*Information and Inference: A Journal of the IMA* (2019) **8**, 273–288
doi:10.1093/imaiai/iay009
Advance Access publication on 21 June 2018



# Duality of graphical models and tensor networks

Elina Robeva
*Department of Mathematics, 77 Massachusetts Avenue, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*
erobeva@mit.edu

and

Anna Seigal†
*Department of Mathematics, University of California Berkeley, Berkeley, CA 94720, USA*
†Corresponding author. Email: seigal@berkeley.edu

[Received on 17 October 2017; revised on 23 March 2018; accepted on 30 April 2018]

In this article we show the duality between tensor networks and undirected graphical models with discrete variables. We study tensor networks on hypergraphs, which we call tensor hypernetworks. We show that the tensor hypernetwork on a hypergraph exactly corresponds to the graphical model given by the dual hypergraph. We translate various notions under duality. For example, marginalization in a graphical model is dual to contraction in the tensor network. Algorithms also translate under duality. We show that belief propagation corresponds to a known algorithm for tensor network contraction. This article is a reminder that the research areas of graphical models and tensor networks can benefit from interaction.

*Keywords*: graphical models; tensor networks; hypergraphs; belief propagation.

## 1. Introduction

Graphical models and tensor networks are very popular, but are mostly separate fields of study. Graphical models are used in artificial intelligence, machine learning and statistical mechanics [20]. Tensor networks show up in areas such as quantum information theory, quantum many-body physics and partial differential equations [7,10].

Tensor network states are tensors that factor according to the adjacency structure of the vertices of a graph. On the other hand, graphical models are probability distributions that factor according to the clique structure of a graph. The joint probability distribution of several discrete random variables is naturally organized into a tensor. Hence, both graphical models and tensor networks are ways to represent families of tensors that factorize according to a graph structure.

The relationship between particular graphical models and particular tensor networks has been studied in the past. For example, in [6] the authors reparametrize a hidden Markov model to make a matrix product state tensor network. In [5], a map is constructed, which sends a restricted Boltzmann machine graphical model to a matrix product state. In [15], an example of a directed graphical model is given with a related tensor network on the same graph, to highlight computational advantages of the graphical model in that setting.

From the outset, there are differences in the graphical description. On the graphical model's side, the factors in the decomposition correspond to cliques in the graph. On the tensor network's side, the factors are associated to the vertices of the graph. For a graphical model we require tensor entries to lie

in $\mathbb{R}_{\geqslant 0}$, so that up to normalization they represent probabilities, while for a tensor network the entries can lie in the field $\mathbb{C}$.

In this article, we show a duality correspondence between graphical models and tensor networks. This correspondence applies to all graphical models and all tensor networks, and does not require reparametrization of either. Our mathematical relationship stems from hypergraph duality. We begin by recalling the definition of a hypergraph.

DEFINITION 1.1 A hypergraph $H = (U, \mathscr{C})$ consists of a set of vertices $U$ and a set of hyperedges $\mathscr{C}$. A hyperedge $C \in \mathscr{C}$ is any subset of the vertices.

There are two ways to construct a hypergraph from a matrix $M$ of size $d \times c$ with entries in $\{0, 1\}$. First, we let the rows index the vertices and the columns index the hyperedges. The non-vanishing entries in each column give the vertices that appear in that hyperedge,

$$M_{uC} = \begin{cases} 1 & u \in C \\ 0 & \text{otherwise.} \end{cases} \tag{1.1}$$

In this case $M$ is the *incidence matrix* of the hypergraph. We allow nested or repeated hyperedges, as well as edges containing one or no vertices, so there are no restrictions on $M$. Alternatively, we can construct a hypergraph with incidence matrix $M^T$. This is the *dual hypergraph* to the one with incidence matrix $M$, see [3, Section 1.1].

We now add extra data to the matrix. We attach positive integers $n_1, \ldots, n_d$ to each row. We assign tensors to each column of $M$ whose size is the product of the $n_i$ as $i$ ranges over the non-vanishing entries in the column. For example, the tensor associated to the column $(1, 1, 0, 1, 0, \ldots, 0)^T$ would have size $n_1 \times n_2 \times n_4$. We explain how this defines the data of both a graphical model and of a tensor network. Filling in the entries of the tensors gives a distribution in a graphical model (if the entries are in $\mathbb{R}_{\geqslant 0}$) or a tensor network state in a tensor network. We see how a graphical model is visualized by the hypergraph with incidence matrix $M$, while the tensor network is visualized by the hypergraph of $M^T$. If, on the other hand, we consider the incidence matrix $M$ as the biadjacency matrix of a bipartite graph, we obtain the factor graph construction; this perspective is studied in [9, 11].

Before stating our duality correspondence, we define graphical models in terms of hypergraphs and introduce tensor hypernetworks. We keep in mind how the definitions translate to the incidence matrix setup from above.

DEFINITION 1.2 Consider a hypergraph $H = (U, \mathscr{C})$ with $U = [d]$. An *undirected graphical model* with respect to $H$ is the set of probability distributions on the random variables $\{X_u, u \in U\}$ which factor according to the hyperedges in $\mathscr{C}$:

$$P(x_1, \ldots, x_d) = \frac{1}{Z} \prod_{C \in \mathscr{C}} \psi_C(x_C). \tag{1.2}$$

Here the random variable $X_u$ takes values $x_u \in \mathscr{X}_u$, the subset $x_C$ equals $\{x_u : u \in C\}$ and the function $\psi_C$ is a *clique potential* with domain $\prod_{u \in C} \mathscr{X}_u$ and range $\mathbb{R}_{\geqslant 0}$. The normalizing constant $Z$ ensures the probabilities sum to one.

