diagrams are given by Penrose and Rindler [82] and by Landsberg [66]. Schrijver [90] provides a brief historical account.

Deviating from Penrose's notation, we work relative to a fixed basis and a corresponding dual basis in each relevant vector space to avoid distinction between primal and dual spaces. This in particular enables a concise treatment of hyperedges and saves us from considering orientation of edges, or the planar layout of edges in a drawing. That is, we will view a tensor network combinatorially as a hypergraph with tensors associated at the vertices, and with a subset of the hyperedges designated to form the boundary of the network (see Section 4 for the precise definitions). A yet further conceptual difference is that we view the execution of a tensor network as a sequence of contractions of sets of *vertices* (tensors) rather than as contractions of hyperedges (modes). This choice enables us to reduce the size of hyperedges gradually before eliminating them during an execution, thus enabling better granularity. For simplicity, we will restrict to a purely multilinear framework and will not consider sums of networks although such a study would be possible, and is pursued, e. g., in Penrose's paper [81].

A large body of existing work in applications (see Section 1.2) studies how to efficiently execute a *given* tensor network D. Our quest in this paper differs from such investigations in that we take a multilinear map A, and seek to *design the most efficient network D that realizes A*, or to establish lower bounds for best-possible designs. In particular, our upper and lower bounds in Theorem 1.1 and Theorem 1.2 are over all tensor networks that realize a particular map A of interest.

Computational problems on tensors and tensor networks Problems on given tensors and tensor networks are known to be mostly computationally hard as soon as the setting changes from matrices to higher-order tensors. Håstad [48] showed that computing the rank of an explicitly given ℓ-tensor is NP-complete over finite fields and NP-hard over rationals whenever $\ell \geq 3$. Hillar and Lim extended the latter result to any field containing $\mathbb{Q}$ [49]. They also showed that many other tensor problems such as the eigenvalue, singular value and spectral norm decision and approximation problems as well as the rank-1 approximation problem for 3-tensors (over $\mathbb{R}$ and in some cases over $\mathbb{C}$) are NP-hard.

The task of finding the value of a given scalar-valued tensor network is known to be #P-complete (see, e. g., [10, 14]). Similarly, it is NP-hard to find the most efficient sequence of contractions for a given network [65, 83].

Tensor networks in applications Beyond our present use of tensor networks as a model of computation to efficiently evaluate multilinear maps, tensor networks have been used across a broad range of applications. Accordingly, the following should be viewed as mere pointers to further literature on tensor networks, not as an exhaustive listing of all applications of tensor networks. Orus [78] gives an introduction to tensor networks in the context of computational physics. Itai and Landau [7] study quantum computation and quantum algorithms for evaluating tensor networks [7]. Solomonik and Hoefler study sparse tensor algebra as a model for parallel programming [91]. The Holant framework introduced by Valiant [96] and studied further by Cai *et al.* [25, 24] involves the study of multilinear sum–product expressions that can be viewed as tensor networks; for the reader familar with the Holant framework, the connection with tensor networks is explained in Appendix A.3. Tensor networks appear naturally in the study of probabilistic graphical models [62, 74, 87, 101], and in the study of various machine-learning problems [29, 30].

Bilinear and multilinear complexity As was concisely outlined in Section 1.4, for bilinear maps our present work is captured by the study of the tensor rank of 3-tensors and an extensive body of work on bilinear complexity, with the arithmetic complexity of the matrix multiplication map as the primary driver. For two starting points to this literature, we refer to the monograph by Bürgisser, Clausen, and Shokrollahi [23] and to the survey by Pan [79]. Our present work can be seen as generalizing this bilinear theory to higher orders of linearity via tensor networks and their executions. The current state of the art for fast matrix multiplication is due to Alman and Vassilevska Williams [4], Le Gall [68, 69], Le Gall and Urrutia [70], Vassilevska Williams [98], Cohn and Umans [34], Cohn, Kleinberg, Szegedy, and Umans [33], Benson and Ballard [12], and Huang, Rice, Matthews, and van de Geijn [50].

1.6 Organization of this paper

Section 2 recalls preliminaries on tensors, multilinear maps, and branchwidth. Section 3 reviews the specific multilinear maps that we study in this paper and describes the associated tensor for each map. In Section 4, tensor networks, execution and cost of a tensor network, and cost of a multilinear map are defined. Section 5 presents tensor-network algorithms for the multilinear maps introduced in Section 3. In Section 6, a general lower bound on the cost of evaluating a multilinear map using a tensor network is obtained. The lower bound is expressed in terms of the socket-width of a multilinear map. In Section 7, lower bounds on socket-width for concrete multilinear maps studied in Sections 3 and 5 are derived. Finally, Appendix A gives some background results on minimum-cost executions.

2 Preliminaries

Throughout the paper $[n]$ denotes $\{1, 2, \ldots, n\}$ and $\mathbb{F}$ denotes an arbitrary fixed field.

2.1 Tensors

This section sets up our notation for tensors and multilinear maps. We work with tensors and multilinear maps relative to fixed bases for the respective vector spaces over $\mathbb{F}$.

Modes, indexing, and positions We will work with the following convention of positioning individual entries inside a tensor. Let E be a finite set of *modes*. Associate with each mode $e \in E$ a finite nonempty *index set* $J(e)$. In this case we say that E is a set of *indexed* modes. The *length* of e is $|J(e)|$. A *position* is an element $j \in \prod_{e\in E} J(e)$. Let us write $J(E) = \prod_{e\in E} J(e)$ for the set of all positions with respect to the indexed modes E. In the special case that the set E of modes is empty we define the set $J(E)$ of positions to consist of a single element.

For example, the 3-tensor α shown on the right in (1.1) (page 4) involves the set $E = \{i', k, \ell\}$ of modes where $J(i') = [2]$, $J(k) = [2]$, and $J(\ell) = [7]$ and a position in the tensor is an element of $J(E) = [2] \times [2] \times [7]$.

Tensors, matrices, vectors, and scalars Let E be a set of indexed modes. A *tensor* with respect to E is a mapping $T : J(E) \to \mathbb{F}$. Equivalently, we write $T \in \mathbb{F}^{J(E)}$ to indicate that T is a tensor with respect to the indexed set E of modes. We view the set $\mathbb{F}^{J(E)}$ of all tensors over E as a vector space over $\mathbb{F}$ with addition and scalar multiplication of tensors defined entrywise. We say that $|E|$ is the *order* of the tensor. A tensor of order zero is called a *scalar*, a tensor of order one is called a *vector*, and a tensor of order two is called a *matrix*. The *volume* of the tensor is $|J(E)|$. The tuple $(|J(e)| : e \in E)$ is the *shape* of the tensor. It is convenient to use the "×" symbol to highlight the shape of a tensor; that is, instead of writing, say $(2, 3, 4)$ for the shape, we write $2 \times 3 \times 4$. For a position $j \in J(E)$ and a tensor $T \in \mathbb{F}^{J(E)}$, we say that $T_j \in \mathbb{F}$ is the *entry* of T at j.

A *flattening* of T induced by a bipartition $E_1 \cup E_2 = E$ of the modes of T is a $|J(E_1)| \times |J(E_2)|$ matrix M where, for $j_1 \in J(E_1)$ and $j_2 \in J(E_2)$ we have $M_{j_1,j_2} = T_{j_1 j_2}$. Given two order-$\ell$ tensors $S \in \mathbb{F}^{[n_1]\times[n_2]\times\cdots\times[n_\ell]}$ and $T \in \mathbb{F}^{[m_1]\times[m_2]\times\cdots\times[m_\ell]}$, their *Kronecker product* $S \otimes T$ is a tensor in $\mathbb{F}^{[n_1 m_1]\times[n_2 m_2]\times\cdots\times[n_\ell m_\ell]}$ defined by

$$(S \otimes T)_{m_1(i_1-1)+j_1, m_2(i_2-1)+j_2, \ldots, m_\ell(i_\ell-1)+j_\ell} = S_{i_1,i_2,\ldots,i_\ell} T_{j_1,j_2,\ldots,j_\ell}.$$

For example, the tensor α shown on the right in (1.1) has order three, volume 28, and shape $(2,2,7)$ or $2\times2\times7$. The flattening of α induced by the bipartition $\{i',k\}\cup\{\ell\}$ is the 4×7 matrix obtained by taking the four rows of the two 2×7 matrices shown. The Kronecker product $\alpha\otimes\alpha$ of α with itself is an order-3 tensor of shape $4\times4\times49$.

2.2 Multilinear maps

Let $E_1, E_2, \ldots, E_\ell, E'$ be pairwise disjoint sets of indexed modes such that $E_1, E_2, \ldots, E_\ell$ are nonempty. We say that a map $A : \mathbb{F}^{J(E_1)} \times \mathbb{F}^{J(E_2)} \times \cdots \times \mathbb{F}^{J(E_\ell)} \to \mathbb{F}^{J(E')}$ is an *ℓ-linear map* if A is linear with respect to each of its ℓ inputs individually when the other inputs remain fixed. In particular, a 1-linear map is a linear map. A multilinear map that gives a scalar output is a multilinear *form*. In particular, A is a form if and only if E' is empty.

The tensors of a multilinear map For an ℓ-linear map $A : \mathbb{F}^{J(E_1)} \times \mathbb{F}^{J(E_2)} \times \cdots \times \mathbb{F}^{J(E_\ell)} \to \mathbb{F}^{J(E')}$, we define two slightly different tensors $T(A)$ and $\hat{T}(A)$. The tensor $T(A)$ is indexed by $J(E_1)\times J(E_2)\times\cdots\times J(E_\ell)\times J(E')$, and the tensor $\hat{T}(A)$ is indexed by $J(E_1\cup E_2\cup\ldots\cup E_\ell\cup E')$. At a position $(j_1, j_2, \ldots, j_\ell, j')$ of $T(A)$ and the corresponding position $j_1 j_2 \ldots j_\ell j'$ of $\hat{T}(A)$ these take the value

$$T(A)_{(j_1,j_2,\ldots,j_\ell,j')} = \hat{T}(A)_{j_1 j_2 \ldots j_\ell j'} = A\big(e^{(j_1)}, e^{(j_2)}, \ldots, e^{(j_\ell)}\big)_{j'},$$

where $e^{(j_i)} \in \mathbb{F}^{J(E_i)}$ denotes the tensor with a 1 in position j_i and 0s in all other position. The only difference between $T(A)$ and $\hat{T}(A)$ is their shape. The shape of $T(A)$ is $|J(E_1)|\times|J(E_2)|\times\cdots\times|J(E_\ell)|\times|J(E')|$, except if A is a form in which case the $|J(E')|$ part is omitted. Thus $T(A)$ is of order $\ell+1$ (or ℓ if A is a form). The shape of $\hat{T}(A)$ is $\bigtimes_{e\in E_i, i\in[\ell]} |J(e)| \times \bigtimes_{e'\in E'} |J(e')|$, thus its order is $|E_1|+|E_2|+\cdots+|E_\ell|+|E'|$.

For example, consider the matrix multiplication map $A : \mathbb{F}^{[n]\times[n]}\times\mathbb{F}^{[n]\times[n]} \to \mathbb{F}^{[n]\times[n]}$, i. e., for two $n\times n$ matrices X and Y, $A(X,Y) = X\cdot Y$. The tensor $\hat{T}(A)$ is a 6-tensor where each position is a 6-tuple $(i_1, k_1, k_2, j_1, i_2, j2) \in [n]^6$, and $T(A)$ is a 3-tensor where each position is a triple $((i_1,k_1),(k_2,j_1),(i_2,j_2)) \in ([n]\times[n])^3$. The value of $\hat{T}(A)$ at position $(i_1,k_1,k_2,j_1,i_2,j_2)$ (and of $T(A)$ at its corresponding position) is the value at position (i_2, j_2) of $A(e^{(i_1k_1)}, e^{(k_2j_1)}) = e^{(i_1k_1)}\cdot e^{(k_2j_1)}$, where $e^{(ab)}$ is the $n\times n$ matrix with a 1 in row a column b and zeroes elsewhere. This is 1 if $i_1 = i_2$, $j_1 = j_2$ and $k_1 = k_2$, otherwise 0.

In other words, each mode of $T(A)$ corresponds to one of the inputs of A, or the output. These inputs are in turn sets of indexed modes so may contain more "fine-grained" structure, but this information is lost at the level of granularity of $T(A)$. When working with tensor networks for evaluating A, we need to keep track of the fine-grained mode structure because this is in many cases what allows us to construct efficient algorithms, hence in most parts of the paper we are more interested in the tensor $\hat{T}(A)$ which contains this fine-grained structure.

On the other hand, $\hat{T}(A)$ does not contain information about which modes are grouped together to form the inputs and output of A, and this information is also important. This leads us to the notion of sockets, defined next.

Sockets Let us study the tensor $\hat{T}(A)$ with respect to the map A. We say that the modes in $E_1 \cup E_2 \cup \cdots \cup E_\ell$ are the *input* modes of $\hat{T}(A)$, and the modes in E' are the *output* modes of $\hat{T}(A)$ with respect to A. Let us say that $E_1, \ldots, E_\ell$ are the *input sockets* of $\hat{T}(A)$ with respect to A. Similarly, E' is the *output socket* in $\hat{T}(A)$ with respect to A. In particular, the output socket is empty if and only if A is a form. To describe a socketing of the modes of a tensor, it is convenient to use parentheses to group a shape of a tensor into sockets, see also Section 2.1. Two multilinear maps, A_1 and A_2, may have the same base tensor $\hat{T}(A_1) = \hat{T}(A_2)$, and from a tensor $\hat{T}$ one may obtain different multilinear maps by varying how the modes of $\hat{T}$ are assigned to input and output sockets.

The form of a multilinear map Let A be a multilinear map with a nonempty output socket. We can turn A into a multilinear form $F(A)$ by turning its output socket into an input socket. Let us say that $F(A)$ is the *multilinear form* of A. We also set $F(A) = A$ when A is a multilinear form.

For example, the form of the $n \times n$ square-matrix multiplication map is the form $F(X, Y, Z) = \sum_{i \in [n]} \sum_{j \in [n]} \sum_{k \in [n]} X_{ij} Y_{jk} Z_{ik}$.

2.3 Branch decompositions and branchwidth

A *branch decomposition* of a (hyper)graph G consists of (i) a tree T whose every vertex has degree either one or three, and (ii) a bijection π between the (hyper)edge set of G and the set of vertices of degree one in T. Deleting an edge $e \in E(T)$ from T partitions T into two subtrees, T_1 and T_2, which via π give rise to a partition of the (hyper)edges of G into two sets, E_1 and E_2. The *width* $w(e)$ of the partition induced by e is the number of vertices of G that are incident to at least one (hyper)edge in E_1 and at least one (hyper)edge in E_2. The *width* of the branch decomposition (T, π) is the maximum of the widths $w(e)$ for $e \in E(T)$. The *branchwidth* $\mathrm{bw}(G)$ of G is the minimum width of a branch decomposition of G.

For graphs, we recall the following upper bound on branchwidth.

Claim 2.1 (Robertson and Seymour [86])**.** *For every $n \geq 3$, $\mathrm{bw}(K_n) = \lceil 2n/3 \rceil$. Consequently, $\mathrm{bw}(G) \leq \lceil 2|V(G)|/3 \rceil$ for every graph G.*

3 Examples of multilinear maps

This section reviews the specific multilinear maps that we study in this paper. Together with each map we describe its associated tensor and a socketing of the tensor that realizes the map.

3.1 Discrete Fourier transform

Let $n \geq 2$ be an integer and let $\rho \in \mathbb{F}$ be a primitive n^{th} root of unity in the field $\mathbb{F}$ if it exists.[5] Define the $n \times n$ symmetric matrix Φ with entries $\Phi_{i,j} = \rho^{(i-1)(j-1)}$ for all $i, j \in [n]$. The *discrete*

[5]That is, ρ satisfies $\rho^n = 1$ and $\rho^{n/k} \neq 1$ for all integer divisors $k \geq 2$ of n.

Fourier transform (DFT) of order n at ρ is the linear map $L : \mathbb{F}^{[n]} \to \mathbb{F}^{[n]}$ defined for all $x \in \mathbb{F}^{[n]}$ by the matrix-vector multiplication $L(x) = \Phi x$. In particular, the matrix Φ is the tensor associated with L. The matrix Φ has two modes, namely its rows and columns. To realize L, take the columns of Φ as the input socket, and the rows as the output socket.

3.2 Determinant and permanent

We write S_n for the group of all permutations $\sigma : [n] \to [n]$ (the symmetric group of degree n). For $\sigma \in S_n$, let $c(\sigma)$ be the number of cycles in the cycle decomposition of σ, where each fixed point of σ is counted as a cycle. Further standard examples of multilinear maps include the determinant and the permanent,

$$\det A = \sum_{\sigma \in S_n} (-1)^{n-c(\sigma)} \prod_{i \in [n]} a_{i,\sigma(i)} \qquad\qquad \operatorname{per} A = \sum_{\sigma \in S_n} \prod_{i \in [n]} a_{i,\sigma(i)}\,, \tag{3.1}$$

both of which are n-linear in the n columns (or the n rows) of the matrix $A = (a_{ij})_{i,j \in [n]}$. The determinant and the permanent are associated with order-n tensors $\hat{T}^{(\mathrm{det})}, \hat{T}^{(\mathrm{per})} \in \mathbb{F}^{[n]\times[n]\times\cdots\times[n]}$ defined for all $j = (i_1, i_2, \ldots, i_n) \in [n] \times [n] \times \cdots \times [n]$ by

$$\hat{T}^{(\mathrm{det})}_j = \begin{cases} (-1)^{n-c(j)} & \text{if } j \in S_n \\ 0 & \text{otherwise} \end{cases} \qquad\qquad \hat{T}^{(\mathrm{per})}_j = \begin{cases} 1 & \text{if } j \in S_n \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

A socketing of $\hat{T}^{(\mathrm{det})}$ and $\hat{T}^{(\mathrm{per})}$ that realizes the determinant and the permanent, respectively, is to take each of the n modes as an input socket corresponding to A. The determinant and permanent tensors are not closed under taking Kronecker products when $n \geq 2$, because the Kronecker product of two tensors has the same order as its input tensors, but for each n there is exactly one determinant (permanent) tensor of order n.

3.3 Matrix multiplication

Let n, r, and m be positive integers. Perhaps the most fundamental example of a bilinear map is the map that multiplies an $n \times r$ matrix $A = (A_{ij})_{i \in [n],\, j \in [r]}$ with an $r \times m$ matrix $B = (B_{ij})_{i \in [r],\, j \in [m]}$ to obtain the $n \times m$ product matrix $C = (C_{ij})_{i \in [n],\, j \in [m]}$ defined for all $i \in [n]$ and $j \in [m]$ by

$$C_{ij} = \sum_{k \in [r]} A_{ik} B_{kj}\,. \tag{3.3}$$

It is natural to view the input $A \in \mathbb{F}^{[n]\times[r]}$ as a 2-tensor, and similarly so for the input $B \in \mathbb{F}^{[r]\times[m]}$, and the output $C \in \mathbb{F}^{[n]\times[m]}$. Thus, (3.3) is naturally associated with the 6-tensor $\hat{T} \in \mathbb{F}^{[n]\times[r]\times[r]\times[m]\times[n]\times[m]}$ with entries defined for all $i_1, i_2 \in [n]$, $j_1, j_2 \in [m]$, and $k_1, k_2 \in [r]$ by

$$\hat{T}_{i_1 k_1 k_2 j_1 i_2 j_2} = \begin{cases} 1 & \text{if } i_1 = i_2 \text{ and } j_1 = j_2 \text{ and } k_1 = k_2; \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

To realize (3.3) using (3.4), we can use the socketing grouped by parentheses in $([n] \times [r]) \times ([r] \times [m]) \times ([n] \times [m])$, where the first two groups are the two input sockets corresponding to A and B, and the last group is the output socket corresponding to C.

Let us write $\langle n, r, m \rangle$ as a shorthand for the tensor (3.4). From (3.4) and the definition of the Kronecker product it is immediate that matrix-multiplication tensors satisfy

$$\langle n_1, r_1, m_1 \rangle \otimes \langle n_2, r_2, m_2 \rangle = \langle n_1 n_2, r_1 r_1, m_1 m_2 \rangle \,. \tag{3.5}$$

That is, matrix multiplication tensors are closed under taking Kronecker products.

3.4 Group algebra product

Let $(\mathbb{A}, +)$ be an Abelian group of order n. Another fundamental example of a bilinear map is to convolve two vectors $f \in \mathbb{F}^{\mathbb{A}}$ and $g \in \mathbb{F}^{\mathbb{A}}$ according to the group operation of $\mathbb{A}$ to obtain the vector $h = f * g \in \mathbb{F}^{\mathbb{A}}$, defined for all $k \in \mathbb{A}$ by

$$h_k = \sum_{j \in \mathbb{A}} f_{(k-j)}\, g_j \,. \tag{3.6}$$

The map (3.6) is associated with the 3-tensor $\hat{T} \in \mathbb{F}^{\mathbb{A} \times \mathbb{A} \times \mathbb{A}}$ defined for all $i, j, k \in \mathbb{A}$ by

$$\hat{T}_{ijk} = \begin{cases} 1 & i + j = k; \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

A socketing of the three modes of (3.7) that realizes (3.6) is to take the first two modes as two input sockets corresponding to f and g, respectively, and to take the last mode as the output socket corresponding to h. The vector space $\mathbb{F}^{\mathbb{A}}$ equipped with the convolution product $*$ is the *group algebra* $\mathbb{F}[\mathbb{A}]$.

3.5 Kruskal operator

Let $n_1, n_2, \ldots, n_\ell$ and r be positive integers. Matrix multiplication generalizes naturally to the ℓ-linear task of multiplying ℓ matrices $A^{(1)} \in \mathbb{F}^{[n_1] \times [r]}$, $A^{(2)} \in \mathbb{F}^{[n_2] \times [r]}$, …, $A^{(\ell)} \in \mathbb{F}^{[n_\ell] \times [r]}$ to obtain the order-ℓ tensor $Y \in \mathbb{F}^{[n_1] \times [n_2] \times \cdots \times [n_\ell]}$ defined for all $(i_1, i_2, \ldots, i_\ell) \in [n_1] \times [n_2] \times \cdots \times [n_\ell]$ by

$$Y_{i_1 i_2 \cdots i_\ell} = \sum_{j \in [r]} A^{(1)}_{i_1 j} A^{(2)}_{i_2 j} \cdots A^{(\ell)}_{i_\ell j} \,. \tag{3.8}$$

This map is known as the *Kruskal operator* [63, 60][6] of the matrices $A^{(1)}, A^{(2)}, \ldots, A^{(\ell)}$.

The map (3.8) is associated with the 3ℓ-tensor

$$\hat{T} \in \mathbb{F}^{[n_1] \times [r] \times [n_2] \times [r] \times \cdots \times [n_\ell] \times [r] \times [n_1] \times [n_2] \times \cdots \times [n_\ell]}$$

[6]Kolda [60] calls this operator the Kruskal operator. Kruskal [63] studied the special case $\ell = 3$.

defined for all $i_1, i'_1 \in [n_1]$, $i_2, i'_2 \in [n_2]$, ..., $i_\ell, i'_\ell \in [n_\ell]$ and $j_1, j_2, \ldots, j_\ell \in [r]$ by

$$\hat{T}_{i_1 j_1 i_2 j_2 \cdots i_\ell j_\ell i'_1 i'_2 \cdots i'_\ell} = \begin{cases} 1 & \text{if } i_1 = i'_1,\ i_2 = i'_2,\ \ldots,\ i_\ell = i'_\ell \text{ and } j_1 = j_2 = \ldots = j_\ell; \\ 0 & \text{otherwise.} \end{cases} \tag{3.9}$$

A socketing of (3.9) that realizes (3.8) is to take each of the ℓ pairs of modes $[n_1] \times [r]$, $[n_2] \times [r]$, ..., $[n_\ell] \times [r]$ as an input socket and the final ℓ modes $[n_1] \times [n_2] \times \cdots \times [n_\ell]$ as the output socket.

Let us write $\langle n_1, n_2, \ldots, n_\ell \mid r \rangle$ for the tensor (3.9). Analogously to the closure property (3.5) for matrix multiplication tensors, we observe that

$$\langle n_1, n_2, \ldots, n_\ell \mid r \rangle \otimes \langle n'_1, n'_2, \ldots, n'_\ell \mid r' \rangle = \langle n_1 n'_1, n_2 n'_2, \ldots, n_\ell n'_\ell \mid r r' \rangle\,. \tag{3.10}$$

That is, Kruskal operator tensors are closed under taking Kronecker products. Furthermore, we observe that $\langle n, r, m \rangle$ and $\langle n, m \mid r \rangle$ are equal after reordering of the modes. That is, the matrix multiplication tensor (3.4) is the special case of the Kruskal operator tensor (3.9) when $\ell = 2$.

3.6 Homomorphism-counting forms

Further multilinear maps arise naturally by algebraization of combinatorial problems. For example, the linear form of the matrix multiplication map

$$\sum_{i,j,k \in [n]} A_{ij} B_{jk} C_{ki} \tag{3.11}$$

can be used to count the triangles in a graph, and the form

$$\sum_{i,j,k,\ell \in [n]} A_{ij} B_{ik} C_{i\ell} D_{jk} E_{j\ell} F_{k\ell} \tag{3.12}$$

can be used to count the occurrences of any 4-vertex subgraph in a graph by varying the six $n \times n$ matrices $A, B, C, D, E, F \in \mathbb{F}^{[n]\times[n]}$. A more complicated variant takes as input four 3-tensors $A, B, C, D \in \mathbb{F}^{[n]\times[n]\times[n]}$ of shape $n \times n \times n$ and considers the linear form

$$\sum_{i,j,k,\ell \in [n]} A_{ijk} B_{ik\ell} C_{ij\ell} D_{jk\ell}\,. \tag{3.13}$$

An $n^{4-\delta}$ time algorithm for the associated trilinear map $(A, B, C) \mapsto D$ would imply an algorithm for the Max 3-Sat problem with running time $(2 - \delta')^n$ [102].

The forms (3.11), (3.12), and (3.13) are all special cases of what we call a *homomorphism-counting form* defined by a hypergraph P, or, succinctly, a *P-linear form*. In more precise terms, let P be a k-uniform hypergraph on $v \geq k$ vertices and write $\binom{[v]}{k}$ for the set of k-element subsets of $[v]$. For each hyperedge $S = \{i_1, i_2, \ldots, i_k\} \in E(P) \subseteq \binom{[v]}{k}$ of P, let $X^{(S)}$ be an input tensor of shape $[n]^S$. The *P-linear form* of *order n* is the form

$$\sum_{\sigma \in [n]^{V(P)}} \prod_{S \in E(P)} X^{(S)}_{\sigma|_S}, \tag{3.14}$$

where $\sigma|_S$ is the restriction of σ to the k indices in S. For example, the forms (3.11), (3.12), and (3.13) are the P-linear forms corresponding to P being a triangle, a K_4, and a complete 3-uniform hypergraph on 4 vertices, respectively. It is immediate that (3.14) is an $|E(P)|$-linear map.

The map (3.14) is associated with a tensor $\hat{T}$ of order $k \cdot |E(P)|$ whose modes are $M = \{ (S, i) \mid S \in E(P), i \in S \}$ with index sets $J((S, i)) = [n]$. The entries of $\hat{T}$ are defined by

$$\hat{T}_j = \begin{cases} 1 & \text{if } \exists \sigma \in [n]^{V(P)} \text{ such that } j_{(S,i)} = \sigma_i \text{ for every } (S, i) \in M; \\ 0 & \text{otherwise.} \end{cases} \tag{3.15}$$

A socketing of (3.15) that realizes (3.14) is given by one input socket for each hyperedge $S \in E(P)$ such that the socket contains the k modes (S, i) for $i \in S$.

Let us observe that the tensors of the forms (3.12) and (3.13) are actually the same—they are both of order 12, have volume n^{12}, and are in fact equal to the outer product of the $n \times n \times n$ identity tensor with itself 4 times after renaming of modes. However, due to the difference in socketing, the forms are computationally very different. We show in Section 7 that while there are non-trivial tensor network algorithms for evaluating (3.12), no such algorithms exist for (3.13).

Let us write $\langle n \rangle_P$ for the tensor $\hat{T}$ as defined in (3.15). Analogously to the closure properties (3.5) and (3.10), we observe the closure property

$$\langle n \rangle_P \otimes \langle n' \rangle_P = \langle nn' \rangle_P \,. \tag{3.16}$$

4 Tensor networks

4.1 Networks

A *network* (or *diagram*) consists of a finite set V of *vertices*, a finite set E of *hyperedges*, an *incidence relation* $I \subseteq V \times E$, and a *boundary* $B \subseteq E$. A network is *nondegenerate* if every hyperedge is incident to at least one vertex. In what follows we assume that all networks are nondegenerate. A hyperedge $e \in E$ is a *loop* if $e \notin B$ and e is incident to exactly one vertex.

For a vertex $v \in V$, let us write $I(v) = \{e \in E : (v, e) \in I\}$ for the set of hyperedges incident to v. Dually, for a hyperedge $e \in E$, let us write $I(e) = \{v \in V : (v, e) \in I\}$ for the set of vertices incident to e. For a network D, we write $V(D)$, $E(D)$, $I(D)$, and $B(D)$ to refer to the vertices of D, the hyperedges of D, the incidence relation of D, and the boundary of D, respectively.

For example, on the left in (1.2) (page 1.2), we see a network with five vertices $V = \{A, B, \alpha, \beta, \gamma\}$, seven hyperedges $E = \{i, j, \ell, i', k, k', j'\}$, and boundary $B = \{i, j\}$. The vertices incident to i' are $I(i') = \{A, \alpha\}$.

Induced networks For a network D and a nonempty subset $W \subseteq V(D)$, the *induced* network $D[W]$ consists of the vertices in W together with the hyperedges of D that are incident to at least one vertex in W; the boundary of $D[W]$ consists of all hyperedges that are at the boundary of D or incident to a vertex outside W. Formally,

$$\begin{aligned} V(D[W]) &= W\,, \\ E(D[W]) &= \{e \in E(D) : \exists w \in W \text{ s.t. } (w,e) \in I(D)\}\,, \\ I(D[W]) &= I(D) \cap (V(D[W]) \times E(D[W]))\,, \\ B(D[W]) &= (B(D) \cap E(D[W])) \cup \{e \in E(D[W]) \colon \exists v \in V(D) \setminus W \text{ s.t. } (v,e) \in I(D)\}. \end{aligned} \tag{4.1}$$

For a vertex $v \in V$, we abbreviate $D[v] = D[\{v\}]$. Note that the boundary of $D[v]$ consists of all non-loop hyperedges incident to v in D.

For example, for the network on the left in (1.2), the induced network $D[\{\alpha, A\}]$ has hyperedges $E(D[\{\alpha, A\}]) = \{i', k, \ell\}$ and boundary $B(D[\{\alpha, A\}]) = \{\ell\}$.

4.2 Tensor networks

Let D be a network. We *index* D by associating with each hyperedge $e \in E$ an *index set* $J(e)$ of size $\ell(e)$. Induced networks inherit indexing by restriction. Next we associate with each vertex $v \in V$ a tensor $T(v) \in \mathbb{F}^{J(I(v))}$. We say that D equipped with the tensors $(T(v))_{v \in V}$ is a *tensor network*.

The *value* of a tensor network D, or the tensor *represented by* D, is a tensor $T(D) \in \mathbb{F}^{J(B)}$, defined for all $i \in J(B)$ by

$$T(D)_i = \sum_{j \in J(E(D) \setminus B)} \prod_{v \in V} T(v)_{ij}\,. \tag{4.2}$$

Observe that in (4.2) the positions i and j together identify a unique entry of $T(v)$ by projection to $J(I(v))$. We also observe that the value of a tensor network with an empty boundary is a scalar.

4.3 Contracting tensors

Let D be a tensor network and let $W \subseteq V(D)$ be a nonempty set of vertices. Let w be a new element not in V. We may *contract* W in D to obtain a tensor network D/W by replacing the sub-network $D[W]$ in D with the single vertex w whose associated tensor $T(w)$ is the tensor represented by $D[W]$. Formally,

$$\begin{aligned} V(D/W) &= (V(D) \setminus W) \cup \{w\}\,, \\ E(D/W) &= E(D) \setminus (E(D[W]) \setminus B(D[W]))\,, \\ I(D/W) &= (I(D) \setminus I(D[W])) \cup \{(w,e) : e \in B(D[W])\}\,, \\ B(D/W) &= B(D)\,, \\ T(w) &= T(D[W])\,. \end{aligned} \tag{4.3}$$

The *cost* of contracting W in D is $c(D, W) = \prod_{e \in E(D[W])} |J(e)|$. The value of a tensor network is invariant under contraction, i. e., for all nonempty $W \subseteq V(D)$ it holds that $T(D) = T(D/W)$ (see Lemma A.1 for a proof).

For example, contracting $W = \{\alpha, A\}$ in the network on the left in (1.2) results in the network shown in the second step, where the tensor $T = T(D[\{\alpha, A\}])$ of the new (unlabelled) vertex is a vector of length ℓ, which in position u takes the value $T_u = \sum_{i',k} \alpha_{i'ku} A_{i'k}$.

4.4 Execution and cost of a tensor network

To compute the tensor $T(D)$ from a given tensor network D, we may proceed by a sequence of contractions on D. Such a process is called executing D, and the cost of D is the cost of a minimum-cost execution of D. We proceed with the details.

Let $D = D_0$ be a tensor network with at least one tensor. For $k = 1, 2, \ldots, t$, select a nonempty subset $W_{k-1} \subseteq V(D_{k-1})$ such that W_{k-1} has at least two tensors or consists of a single tensor with a loop. Set $D_k = D_{k-1}/W_{k-1}$ and observe that the number of tensors and/or modes decreases by at least one in the contraction. Suppose that D_t is loopless and consists of a single tensor. We say that such a sequence of contractions is an *execution* of D in t *steps*. The *cost* of the execution is $\max_{k=1,2,\ldots,t} c(D_{k-1}, W_{k-1})$. The cost of an execution in zero steps is defined to be 0.

It is immediate that D has at least one execution and every execution consists of at most $2|V(D)| - 1$ steps. By invariance under contractions, we have $T(D_t) = T(D)$. The *cost* $c(D)$ of D is the cost of a minimum-cost execution of D.

An execution of D of cost $c(D)$ immediately translates into an algorithm that computes $T(D)$ using $O(c(D)|V(D)|)$ arithmetic operations in $\mathbb{F}$, since the contraction step $D_k = D_{k-1}/W_{k-1}$ takes $O(c(D_{k-1}, W_{k-1})) \leq c(D)$ time to evaluate, and there are $O(V(D))$ steps.

Lemma 4.1. *Let D be a tensor network. There exists a minimum-cost execution of D such that each contracted set has size at most two. Furthermore, if D is loopless, we can assume that each contracted set has size exactly two.*

Lemma 4.1 is proven in Appendix A.2. In what follows we restrict to consider loopless D only. Thus while a general execution may contract arbitrary vertex sets in D in each step, we may assume without loss of generality that the minimum-cost execution has the structure of a rooted binary tree, whose leaves are the vertices of the tensor network, and each internal vertex is the tensor obtained by contracting its two children.

For example, (1.2) illustrates an execution of the network shown there, both as a sequence of contractions of pairs of vertices, and more succinctly as an execution tree overlaid on top of the network.

4.5 Cost of a multilinear map

Let us now define the cost of a multilinear map via the minimum-cost tensor networks (and socketing) for evaluating the map. That is, the cost of a multilinear map is defined in terms of the best tensor network that implements the map. In more precise terms, let

$$A : \mathbb{F}^{J(E_1)} \times \mathbb{F}^{J(E_2)} \times \cdots \times \mathbb{F}^{J(E_\ell)} \longrightarrow \mathbb{F}^{J(E')}$$

be an ℓ-linear map. Consider the tensor $\hat{T}(A)$ of A and the associated input sockets $E_1, E_2, \ldots, E_\ell$ and the output socket E'. Let D^* be an arbitrary tensor network such that $T(D^*) = \hat{T}(A)$ and the boundary satisfies $B(D^*) = E_1 \cup E_2 \cup \cdots \cup E_\ell \cup E'$. Modify the network D^* as follows. For each $k = 1, 2, \ldots, \ell$, introduce a new vertex to D^*, make the new vertex incident to each of the modes in the input socket E_k, and associate the new vertex with a tensor $X^{(k)} \in \mathbb{F}^{J(E_k)}$. Remove the modes $E_1 \cup E_2 \cup \cdots \cup E_\ell$ from the boundary of D^*. Let us denote the resulting network by D and call the introduced ℓ new vertices the *socket vertices* of D. We observe that $B(D) = E'$ and $A(X^{(1)}, X^{(2)}, \ldots, X^{(\ell)}) = T(D)$. Furthermore, the cost $c(D)$ is independent of the value of $X^{(k)} \in \mathbb{F}^{J(E_k)}$ for $k = 1, 2, \ldots, \ell$. We say that D is a *realization* of A if it can be obtained from A by this process, and write $\mathscr{D}(A)$ for the set of all tensor network realizations D of A.

The *cost* of the map A is $c(A) = \min_{D \in \mathscr{D}(A)} c(D)$. In particular, we observe that the minimum exists since the cost of a tensor network is a nonnegative integer and the family $\mathscr{D}(A)$ is nonempty.

5 Upper bounds

This section presents tensor-network algorithms for the maps introduced in Section 3. We start with our key technical result that cost is submultiplicative (Theorem 1.3, stated formally as Theorem 5.1), which then enables us to represent essentially the fastest known algorithms using tensor networks, and, in the case of P-linear forms, also improve on earlier work as reviewed in Section 1.3.

Recall that the cost of an execution of a network is the maximum number of operations of any step (contraction) in that execution, rather than the total running time (number of operations). With the exception of Theorem 1.3, the results in this section are stated as the resulting upper bound on running time but they are all derived by obtaining upper bounds on cost.

5.1 Submultiplicativity of cost

Let $E_1, E_2, \ldots, E_\ell, E'$ be pairwise disjoint sets of indexed modes such that $E_1, E_2, \ldots, E_\ell$ are nonempty. Let

$$A : \mathbb{F}^{J(E_1)} \times \mathbb{F}^{J(E_2)} \times \cdots \times \mathbb{F}^{J(E_\ell)} \to \mathbb{F}^{J(E')}$$

be an ℓ-linear map. For a positive integer k, we define the ℓ-linear map $A^{\otimes k}$ such that its tensor satisfies $T(A^{\otimes k}) = T(A)^{\otimes k}$. Then

$$A^{\otimes k} : \mathbb{F}^{J(E_1)^k} \times \mathbb{F}^{J(E_2)^k} \times \cdots \times \mathbb{F}^{J(E_\ell)^k} \to \mathbb{F}^{J(E')^k}.$$

Note that $T(A^{\otimes k}) = T(A)^{\otimes k}$ is the k-fold Kronecker product of $T(A)$ with itself—that is, it has the same order, but the index sets are larger—whereas $\hat{T}(A^{\otimes k})$ is the k-fold outer product of $\hat{T}(A)$ with itself—that is, its index sets have the same sizes, but its order is k times larger.