When all random variables are discrete, the joint probabilities form a tensor $P$ of size $\times_{u \in U} |\mathscr{X}_u|$ and the clique potentials are tensors of size $\times_{u \in C} |\mathscr{X}_u|$, all with entries in $\mathbb{R}_{\geqslant 0}$. The graphical model

is depicted as the hypergraph whose incidence matrix has rows represented by the random variables $\{X_u : 1 \leqslant u \leqslant d\}$ and columns indexed by the hyperedges.

If we fix the values in the clique potentials, we obtain a particular distribution in the graphical model. We recover the usual depiction of the graphical model by a graph instead of a hypergraph by connecting pairs of vertices by an edge if they lie in the same hyperedge.

REMARK 1.1 Graphical models are sometimes required to factorize according to the *maximal* cliques of a graph. We see later how our setup specializes to this case. Models with cliques that are not necessarily maximal can be called *hierarchical models* [19].

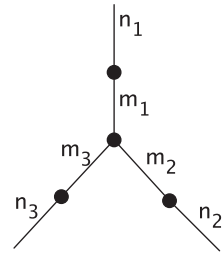
We now define tensor hypernetworks: families of tensors that factor according to a hypergraph. They have been defined this way in the literature, though it is not common [1,2].

DEFINITION 1.3 Consider a hypergraph $G = (V, E)$. To each hyperedge $e \in E$ we associate a positive integer $n_e$, called the size of the hyperedge. To each vertex $v \in V$ we assign a tensor $T_v \in \bigotimes_{e \ni v} \mathbb{K}^{n_e}$, where $\mathbb{K}$ is usually $\mathbb{R}$ or $\mathbb{C}$. The *tensor hypernetwork state* is obtained from $\bigotimes_{v \in V} T_v$ by contracting (summing over) the indices of all hyperedges in the graph that contain two or more vertices. We call hyperedges containing only one vertex dangling edges.

The *data* of a tensor hypernetwork (up to global scaling constant) is its hypergraph along with the tensor at each vertex of the hypergraph. This is the tensor network before contracting the hyperedges. The distinction between the contracted and uncontracted tensor network generalizes the fact that a rank $r$ matrix $M$ of size $n \times m$ can be represented either by its entries, or by two matrices, of sizes $n \times r$ and $r \times m$, whose product is $M$.

Note that as opposed to graphical models, in tensor hypernetworks we assign tensors to the vertices of the graph rather than to the hyperedges. Restricting the definition of a tensor hypernetwork to hyperedges with at most two vertices gives the usual definition of a tensor network. Tensor networks are sometimes assumed to have exactly one dangling edge per vertex, but we do not make that assumption here. The following example illustrates a widely used tensor network.

EXAMPLE 1.1 (Tucker decomposition) Consider the following graph:



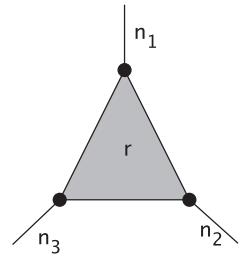

We have a core tensor $T_0 \in \mathbb{K}^{m_1} \otimes \cdots \otimes \mathbb{K}^{m_d}$ and matrices $T_i \in \mathbb{K}^{n_i} \otimes \mathbb{K}^{m_i}$. The entries of the tensor $T \in \mathbb{K}^{n_1} \otimes \cdots \otimes \mathbb{K}^{n_d}$ are

$$T_{i_1,\ldots,i_d} = \sum_{j_1,\ldots,j_d} (T_0)_{j_1,\ldots,j_d} (T_1)_{i_1,j_1} \cdots (T_d)_{i_d,j_d}. \tag{1.3}$$

For suitable weights $m_i$ and orthogonal matrices $T_j$, this is the *Tucker decomposition* of $T$.

An important reason to extend the definition of tensor networks to tensor hypernetworks, other than the duality with graphical models explained in the next section, is that significant classes of tensors naturally arise from tensor hypernetworks.

EXAMPLE 1.2 (Tensor rank (CP rank)) Consider this hypergraph on vertex set $\{1, 2, 3\}$:




There is one dangling edge for each vertex, with sizes $n_1, n_2, n_3$. There is one more hyperedge of size $r$, represented by a shaded triangle, that connects all three vertices. The tensors $T_v$ attached to each of the three vertices are matrices of size $n_v \times r$. The tensor hypernetwork state has size $n_1 \times n_2 \times n_3$ with entries

$$T_{ijk} = \sum_{l=1}^{r} (T_1)_{il}(T_2)_{jl}(T_3)_{kl}. \tag{1.4}$$

The set of tensors given by this tensor hypernetwork equals the set of tensors of rank at most $r$. The same structure on $d$ vertices, with weights $n_1, \ldots, n_d, r$, gives tensors of size $n_1 \times \cdots \times n_d$ and rank at most $r$. Tensor rank is the most direct generalization of matrix rank to the setting of tensors [7]. The set of tensors of rank at most $r$ is naturally parametrized by this tensor hypernetwork without requiring special structure (such as diagonal structure) on the tensors at the vertices.

The rest of the paper is organized as follows. We describe the duality correspondence between graphical models and tensor networks in Section 2. In Section 3 we explain how certain structures (graphs, trees and homotopy types) and operators (marginalization, conditioning and entropy) translate under the duality map. In Section 4 we give an algorithmic application of our duality correspondence.