Let D be a network that realizes A and let $\mathcal{T}_D$ be an execution tree for D. For each internal vertex x in $\mathcal{T}_D$ (that is, a vertex obtained by contraction), define the *amortized cost* of x by splitting into the following three cases:

1. if neither of the two subtrees of x contains a socket vertex, the amortized cost of x is 1;

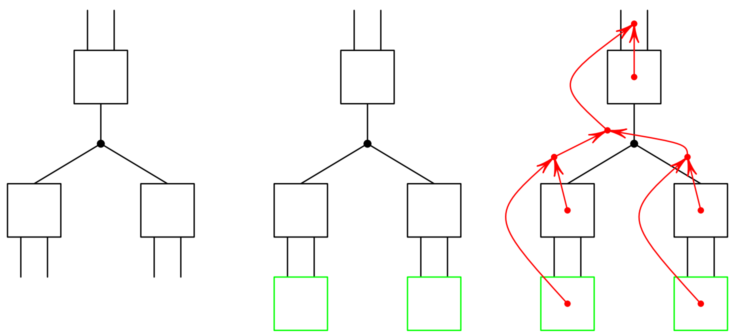
2. if exactly one of the subtrees of x, say, the subtree rooted at y (where x and y are adjacent in $\mathcal{T}_D$), contains at least one socket vertex, the amortized cost of x is the maximum of the volume of the tensor at x and the volume of the tensor at y;[7]

[7]Here, it is crucial to note that the volume of the other subtree rooted at x, only containing non-socket vertices, does not contribute directly to the amortized cost of x.

3. if both of the subtrees of x contain at least one socket vertex, the amortized cost of x is the cost of the contraction to obtain x.

The *amortized cost* $a(\mathcal{T}_D)$ of $\mathcal{T}_D$ is the maximum of the amortized costs of the internal vertices of $\mathcal{T}_D$. Since the amortized cost of each internal vertex of $\mathcal{T}_D$ is at most its cost, we have $a(\mathcal{T}_D) \leq c(\mathcal{T}_D)$. Furthermore, we observe that the amortized cost of x in case (ii) above may be strictly less than the cost of the contraction to obtain x. In particular, in (ii) the amortized cost is defined *not by the cost of the contraction but rather by volume*. This is because in a k^{th} Kronecker power we can amortize the cost of the aggregate transformation in case (ii) not with a single contraction but with a sequence of k contractions. This observation will form the heart of the proof of Theorem 1.3.

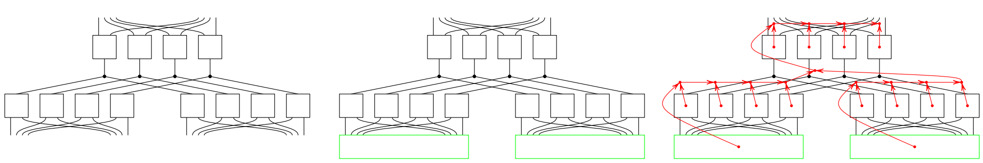
Before proceeding with the proof, let us illustrate the key ideas in visual terms. Let us start with the three illustrations in (5.1).



(5.1)

Suppose the leftmost network in (5.1) is socketed so that the two modes at the top form the output socket, and the four modes at the bottom form two input sockets so that modes in the same socket are incident to the same vertex. In the middle in (5.1), we have adjoined two socket vertices to the input sockets to obtain a realization D. On the right in (5.1), we display an execution tree $\mathcal{T}_D$ for D. Observe that the bottom-most internal vertices of $\mathcal{T}_D$, and the top-most internal vertex of $\mathcal{T}_D$, have type (ii). The internal vertex in the center has type (iii). (There are no internal vertices of type (i).) Supposing that all the modes have length at least 2, we also observe that the vertices of type (ii) have amortized cost strictly less than their contraction cost.

Let us now consider the k^{th} power of (5.1) visually, for $k = 4$:



(5.2)

The leftmost network in (5.2) depicts the k-fold outer product of the network on the left in (5.1) with itself. Observe that we simply take k copies of the network, but that for the purposes of the visualization we have taken care to draw the k copies of each mode together for the socketing. In the middle in (5.2), we have adjoined two socket vertices to the input sockets to obtain a realization $D^{\otimes k}$ of $A^{\otimes k}$. On the right in (5.2), we display an execution tree $\mathcal{T}_{D^{\otimes k}}$ for

$D^{\otimes k}$. Observe how each of the internal vertices of type (ii) in $\mathcal{T}_D$ gets expanded to a sequence of k internal vertices in $\mathcal{T}_{D^{\otimes k}}$. This transformation from $\mathcal{T}_D$ to $\mathcal{T}_{D^{\otimes k}}$ is the gist of the following theorem.

Theorem 5.1 (Formal statement of Theorem 1.3)**.** *Let D be an arbitrary realization of A and let $\mathcal{T}_D$ be an arbitrary execution tree for D. For all positive integers k, we have*

$$c(A^{\otimes k}) \leq a(\mathcal{T}_D)^{k-1} c(\mathcal{T}_D)\,. \tag{5.3}$$

Furthermore, this realization of $A^{\otimes k}$ consists of at most $k \cdot |V(D)|$ vertices.

Proof. Let D^* be the subnetwork of D with $T(D^*) = \hat{T}(A)$. That is, D^* is the network induced by all the non-socket vertices of D. Taking k disjoint copies of D^*, we obtain a network whose tensor is $\hat{T}(A^{\otimes k})$. Attaching the resulting network to tensors at sockets gives a realization of $A^{\otimes k}$. Let us write $D^{\otimes k}$ for this realization.

To establish (5.3), it suffices to construct an execution tree $\mathcal{T}_{D^{\otimes k}}$ for $D^{\otimes k}$ whose cost satisfies $c(\mathcal{T}_{D^{\otimes k}}) \leq a(\mathcal{T}_D)^{k-1} c(\mathcal{T}_D)$. We construct $\mathcal{T}_{D^{\otimes k}}$ by rewriting $\mathcal{T}_D$ from leaves towards the root to consider the k copies of each vertex in D^*. We start with leaf vertices which are the vertices of $D^{\otimes k}$. We split the process into cases (i), (ii), and (iii) as in the definition of amortized cost. Let x be the internal vertex of $\mathcal{T}_D$ that we are currently considering.

In case (i), we perform the contraction indicated by x in each of the k copies of D^* in $D^{\otimes k}$ individually. This creates k new internal vertices in $\mathcal{T}_{D^{\otimes k}}$ that are all copies of x. We set these k vertices as the vertices that correspond to x in the subsequent steps. Each of these contractions in $\mathcal{T}_{D^{\otimes k}}$ has the same cost as the contraction indicated by x in $\mathcal{T}_D$. This cost is less than or equal to $c(\mathcal{T}_D)$.

In case (ii), let y be the child of x in $\mathcal{T}_D$ such that the subtree rooted at y contains a socket vertex, and let z be the other child of x in $\mathcal{T}_D$. There is a single vertex in $\mathcal{T}_{D^{\otimes k}}$ corresponding to y and k identical vertices in $\mathcal{T}_{D^{\otimes k}}$ corresponding to z. We contract these k vertices individually each with the vertex that corresponds to y. This creates k new internal vertices in $\mathcal{T}_{D^{\otimes k}}$, where we set the topmost vertex as the vertex that corresponds to x in the subsequent steps. After the ith step, the corresponding tensor has i copies of modes of x and $k-i$ copies of modes of y. The cost of the contraction in the ith step is the cost of contracting y and z in $\mathcal{T}_D$ multiplied by the the volume of y to the power $k-i$ and the volume of x to the power $i-1$. Since the volumes of x and y are less than or equal to $a(\mathcal{T}_D)$, this cost is less than or equal to $a(\mathcal{T}_D)^{k-1}c(\mathcal{T}_D)$.

In case (iii), let y and z be the two child vertices of x in $\mathcal{T}_D$. By the structure of the earlier steps, we have that a single vertex in $D^{\otimes k}$ corresponds to y, and similarly for z. We contract these two vertices. This creates one new internal vertex in $\mathcal{T}_{D^{\otimes k}}$, which we set as the vertex that corresponds to x in the subsequent steps. This tensor has k copies of modes of x. The cost of this contraction in $\mathcal{T}_{D^{\otimes k}}$ is the cost of the corresponding contraction in $\mathcal{T}_D$ to the k^{th} power, because both tensors have k copies of all modes compared to y and z. By definition, in case (iii) the amortized cost of contracting y and z is the same as the cost of contracting y and z. Hence the cost of this contraction in $\mathcal{T}_{D^{\otimes k}}$ is less than or equal to $a(\mathcal{T}_D)^k \leq a(\mathcal{T}_D)^{k-1}c(\mathcal{T}_D)$.

This rewriting process produces an execution tree $\mathcal{T}_{D^{\otimes k}}$ for $D^{\otimes k}$ with $c(\mathcal{T}_{D^{\otimes k}}) \leq a(\mathcal{T}_D)^{k-1}c(\mathcal{T}_D)$. □

An immediate corollary is that tensor networks can use low rank decompositions of $T(A)$ to efficiently evaluate $A^{\otimes k}$.

Corollary 5.2 (Submultiplicativity of low-rank executions). *Let* $A : \mathbb{F}^{J(E_1)} \times \mathbb{F}^{J(E_2)} \times \cdots \times \mathbb{F}^{J(E_\ell)} \to \mathbb{F}^{J(E')}$ *be a multilinear map. Define* $m = \max\{|J(E_1)|, |J(E_2)|, \ldots, |J(E_\ell)|, |J(E')|\}$ *and* $r = \operatorname{rk} T(A)$. *Then* $c(A^{\otimes k}) \le \max(r, m)^k \min(r, m)$

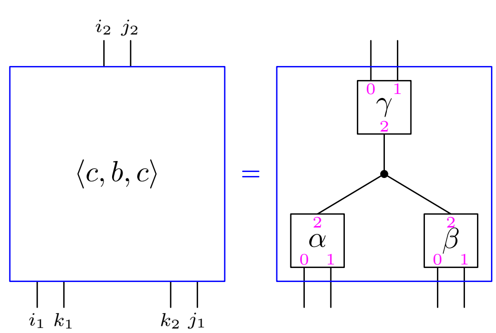
Proof. By taking a star-like network topology (as in (5.1)) we get an execution with $a(\mathcal{T}_D) = \max(r, m)$ and cost $c(\mathcal{T}_D) = m \cdot r$. □

5.2 Fast matrix multiplication

Let us now illustrate the use of Theorem 5.1 by capturing fast matrix multiplication with tensor networks. We start by recalling that for a constant $h > 0$, we write $\omega(h)$ for the rectangular matrix multiplication exponent (see Section 1.3). It is immediate that $\max(2, h+1) \le \omega(h) \le h+2$; for the state-of-the-art bounds on $\omega(h)$, see Le Gall and Urrutia [70].

Lemma 5.3. *For all constants* $h > 0$ *and* $\epsilon > 0$ *it holds that an* $n \times \lfloor n^h \rfloor$ *matrix may be multiplied with an* $\lfloor n^h \rfloor \times n$ *matrix in* $O(n^{\omega(h)+\epsilon})$ *operations by executing a tensor network.*

Proof. Fix an $0 < \epsilon < 1$. Let $0 < \delta_1, \delta_2 < 1$ be constants whose precise values we will fix later. By definition of $\omega(h)$, for all $t > \omega(h)$ there exists a constant $U > 0$ such that the arithmetic complexity of multiplying an $c \times \lfloor c^h \rfloor$ matrix with an $\lfloor c^h \rfloor \times c$ matrix is at most Uc^t for all c. Since the rank of a bilinear map is bounded from above by two times its (multiplicative) complexity (see [23, §14.1]), it follows that there exist three 3-tensors α, β, γ respective shapes $(c \times b) \times d$, $(b \times c) \times d$, and $(c \times c) \times d$ for positive integer constants b, c, and d with $\max(c^2, cb) \le d \le c^{\omega(h)+\delta_1}$ and $c^{h-\delta_2} \le b \le c^h$ such that α, β, γ decompose the matrix multiplication tensor $\langle c, b, c \rangle$ defined by (3.4) as depicted below; the indices $i_1, k_1, k_2, j_1, i_2, j_2$ refer to the tensor (3.4). The mode shared by α, β, γ in (5.4) has length d, the modes i_1, i_2, j_1, j_2 each have length c, and the modes k_1, k_2 have length b.

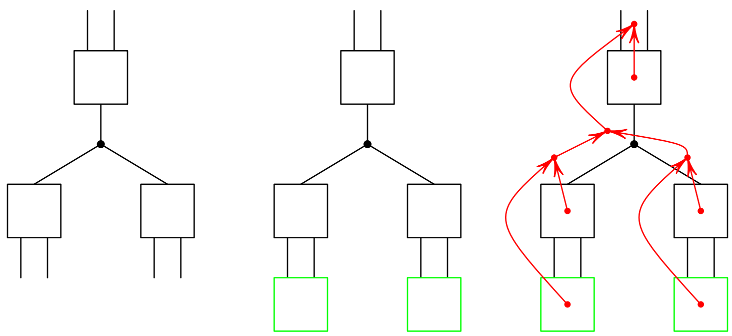


(5.4)

For example, for $h = 1$, Strassen's decomposition [93] as depicted in (5.5) below realizes (5.4) with $c = b = 2$ and $r = 7$. We use the numbering in magenta to indicate correspondence between modes in (5.4) and (5.5).

$$\alpha = \underset{2}{\begin{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}_{1} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & -1 \end{bmatrix}_{1} \end{bmatrix}_{0}} \quad \beta = \underset{2}{\begin{bmatrix} \begin{bmatrix} 1 & 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}_{1} \\ \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & 1 \end{bmatrix}_{1} \end{bmatrix}_{0}} \quad \gamma = \underset{2}{\begin{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}_{1} \\ \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}_{1} \end{bmatrix}_{0}} \tag{5.5}$$

Let us next set up an application of Theorem 5.1. Let $A : \mathbb{F}^{[c]\times[b]} \times \mathbb{F}^{[b]\times[c]} \to \mathbb{F}^{[c]\times[c]}$ be the bilinear map that multiplies a $c \times b$ matrix with a $b \times c$ matrix. Observe that the tensor of A is $\hat{T}(A) = \langle c, b, c\rangle$. To realize A, define two input sockets in (5.4), namely $\{i_1, k_1\}$ and $\{k_2, j_1\}$ to obtain a realization D and an execution tree $\mathcal{T}_D$ as follows:



(5.6)

Since $\max(c^2, cb) \le d$, the amortized cost of $\mathcal{T}_D$ satisfies $a(\mathcal{T}_D) = d$. The cost is $c(\mathcal{T}_D) = \max(c^2, cb)d$.

Let the matrices $X \in \mathbb{F}^{[n]\times[\lfloor n^h \rfloor]}$ and $Y \in \mathbb{F}^{[\lfloor n^h \rfloor]\times[n]}$ be given. We construct a tensor network that multiplies X and Y. We may assume that $\delta_2 < h$. Let $k = \lceil \frac{h \log n}{(h-\delta_2)\log c} \rceil$. We observe that $n^h \le c^{(h-\delta_2)k} \le b^k$ and $n \le c^{(1-\delta_2/h)k} \le c^k$. Extend the matrices X, Y to $X \in \mathbb{F}^{[c^k]\times[b^k]}$ and $Y \in \mathbb{F}^{[b^k]\times[c^k]}$ by inserting rows and columns with zero-entries.

Since $\langle c, b, c\rangle^{\otimes k} = \langle c^k, b^k, c^k\rangle$ by (3.5), we have that $A^{\otimes k}$ is the map that multiplies a $c^k \times b^k$ matrix with a $b^k \times c^k$ matrix. Using Theorem 5.1 with D and $\mathcal{T}_D$, we obtain $c(A^{\otimes k}) \le a(\mathcal{T}_D)^{k-1} c(\mathcal{T}_D) = d^k \max(c^2, cb)$. Moreover, the realization $D^{\otimes k}$ of $A^{\otimes k}$ given by Theorem 5.1 consists of $|V(D^{\otimes k})| = O(k)$ vertices. We can now associate X and Y with the two socket vertices of $D^{\otimes k}$, taking care to associate X with the left socket (originating from $\{i_1, k_1\}$ and α) and Y with the right socket (originating from $\{k_1, j_1\}$ and β). (Cf. (5.2) for an illustration how D and $\mathcal{T}_D$ in (5.6) yield $D^{\otimes k}$ and $\mathcal{T}_{D^{\otimes k}}$.) Executing $D^{\otimes k}$ then results in the product matrix $A^{\otimes k}(X, Y) = XY$ in

$$\begin{aligned} d^k \max(c^2, cb)|V(D^{\otimes k})| &\le c^{(\omega(h)+\delta_1)k} c^2 b |V(D^{\otimes k})| \\ &\le n^{(\omega(h)+\delta_1)(h/(h-\delta_2))} c^{5+h} b |V(D^{\otimes k})| \\ &= O(n^{\omega(h)+\epsilon}) \end{aligned}$$

operations for all large enough n; here we have used the elementary upper bound $\omega(h) \le h + 2$ and selected δ_1, δ_2 to be small enough constants (that depend on the constants h and ϵ). $\square$

The following lemma illustrates that we can also capture rectangular matrix multiplication by reduction to square matrix multiplication using tensor networks. This lemma in particular covers the cases when r does not grow as a polynomial function of n. We also observe that if $\omega = 2$, the upper bounds in the lemma are optimal up to the choice of $\epsilon > 0$ because of the size of the input/output.

Lemma 5.4. *For all $\epsilon > 0$ it holds that we may multiply an $n \times r$ matrix with an $r \times n$ matrix by executing a tensor network in*

1. *$O(n^{\omega+\epsilon-1}r)$ operations when $r \geq n$, and*
2. *$O(n^2 r^{\omega+\epsilon-2})$ operations when $r \leq n$.*

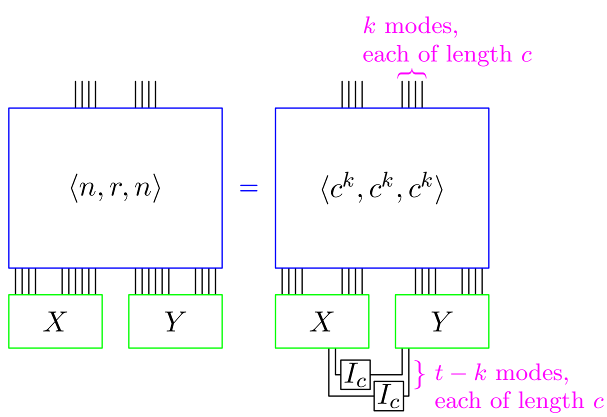
Proof. Fix an $\epsilon > 0$ and let α, β, γ be three 3-tensors of shape $(c \times c) \times d$ for constants $b = c$ and d as in the proof of Lemma 5.3. Let the matrices $X \in \mathbb{F}^{[n]\times[r]}$ and $Y \in \mathbb{F}^{[r]\times[n]}$ be given. We construct tensor networks that compute the product XY.

To establish (i), first pad X and Y using rows and columns of zero-entries so that both n and r become positive integer powers of c and n divides r. This increases n and r by at most a multiplicative factor c. We now have $n = c^k$ and $r = c^t$ for positive integers $t \geq k$.

Observe that we can compute the $n \times n$ product matrix XY by taking the sum of $\frac{r}{n} = c^{t-k}$ matrix products of size $n \times n$.

Let us implement this computation with a tensor network. Reshape X to a $(k+t)$-tensor whose all modes have length c. The first k modes index the rows, the last t modes index the columns. Reshape Y to a $(t+k)$-tensor whose all modes have length c. The first t modes index the rows, the last k modes index the columns.

Connect X and Y into a network as displayed in (5.7) on the right.