## 2. Duality

In this section we give the duality between graphical models and tensor networks.

THEOREM 2.1 A distribution in a discrete graphical model associated to a hypergraph $H = (U, \mathscr{C})$ with clique potentials $\psi_C : \prod_{u \in C} \mathscr{X}_u \to \mathbb{K}$ is the same as the data of a tensor hypernetwork associated to its dual hypergraph $H^*$ with tensors $T_C = \psi_C$ at each vertex of $H^*$.

*Proof*. Consider a joint distribution (or tensor) $P$ in the graphical model defined by the hypergraph $H$. As described above, the incidence matrix $M$ of $H$ has rows corresponding to the variables $u \in U$ and columns corresponding to the cliques $C \in \mathscr{C}$. The data of the distribution $P$ also contains a potential function $\psi_C : \prod_{u \in C} \mathscr{X}_u \to \mathbb{K}$ for each clique $C \in \mathscr{C}$, which is equivalently a tensor of size $\times_{u \in C} |\mathscr{X}_u|$.

The dual hypergraph $H^*$ has incidence matrix $M^T$. It is a hypergraph with vertices $\{v_C : C \in \mathscr{C}\}$ and hyperedges $\{e_u : u \in U\}$. By definition of the dual hypergraph, $u \in C$ is equivalent to $v_C \in e_u$. Associating the tensors $T_C = \psi_C \in \bigotimes_{e_u \ni v_C} \mathbb{K}^{|\mathscr{X}_u|}$ to each vertex $v_C$ of $H^*$ gives a tensor hypernetwork

for $H^*$. Moreover, up to scaling by the normalization constant $Z$, the joint probability tensor $P$ is given by

$$P(x_u : u \in U) \cdot Z = \prod_{C \in \mathscr{C}} \psi_C(x_C) = \prod_{C \in \mathscr{C}} (T_C)_{x_C}. \tag{2.1}$$

The last expression is the tensor hypernetwork state before contracting the hyperedges. □

Since $(M^T)^T = M$, the dual of the dual $(H^*)^*$ is equal to $H$. This implies the following one-to-one correspondence. Before we state it, let us denote the set of distributions on $\mathscr{X} = \prod_{u \in U} \mathscr{X}_u$ that are in the graphical model defined by the hypergraph $H = (U, F)$ by $\mathscr{G}(H, \mathscr{X})$. These are the distributions that factor according to the hypergraph $H$, whose factor has sizes determined by $\{|\mathscr{X}_u| : u \in U\}$, and for which the entries of the factors can vary. Analogously, we denote the set of non-contracted tensor hypernetwork states from a hypergraph $G = (V, E)$ with weights $\mathbf{n} = \{n_e : e \in E\}$ by $\mathscr{T}(G, \mathbf{n})$.

Note that while clique potentials are required to take values in $\mathbb{R}_{\geq 0}$ for probabilistic reasons, the definition and factorization structure of graphical models carries over to the case where the entries of these tensors belong to a general field $\mathbb{K}$, and the statement of the following Corollary refers to this generalized setting.

COROLLARY 2.1 There is a one-to-one correspondence between the graphical models $\mathscr{G}(H, \mathscr{X})$ and the tensor hypernetwork states $\mathscr{T}(H^*, \{|\mathscr{X}_u| : u \in U\})$ up to global scaling constant.

Imposing that the entries of the tensors in both the graphical model and the tensor hypernetwork states be non-negative specializes the Corollary to the probabilistic setting. In the rest of this section we illustrate our results by showing the duals to some familiar examples of tensor network states and graphical models.We consider those tensor networks, or graphical models, which factor according to the hypergraphs shown. We do not fix the sizes of the factors or the entries of the tensors.

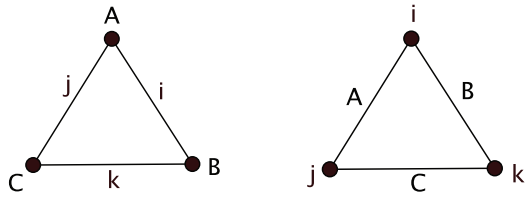
EXAMPLE 2.1 (Matrix Product States (MPS)/Tensor Trains) These are a popular family of tensor networks in quantum physics [**14**] and numerical applications [**7**] (where the two names come from). We return to them in detail in Section 4. The MPS network on the left is dual to the graphical model on the right.

The top row of edges in the tensor network is contracted. We see later that this corresponds to the top row of variables in the graphical model being hidden.

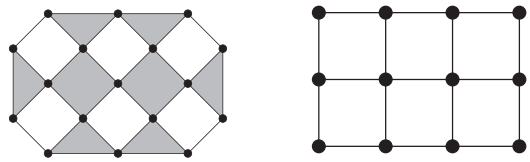
EXAMPLE 2.2 (No three-way interaction model) This graphical model consists of all probability distributions that factor as $p_{ijk} = A_{ij}B_{ik}C_{jk}$, for clique potential matrices $A, B, C$. It is represented by a hypergraph in which all hyperedges have two vertices. The incidence matrix of the hypergraph is as follows:

$$\begin{array}{c} \\ i \\ j \\ k \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \end{array} \tag{2.2}$$

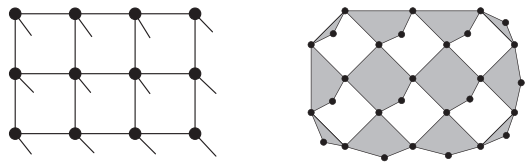
This matrix is symmetric. Hence, the tensor network corresponding to this graphical model is given by the same triangle graph. We note that, up to dangling edges, this is also the shape of the tensor network of the tensor that represents the matrix multiplication operator [10].



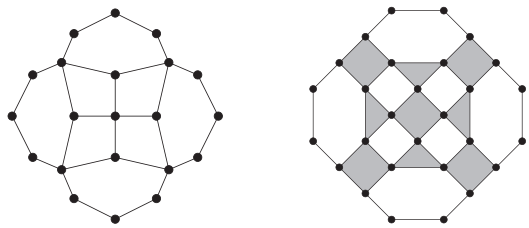

EXAMPLE 2.3 (The Ising Model) This graphical model is defined by cliques, which are the edges of a two-dimensional lattice such as the grid on the right. Its dual is the hypergraph on the left.



EXAMPLE 2.4 (Projected Entangled Pair States (PEPS)) This tensor network is a two-dimensional analogue of MPS. It depicts two-dimensional quantum spin systems. Its hypergraph is depicted on the left, with its dual graphical model on the right. Note the structural similarity with Example 2.3.



EXAMPLE 2.5 (The Multi-scale Entanglement Renormalization Ansatz) This tensor network is popular in the quantum community, due to its favorable abilities to represent relevant tensors and compute efficiently with them. It is on the left, with its dual graphical model on the right.



Finally, we point out the following fun fact.

REMARK 2.1 (Duality of Tucker and CP decomposition) Consider the hypergraphs in Examples 1.1 and 1.2 that give the graph structure of Tucker decomposition and CP decomposition respectively. Up to removal of dangling edges, the hypergraph corresponding to CP decomposition is dual to the one for Tucker decomposition.

## 3. Properties

Tensor networks and graphical models are often given special structure. For example, one can restrict to tensor networks that use a graph rather than a hypergraph. In this section we show how properties and operations for graphical models and tensor hypernetworks behave under the duality map.

### 3.1 *Restricting to graphs*

Graphs are special hypergraphs in which every hyperedge contains two vertices. They are also known as 2-uniform hypergraphs. Each column of the incidence matrix of such a hypergraph sums to two. Taking the dual of a graph gives a hypergraph in which every vertex has degree two, also known as a 2-regular hypergraph [3]. We call a hypergraph at-most-2-regular if every vertex has degree at most 2.

PROPOSITION 3.1 Tensor networks are dual to at-most-2-regular graphical models. Graph models (graphical models whose cliques are the edges of a graph) are dual to 2-regular tensor hypernetworks.

Graphical models defined by the maximal cliques of a graph correspond to hypergraphs in which we introduce a hyperedge for each maximal clique. Their dual tensor hypernetworks have the following property.

PROPOSITION 3.2 Graphical models defined by the maximal cliques of a graph correspond to tensor hypernetworks whose hypergraphs have the property that, whenever a set of hyperedges meet pairwise, the intersection of all of them is non-empty.

*Proof*. Let $E' \subseteq E$ be a set of hyperedges of the hypergraph of the tensor hypernetwork that meet pairwise. Then, for all $e_1, e_2 \in E'$, the corresponding vertices $u_{e_1}, u_{e_2}$ in the dual hypergraph (i.e. in the graphical model) are connected by an edge. Thus, the vertices $\{u_e : e \in E'\}$ form a clique in the graphical model, so there exists a maximal clique $C$ in which this clique is contained. Thus, all hyperedges in $E'$ contain the vertex corresponding to $C$. □

### 3.2 *Trees on each side*

The *homotopy type* of a hypergraph is the homotopy type of the simplicial complex whose maximal simplices are the maximal hyperedges. For topological purposes, we associate hypergraphs with their simplicial complexes. We show that the homotopy type of a hypergraph and its dual agree.

DEFINITION 3.1 (see [8]) Consider an open cover $\mathscr{V} = \{V_i : i \in I\}$ of a topological space $X$. The *nerve* $N(\mathscr{V})$ of the cover is a simplicial complex with one vertex for each open set. A subset $\{V_j : j \in J\}$ spans a simplex in the nerve whenever $\cap_{j\in J} V_j \neq \emptyset$.

THEOREM 3.1 (The Nerve Lemma [4]) The homotopy type of a space $X$ equals the homotopy type of the nerve of an open cover of $X$, provided that all intersections $\cap_{j\in J} V_j$ of sets in the open cover are contractible.

We consider the open cover of the simplicial complex corresponding to a hypergraph in which open sets of the complex are $\varepsilon$-neighborhoods of the maximal simplices. For $\varepsilon$ sufficiently small, such an open cover has contractible intersections, since they are homotopy equivalent to intersections of simplices. Hence, the homotopy type of the hypergraph is equal to that of its nerve. The following proposition relates the nerve to the dual hypergraph.

PROPOSITION 3.3 The nerve of the above open cover of the simplicial complex of a hypergraph is the simplicial complex of its dual hypergraph.

*Proof*. Consider a hypergraph $H$ with vertex set $U$ and hyperedge set $\mathscr{C}$. We now construct the dual hypergraph. The edges are represented by rows in the original incidence matrix. A subset $\{C_j : j \in J\} \subseteq \mathscr{C}$ is contained in a hyperedge if there exists a vertex $u \in U$ that is in all hyperedges $C_j$ in the subset. Hence, the simplices that arise in the dual simplicial complex are given by subsets of hyperedges for which the intersection $\cap_{j\in J} C_j$ is non-empty. This is exactly the definition of the nerve. □

From this, the Nerve Lemma implies the following.