(5.7)

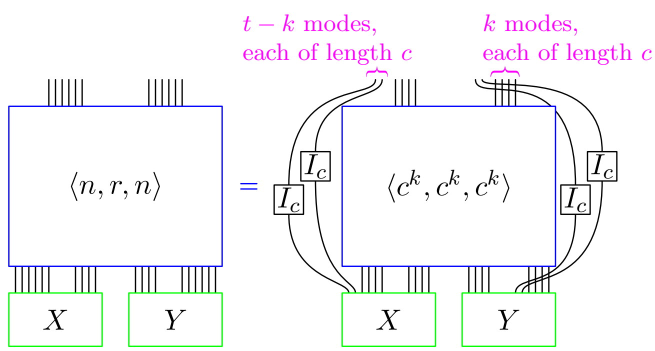
That is, we join $t-k$ column modes of X with the matching $t-k$ row modes of Y using $t-k$ identity matrices I_c (to avoid degeneracy of the network if X and Y are removed). Then we connect the remaining modes of X and Y to the left and right sockets of a matrix multiplication network for $c^k \times c^k$ matrices as depicted by $\langle c^k, c^k, c^k \rangle$ in (5.7); this matrix multiplication network is obtained as in the proof of Lemma 5.3 with $b = c$. The result is a multiplication network

$\langle n, r, n\rangle$ as depicted in (5.7) on the left. We execute this network using the structure on the right in (5.7) by first contracting (with zero cost) the identity matrices I_c with Y. We then execute the remaining network as in the proof of Lemma 5.3. For large enough n, the cost of the execution is at most $c^{k(\omega+\epsilon/2)+2}c^{t-k+1}$, which translates to $O(n^{\omega+\epsilon}r)$ operations since the network has $O(k)$ vertices whose contractions have nonzero cost.

To establish (ii), first pad X and Y using rows and columns of zero-entries so that both n and r become positive integer powers of c and r divides n. This increases r and n by at most a multiplicative factor c. We now have $r = c^k$ and $n = c^t$ for positive integers $t \geq k$.

Observe that we can compute the $n \times n$ product matrix XY by taking $(\frac{n}{r})^2 = c^{2(t-k)}$ matrix products of size $r \times r$.

Let us implement this computation with a tensor network. Reshape X to a $(t+k)$-tensor whose all modes have length c. The first t modes index the rows, the last k modes index the columns. Reshape Y to a $(k+t)$-tensor whose all modes have length c. The first k modes index the rows, the last t modes index the columns. Connect X and Y into a network as displayed in (5.7) on the right.



(5.8)

That is, we join $t-k$ row modes of X each to an identity matrix I_c whose other mode is at the boundary. Similarly, we join matching $t-k$ column modes of Y each to an identity matrix I_c whose other mode is at the boundary. (This is to avoid degeneracy of the network if X and Y are removed.) Then connect the remaining modes of X and Y to the left and right sockets of a matrix multiplication network for $c^k \times c^k$ matrices as depicted by $\langle c^k, c^k, c^k\rangle$ in (5.8); this matrix multiplication network is obtained as in the proof of Lemma 5.3 with $b = c$. The result is a multiplication network $\langle n, r, n\rangle$ as depicted in (5.8) on the left. We execute this network using the structure on the right in (5.8) by first contracting (with zero cost) the identity matrices I_c with X and Y, respectively. We then execute the remaining network as in the proof of Lemma 5.3. For large enough r, the cost of the execution is at most $c^{k(\omega+\epsilon/2)+2}c^{2(t-k)+1}$, which translates to $O(n^2 r^{\omega-1+\epsilon})$ operations since the network has $O(k)$ vertices whose contractions have nonzero cost. ◻

Let us conclude this subsection with a well-known lemma on rectangular matrix multiplication that we can also capture with tensor networks.

Lemma 5.5. *For all non-negative integers a, b, c and $\epsilon > 0$ it holds that we may multiply an $n^a \times n^b$ matrix by an $n^b \times n^c$ matrix via a tensor network in $O(n^{\max(a+b,b+c,a+c)(\omega+\epsilon)/2})$ operations.*

Proof. By symmetry we may assume that $a \le c$. Thus there are three cases to consider, namely (i) $a \le b \le c$, (ii) $a \le c \le b$, and (iii) $b \le a \le c$.

When $a \le b \le c$, we need to achieve $O(n^{(b+c)(\omega+\epsilon)/2})$ operations. Toward this end, it suffices to multiply an $n^b \times n^b$ matrix with an $n^b \times n^c$ matrix, which can be implemented as n^{c-b} multiplications of two square matrices of size $n^b \times n^b$. Proceeding analogously as in Lemma 5.4, we obtain a network that can be executed in $O(n^{c-b} n^{b(\omega+\epsilon/2)})$ operations. We have $c - b + b(\omega + \epsilon) \le (b + c)(\omega + \epsilon)/2$ as desired since $c \ge b$, $\omega \ge 2$, and $\epsilon \ge 0$.

When $a \le c \le b$, we need to achieve $O(n^{(b+c)(\omega+\epsilon)/2})$ operations. Toward this end, it suffices to multiply an $n^c \times n^b$ matrix with an $n^b \times n^c$ matrix, and apply part (i) of Lemma 5.4 to obtain a network that can be executed in $O(n^{b+c(\omega+\epsilon-1)})$ operations. We have $b + c(\omega + \epsilon - 1) \le (b + c)(\omega + \epsilon)/2$ as desired since $b \ge c$, $\omega \ge 2$, and $\epsilon \ge 0$.

When $b \le a \le c$, we need to achieve $O(n^{(a+c)(\omega+\epsilon)/2})$ operations. Toward this end, it suffices to multiply an $n^a \times n^b$ matrix with an $n^b \times n^c$ matrix. An easy modification of part (ii) of Lemma 5.4 gives a network that can be executed in $O(n^{a+c+b(\omega+\epsilon-2)})$ operations. We have $a + c + b(\omega + \epsilon - 2) \le (a + c)(\omega + \epsilon)/2$ as desired since $b \le a$, $b \le c$, $\omega \ge 2$, and $\epsilon \ge 0$. □

5.3 Homomorphism-counting for pattern graphs of small branchwidth

Our main result in this section is the following upper bound for P-linear forms when P is a graph of small branchwidth.

Lemma 5.6. *For any fixed pattern graph P and every $\epsilon > 0$, there is a tensor network that evaluates the P-linear form of order n in $O(n^2 + n^{\mathrm{bw}(P)(\omega+\epsilon)/2})$ operations.*

Proof. Consider any branch decomposition of P of width $\mathrm{bw}(P)$. Rooting this decomposition arbitrarily by subdividing any edge and taking the newly added vertex as root, we obtain a binary rooted tree T with $|E(P)|$ leaves where the leaves are identified by the edges of P. Let r be the root of T and for each vertex u of T, let:

1. $C_u \subseteq V(P)$ (mnemonic: C for "crossing" or "cut") be the set of all vertices of P that appear both in some leaf in the subtree rooted at u, and in some leaf outside the subtree. By definition $|C_u| \le \mathrm{bw}(P)$ for all u, and $C_r = \emptyset$.

2. $D_u \subseteq V(P)$ (mnemonic: D for "done") be the set of all vertices of P that appear only in leaves in the subtree rooted at u, and not outside. Note that C_u and D_u form a partition of the set of all vertices appearing in some leaf in the subtree rooted at u, and that $D_r = V(P)$.

3. $E_u \subseteq E(P)$ be the set of all leaves of the subtree of T rooted at u (recall that each leaf of T corresponds to an edge). Note that $E_r = E(P)$.

4. $A_u \in \mathbb{F}^{[n]^{C_u}}$ be the following order-$|C_u|$ tensor of shape $n \times n \times \cdots \times n$ defined for a position $i \in [n]^{C_u}$ by
$$(A_u)_i = \sum_{j \in [n]^{D_u}} \prod_{S \in E_u} X^{(S)}_{(ij)_{|S}},$$
where $ij \in [n]^{C_u \cup D_u}$ is the Cartesian product of i and j.

Note that A_r equals the value of the P-linear form (3.14). Furthermore, for a leaf u of T corresponding to an edge $\{x_1, x_2\}$ of P, the tensor A_u is easy to compute: if both x_1 and x_2 have degree at least 2, then A_u simply equals the input tensor $X^{\{x_1,x_2\}}$. On the other hand if x_1 and/or x_2 has degree 1, then A_u is either a vector or a scalar, being either the row sums, column sums, or sum of all entries of $X^{\{x_1,x_2\}}$. Each of these is readily computed in $O(n^2)$ time by a contraction of $X^{\{x_1,x_2\}}$ with an appropriate tensor of ones.

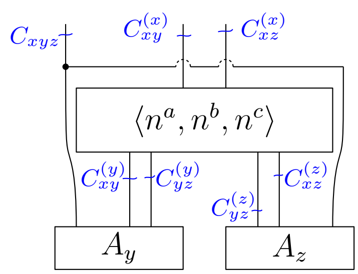
As we shall see below, for every non-leaf vertex x of T with children y and z, the tensor A_x equals the contraction of A_y and A_z. Thus, the desired result would follow if we can show that the contraction of two siblings A_y and A_z can be computed by a tensor network in $O(n^{\mathrm{bw}(P)(\omega+\epsilon)}/2)$ operations. We now proceed to establish this.

Partition C_x into $C_{xy} \cup C_{xz} \cup C_{xyz}$ where $C_{xyz} = C_x \cap C_y \cap C_z$, $C_{xy} = C_x \setminus C_z$ and $C_{xz} = C_x \setminus C_y$ (note that every element of C_x must appear in exactly one of C_y and C_z so these three sets indeed partition C_x). Symmetrically partition C_y into $C_{xy} \cup C_{yz} \cup C_{xyz}$ and C_z into $C_{xz} \cup C_{yz} \cup C_{xyz}$. Note that $C_{yz} \subseteq D_x$ and that the contraction of A_y and A_z at position (i, j, k) for $i \in [n]^{C_{xy}}$, $j \in [n]^{C_{xz}}$, $k \in [n]^{C_{xyz}}$ is exactly

$$\sum_{\ell \in [n]^{C_{yz}}} A_y(i,k,\ell) A_z(j,k,\ell) = A_x(i,j,k)$$

Let $a = |C_{xy}|$, $b = |C_{yz}|$, and $c = |C_{xz}|$. Split each mode in C_{xy} into two separate modes $C_{xy}^{(x)}$ and $C_{xy}^{(y)}$ where the former is used to index A_x and the latter is used to index A_y. Similarly split the modes in C_{xz} and C_{yz}, but *not* the modes in C_{xyz}.

The contraction of A_y and A_z can then be evaluated using rectangular matrix multiplication with the following tensor network.



(5.9)

By Lemma 5.5, this network can be evaluated in $O(n^{|C_{xyz}|} n^{\max(a+b, b+c, a+c)(\omega+\epsilon)/2})$ operations. Since $a+b+|C_{xyz}| = |C_y| \le \mathrm{bw}(P)$, $b+c+|C_{xyz}| = |C_z| \le \mathrm{bw}(P)$, and $a+c+|C_{xyz}| = |C_x| \le \mathrm{bw}(P)$, the number of operations used to contract A_y and A_z is bounded by $O(n^{\mathrm{bw}(P)(\omega+\epsilon)/2})$. □

5.4 Clique-counting forms

For counting v-cliques, the general upper bound for P-linear forms (Lemma 5.6) with $P = K_v$ gives a running time of $O(n^{(\omega+\epsilon)\lceil 2v/3 \rceil/2}) = O(n^{(\omega+\epsilon)\lfloor v/3 \rfloor + \frac{(\omega+\epsilon)}{2}(v \bmod 3)})$. In this section, we give a slightly improved tensor-network algorithm, matching the running time of the currently fastest known algorithm of Eisenbrand and Grandoni [44]. That this is possible is more or less immediate given that tensor networks can do fast rectangular matrix multiplication. It can also be viewed as a consequence of a more careful analysis of the proof of Lemma 5.6 where, towards the end, instead of using the generic upper bound of Lemma 5.5 for rectangular matrix multiplication, we use the precise shape of the bottleneck matrix multiplication needed for the v-clique case. For completeness, we give a more direct argument below.

For brevity, throughout this section we refer to the P-linear form for $P = K_v$ as the $\binom{v}{2}$-linear form.

Lemma 5.7. *For all constants $\epsilon > 0$ and $v \geq 3$, the $\binom{v}{2}$-linear form of order n can be evaluated by executing a tensor network in $O(n^{\beta+\epsilon})$ operations, where $\omega v/3 \leq \beta \leq \omega v/3 + (v \bmod 3)$ is the exponent of the arithmetic complexity of the $\langle n^{\lfloor v/3 \rfloor}, n^{\lfloor (v+1)/3 \rfloor}, n^{\lfloor (v+2)/3 \rfloor}\rangle$ matrix multiplication map.*

Proof. Let S_1, S_2, and S_3 be an arbitrary partition of the v vertices of K_v into sets of sizes $\lfloor v/3 \rfloor$, $\lfloor (v+1)/3 \rfloor$ and $\lfloor (v+2)/3 \rfloor$, respectively. Let $E_{ij} = E(S_i, S_i \cup S_j)$ be the edges inside S_i and between S_i and S_j. In particular, E_{12}, E_{23} and E_{31} form a partition of the edges of K_v.

We construct a network for evaluating the clique-counting form as follows. First, contract all inputs $X^{(ab)}$ with $ab \in E_{12}$ naïvely. This results in a tensor $Y^{(12)}$ with volume $n^{|S_1|+|S_2|}$ which we view as an $n^{|S_1|} \times n^{|S_2|}$ matrix where the rows are indexed by multisubsets R of $[n]$ of size $|S_1|$ and the columns by multisubsets C of size $|S_2|$; the entry at row R, column C is the 0-1 indicator of whether $R \cup C$ is a set of size $|S_1| + |S_2|$ where all vertices of R have edges to all other vertices in $R \cup C$. Contract all of E_{23} and E_{31} similarly, obtaining an $n^{|S_2|} \times n^{|S_3|}$ matrix $Y^{(23)}$ and an $n^{|S_3|} \times n^{|S_1|}$ matrix $Y^{(31)}$. The cost of creating $Y^{(ij)}$ is simply its volume $n^{|S_i|+|S_j|} \leq n^{\beta}$.

Next, perform the matrix multiplication $Z = Y^{(12)} \cdot Y^{(23)}$ in time $O(n^{\beta+\epsilon})$ using the same method as in the proof of Lemma 5.3. The matrix Z at row $R \in [n]^{|S_1|}$, column $C \in [n]^{|S_3|}$ counts the subsets $U \in [n]^{|S_2|}$ such that $W = R \cup C \cup U$ is a set of size $|S_1| + |S_2| + |S_3|$ (viewing R, C and U as multisubsets of $[n]$ in the obvious way) where all vertices in $R \cup U$ have edges to all other vertices in W.

Finally, contract Z with $Y^{(31)}$ (identifying the rows of Z with the columns of $Y^{(31)}$ and vice versa) in time $n^{|S_3|+|S_1|} \leq n^{\beta}$ to obtain the number of v-cliques in the input graph. □

5.5 Fast Fourier transforms and fast convolution

Let us next express fast Fourier transforms as tensor networks and their executions. We start by observing that the Cooley–Tukey [35] fast Fourier transform on $\mathbb{Z}_{2^k}$ can be implemented as a tensor network. We assume ρ is a primitive $(2^k)^{\text{th}}$ root of unity in the field $\mathbb{F}$.

Lemma 5.8. *The discrete Fourier transform for the Abelian group $\mathbb{Z}_{2^k}$ can be computed by executing a tensor network in $O(2^k k)$ operations.*

Proof. The case $k = 0$ is immediate so let us assume that $k \geq 1$. We construct a tensor network whose execution multiplies a vector $x \in \mathbb{F}^{[2^k]}$ with the DFT matrix Φ in Section 3.1 for $n = 2^k$ to yield the result $\Phi x \in \mathbb{F}^{[2^k]}$. Toward this end, let us write I_m for an $m \times m$ identity matrix,

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{5.10}$$

for the 2×2 Hadamard-Walsh matrix, and $R^{(k,j)} \in \mathbb{F}^{[2^k]}$ for the vector obtained by concatenating 2^j copies of the vector

$$[\underbrace{1,1,\ldots,1}_{2^{k-1-j}},\underbrace{\rho^{2^j\cdot 0},\rho^{2^j\cdot 1},\ldots,\rho^{2^j\cdot(2^{k-1-j}-1)}}_{2^{k-1-j}}]$$

for $j = 0,1,\ldots,k-2$. We can decompose Φ into a sequence of $2^k \times 2^k$ matrices (see, e. g., [97])

$$\Phi = P^{(k)}B^{(k,k-1)}T^{(k,k-2)}B^{(k,k-2)}\cdots T^{(k,1)}B^{(k,1)}T^{(k,0)}B^{(k,0)} \tag{5.11}$$

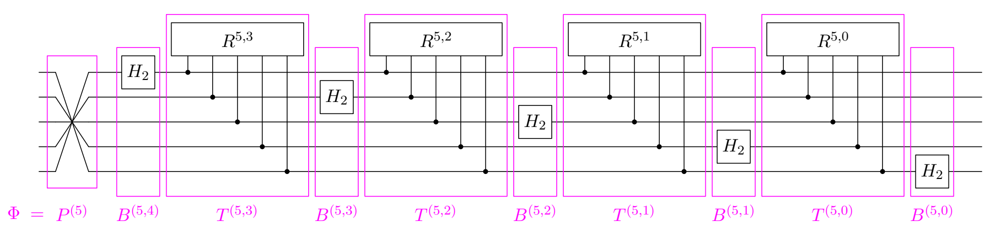
consisting of alternating butterfly matrices

$$B^{(k,j)} = \underbrace{I_2 \otimes I_2 \otimes \cdots \otimes I_2}_{j} \otimes H_2 \otimes \underbrace{I_2 \otimes I_2 \otimes \cdots \otimes I_2}_{k-j-1}$$

and diagonal twiddle matrices

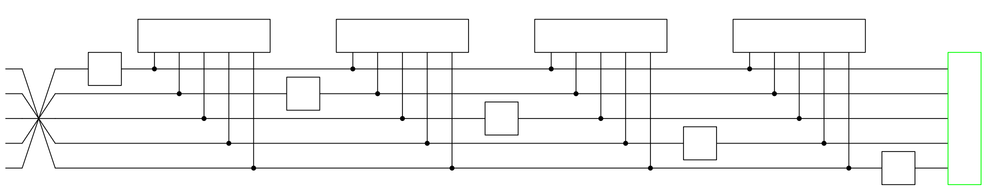
$$T^{(k,j)} = \operatorname{diag}(R^{(k,j)})$$

followed by a permutation matrix $P^{(k)}$ that permutes the indices in $[2^k]$ viewed as k-bit strings by reversing the bit-order. Since only the $(j+1)^{\text{th}}$ Kronecker component in the butterfly matrix $B^{(k,j)}$ is a nonidentity matrix, and multiplication of a vector with the diagonal twiddle matrix $T^{(k,j)}$ corresponds to pointwise (Hadamard) multiplication with the vector $R^{(k,j)}$ on the diagonal, we observe that the sequence (5.11) can be represented as a tensor network D^* as depicted below (for $k = 5$) so that all the modes have length 2.