THEOREM 3.2 The hypergraph of a tensor hypernetwork and the hypergraph of its dual graphical model have the same homotopy type.

A hypergraph cycle (see [3, Chapter 5]) is a sequence $(x_1, E_1, x_2, E_2, x_3, \ldots, x_k, E_k, x_1)$, where the $E_i$ are distinct hyperedges and the $x_j$ are distinct vertices, such that $\{x_i, x_{i+1}\} \subseteq E_i$ for all $i = 1, \ldots, k-1$, and $\{x_1, x_k\} \subseteq E_k$. A tree is a hypergraph with no cycles. The simplicial complexes corresponding to trees are contractible. Theorem 3.2 implies that trees are preserved under the duality correspondence.

### 3.3 *Marginalization and contraction*

Let $H = (U, \mathscr{C})$ be a hypergraph and $H^*$ its dual. Let $P$ be a distribution in the graphical model on $H$ with clique potentials $\psi_C : \prod_{u\in C}[n_u] \to \mathbb{K}$. The dual tensor hypernetwork has tensors $T_C = \psi_C \in \bigotimes_{u\in C} \mathbb{K}^{n_u}$ at the vertices of $H^*$.

PROPOSITION 3.4 (Marginalization Equals Contraction) Let $W \subseteq U$ be a subset of the vertices of the graph $H$. Then, the marginal distribution of $\{X_u\}_{u\in W}$ equals

$$P(x_W) = \sum_{\substack{x_u\in[n_u]:\\ u\notin W}} \prod_{C\in\mathscr{C}} (T_C)_{\{x_C : u\in C\}}, \tag{3.1}$$

which is the contracted tensor hypernetwork along the hyperedges corresponding to $W^c$.

*Proof*. The proof follows from the chain of equalities:

$$P(x_W) = \sum_{\substack{x_u\in[n_u]:\\ u\notin W}} P(x) = \sum_{\substack{x_u\in[n_u]:\\ u\notin W}} \prod_{C\in\mathscr{C}} \psi_C(x_C) = \sum_{\substack{x_u\in[n_u]:\\ u\notin W}} \prod_{C\in\mathscr{C}} (T_C)_{\{x_C : u\in C\}}. \tag{3.2}$$

In other words, summing over the values of all variables in $W^c$ is the same as contracting the tensor hypernetwork along all hyperedges in $W^c$. □

The interpretations of marginalization and contraction are also very similar in nature. The variables of a graphical model that are marginalized are often considered to be hidden, and the contracted edges of a tensor network represent entanglement ('unseen interaction').

The correspondence described in Proposition 3.4 allows us to translate algorithms for marginalization in graphical models to algorithms for contraction in tensor networks, see Section 4. Without care to order indices, marginalization and contraction involve summing exponentially many terms. In many cases more efficient methods are possible.

### 3.4 *Conditional distributions*

Consider a probability distribution given by a fully observed graphical model. Conditioning a variable $X_u$ to only take values in a given set $\mathscr{Y}_u \subseteq \mathscr{X}_u$ means restricting the probability tensor $P$ to the slice $\mathscr{Y}_u \times \prod_{b\in U\setminus\{u\}} \mathscr{X}_b$ which contains only the values $\mathscr{Y}_u$ for the variable $X_u$. This in turn corresponds to restricting each of the potentials for hyperedges $C$ containing $u$ to the given subset of elements $\mathscr{Y}_u \times \prod_{b\in C\setminus\{u\}} \mathscr{X}_b$. On the tensor network's side, we restrict the tensor corresponding to the given clique potential to the slice $\mathscr{Y}_u \times \prod_{b\in C\setminus\{u\}} \mathscr{X}_b$.

We wish to remark that the equivalence of conditioning and restriction to a slice of the probability tensor is due to the fact that the basis in which we view the probability tensor is fixed. The basis is given by the states of the random variables: graphical models are not basis invariant. On the other hand, basis invariance is a key property of tensor networks that crops up in many applications, e.g. often a gauge (basis) is selected to make the computations efficient [14].

### 3.5 *Entanglement entropy and Shannon entropy*

Given a tensor network state represented by a tensor $T$, the *entanglement entropy* [14] equals

$$-\operatorname{trace}(T\log T), \tag{3.3}$$

where $T\log T$ is a tensor, the same size as $T$, whose entry indexed by $\mathbf{i}$ is $T_{\mathbf{i}}\log T_{\mathbf{i}}$. On the other hand, if $T$ represents the corresponding marginal distribution of the graphical model, the *Shannon entropy* [20] of $T$ is defined as

$$H(T) = -\sum_{\mathbf{i}\in\mathscr{I}} T_{\mathbf{i}}\log T_{\mathbf{i}}, \tag{3.4}$$

where $\mathscr{I}$ indexes all entries of $T$. Expanding out the formula $-\operatorname{trace}(T\log T)$ shows that these two notions of entropy are the same.

## 4. Algorithms for marginalization and contraction

The *belief propagation* (or *sum–product*) algorithm is a dynamic programming method for computing marginals of a distribution [20]. The *junction tree algorithm* [20] extends it to graphs with cycles. The equivalence between marginalization in graphical models and contraction in tensor hypernetworks was given in Proposition 3.4. It means that we can use methods for marginalization to contract tensor hypernetworks and vice versa. For example, we can compute expectation values of tensor hypernetwork states (obtained by contracting a tensor hypernetwork along all edges) [14] as well as contracted tensor hypernetwork states. In this section, we apply the junction tree algorithm to these tasks for the MPS tensor networks from Example 2.1. We first recall the algorithm.