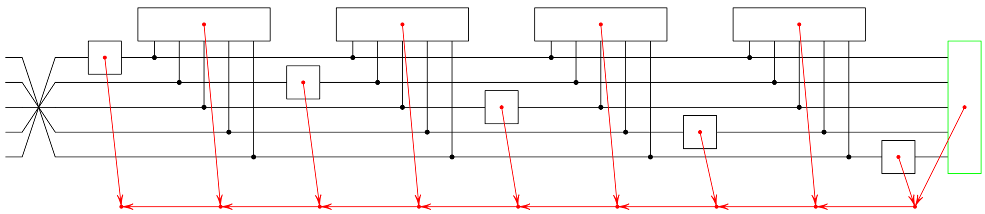
(5.12)

We can now connect the network (5.12) to a data vector $x \in \mathbb{F}^{[2^k]}$ to obtain the network D below:



(5.13)

Below we depict in red an execution tree $\mathcal{T}_D$ with cost 2^{k+1} for the network (5.13). Observe that the mode permutation (multiplication with the matrix $P^{(5)}$) is not part of the execution since the permutation amounts to merely rearranging the modes.



(5.14)

Since the network has $O(k)$ tensors, the execution (5.14) can be carried out in $O(2^k k)$ operations in $\mathbb{F}$. □

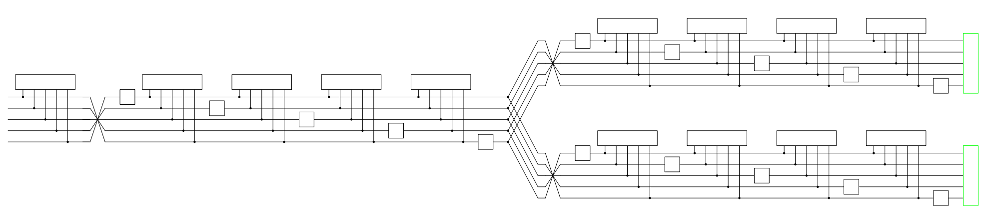
Lemma 5.9. *The discrete Fourier transform for the elementary Abelian group $\mathbb{Z}_2^k$ can be computed by executing a tensor network in $O(2^k k)$ operations.*

Proof. This proof is analogous to the proof of Lemma 5.8 but omits the twiddle matrices $T^{(k,j)}$ and the final permutation matrix $P^{(k)}$ from the decomposition of the tensor network. □

The following corollary is immediate from Lemma 5.8.

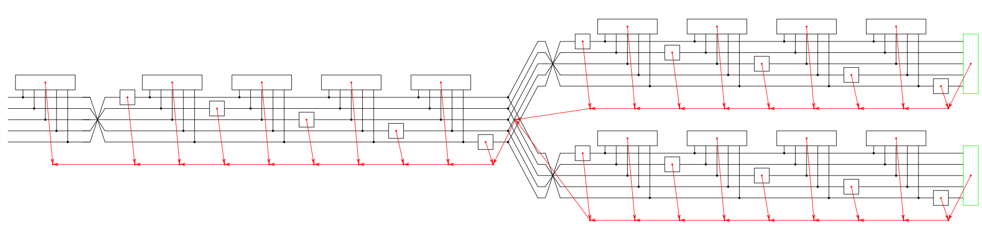
Lemma 5.10. *The group algebra products on $\mathbb{F}[\mathbb{Z}_{2^k}]$ and $\mathbb{F}[\mathbb{Z}_2^k]$ can be computed by executing a tensor network in $O(2^k k)$ operations whenever 2 is a unit in $\mathbb{F}$.*

Proof. We start with $\mathbb{F}[\mathbb{Z}_{2^k}]$. Let $f, g \in \mathbb{F}^{[2^k]}$ be two vectors given as input. Our task is to compute the $\mathbb{Z}_{2^k}$-convolution $f * g \in \mathbb{F}^{[2^k]}$. The case $k = 0$ is immediate so suppose that $k \geq 1$. Recalling that $f * g = \Phi^{-1}((\Phi f) \cdot (\Phi g))$, where "$\cdot$" denotes an elementwise (Hadamard) product of two vectors of length 2^k, let us construct a tensor network as follows. First take the FFT of f and g using Lemma 5.8, then multiply the resulting vectors elementwise, and finally take the inverse FFT by replacing ρ with ρ^{-1} in Lemma 5.8 and multiplying with the diagonal matrix $\frac{1}{2^k} I_{2^k}$. Below we display (for $k = 5$) the resulting network D that executes to $f * g$.



(5.15)

The execution of the network proceeds from right to left analogously to (5.14).



(5.16)

The cost of this execution is 2^{k+1}. Since the network has $O(k)$ tensors, the execution (5.16) can be carried out in $O(2^k k)$ operations in $\mathbb{F}$.

The case of $\mathbb{F}[\mathbb{Z}_2^k]$ is analogous but replacing Lemma 5.8 with Lemma 5.9 and modifying the networks accordingly. □

5.6 Yates' algorithm

A particularly simple use case for Theorem 5.1 occurs when $A : \mathbb{F}^{[s]} \to \mathbb{F}^{[t]}$ is a linear map. It is immediate that we can realize A with a two-vertex network and an execution tree that has amortized cost $\max(s,t)$ and cost st.

Then, Theorem 5.1 immediately implies that we can evaluate $A^{\otimes k} : \mathbb{F}^{[s]^k} \to \mathbb{F}^{[t]}$ using a tensor network with cost $\max(s^{k+2}, t^{k+2})$ and $O(k)$ vertices. In particular, the network can be executed in $O(\max(s^{k+2}, t^{k+2})k)$ operations in $\mathbb{F}$. This network in essence realizes Yates' algorithm [104] for multiplying an s^k-length vector with the k^{th} Kronecker power of an $s \times t$ matrix to obtain t^k-length vector.

Applying the previous observation to $A = H_2$ in (5.10) with $s = t = 2$, we obtain Lemma 5.9 as an immediate corollary. Similarly, other choices of 2×2 matrices yield the algebraic core of currently the fastest known algorithms for problems such as graph coloring and its generalizations such as computing the Tutte polynomial of a graph [16, 17, 19]. In particular, the pair of mutually inverse 2×2 matrices

$$Z_2 = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \qquad M_2 = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$$

yield, as $Z_2^{\otimes k}$ and $M_2^{\otimes k}$, the zeta and Möbius tranforms for the lattice $(\{0,1\}^k, \subseteq, \cap, \cup)$ of all subsets of a k-element set, partially ordered by subset inclusion. Theorem 5.1 yields immediately the standard algorithms (normally developed via Yates's algorithm) for the fast zeta and the fast Möbius transforms via tensor networks. These networks can then be combined as in the proof of Lemma 5.10 to yield the associated bilinear convolution maps (multiplication maps in the semigroup algebra $\mathbb{F}[(\{0,1\}^k, \subseteq, \cap, \cup)]$) to realize these maps in $O(2^k k)$ operations. We omit the details due to similarity with Lemma 5.10.

5.7 Kruskal operator

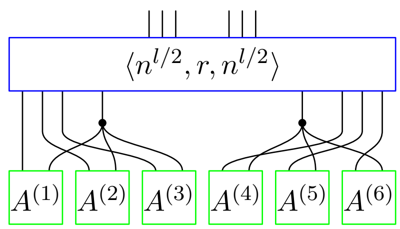
We proceed to implement the Kruskal operator by taking Kronecker products and reducing to fast rectangular matrix multiplication. Let us first proceed via Lemma 5.4 relying on square matrix multiplication.

Lemma 5.11. *For all constants $\epsilon > 0$ and $\ell = 1, 2, \ldots$ it holds that we may evaluate the Kruskal operator of ℓ matrices of shape $n \times r$ by executing a tensor network in*

1. $O(n^{\lceil \ell/2 \rceil(\omega+\epsilon-1)} r)$ *operations when* $r \geq n^{\lceil \ell/2 \rceil}$, *and*
2. $O(n^{2\lceil \ell/2 \rceil} r^{\omega+\epsilon-2})$ *operations when* $r \leq n^{\lceil \ell/2 \rceil}$.

Proof. Fix an $\epsilon > 0$ and let α, β, γ be three 3-tensors of shape $(c \times c) \times d$ for constants $b = c$ and d as in the proof of Lemma 5.3. Let $A^{(1)}, A^{(2)}, \ldots, A^{(\ell)} \in \mathbb{F}^{[n]\times[r]}$ be given as input. We construct a tensor network that computes the output Y of the Kruskal operator (3.8).

Without loss of generality we may assume that ℓ is even by introducing a matrix $A^{(\ell+1)} \in \mathbb{F}^{[n]\times[r]}$ filled with 1-entries and setting $\ell \leftarrow \ell + 1$. By inserting rows and columns with zero-entries as necessary, we can assume that both n and r are positive integer powers of c. The key idea is now to take Kronecker products of the matrices $A^{(1)}, A^{(2)}, \ldots, A^{(\ell/2)}$ and $A^{(\ell/2+1)}, A^{(\ell/2+2)}, \ldots, A^{(\ell)}$ *in the vertical dimension only* to obtain two matrices $B \in \mathbb{F}^{[n^{\ell/2}]\times[r]}$ and $C \in \mathbb{F}^{[n^{\ell/2}]\times[r]}$, respectively. We then multiply B and the transpose of C using fast rectangular matrix multiplication from Lemma 5.4 to obtain $Y \in \mathbb{F}^{[n^{\ell/2}]\times[n^{\ell/2}]}$. It is immediate from Lemma 5.4 that this results in the operation counts claimed in (i) and (ii) as long as we can realize the idea using tensor networks. Taking Kronecker product in the vertical dimension only can be realized by joining all the *horizontal* dimensions to a common mode, which becomes the inner mode for matrix multiplication. The resulting network is depicted below (for $\ell = 6$), where either (5.7) or (5.8) is used for the subnetwork indicated with $\langle n^{\ell/2}, r, n^{\ell/2}\rangle$ depending on whether (i) or (ii) holds.



(5.17)

In drawing (5.17) we have made two abstractions. First, each drawn mode in fact is a bundle of modes, each of length c. Second, each mode in a bundle that is incident to one of the input matrices $A^{(1)}, A^{(2)}, \ldots, A^{(\ell)}$ is in fact subdivided by inserting an identity matrix I_c just before the incidence to the input matrix. The network (5.17) is executed first by contracting the (zero-cost) identity matrices I_c, then contracting $A^{(1)}, A^{(2)}, \ldots, A^{(\ell/2)}$ and $A^{(\ell/2+1)}, A^{(\ell/2+2)}, \ldots, A^{(\ell)}$, and finally proceeding to execute the subnetwork $\langle n^{\ell/2}, r, n^{\ell/2}\rangle$ as in Lemma 5.4. □

The next lemma gives a sharper conclusion using the rectangular matrix multiplication exponents $\omega(h)$ when r is assumed to grow polynomially as a function of n.

Lemma 5.12. *For all constants $\epsilon > 0$, $\rho > 0$, and $\ell = 1, 2, \ldots$ it holds that we may evaluate the Kruskal operator of ℓ matrices of shape $n \times r$, where $r = \lfloor n^{\rho} \rfloor$, by executing a tensor network in $O\big(n^{\lceil \ell/2\rceil \omega\left(\frac{\rho}{\lceil \ell/2\rceil}\right)+\epsilon}\big)$ operations.*

Proof. Proceed as in the proof of Lemma 5.11, but implement the subnetwork for $\langle n^{\lceil \ell/2\rceil}, r, n^{\lceil \ell/2\rceil}\rangle$ as in Lemma 5.3. More precisely, take $N = n^{\lceil \ell/2\rceil}$, $h = \rho/\lceil \ell/2\rceil$, and observe that $\langle N, \lfloor N^h \rfloor, N\rangle = \langle n^{\lceil \ell/2\rceil}, r, n^{\lceil \ell/2\rceil}\rangle$. □

To our knowledge, Lemmas 5.11 and 5.12 capture the state of the art for computing the Kruskal operator.

5.8 The permanent

The following lemma observes that essentially the fastest known algorithm for the permanent, namely Ryser's algorithm [89], can be realized as a tensor network. Here by essentially fastest known we mean the base of the exponential running time. Sub-exponential speed-ups to Ryser's algorithm are known, see, e. g., Björklund [15].

Lemma 5.13. *The permanent of an $n \times n$ matrix can be computed by executing a tensor network in $O(2^n n)$ operations.*

Proof. We observe that Ryser's algorithm [89] for the permanent (3.1), namely the inclusion–exclusion expression

$$\operatorname{per} A = \sum_{S \subseteq [n]} (-1)^{n-|S|} \prod_{i \in [n]} \sum_{j \in S} a_{ij}$$

is implementable with a star-shaped tensor network consisting of n matrices of shape $2^n \times n$ joined together by a common mode (of length 2^n), with the n modes of length n being the boundary of the network. Each of the n matrices consists of the $\{0, 1\}$-valued incidence vectors of the 2^n subsets $S \subseteq [n]$, with one of the matrices arbitrarily selected to contain signed rows determined by $(-1)^{n-|S|}$. The input to the network consists of n vectors of length n, namely the rows of the $n \times n$ input matrix A. The network is executed by first executing the n matrix-vector multiplications, and then contracting the resulting n vectors of length 2^n until the scalar per A remains. □

6 A lower bound for the cost of a multilinear map

In this section, we prove a general lower bound on the cost of evaluating a multilinear map using tensor networks, as defined in Section 4.5. The lower bound is expressed in terms of the *socket-width* of a multilinear map, which we now proceed to define.

Let $A : \mathbb{F}^{J(E_1)} \times \mathbb{F}^{J(E_2)} \times \cdots \times \mathbb{F}^{J(E_\ell)} \to \mathbb{F}^{J(E')}$ be an ℓ-linear map. A *socket-tree* of A is a tree $\mathcal{T}_S$ whose $\ell + 1$ leaf vertices are the sockets $E_1, E_2, \ldots, E_\ell, E'$ of A and whose internal vertices all have degree exactly 3. Associate with each edge $e = \{x_R, x_C\}$ of $\mathcal{T}_S$ the two subtrees $\mathcal{T}_S(x_R, e)$ and $\mathcal{T}_S(x_C, e)$ obtained by removing e, where $\mathcal{T}_S(x_R, e)$ is the subtree containing x_R and $\mathcal{T}_S(x_C, e)$ is the subtree containing x_C. Let $L(x_R, e)$ be the set of leaves in $\mathcal{T}_S(x_R, e)$ and let $L(x_C, e)$ be the set of leaves in $\mathcal{T}_S(x_C, e)$.

The sets $L(x_R, e)$ and $L(x_C, e)$ are both nonempty and together partition the set of sockets. Consider the flattening $M(\mathcal{T}_S, e)$ of the tensor $T(A)$ such that the modes in $L(x_R, e)$ index the rows and the modes in $L(x_C, e)$ index the columns of $M(\mathcal{T}_S, e)$. The *width* of $\mathcal{T}_S$ at e is the rank of $M(\mathcal{T}_S, e)$, and the *width* of $\mathcal{T}_S$ is $w(\mathcal{T}_S) = \max_{e \in E(\mathcal{T}_S)} \operatorname{rk}(M(\mathcal{T}_S, e))$.

Let us write $\mathscr{S}(A)$ for the set of all socket-trees of the multilinear form A. We define the *socket-width* of A to be $w(A) = \min_{\mathcal{T}_S \in \mathscr{S}(A)} w(\mathcal{T}_S)$.

The rest of this section is devoted to proving Theorem 1.4:

Theorem 1.4. *For every multilinear map A, it holds that $c(A) \geq w(A)$.*

First, we prove that without loss of generality, we may restrict attention to forms rather than general maps.

Claim 6.1. *For any multilinear map A, it holds that $c(A) \geq c(F(A))$.*

Proof. We observe that A and $F(A)$ satisfy $\hat{T}(A) = \hat{T}(F(A))$. Any network $D \in \mathscr{D}(A)$ can be modified to a network $D' \in \mathscr{D}(F(A))$ by attaching a tensor $X' \in \mathbb{F}^{J(E')}$ to the boundary of D. Let $D \in \mathscr{D}(A)$ be such that $c(D) = c(A)$. The minimum-cost execution of D, followed by contracting $T(D)$ and X', is an execution of D'. Its cost is $c(A)$, since the cost of contracting of $T(D)$ and X' is $\prod_{e \in B(D)} |J(e)|$ and $\prod_{e \in B(D)} |J(e)| \leq c(A)$, because the last step of the minimum-cost execution of D contracts a set W with all modes $e \in B(D)$ incident to W. Thus, $c(A) \geq c(F(A))$. $\square$

Furthermore, $w(A) = w(F(A))$ for every multilinear map A, since $w(A)$ only depends on the tensor $T(A)$, but not on which of its coordinates (if any) is the output. Thus it suffices to prove Theorem 1.4 for multilinear forms, which we now proceed to do.

Lemma 6.2. *For any multilinear form F, it holds that $c(F) \geq w(F)$.*

Proof. Let $D \in \mathscr{D}(F)$ be such that $c(D) = c(F)$. It is a tensor network with empty boundary and a socket vertex $S_i \in V(D)$ for each input socket E_i, where $i = 1, 2, \ldots, \ell$. Its tensor is $T(D) = F(X^{(1)}, X^{(2)}, \ldots, X^{(\ell)})$ where $X^{(i)} = T(S_i)$ for $i = 1, 2, \ldots, \ell$.

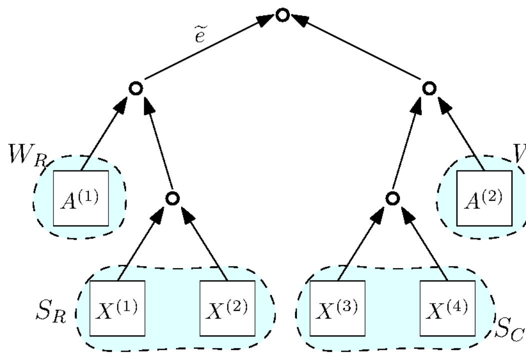
By Lemma 4.1, a minimum-cost execution of D can be represented by a rooted binary tree $\mathcal{T}_D$, where the set of leaves of $\mathcal{T}_D$ are $V(D)$ and each inner vertex represents the vertex obtained by contracting its two children. Let $\mathcal{T}_S$ be the unique socket-tree of F that is obtained as a topological minor of $\mathcal{T}_D$. Slightly abusing the notation, we assume that the leaves of $\mathcal{T}_S$ are the socket vertices $S_1, S_2, \ldots, S_\ell$ instead of the sockets $E_1, E_2, \ldots, E_\ell$. To establish the lemma, it suffices to show that $\mathcal{T}_D$ has cost at least $w(\mathcal{T}_S)$, since $w(\mathcal{T}_S) \geq w(F)$.

Let $e = \{x_R, x_C\} \in E(\mathcal{T}_S)$ be an edge of the socket tree $\mathcal{T}_S$ with $\mathrm{rk}(M(\mathcal{T}_S, e)) = w(\mathcal{T}_S)$, and let $\widetilde{e}$ be an edge of the execution tree $\mathcal{T}_D$ in the subdivision of e appearing in $\mathcal{T}_D$. Without loss of generality we may assume that $\widetilde{e}$ is directed from the part of $\mathcal{T}_D$ corresponding to x_R towards the part corresponding to x_C (if not, simply switch names of x_R and x_C). Define $S_R = L(x_R, e)$ and $S_C = L(x_C, e)$. Let $W_R \subseteq V(D)$ be the set of non-socket vertices of D that appear on the same side of $\widetilde{e}$ in $\mathcal{T}_D$ with socket vertices S_R and let W_C be the set of remaining non-socket vertices of D. See Figure 1 for an illustration of all these definitions. Finally, let $D' = D/S_R/S_C/W_R/W_C$ be the result of contracting each of these four sets of vertices of D. For notational convenience, we identify the four vertices of the new network with the four subsets S_R, S_C, W_R, W_C.

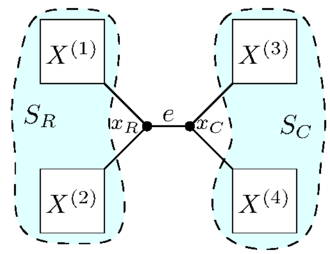
Now, the tensor $P = T(D'[W_R \cup S_R])$ appears as an intermediate result in the execution $\mathcal{T}_D$,[8] hence the volume of P is a lower bound on the cost of $\mathcal{T}_D$.