### 4.1 *The junction tree algorithm*

The input and output data of the junction tree algorithm are as follows.
*Input:* A graphical model defined by a hypergraph $H$ with clique potentials $\psi_C(x_C)$.
*Output:* The marginals at each hyperedge, $P(x_C) = \sum_{x_u : u\notin C} P(x)$.

We now recall how this algorithm works. First, we construct the graph $G$ associated to the hypergraph $H$ by adding edge $(i,j)$ whenever vertices $i$ and $j$ belong to the same hyperedge. If $G$ is not chordal (or triangulated) we add edges until all cycles of length four or more have a chord, i.e. $G$

becomes chordal. Then we can form a junction tree. This is a tree whose nodes are the maximal cliques of the graph. It has the *running intersection property*: the subset of cliques of $G$ containing a given vertex forms a connected subtree. Note that there are often multiple ways to construct a junction tree of a given graph $G$.

To each maximal clique $C$ in $G$ we associate a clique potential which equals the product of the potentials of the hyperedges contained in $C$. If a hyperedge is contained in more than one maximal clique, its clique potential is assigned to one of them. Each edge of the junction tree connects two cliques $C_1, C_2 \in \mathscr{C}$ in $G$. We associate to such an edge the *separator* set $S = C_1 \cap C_2$. We also assign a separator potential $\psi_S(x_S)$ to each $S$. It is initialized to the constant value 1. A *basic message passing operation* from $C_1$ to a neighboring $C_2$ updates the potential functions at clique $C_2$ and separator $S = C_1 \cap C_2$:

$$\tilde{\psi}_S(x_S) \leftarrow \sum_{x_{C_1 \setminus S}} \psi_{C_1}(x_{C_1}), \tag{4.1}$$

$$\tilde{\psi}_{C_2}(x_{C_2}) \leftarrow \frac{\tilde{\psi}_S(x_S)}{\psi_S(x_S)} \psi_{C_2}(x_{C_2}). \tag{4.2}$$

The algorithm chooses a root of the junction tree and orients all edges to point from the root outwards. It then applies basic message passing operations step-by-step from the root to the leaves until every node has received a message. Then we reverse the orientation of all edges and update the clique and separator potentials from the leaves back to the root obeying the partial order given by the new orientations of the edges. After all messages have been passed, the final clique potentials equal the marginals, $\tilde{\psi}_C(x_C) = \sum_{x_u \notin C} \prod_{B \in \mathscr{C}} \psi_B(x_B)$, and likewise for the final separator potentials.

REMARK 4.1 When the junction tree algorithm is used for probability distributions the clique potential functions are positive, but it works just as well for complex valued functions.

The complexity of the junction tree algorithm depends on the triangulation that has been computed. It is exponential in the size of the largest clique of the chosen triangulation. Thus, at best, it is exponential in the *treewidth* of the graph, which is one less than the smallest size of the largest clique over all possible triangulations [**20**, Chapter 2].

### 4.2 *Contracting tensor networks via duality*

To compute a contracted tensor network state, we contract all edges in its tensor network that are not dangling. Our framework allows us to do this via duality and to provably show the hardness of this computation since computing marginals on the graphical model's side is widely studied [**20**]. The recipe is as follows. We consider the dual graphical model to the tensor hypernetwork. We make a new clique in the graphical model consisting of all vertices corresponding to the dangling edges of the tensor hypernetwork. The tensor hypernetwork state is the marginal distribution of that clique. We can then use, e.g., the junction tree algorithm to compute it. The following example illustrates how to use the junction tree algorithm to contract a tensor network.

EXAMPLE 4.1 Consider a PEPS tensor network on 4 vertices (see left of Fig. 1, cf. Example 2.4).

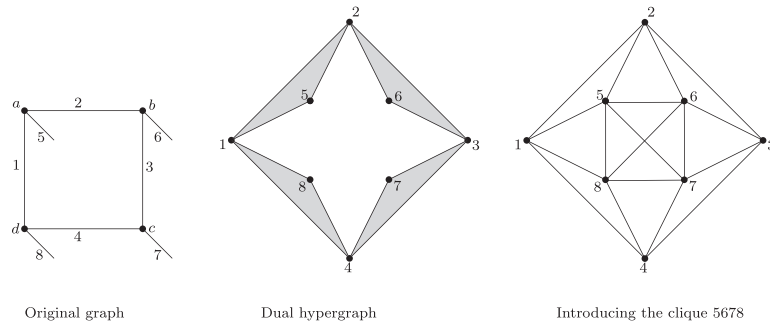



FIG. 1. The PEPS tensor network on four states, its transformation via duality and the addition of the 5678 clique.

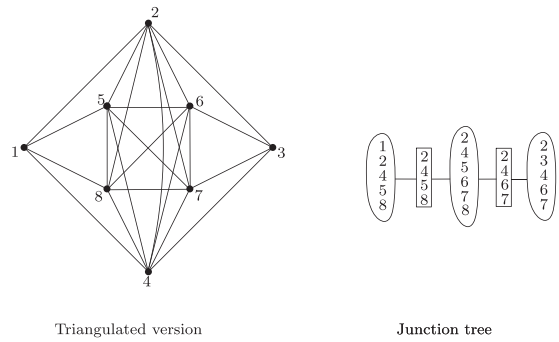



FIG. 2. A triangulation of the graph and its junction tree.

The tensor network consists of four arrays

$$T_a \in \mathbb{C}^{n_1 \times n_2 \times n_5}, \quad T_b \in \mathbb{C}^{n_2 \times n_3 \times n_6}, \quad T_c \in \mathbb{C}^{n_3 \times n_4 \times n_7}, \quad T_d \in \mathbb{C}^{n_1 \times n_4 \times n_8},$$

where $n_1, n_2, \ldots, n_8$ are the dimensions of the vector spaces at the eight edges. Contracting the tensor network gives $T \in \mathbb{C}^{n_5 \times n_6 \times n_7 \times n_8}$ with entries:

$$T_{i_5,i_6,i_7,i_8} = \sum_{i_1,i_2,i_3,i_4} (T_a)_{i_1,i_2,i_5} (T_b)_{i_2,i_3,i_6} (T_c)_{i_3,i_4,i_7} (T_d)_{i_4,i_1,i_8}. \tag{4.3}$$

The junction tree algorithm gives a fast way to compute this sum. If the entanglement edge dimensions are $n_1 = n_2 = n_3 = n_4 = r$ and the dangling edge dimensions are $n_5 = n_6 = n_7 = n_8 = n$, we can compute the contracted tensor in time and space $\mathcal{O}(n^3 r^2 + n^2 r^4)$, whereas summing term-by-term is $\mathcal{O}(n^4 r^4)$.

First, we find the dual hypergraph to the original graph (see middle of Fig. 1). We consider its graph skeleton and add a clique corresponding to the vertices 5, 6, 7 and 8, whose marginal $T$ we seek (right of Fig. 1). We triangulate the new graph (left of Fig. 2) and form its junction tree (right of Fig. 2). We

assign clique potentials to each of the cliques and separators of the junction tree as follows:

$$\psi_{12458}(i_1,i_2,i_4,i_5,i_8) = (T_a)_{i_1,i_2,i_5}(T_d)_{i_4,i_1,i_8}, \quad \psi_{2458} = \psi_{245678} = \psi_{2467} = 1,$$

$$\psi_{23467}(i_2,i_3,i_4,i_6,i_7) = (T_b)_{i_2,i_3,i_6}(T_c)_{i_3,i_4,i_7}.$$

We now perform the junction tree algorithm. We choose the left vertex of the junction tree, i.e. 12458, as the root and proceed from left to right:

$$\tilde{\psi}_{2458}(i_2,i_4,i_5,i_8) = \sum_{i_1} \psi_{12458}(i_1,i_2,i_4,i_5,i_8),$$

$$\tilde{\psi}_{245678}(i_2,i_4,i_5,i_6,i_7,i_8) = \frac{\tilde{\psi}_{2458}(i_2,i_4,i_5,i_8)}{\psi_{2458}(i_2,i_4,i_5,i_8)}\psi_{245678}(i_2,i_4,i_5,i_6,i_7,i_8),$$

$$\tilde{\psi}_{2467}(i_2,i_4,i_6,i_7) = \sum_{i_5,i_8} \tilde{\psi}_{245678}(i_2,i_4,i_5,i_6,i_7,i_8),$$

$$\tilde{\psi}_{23467}(i_2,i_3,i_4,i_6,i_7) = \frac{\tilde{\psi}_{2467}(i_2,i_4,i_6,i_7)}{\psi_{2467}(i_2,i_4,i_6,i_7)}\psi_{23467}(i_2,i_3,i_4,i_6,i_7).$$

Then, we repeat the process going back to the root 12458. The first two steps of the updates, returning to the root, are as follows:

$$\tilde{\tilde{\psi}}_{2467}(i_2,i_4,i_6,i_7) = \sum_{i_3} \tilde{\psi}_{23467}(i_2,i_3,i_4,i_6,i_7),$$

$$\tilde{\tilde{\psi}}_{245678}(i_2,i_4,i_5,i_6,i_7,i_8) = \frac{\tilde{\tilde{\psi}}_{2467}(i_2,i_4,i_6,i_7)}{\tilde{\psi}_{2467}(i_2,i_4,i_6,i_7)}\tilde{\psi}_{245678}(i_2,i_4,i_5,i_6,i_7,i_8).$$

The desired marginal over $5,6,7,8$ equals $\sum_{i_2,i_4} \tilde{\tilde{\psi}}_{245678}(i_2,i_4,i_5,i_6,i_7,i_8)$.

Note that for this particular graph, the complexity $\mathcal{O}(n^3r^2 + n^2r^4)$ of the junction tree algorithm can also be achieved by factorizing $T$ as

$$T_{i_5,i_6,i_7,i_8} = \sum_{i_2,i_4} \left( \sum_{i_1} (T_a)_{i_1,i_2,i_5}(T_d)_{i_4,i_1,i_8} \right) \left( \sum_{i_3} (T_b)_{i_2,i_3,i_6}(T_c)_{i_3,i_4,i_7} \right).$$

For more general graphs, finding a way to factor the contracted tensor network in order to match the performance of the junction tree algorithm is more difficult.

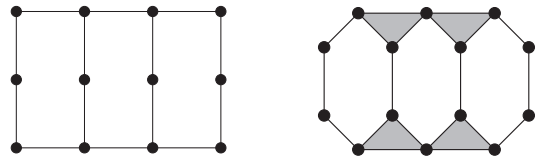


FIG. 3. The MPS tensor network on four states contracted with itself (left). Its dual graphical model (right).

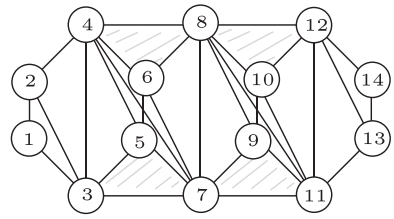



FIG. 4. A triangulation of the dual of a matrix product state.