We group the modes of D' incident on S_R or W_R as shown in Figure 2: E_{SW} are all modes in D' incident exactly upon S_R and W_R, E_{WC} are all modes incident on W_R but not on S_R, E_{SC}

[8]Note that this follows from our choice that the R part of D is below $\tilde{e}$ in the execution tree, and that the same is not necessarily true for the tensor $T(D'[W_C \cup S_C])$. E.g. if in Figure 1 we had instead taken $\tilde{e}$ to be the edge further down which joins $X^{(1)}$ and $X^{(2)}$ with $A^{(1)}$ then $T(D'[W_C \cup S_C])$ would be the contraction of $A^{(1)}$, $A^{(2)}$, $X^{(3)}$ and $X^{(4)}$, which is not an intermediate result in the execution of $\mathcal{T}_D$.



(a) Example of a possible execution tree $\mathcal{T}_D$. Given the choice of e in the corresponding socket tree $\mathcal{T}_S$ shown on the right there are four possible choices of $\widetilde{e}$.



(b) The corresponding socket tree $\mathcal{T}_S$. The exact choice of $\widetilde{e}$ in $\mathcal{T}_D$ determines which part of the cut is the x_R part, and which is the x_C part.

Figure 1: Illustration of the notation used for the execution and socket trees.

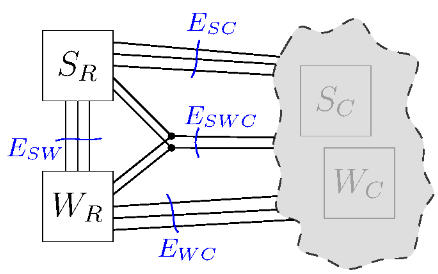


Figure 2: Illustration of D'. We group the modes of D' based on how they connect S_R, S_C, and the "C part" of D'.

are all modes incident on S_R but not W_R, and finally E_{SWC} are all modes incident upon S_R, W_R, and at least one of S_C or W_C. Write $E_S = E_{SW} \cup E_{SC} \cup E_{SWC}$ for the modes incident on S_R, and similarly $E_C = E_{WC} \cup E_{SC} \cup E_{SWC}$ for all modes incident upon at least one of S_R or W_R, and at least one of S_C or W_C. Note that $|J(E_C)|$ is precisely the volume of P which we aim to lower bound.

Define a matrix $A \in \mathbb{F}^{J(E_S)} \times \mathbb{F}^{J(E_C)}$ as follows. We identify its row indices $i \in J(E_S)$ as being triples $i = (i_{SW}, i_{SC}, i_{SWC}) \in J(E_{SW}) \times J(E_{SC}) \times J(E_{SWC})$ and similarly its column indices $j \in J(E_C)$ are triples $j = (j_{SC}, j_{WC}, j_{SWC}) \in J(E_{SC}) \times J(E_{WC}) \times J(E_{SWC})$. Then the entries of A are

$$A_{(i_{SW}, i_{SC}, i_{SWC}),(j_{SC}, j_{WC}, j_{SWC})} = \begin{cases} T(D'[W_R])_{i_{SW}, j_{WC}, j_{SWC}} & \text{if } i_{SC} = j_{SC} \wedge i_{SWC} = j_{SWC}, \\ 0 & \text{otherwise,} \end{cases}$$

In the case when $E_S = E_{SW}$ (i. e., all modes incident on S_R connect only to W_R), A is simply a flattening of $T(D'[W_R])$. Recall that $T(D'[S_R]) \in \prod_{e \in E_S} \mathbb{F}^{J(e)}$. Then for every

$j = (j_{SC}, j_{WC}, j_{SWC}) \in J(E_C)$, we have

$$\begin{aligned}\sum_{i \in J(E_S)} A_{i,j} T(D'[S_R])_i &= \sum_{i_{SW} \in J(E_{SW})} A_{(i_{SW}, j_{SC}, j_{SWC}), j} T(D'[S_R])_{i_{SW}, j_{SC}, j_{SWC}} \\ &= \sum_{i_{SW}} T(D'[W_R])_{i_{SW}, j_{WC}, j_{SWC}} T(D'[S_R])_{i_{SW}, j_{SC}, j_{SWC}} \\ &= P_{j_{SC}, j_{WC}, j_{SWC}} = P_j\end{aligned}$$

(recall that P is the contraction of $T(D'[W_R])$ and $T(D'[S_R])$). Viewing $T(D'[S_R])$ as a row vector in $\mathbb{F}^{J(E_S)}$ we see that P is simply the vector-matrix product $P = T(D'[S_R]) \cdot A \in \mathbb{F}^{J(E_C)}$.

Symmetrically, for the other half of D', we can write $Q = T(D'[W_C \cup S_C])$ as a matrix-vector product $Q = B \cdot T(D'[S_C]) \in \mathbb{F}^{J(E_C)}$ where B is a matrix corresponding to $T(D'[W_S])$ analogously to how A corresponds to $T(D'[W_R])$.

Thus we have $T(D) = T(D'[S_R]) \cdot A \cdot B \cdot T(D'[S_C])$. Recall that for each socket vertex S_i in the original network D, we have $T(S_i) = X^{(i)}$. Denoting $X_R = T(D'[S_R])$ and $X_C = T(D'[S_C])$, we get $X_R = \bigotimes_{S_i \in S_R} X^{(i)}$ and $X_C = \bigotimes_{S_i \in S_C} X^{(i)}$.[9] Hence

$$F(X^{(1)}, X^{(2)}, \ldots, X^{(\ell)}) = X_R \cdot A \cdot B \cdot X_C.$$

It follows that $A \cdot B$ is the flattening of $T(F)$ to a matrix with rows indexed by the sockets in S_R and columns indexed by the sockets in S_C. But this flattening is precisely the matrix $M(\mathcal{T}_S, e)$, implying that $|J(E_{C})| \geq \operatorname{rk}(M(\mathcal{T}_S, e)) = w(\mathcal{T}_S)$, as desired. □

7 Lower bounds for socket-width

In this section we establish lower bounds on socket-width for concrete maps.

7.1 Determinant and permanent

We now prove lower bounds for the socket width of the determinant and permanent. Let us start with the following trivial observation.

Claim 7.1. *For any socket tree $\mathcal{T}_S$ with $n \geq 2$ leaves, there is an edge $e = \{x_R, x_C\}$ such that $n/3 \leq |L(x_R, e)| < 2n/3$.*

Proof. Consider all edges $e = \{x_R, x_C\}$ such that $|L(x_R, e)| \geq n/3$. At least one such edge certainly exists since $n \geq 2$. Indeed, an edge $e = \{x_R, x_C\}$ incident to a leaf x_C has $|L(x_R, e)| = n - 1$ leaves. Among these, choose an edge such that $\mathcal{T}_S(x_R, e)$ is of minimal size. Assume for contradiction that $|L(x_R, e)| \geq 2n/3$. But then one of the two subtrees of x_R in $\mathcal{T}_S(x_R, e)$ must have at least $n/3$ leaves, and since they are smaller than $\mathcal{T}_S(x_R, e)$, this contradicts the minimality of $\mathcal{T}_S(x_R, e)$. □

[9] These identities use the fact that D is derived from a *non-degenerate* network D^* for $\hat{T}(F)$. In particular, every mode in the network D is incident upon at least one non-socket vertex, hence all modes incident upon S_R are boundary modes in $D'[S_R]$, and similarly for S_C.

Lemma 7.2. *For every positive integer $n \geq 2$, the socket-width of both the determinant and the permanent of an $n \times n$ matrix is at least $\binom{n}{\lceil n/3 \rceil}$.*

Proof. Let $\mathcal{T}_S$ be a socket tree for the permanent, let $e = \{x_R, x_C\}$ be an arbitrary edge in $\mathcal{T}_S$, and let $k = |L(x_R, e)|$ be the number of leaves in $\mathcal{T}_S(x_R, e)$. We now show that $\mathrm{rk}(M(\mathcal{T}_S, e)) \geq \binom{n}{k}$.

Recall that the sockets of $\mathcal{T}_S$ are the n rows of the input matrix. Without loss of generality, number the rows so that the leaves of $\mathcal{T}_S(x_R, e)$ are rows 1 to k, and the leaves of $\mathcal{T}_S(x_C, e)$ are rows $k+1$ to n. The rows of $M(\mathcal{T}_S, e)$ are indexed by a tuple $j_R = (i_1, \ldots, i_k) \in [n]^k$, and the columns are indexed by a tuple $j_C = (i_{k+1}, \ldots, i_n) \in [n]^{n-k}$. The entry of $M(\mathcal{T}_S, e)$ at position (j_R, j_C) is 1 if $(i_1, i_2, \ldots, i_n)$ is a permutation. For any set $U \in \binom{[n]}{k}$, let $u_1 < u_2 < \ldots < u_k$ be the elements of U in ascending order and define $j_R(U) = (u_1, \ldots, u_k)$; let $u_{k+1} < u_{k+2} < \ldots < u_n$ be the elements of $[n] \setminus U$ in ascending order and define $j_C(U) = (u_{k+1}, u_{k+2}, \ldots, u_n)$.

For $U_1, U_2 \in \binom{[n]}{k}$, the entry of $M(\mathcal{T}_S, e)$ at position $(j_R(U_1), j_C(U_2))$ equals 1 if $U_1 = U_2$, and 0 otherwise. This induces a $\binom{n}{k} \times \binom{n}{k}$ identity submatrix of $M(\mathcal{T}_S, e)$, implying $\mathrm{rk}(M(\mathcal{T}_S, e)) \geq \binom{n}{k}$.

By Claim 7.1, $\mathcal{T}_S$ has an edge e with $n/3 \leq k < 2n/3$. For that edge, $\mathrm{rk}(M(\mathcal{T}_S, e)) \geq \binom{n}{k} \geq \binom{n}{\lceil n/3 \rceil}$, which completes the proof for the permanent.

For the determinant, the only change is that some entries of $M(\mathcal{T}_S, e)$ become -1 instead of 1, but this only changes the identified submatrix from an identity matrix to a diagonal matrix and in particular does not change its rank. ◻

The preceding proof is similar to a lower bound by Nisan ([76], Lemma 2) used to obtain lower bounds for algebraic branching programs. But the lower bounds obtained there can be made as sharp as $\Omega(2^n)$ whereas in our setting, we cannot rule out the possibility of a tensor network that avoids splitting the n variables in two approximately equal size parts. This means that the best we can obtain with our current method is $\binom{n}{\lceil n/3 \rceil}$ instead of $\binom{n}{n/2}$.

7.2 P-linear forms

Suppose $\mathcal{T}_S$ is a socket tree for a P-linear form for a k-uniform hypergraph P on v vertices. Recall that the sockets of this form correspond to the elements of $E(P) \subseteq \binom{[v]}{k}$. Given an edge $e \in E(\mathcal{T}_S)$ and a vertex $x \in V(\mathcal{T}_S)$, we write

$$U(x, e) = \bigcup_{S \in L(x,e)} S \subseteq [v].$$

Claim 7.3. *Let $\mathcal{T}_S$ be a socket tree for the P-linear form of order n. Let $e = \{x_R, x_C\} \in E(\mathcal{T}_S)$ and suppose $|U(x_C, e) \cap U(x_R, e)| = u$. Then the socket width of $\mathcal{T}_S$ is at least $w(\mathcal{T}_S) \geq n^u$.*

Proof. Let $m = n^u$. We show that $\mathrm{rk}(M(\mathcal{T}_S, e)) \geq m$ by identifying an $m \times m$ identity submatrix of $M(\mathcal{T}_S, e)$.

Define $E_R = \{\, (S, i) \mid S \in L(x_R, e), i \in S \,\}$ to be the modes contained in the sockets on the x_R side of e, and analogously $E_C = \{\, (S, i) \mid S \in L(x_C, e), i \in S \,\}$.

The rows of $M(\mathcal{T}_S, e)$ are indexed by $J(E_R)$ and the columns are indexed by $J(E_C)$. We will consider an $m \times m$ submatrix of $M(\mathcal{T}_S, e)$ whose rows and columns are indexed by the elements

$\sigma \in \prod_{i \in A}[n_i]$. More specifically, each row of the submatrix is indexed by j_R where $j_R|_{(S,i)} = \sigma|_i$ for $i \in A$, and $j_R|_{(S,i)} = 1$ for $i \notin A$. Each column is indexed by j_C where $j_C|_{(S,i)} = \sigma|_i$ for $i \in A$ and $j_C|_{(S,i)} = 1$ for $i \notin A$.

The value of the submatrix at position (j_R, j_C) is 1 if there exists $\sigma \in \prod_{i \in A}[n_i]$ such that $j_R|_{(S,i)} = \sigma|_i$ and $j_C|_{(S,i)} = \sigma|_i$ for all $i \in A$ and 0 otherwise. We obtain an $m \times m$ identity matrix as desired. □

From this claim, the branchwidth-based lower bound is immediate.

Lemma 7.4. *For any hypergraph P, the socket width of the P-linear form of order n is at least $n^{\mathrm{bw}(P)}$.*

Proof. Any socket tree for a P-linear form can be directly viewed as a branch decomposition of P. Thus, by definition every socket tree for the form has an edge $e = \{x_R, x_C\}$ where $|U(x_C, e) \cap U(x_R, e)| \geq \mathrm{bw}(P)$, and the lemma now follows from Claim 7.3. □

For counting homomorphisms of hypergraph cliques, that is, the P-linear form for the complete k-uniform hypergraph $P = \binom{[v]}{k}$ on v vertices, we need the following simple lower bound on the branchwidth of complete hypergraphs (which is most likely known, but we are not aware of a reference).

Lemma 7.5. *For $v > k \geq 3$, the complete k-uniform hypergraph on v vertices has branchwidth v.*

Proof. Let T be an arbitrary branch decomposition of the hypergraph. We claim that there always exists an edge $e^* = \{x_R^*, x_C^*\}$ such that $U(x_R^*, e^*) = U(x_C^*, e^*) = [v]$, implying that the width of the decomposition is at least v. Towards establishing this claim, we first observe that for every edge $e = \{x_R, x_C\}$, at least one of $U(x_R, e)$ and $U(x_C, e)$ is equal to $[v]$. Indeed, assume towards contradiction that there is an edge $e = \{x_R, x_C\}$ with $i_1 \notin U(x_R, e)$ and $i_2 \notin U(x_C, e)$ for some not necessarily distinct $i_1, i_2 \in [v]$. Since $k \geq 2$, there exists an edge $S \supseteq \{i_1, i_2\}$, and that edge must appear in either $\mathcal{T}_S(x_R, e)$ or $\mathcal{T}_S(x_C, e)$, violating the assumption.

Now suppose for contradiction that the claim about the existence of e^* does not hold. Direct each edge towards the subtree which covers $[v]$ (which, by the observation above, always exists). For example, if $U(x_R, e) = [v]$ and $U(x_C, e) \neq [v]$ then e is directed towards x_R.

Since a tree is acyclic, there must be some vertex x^* such that all edges incident to x^* are directed towards x^*. The vertex x^* cannot be a leaf, because the subtree consisting of a leaf contains a single edge, which covers $k < v$ elements. Thus x^* has degree 3. Let the three edges incident to x^* be $e_1 = \{x^*, y_1\}$, $e_2 = \{x^*, y_2\}$ and $e_3 = \{x^*, y_3\}$. Since all the edges are directed towards x^* there are $i_1, i_2, i_3 \in [v]$ such that $i_1 \notin S(y_1, e_1)$, $i_2 \notin S(y_2, e_2)$ and $i_3 \notin S(y_3, e_3)$. Since $k \geq 3$ there exists an edge $S \supseteq \{i_1, i_2, i_3\}$. But now the edge S cannot appear in any of the subtrees rooted at y_1, y_2, or y_3. Since these three subtrees together cover all leaves, this yields the desired contradiction and we conclude that there must exist an edge $e^* = \{x_R^*, x_C^*\}$ such that $U(x_R^*, e^*) = U(x_C^*, e^*) = [v]$ and therefore the branchwidth is v. □

7.3 Kruskal operator

We say that an ℓ-linear Kruskal operator is *n-uniform* if the lengths of the modes satisfy $n = n_1 = n_2 = \cdots = n_\ell$.

Lemma 7.6. *For positive integers n and r, the socket-width of an n-uniform ℓ-linear Kruskal operator is at least* $\max\left(n^\ell, n^{\lceil \ell/2 \rceil} r\right)$.

Proof. Let $\mathcal{T}_S$ be an arbitrary socket tree for the operator. One of the leaves is the output socket representing the tensor B of shape $n \times n \times \cdots \times n$ (ℓ times) that is obtained by applying the Kruskal operator to the input matrices $A^{(i)}$ of shape $n \times r$, where $1 \le i \le \ell$.

Consider the neighbor x of the output socket. It has three neighbors: the output socket leaf, and two other vertices x_R and x_C. Either the subtree rooted at x_R or the one rooted at x_C must contain $u \ge \lceil \ell/2 \rceil$ input sockets (since together they contain all ℓ input sockets). Let e be the edge leading to that subtree.

We claim that $\mathrm{rk}(M(\mathcal{T}_S, e)) \ge n^u r$. Suppose without loss of generality that the input sockets $A^{(1)}, \ldots, A^{(u)}$ are in the subtree rooted at x_R, and that the input sockets $A^{(u+1)}, \ldots, A^{(\ell)}$ are in the subtree rooted at x together with the output socket.

Each row of $M(\mathcal{T}_S, e)$ is indexed by sequences of $2u$ indices $(i_1, j_1, \ldots, i_u, j_u)$ where each $i_s \in [n]$ and each $j_t \in [r]$. Each column is indexed by a sequence of $\ell + 2(n-u)$ indices $(i'_1, \ldots, i'_\ell; i_{u+1}, j_{u+1}, \ldots, i_\ell, j_\ell)$ where each $i_s, i'_{s'} \in [n]$ and each $j_t \in [r]$. An entry of $M(\mathcal{T}_S, e)$ is 1 if and only if $i'_1 = i_1, i'_2 = i_2, \ldots, i'_\ell = i_\ell$, and $j_1 = j_2 = \ldots = j_\ell$.

For any $i_1, \ldots, i_u \in [n]$ and $j \in [r]$, consider the row index $(i_1, j, i_2, j, \ldots, i_u, j)$ and the column index $(i_1, i_2, \ldots, i_u, 1, 1, \ldots, 1; 1, j, 1, j, \ldots, 1, j)$. The $n^u r \times n^u r$ submatrix of $M(\mathcal{T}_S, e)$ induced by these sets of row and column indices is the identity matrix, thus $\mathrm{rk}(M(\mathcal{T}_S, e)) \ge n^u r$ as desired.

For the n^ℓ lower bound, we instead consider the edge e' joining x with the output socket. Now the rows are indexed by $(i_1, j_1, i_2, j_2, \ldots, i_\ell, j_\ell)$ and the columns by $(i'_1, \ldots, i'_\ell)$. The $n^\ell \times n^\ell$ identity submatrix is obtained by taking for every $i_1, \ldots, i_\ell \in [n]$, the row $(i_1, 1, i_2, 1, \ldots, i_\ell, 1)$ and the column $(i_1, \ldots, i_\ell)$. ∎

A Background on tensor networks

This appendix summarizes some results on tensor network contractions, minimum-cost executions, and explains the connection between the Holant framework and tensor networks. In the language of graphical models, computing the value of a tensor network is known as the sum-product inference task and the results in Appendices A.1 and A.2 can be found in [62, Part II].

A.1 Invariance property

In the following lemma, we will show that the tensor of a network is equal to the tensor of any network that is obtained by a contraction from the original network. In particular, this implies that any execution gives the same tensor.

Lemma A.1 (Invariance). *Let D be a tensor network. For all nonempty $W \subseteq V(D)$ it holds that $T(D) = T(D/W)$.*

Proof. Let $W \subseteq V(D)$ be nonempty, let w be the vertex that replaces $D[W]$ in D/W and let $i \in J(B(D/W)) = J(B(D))$. From (4.1), (4.2), and (4.3), it follows that

$$\begin{aligned}
T(D/W)_i &= \sum_{j \in J(E(D/W) \setminus B(D/W))} \prod_{v \in (V(D) \setminus W) \cup \{w\}} T(v)_{ij} \\
&= \sum_{j \in J(E(D/W) \setminus B(D/W))} T(w)_{ij} \prod_{v \in V(D) \setminus W} T(v)_{ij} \\
&= \sum_{j \in J(E(D/W) \setminus B(D))} T(D[W])_{ij} \prod_{v \in V(D) \setminus W} T(v)_{ij} \\
&= \sum_{j \in J(E(D/W) \setminus B(D))} \sum_{j' \in J(E(D[W]) \setminus B(D[W]))} \prod_{w \in W} T(w)_{ijj'} \prod_{v \in V(D) \setminus W} T(v)_{ij} \\
&= \sum_{j \in J(E(D/W) \setminus B(D))} \sum_{j' \in J(E(D[W]) \setminus B(D[W]))} \prod_{v \in V(D)} T(v)_{ijj'} \\
&= \sum_{j'' \in J(E(D) \setminus B(D))} \prod_{v \in V(D)} T(v)_{ij''} \\
&= T(D)_i \, .
\end{aligned}$$

□

A.2 The structure of a minimum-cost execution

In this section, we analyze the structure of a minimum-cost execution. In particular, we prove Lemma 4.1, which states that each contracted set has size at most two, and also show that one can always contract adjacent vertices in a network.

Lemma A.2. *Let D be a tensor network. There exists a minimum-cost execution of D such that each contracted set has size at most two. Furthermore, if D is loopless, we can assume that each contracted set has size exactly two.*

Proof. If D contains loops, we may assume that a minimum-cost execution first removes all the loops by contracting singleton vertices incident to loops. Indeed, the cost of contracting a singleton vertex is the volume of the tensor associated to it. Since the result of an execution is a single tensor, then every vertex has to be contained in a contracted set of an execution and none of the hyperedges incident to a vertex can be removed before the vertex is contained in a contracted set. Hence, the volume of any tensor in the tensor network is a lower bound for the cost of any execution of D and we may contract singleton vertices.

So let us assume that D is loopless. Suppose that a minimum-cost execution of D contains a contraction by a set $W = \{w_1, w_2, \ldots, w_s\}$ of size at least $s \geq 3$, and let w be the new vertex after this contraction. Then we can replace the contraction by W with two contractions by $W' = \{w_1, w_2, \ldots, w_{s-1}\}$ and $W'' = \{w, w_s\}$ without increasing the cost of the execution. The cost of contracting W' is less than or equal to the cost of contracting W, because every hyperedge incident to W' is also incident to W. The cost of contracting W'' is less than or equal to the cost of contracting W, because the set of hyperedges incident to w is contained in the set of hyperedges incident to W'. We repeat this procedure until all contracted sets in the execution have size at most two. $\square$

Two tensors in a tensor network are called *adjacent* if they are incident to a common mode.

Lemma A.3 (Execution by contracting adjacent tensors)**.** *Let D be a loopless tensor network that is connected as a hypergraph. Then, there exists a minimum-cost execution of D such that each contracted set has size two and consists of adjacent vertices.*

Proof. Consider a minimum-cost execution of D such that each contracted set has size two; such an execution exists by Lemma A.2. If all contracted sets in the execution consist of adjacent vertices, we are done. Otherwise, consider a contraction of two vertices, u and v, such that u and v are not adjacent when they are contracted to yield the vertex uv. Consider the steps of the execution after this contraction step. Let us call such a step a *relevant* step if it involves a descendant of the vertex uv. Let us modify the execution as follows. First, delete the contraction of u and v from the execution. Then, for each relevant step in execution order, replace the descendant of the vertex uv with the current descendant of either u or v so that the contraction becomes a contraction of adjacent vertices whenever possible. If the descendants of u and v become adjacent after a relevant step, contract the descendants of u and v, and then continue the execution without further changes. Since D is connected, the descendants of u and v must eventually become adjacent.

We will show that this modification of the execution gives again an execution and that it has cost no larger than the original minimum-cost execution. In the modification of the execution, let the contraction sets containing u or a descendant of u and not containing a descendant of v be $\{u, w_1\}, \{uw_1, w_2\}, \ldots, \{uw_1 \ldots w_{s-1}, w_s\}$; similarly, let the contraction sets containing v or a descendant of v and not containing a descendant of u be $\{v, z_1\}, \{vz_1, z_2\}, \ldots, \{vz_1 \ldots z_{t-1}, z_t\}$. Here $w_1, \ldots, w_s, z_1, \ldots, z_t$ can be vertices of D or vertices obtained after contraction steps. Contracting $\{uw_1 \ldots w_s, vz_1 \ldots z_t\}$ gives the vertex $uw_1 \ldots w_s vz_1 \ldots z_t$, which appears also in the original execution. Hence, after a certain number of steps the tensor networks in the original execution and in the modification are the same (after making necessary non-relevant contractions) and the modification of the original execution is also an execution.

First we consider the cost of contracting $\{uw_1 \ldots w_{i-1}, w_i\}$ in the modified execution with $1 \le i \le s$. There is a j satisfying $1 \le j \le t$ so that the original execution contracts $\{uw_1 \ldots w_{i-1} vz_1 \ldots z_j, w_i\}$. Then the cost of contracting $\{uw_1 \ldots w_{i-1}, w_i\}$ is at most the cost of contracting $\{uw_1 \ldots w_{i-1} vz_1 \ldots z_j, w_i\}$, because every hyperedge incident to the vertex $uw_1 \ldots w_{i-1}$ in the modified execution has to be incident to $uw_1 \ldots w_{i-1} vz_1 \ldots z_j$ in the original execution. Otherwise, there would be a hyperedge in D that is incident only to vertices in $\{u, w_1, \ldots, w_{i-1}, v, z_1, \ldots, z_j\}$, and in particular both to vertices in $\{u, w_1, \ldots, w_{i-1}\}$ and in $\{v, z_1, \ldots, z_j\}$. This is impossible, because then $uw_1 \ldots w_{i-1}$ and $vz_1 \ldots z_j$ would be adjacent vertices in the modified execution, but by assumption $uw_1 \ldots w_s$ and $vz_1 \ldots z_t$ are the first adjacent descendants of u and v in the modified execution. Similarly, we can show that the cost of contracting $\{vz_1 \dots z_{j-1}, z_j\}$ in the modified execution for $1 \leq j \leq t$ is less than the cost of contracting a set in the original execution.

Second we consider the cost of contracting $\{uw_1 \dots w_s, vz_1 \dots z_t\}$ in the modified execution. Without loss of generality, we can assume that in the original execution the vertex $uw_1 \dots w_s vz_1 \dots z_t$ is obtained from contracting $\{uw_1 \dots w_s vz_1 \dots z_{t-1}, z_t\}$. We will show that the cost of contracting $\{uw_1 \dots w_s, vz_1 \dots z_t\}$ in the modified execution is less than or equal to the cost of contracting $\{uw_1 \dots w_s vz_1 \dots z_{t-1}, z_t\}$ in the original execution. Indeed, every hyperedge incident to $uw_1 \dots w_s$ in the modified execution is incident to $uw_1 \dots w_s vz_1 \dots z_{t-1}$ in the original execution, because otherwise there would be a hyperedge in D that is incident only to vertices in $\{u, w_1, \dots, w_s, v, z_1, \dots, z_{t-1}\}$, and in particular both to vertices in $\{u, w_1, \dots, w_s\}$ and $\{v, z_1, \dots, z_{t-1}\}$. However, by assumption $uw_1 \dots w_s$ and $vz_1 \dots z_t$ are the first adjacent descendants of u and v in the modified execution. Similarly, every hyperedge incident to $vz_1 \dots z_t$ in the modified execution is incident to $uw_1 \dots w_s vz_1 \dots z_{t-1}$ or z_t in the original execution, because otherwise there would be a hyperedge in D that is incident only to vertices in $\{u, w_1, \dots, w_s, v, z_1, \dots, z_{t-1}\}$, and in particular both to vertices in $\{u, w_1, \dots, w_s\}$ and $\{v, z_1, \dots, z_{t-1}\}$. This contradicts that $uw_1 \dots w_s$ and $vz_1 \dots z_t$ are the first adjacent descendants of u and v in the modified execution.

The modified execution consists of at least one less contraction of nonadjacent vertices. Repeating this procedure until there are no contractions of nonadjacent vertices completes the lemma. □

A.3 Holant framework

In this section, we elaborate on the connection between tensor networks and the Holant framework. We follow the exposition in [25]. The main object of the Holant framework is a signature grid. A *signature grid* is a triple $\Omega = (G, \mathcal{F}, \pi)$, where $G = (V, E)$ is an undirected graph, $\mathcal{F}$ is a set of functions and π assigns to each vertex $v \in V$ a multivariate function $f_v \in \mathcal{F}$ with input variables corresponding to the edges incident to v and taking values in a finite set S. The function f_v is called the signature of v. The edges of G are considered to be variables that take values in the set S. An assignment $\sigma : E \to S$ gives an evaluation $\prod_{v \in V} f_v(\sigma|E(v))$, where $E(v)$ is the set of edges incident to a vertex v. The Holant of a signature grid is the sum of these evaluations over all the assignments σ:

$$\mathrm{Holant}_\Omega = \sum_{\sigma: E \to S} \prod_{v \in V} f_v(\sigma|E(v)).$$

A generalization of a signature grid is an *$\mathcal{F}$-gate* that additionally allows dangling edges for input and output variables.

Signature grids are special instances of tensor networks with the restriction that all hyperedges are of size two and the index sets associated to different edges are equal. Signature grids correspond to tensor networks with empty boundary; $\mathcal{F}$-gates do not have the restriction. A signature f_v associated to a vertex v in a signature grid corresponds precisely to a tensor $T(v)$ associated to a vertex v in a tensor network. The Holant of a signature grid is the same as the value of the corresponding tensor network.

Acknowledgements. We are grateful to Andreas Björklund for highlighting branchwidth to us as a natural parameter to generalize from clique-counting to counting homomorphisms, and to our anonymous reviewers for their extensive reading of the paper and many helpful suggestions.

References

[1] AMIR ABBOUD, ARTURS BACKURS, KARL BRINGMANN, AND MARVIN KÜNNEMANN: Fine-grained complexity of analyzing compressed data: Quantifying improvements over decompress-and-solve. In *Proc. 58th FOCS*, pp. 192–203. IEEE Comp. Soc., 2017. [doi:10.1109/FOCS.2017.26, arXiv:1803.00796] 7

[2] AMIR ABBOUD, ARTURS BACKURS, AND VIRGINIA VASSILEVSKA WILLIAMS: If the current clique algorithms are optimal, so is Valiant's parser. *SIAM J. Comput.*, 47(6):2527–2555, 2018. Preliminary version in FOCS'15. [doi:10.1137/16M1061771, arXiv:1504.01431] 3, 7

[3] MICHAEL ALEKHNOVICH, ALLAN BORODIN, JOSHUA BURESH-OPPENHEIM, RUSSELL IMPAGLIAZZO, AVNER MAGEN, AND TONIANN PITASSI: Toward a model for backtracking and dynamic programming. *Comput. Complexity*, 20(4):679–740, 2011. Preliminary version in CCC'05. [doi:10.1007/s00037-011-0028-y, ECCC:TR09-038] 2

[4] JOSH ALMAN AND VIRGINIA VASSILEVSKA WILLIAMS: A refined laser method and faster matrix multiplication. In *Proc. 32nd Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA'21)*, pp. 522–539. SIAM, 2021. [doi:10.1137/1.9781611976465.32, arXiv:2010.05846] 5, 11

[5] NOGA ALON AND SHAI GUTNER: Balanced families of perfect hash functions and their applications. *ACM Trans. Algorithms*, 6(3):54:1–12, 2010. Preliminary version in ICALP'07. [doi:10.1145/1798596.1798607, arXiv:0805.4300] 7

[6] NOGA ALON, RAPHAEL YUSTER, AND URI ZWICK: Finding and counting given length cycles. *Algorithmica*, 17(3):209–223, 1997. Preliminary version in ESA'94. [doi:10.1007/BF02523189] 7

[7] ITAI ARAD AND ZEPH LANDAU: Quantum computation and the evaluation of tensor networks. *SIAM J. Comput.*, 39(7):3089–3121, 2010. [doi:10.1137/080739379] 3, 11

[8] SANJEEV ARORA AND BOAZ BARAK: *Computational Complexity: A Modern Approach*. Cambridge Univ. Press, 2009. [doi:10.1017/CBO9780511804090] 2

[9] PER AUSTRIN, PETTERI KASKI, AND KAIE KUBJAS: Tensor network complexity of multilinear maps. In *Proc. 9th Innovations in Theoret. Comp. Sci. Conf. (ITCS'18)*, pp. 7:1–21. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018. [doi:10.4230/LIPIcs.ITCS.2019.7] 1

[10] FAHIEM BACCHUS, SHANNON DALMAO, AND TONIANN PITASSI: Algorithms and complexity results for #SAT and Bayesian inference. In *Proc. 44th FOCS*, pp. 340–351. IEEE Comp. Soc., 2003. [doi:10.1109/SFCS.2003.1238208] 11

[11] MARTIN BEAUDRY AND MARKUS HOLZER: The complexity of tensor circuit evaluation. *Comput. Complexity*, 16(1):60–111, 2007. Preliminary version in MFCS'01. [doi:10.1007/s00037-007-0222-0] 3

[12] AUSTIN R. BENSON AND GREY BALLARD: A framework for practical parallel fast matrix multiplication. In *Proc. 20th SIGPLAN Symp. Principles and Practice of Parallel Programming (PPoPP'15)*, pp. 42–53. ACM Press, 2015. [doi:10.1145/2688500.2688513, arXiv:1409.2908] 11

[13] STUART J. BERKOWITZ: On computing the determinant in small parallel time using a small number of processors. *Inform. Process. Lett.*, 18(3):147–150, 1984. [doi:10.1016/0020-0190(84)90018-8] 7

[14] JACOB D. BIAMONTE, JASON MORTON, AND JACOB TURNER: Tensor network contractions for #SAT. *J. Stat. Phys.*, 160(5):1389–1404, 2015. [doi:10.1007/s10955-015-1276-z, arXiv:1405.7375] 11

[15] ANDREAS BJÖRKLUND: Below all subsets for some permutational counting problems. In *Proc. 15th Scand. Symp. Algorithm Theory (SWAT'16)*, pp. 17:1–11. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016. [doi:10.4230/LIPIcs.SWAT.2016.17] 35

[16] ANDREAS BJÖRKLUND, THORE HUSFELDT, PETTERI KASKI, AND MIKKO KOIVISTO: Fourier meets Möbius: fast subset convolution. In *Proc. 39th STOC*, pp. 67–74. ACM Press, 2007. [doi:10.1145/1250790.1250801, arXiv:cs/0611101] 6, 33

[17] ANDREAS BJÖRKLUND, THORE HUSFELDT, PETTERI KASKI, AND MIKKO KOIVISTO: Computing the Tutte polynomial in vertex-exponential time. In *Proc. 49th FOCS*, pp. 677–686. IEEE Comp. Soc., 2008. [doi:10.1109/FOCS.2008.40, arXiv:0711.2585] 3, 6, 33

[18] ANDREAS BJÖRKLUND, THORE HUSFELDT, PETTERI KASKI, AND MIKKO KOIVISTO: Counting paths and packings in halves. In *Proc. 17th Eur. Symp. Algorithms (ESA'09)*, pp. 578–586. Springer, 2009. [doi:10.1007/978-3-642-04128-0_52, arXiv:0904.3093] 7

[19] ANDREAS BJÖRKLUND, THORE HUSFELDT, AND MIKKO KOIVISTO: Set partitioning via inclusion-exclusion. *SIAM J. Comput.*, 39(2):546–563, 2009. [doi:10.1137/070683933] 3, 6, 33

[20] ANDREAS BJÖRKLUND, PETTERI KASKI, AND ŁUKASZ KOWALIK: Counting thin subgraphs via packings faster than meet-in-the-middle time. *ACM Trans. Algorithms*, 13(4):48:1–26, 2017. Preliminary version in SODA'14. [doi:10.1145/3125500, arXiv:1306.4111] 7

[21] HANS L. BODLAENDER, ERIK JAN VAN LEEUWEN, JOHAN M. M. VAN ROOIJ, AND MARTIN VATSHELLE: Faster algorithms on branch and clique decompositions. In *Proc. Internat. Symp. Math. Foundations of Comp. Sci. (MFCS'10)*, pp. 174–185. Springer, 2010. [doi:10.1007/978-3-642-15155-2_17] 6

[22] JAMES R. BUNCH AND JOHN E. HOPCROFT: Triangular factorization and inversion by fast matrix multiplication. *Math. Comput.*, 28(125):231–236, 1974. [doi:10.1090/S0025-5718-1974-0331751-8] 7

[23] PETER BÜRGISSER, MICHAEL CLAUSEN, AND M. AMIN SHOKROLLAHI: *Algebraic Complexity Theory*. Springer, 1997. [doi:10.1007/978-3-662-03338-8] 11, 24

[24] JIN-YI CAI, HENG GUO, AND TYSON WILLIAMS: A complete dichotomy rises from the capture of vanishing signatures. *SIAM J. Comput.*, 45(5):1671–1728, 2016. Preliminary version in STOC'13. [doi:10.1137/15M1049798, arXiv:1204.6445] 3, 11

[25] JIN-YI CAI, PINYAN LU, AND MINGJI XIA: Computational complexity of Holant problems. *SIAM J. Comput.*, 40(4):1101–1132, 2011. [doi:10.1137/100814585] 3, 11, 44

[26] FLORENT CAPELLI, ARNAUD DURAND, AND STEFAN MENGEL: The arithmetic complexity of tensor contraction. *Theory Computing Sys.*, 58(4):506–527, 2016. Preliminary version in STACS'13. [doi:10.1007/s00224-015-9630-8, arXiv:1209.4865] 3

[27] ARTHUR CAYLEY: On the theory of the analytical forms called trees. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 13(85):172–176, 1857. [doi:10.1080/14786445708642275] 3, 10

[28] ARTHUR CAYLEY: On the analytical forms called trees, part II. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 18(121):374–378, 1859. [doi:10.1080/14786445908642782] 3, 10

[29] ANDRZEJ CICHOCKI, NAMGIL LEE, IVAN V. OSELEDETS, ANH HUY PHAN, QIBIN ZHAO, AND DANILO P. MANDIC: Tensor networks for dimensionality reduction and large-scale optimization: Part 1 Low-rank tensor decompositions. *Found. Trends Mach. Learning*, 9(4–5):249–429, 2016. [doi:10.1561/2200000059] 11

[30] ANDRZEJ CICHOCKI, ANH HUY PHAN, QIBIN ZHAO, NAMGIL LEE, IVAN V. OSELEDETS, MASASHI SUGIYAMA, AND DANILO P. MANDIC: Tensor networks for dimensionality reduction and large-scale optimization: Part 2 Applications and future perspectives. *Found. Trends Mach. Learning*, 9(6):431–673, 2017. [doi:10.1561/2200000067, arXiv:1708.09165] 11

[31] ALFRED CLEBSCH: Ueber symbolische Darstellung algebraischer Formen. *J. reine angew. Math.*, 59:1–62, 1861. [doi:10.1515/crll.1861.59.1] 3, 10

[32] WILLIAM K. CLIFFORD: Extract of a letter to Mr. Sylvester from Prof. Clifford of University College, London. *Amer. J. Math.*, 1(2):126–128, 1878. [doi:10.2307/2369303] 3, 10

[33] HENRY COHN, ROBERT D. KLEINBERG, BALÁZS SZEGEDY, AND CHRISTOPHER UMANS: Group-theoretic algorithms for matrix multiplication. In *Proc. 46th FOCS*, pp. 379–388. IEEE Comp. Soc., 2005. [doi:10.1109/SFCS.2005.39] 11

[34] HENRY COHN AND CHRISTOPHER UMANS: Fast matrix multiplication using coherent configurations. In *Proc. 24th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA'13)*, pp. 1074–1087. SIAM, 2013. [doi:10.1137/1.9781611973105.77, arXiv:1207.6528] 11

[35] JAMES W. COOLEY AND JOHN W. TUKEY: An algorithm for the machine calculation of complex Fourier series. *Math. Comput.*, 19(90):297–301, 1965. [doi:10.1090/S0025-5718-1965-0178586-1] 2, 5, 30

[36] LÁSZLÓ CSÁNKY: Fast parallel matrix inversion algorithms. *SIAM J. Comput.*, 5(4):618–623, 1976. Preliminary version in FOCS'75. [doi:10.1137/0205040] 7

[37] RADU CURTICAPEAN, HOLGER DELL, AND DÁNIEL MARX: Homomorphisms are a good basis for counting small subgraphs. In *Proc. 49th STOC*, pp. 210–223. ACM Press, 2017. [doi:10.1145/3055399.3055502, arXiv:1705.01595] 6, 7

[38] PREDRAG CVITANOVIĆ: Group theory for Feynman diagrams in non-Abelian gauge theories. *Phys. Rev. D*, 14(6):1536–1553, 1976. [doi:10.1103/PhysRevD.14.1536] 10

[39] PREDRAG CVITANOVIĆ AND ANTHONY D. KENNEDY: Spinors in negative dimensions. *Physica Scripta*, 26(1):5–14, 1982. [doi:10.1088/0031-8949/26/1/001] 10

[40] CARSTEN DAMM, MARKUS HOLZER, AND PIERRE MCKENZIE: The complexity of tensor calculus. *Comput. Complexity*, 11(1–2):54–89, 2002. Preliminary version in CCC'00. [doi:10.1007/s00037-000-0170-4, ECCC:TR00-036] 3

[41] SASHKA DAVIS AND RUSSELL IMPAGLIAZZO: Models of greedy algorithms for graph problems. *Algorithmica*, 54(3):269–317, 2009. Preliminary version in SODA'04. [doi:10.1007/s00453-007-9124-4, ECCC:TR05-120] 2

[42] JOSEP DÍAZ, MARIA J. SERNA, AND DIMITRIOS M. THILIKOS: Counting H-colorings of partial k-trees. *Theoret. Comput. Sci.*, 281(1–2):291–309, 2002. Preliminary version in COCOON'01. [doi:10.1016/S0304-3975(02)00017-8] 6

[43] FREDERIC DORN: Dynamic programming and fast matrix multiplication. In *Proc. 14th Eur. Symp. Algorithms (ESA'06)*, pp. 280–291. Springer, 2006. [doi:10.1007/11841036_27] 6

[44] FRIEDRICH EISENBRAND AND FABRIZIO GRANDONI: On the complexity of fixed parameter clique and dominating set. *Theoret. Comput. Sci.*, 326(1–3):57–67, 2004. [doi:10.1016/j.tcs.2004.05.009] 3, 6, 7, 30

[45] PETER FLODERUS, MIROSLAW KOWALUK, ANDRZEJ LINGAS, AND EVA-MARTA LUNDELL: Detecting and counting small pattern graphs. *SIAM J. Discr. Math.*, 29(3):1322–1339, 2015. Preliminary version in ISAAC'13. [doi:10.1137/140978211] 7

[46] FEDOR V. FOMIN, DANIEL LOKSHTANOV, VENKATESH RAMAN, SAKET SAURABH, AND B. V. RAGHAVENDRA RAO: Faster algorithms for finding and counting subgraphs. *J. Comput. System Sci.*, 78(3):698–706, 2012. [doi:10.1016/j.jcss.2011.10.001, arXiv:0912.2371] 7

[47] WILLIAM I. GASARCH: The second P =? NP poll. *SIGACT News*, 43(2):53–77, 2012. [doi:10.1145/2261417.2261434] 2

[48] JOHAN HÅSTAD: Tensor rank is NP-complete. *J. Algorithms*, 11(4):644–654, 1990. Preliminary version in ICALP'89. [doi:10.1016/0196-6774(90)90014-6] 11

[49] CHRISTOPHER J. HILLAR AND LEK-HENG LIM: Most tensor problems are NP-hard. *J. ACM*, 60(6):45:1–39, 2013. [doi:10.1145/2512329, arXiv:0911.1393] 11

[50] JIANYU HUANG, LESLIE RICE, DEVIN A. MATTHEWS, AND ROBERT A. VAN DE GEIJN: Generating families of practical fast matrix multiplication algorithms. In *Proc. Internat. Parallel and Distrib. Processing Symp. (IPDPS'17)*, pp. 656–667. IEEE Comp. Soc., 2017. [doi:10.1109/IPDPS.2017.56, arXiv:1611.01120] 11

[51] RUSSELL IMPAGLIAZZO AND RAMAMOHAN PATURI: On the complexity of k-SAT. *J. Comput. System Sci.*, 62(2):367–375, 2001. [doi:10.1006/jcss.2000.1727] 2

[52] RUSSELL IMPAGLIAZZO, RAMAMOHAN PATURI, AND FRANCIS ZANE: Which problems have strongly exponential complexity? *J. Comput. System Sci.*, 63(4):512–530, 2001. Preliminary version in FOCS'98. [doi:10.1006/jcss.2001.1774] 2

[53] ALON ITAI AND MICHAEL RODEH: Finding a minimum circuit in a graph. *SIAM J. Comput.*, 7(4):413–423, 1978. Preliminary version in STOC'77. [doi:10.1137/0207033] 3, 7

[54] NORMAN P. JOUPPI, CLIFF YOUNG, NISHANT PATIL, AND DAVID PATTERSON +72: In-datacenter performance analysis of a tensor processing unit. In *Proc. 44th Internat. Symp. Computer Architecture (ISCA'17)*, pp. 1–12. ACM Press, 2017. [doi:10.1145/3079856.3080246, arXiv:1704.04760] 3

[55] MATTI KARPPA, PETTERI KASKI, AND JUKKA KOHONEN: A faster subquadratic algorithm for finding outlier correlations. *ACM Trans. Algorithms*, 14(3):31:1–26, 2018. Preliminary version in SODA'16. [doi:10.1145/3174804, arXiv:1510.03895] 6

[56] LOUIS H. KAUFFMAN: Knots, abstract tensors and the Yang-Baxter equation. In L. LUSSANA, editor, *Knots, Topology and Quantum Field Theories*, pp. 179–334. World Scientific, 1989. 3

[57] ALFRED BRAY KEMPE: On the application of Clifford's graphs to ordinary binary quantics. *Proc. London Math. Soc.*, 17(1):107–123, 1885. [doi:10.1112/plms/s1-17.1.107] 3, 10

[58] ALFRED BRAY KEMPE: On the application of the Sylvester–Clifford graphs to ordinary binary quantics. (Second part.). *Proc. London Math. Soc.*, 24(1):97–118, 1892. [doi:10.1112/plms/s1-24.1.97] 3, 10

[59] TON KLOKS, DIETER KRATSCH, AND HAIKO MÜLLER: Finding and counting small induced subgraphs efficiently. *Inform. Process. Lett.*, 74(3–4):115–121, 2000. Preliminary version in WG'95. [doi:10.1016/S0020-0190(00)00047-8] 7

[60] TAMARA GIBSON KOLDA: Multilinear operators for higher-order decompositions. Technical Report SAND2006-2081, Sandia National Laboratories, 2006. [doi:10.2172/923081] 6, 16

[61] TAMARA GIBSON KOLDA AND BRETT W. BADER: Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, 2009. [doi:10.1137/07070111X] 6

[62] DAPHNE KOLLER AND NIR FRIEDMAN: *Probabilistic Graphical Models*. MIT Press, 2009. MIT Press. 3, 11, 41

[63] JOSEPH B. KRUSKAL: Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Lin. Alg. Appl.*, 18(2):95–138, 1977. [doi:10.1016/0024-3795(77)90069-6] 6, 16

[64] GREG KUPERBERG: Involutory Hopf algebras and 3-manifold invariants. *Internat. J. Mathematics*, 2(1):41–66, 1991. [doi:10.1142/S0129167X91000053] 3, 10

[65] CHI-CHUNG LAM, P. SADAYAPPAN, AND REPHAEL WENGER: On optimizing a class of multi-dimensional loops with reduction for parallel execution. *Parallel Process. Lett.*, 7(2):157–168, 1997. [doi:10.1142/S0129626497000176] 11

[66] JOSEPH M. LANDSBERG: *Tensors: Geometry and Applications*. Amer. Math. Soc., 2012. [doi:10.1090/gsm/128] 3, 10

[67] JOSEPH M. LANDSBERG, YANG QI, AND KE YE: On the geometry of tensor network states. *Quantum Inf. Comput.*, 12(3& 4):346–354, 2012. [doi:10.26421/QIC12.3-4-12] 3

[68] FRANÇOIS LE GALL: Faster algorithms for rectangular matrix multiplication. In *Proc. 53rd FOCS*, pp. 514–523. IEEE Comp. Soc., 2012. [doi:10.1109/FOCS.2012.80, arXiv:1204.1111] 11

[69] FRANÇOIS LE GALL: Powers of tensors and fast matrix multiplication. In *Proc. 39th Internat. Symp. Symbolic and Algebraic Computation (ISSAC'14)*, pp. 296–303. ACM Press, 2014. [doi:10.1145/2608628.2608664, arXiv:1401.7714] 5, 11

[70] FRANÇOIS LE GALL AND FLORENT URRUTIA: Improved rectangular matrix multiplication using powers of the Coppersmith-Winograd tensor. In *Proc. 29th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA'18)*, pp. 1029–1046. SIAM, 2018. [doi:10.1137/1.9781611975031.67, arXiv:1708.05622] 5, 11, 24

[71] JAMES R. LEE, PRASAD RAGHAVENDRA, AND DAVID STEURER: Lower bounds on the size of semidefinite programming relaxations. In *Proc. 47th STOC*, pp. 567–576. ACM Press, 2015. [doi:10.1145/2746539.2746599, arXiv:1411.6317] 2

[72] ANDREA LINCOLN, VIRGINIA VASSILEVSKA WILLIAMS, AND R. RYAN WILLIAMS: Tight hardness for shortest cycles and paths in sparse graphs. In *Proc. 29th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA'18)*, pp. 1236–1252. SIAM, 2018. [doi:10.1137/1.9781611975031.80, arXiv:1712.08147] 8

[73] STEFANO MARKIDIS, STEVEN WEI DER CHIEN, ERWIN LAURE, IVY BO PENG, AND JEFFREY S. VETTER: NVIDIA tensor core programmability, performance & precision. In *Proc. Internat. Parallel and Distrib. Processing Symp. (IPDPS'18)*, pp. 522–531. IEEE Comp. Soc., 2018. [doi:10.1109/IPDPSW.2018.00091, arXiv:1803.04014] 3

[74] ANKUR MOITRA AND ALEXANDER S. WEIN: Spectral methods from tensor networks. In *Proc. 51st STOC*, pp. 926–937. ACM Press, 2019. [doi:10.1145/3313276.3316357, arXiv:1811.00944] 11

[75] JARIK NEŠETŘIL AND SVATOPLUK POLJAK: On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985. EuDML. 3, 6, 7, 9

[76] NOAM NISAN: Lower bounds for non-commutative computation (extended abstract). In *Proc. 23rd STOC*, pp. 410–418. ACM Press, 1991. [doi:10.1145/103418.103462] 39

[77] STEPHAN OLARIU: Paw-free graphs. *Inform. Process. Lett.*, 28(1):53–54, 1988. [doi:10.1016/0020-0190(88)90143-3] 7

[78] ROMÁN ORÚS: A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Ann. Phys.*, 349:117–158, 2014. [doi:10.1016/j.aop.2014.06.013, arXiv:1306.2164] 3, 11

[79] VICTOR Y. PAN: Matrix multiplication, trilinear decompositions, APA algorithms, and summation, 2014. [arXiv:1412.1145] 11

[80] ROGER PENROSE: *Tensor Methods in Algebraic Geometry*. Ph. D. thesis, St. John's College, 1956. 3

[81] ROGER PENROSE: Applications of negative dimensional tensors. In D. J. A. WELSH, editor, *Combinatorial Mathematics and its Applications*, pp. 221–244. Academic Press, 1971. LINK. 3, 10

[82] ROGER PENROSE AND WOLFGANG RINDLER: *Spinors and space-time. Vol. 1*. Cambridge Univ. Press, 1984. [doi:10.1017/CBO9780511564048] 10

[83] ROBERT N. C. PFEIFER, JUTHO HAEGEMAN, AND FRANK VERSTRAETE: Faster identification of optimal contraction sequences for tensor networks. *Phys. Rev. E*, 90(3), 2014. [doi:10.1103/PhysRevE.90.033315] 3, 11

[84] RAN RAZ: Tensor-rank and lower bounds for arithmetic formulas. *J. ACM*, 60(6):40:1–15, 2013. Preliminary version in STOC'10. [doi:10.1145/2535928, ECCC:TR10-002] 7

[85] JÜRGEN RICHTER-GEBERT AND PETER LEBMEIR: Diagrams, tensors and geometric reasoning. *Discr. Comput. Geom.*, 42(2):305–334, 2009. [doi:10.1007/s00454-009-9188-9] 3

[86] NEIL ROBERTSON AND PAUL D. SEYMOUR: Graph minors. X. Obstructions to tree-decomposition. *J. Combin. Theory–B*, 52(2):153–190, 1991. [doi:10.1016/0095-8956(91)90061-N] 14

[87] ELINA ROBEVA AND ANNA SEIGAL: Duality of graphical models and tensor networks. *Inf. Inference*, 8(2):273–288, 2018. [doi:10.1093/imaiai/iay009, arXiv:1710.01437] 3, 11

[88] GÜNTER ROTE: Division-free algorithms for the determinant and the Pfaffian: Algebraic and combinatorial approaches. In HELMUT ALT, editor, *Computational Discrete Mathematics, Advanced Lectures*, pp. 119–135. Springer, 2001. [doi:10.1007/3-540-45506-X_9] 7

[89] HERBERT JOHN RYSER: *Combinatorial Mathematics*. Amer. Math. Soc., 1963. 3, 6, 35

[90] ALEXANDER SCHRIJVER: On traces of tensor representations of diagrams. *Lin. Alg. Appl.*, 476:28–41, 2015. [doi:10.1016/j.laa.2015.02.037] 3, 10

[91] EDGAR SOLOMONIK AND TORSTEN HOEFLER: Sparse tensor algebra as a parallel programming model, 2015. [arXiv:1512.00066] 3, 11

[92] EDGAR SOLOMONIK, DEVIN MATTHEWS, JEFF R. HAMMOND, JOHN F. STANTON, AND JAMES DEMMEL: A massively parallel tensor contraction framework for coupled-cluster computations. *J. Parallel Distrib. Comput.*, 74(12):3176–3190, 2014. [doi:10.1016/j.jpdc.2014.06.002] 3

[93] VOLKER STRASSEN: Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969. [doi:10.1007/BF02165411] 4, 24

[94] JAMES J. SYLVESTER: On an application of the new atomic theory to the graphical representation of the invariants and covariants of binary quantics, with three appendices. *Amer. J. Math.*, 1(1):64–104, 1878. [doi:10.2307/2369436] 3, 10

[95] LESLIE G. VALIANT: The complexity of computing the permanent. *Theoret. Comput. Sci.*, 8(2):189–201, 1979. [doi:10.1016/0304-3975(79)90044-6] 3

[96] LESLIE G. VALIANT: Holographic algorithms. *SIAM J. Comput.*, 37(5):1565–1594, 2008. Preliminary version in FOCS'04. [doi:10.1137/070682575, ECCC:TR05-099] 3, 11

[97] CHARLES VAN LOAN: *Computational Frameworks for the Fast Fourier Transform*. SIAM, 1992. [doi:10.1137/1.9781611970999] 2, 5, 31

[98] VIRGINIA VASSILEVSKA WILLIAMS: Multiplying matrices faster than Coppersmith–Winograd. In *Proc. 44th STOC*, pp. 887–898. ACM Press, 2012. [doi:10.1145/2213977.2214056] 5, 11

[99] VIRGINIA VASSILEVSKA WILLIAMS, JOSHUA R. WANG, R. RYAN WILLIAMS, AND HUACHENG YU: Finding four-node subgraphs in triangle time. In *Proc. 26th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA'15)*, pp. 1671–1680. SIAM, 2015. [doi:10.1137/1.9781611973730.111] 7

[100] VIRGINIA VASSILEVSKA WILLIAMS AND R. RYAN WILLIAMS: Finding, minimizing, and counting weighted subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013. Preliminary version in STOC'09. [doi:10.1137/09076619X] 7

[101] MARTIN J. WAINWRIGHT AND MICHAEL I. JORDAN: Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learning*, 1(1–2):1–305, 2008. [doi:10.1561/2200000001] 11

[102] R. RYAN WILLIAMS: A new algorithm for optimal 2-constraint satisfaction and its implications. *Theoret. Comput. Sci.*, 348(2–3):357–365, 2005. Preliminary version in ICALP'04. [doi:10.1016/j.tcs.2005.09.023] 3, 7, 17

[103] R. RYAN WILLIAMS: Faster all-pairs shortest paths via circuit complexity. *SIAM J. Comput.*, 47(5):1965–1985, 2018. Preliminary version in STOC'14. [doi:10.1137/15M1024524] 3

[104] FRANK YATES: *The Design and Analysis of Factorial Experiments*. Imperial Bureau of Soil Science, 1937. Rothamsted Repository. 33

AUTHORS

Per Austrin
Associate professor
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology
Sweden
austrin@kth.se
`https://www.csc.kth.se/~austrin/`

Petteri Kaski
Associate professor
Department of Computer Science
Aalto University
Finland
petteri.kaski@aalto.fi
`https://www.aalto.fi/en/people/petteri-kaski`

Kaie Kubjas
Assistant professor
Department of Mathematics and Systems Analysis
Aalto University
Finland
kaie.kubjas@aalto.fi
`https://www.kaiekubjas.com`

ABOUT THE AUTHORS

PER AUSTRIN graduated from KTH Royal Institute of Technology in 2008; his advisor was Johan Håstad. The subject of his thesis was hardness of approximation and discrete harmonic analysis. In his spare time he sometimes enjoys algorithmic programming contests.

PETTERI KASKI graduated from the Helsinki University of Technology in 2005; his advisor was Patric Östergård. The subject of his thesis was algorithms for classification of combinatorial objects. He enjoys the interplay of algorithms, algebra, and combinatorics.

KAIE KUBJAS graduated from the Free University of Berlin in 2013; her advisors were Christian Haase and Klaus Altmann. The subject of her thesis was algebraic and combinatorial aspects of group-based models. She enjoys applying algebraic geometry to other fields. Kubjas discovered her love for mathematics in her school years in Tartu, Estonia under the guidance and mentoring of her mathematics teachers Reene Õigus and Hele Kiisel. Thanks to the encouragement and support of her teachers, Kubjas got involved in mathematics competitions.