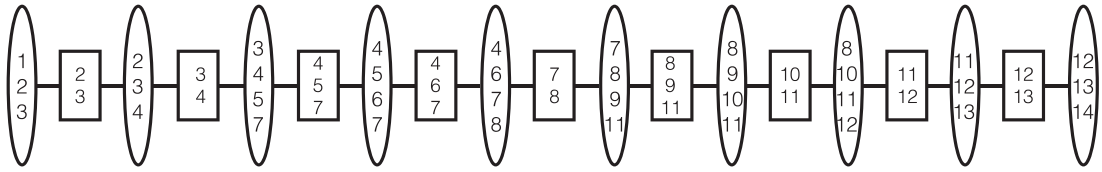



FIG. 5. The junction tree of the chordal graph in Fig. 4. The cliques are in ovals; the separators are boxed.

### 4.3 *Computing expectation values for matrix product states*

We now explain how to compute *expectation values* of MPS tensor networks (see Fig. 3) using the junction tree algorithm for tensor hypernetworks. Using Theorem 2.1 we compute the family of graphical models that is dual to matrix product states. In the figures below, we draw the MPS with four observable indices, but repeating the pattern gives the results in the general case. We show that the junction tree algorithm used to compute marginalizations of the dual graphical model corresponds to the *bubbling* algorithms that are used to compute expectation values of an MPS [**14**].

In quantum applications a tensor network state is denoted $|\psi\rangle$. Its expectation value is the inner product $\langle\psi|A|\psi\rangle$ for some operator $A$. We consider the case that $A$ is block diagonal, which means $A$ transforms a tensor network state $|\psi\rangle$ by a linear transformation in each of its vector spaces of observable indices. The method we describe can be extended to operators of interest which are not block diagonal.

Computing the expectation value of an MPS means contracting the tensor network on the left in Fig. 3, where the middle row of vertices correspond to the blocks of $A$. Equivalently, it means marginalizing all variables of the graphical model on the right (or, computing the normalization constant of this graphical model). We contract the tensor network by applying the junction tree algorithm to the graphical model.

The first step of the algorithm is to triangulate the graph of the graphical model, by adding edges until it is chordal (or triangulated), see Fig. 4. Next, we form a junction tree for the triangulated graph, see Fig. 5.

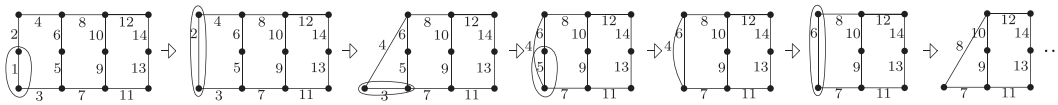


FIG. 6. Order of contraction in the MPS tensor network to compute its expectation value.

We choose the root of the tree to be the left-most vertex in Fig. 5. We first orient all edges to point away from the root, i.e. in this case from left to right, and then we perform basic message passing operations along the directed edges until every vertex has received a message from its parent. We arrive at the right-most clique $\{12, 13, 14\}$. At this point, the function recorded at the clique $\{12, 13, 14\}$ in fact equals the marginal at that clique. Thus, in order to compute the total sum, we can simply sum over the three vertices 12, 13 and 14. Therefore, in this case, we do not need to run the second step of the junction tree algorithm in which we have to reverse the orientations of all edges and pass messages all the way back to the root.

We now translate the junction tree algorithm to the language of tensor networks. The junction tree determines the order in which to contract the indices of the tensor network, see Fig. 6. We contract edges in the tensor network until it is completely contracted.

At each step we sum over just one vertex of the dual graphical model (due to the structure of the junction tree in this case). This means we contract one edge at a time from the tensor network. In the first message passing operation we have $C_1 = \{1, 2, 3\}$, $C_2 = \{2, 3, 4\}$, $S = \{2, 3\}$. We sum over the values of vertex 1, since it is the only variable in $C_1 \backslash S$. This corresponds to contracting the tensor along the edge corresponding to vertex 1 of the graphical model (see step one of Fig. 6 for the corresponding tensor network operation). In the second message passing operation we sum over the values of vertex 2 of the graphical model. This corresponds to contracting the tensor network along the left edge (see the second step of Fig. 4). The subsequent steps of the junction tree algorithm correspond to the steps shown in Fig. 4.

The triangulated graph of the dual graph of MPS has a treewidth of size four, since we can continue the triangulation given in Fig. 4. We can compute the complexity of the junction tree algorithm to be $O(|V|(nr^3 + n^2r^2))$, where $|V|$ is the number of vertices in the MPS, $n$ is the size of the dangling edges and $r$ is the size of the entanglement edges. It turns out that contracting the tensor in this way is what is usually done by the tensor network's community as well, a method sometimes called bubbling [**14**]. Similar algorithms are used in the case of MPS with periodic boundary conditions, e.g. the algorithm in [**16**] which runs in $O(|V|nr^5)$. A numerical algorithm for an infinitely long MPS chain which runs in $O(nr^3)$ is given in [**18**].

Our duality results, together with the junction tree algorithm, provide a method for computing expectation values for any tensor network. In contrast, existing methods for computing expectation values are usually addressed on a case-by-case basis depending on the structure of the tensor network. We conclude the paper with a brief discussion of another main example, the two-dimensional generalization of MPS.

### 4.4 *Extending to larger dimensions*

The higher-dimensional analogue of matrix product states/tensor trains is called the PEPS, see Example 2.4. They are based on a two-dimensional lattice of entanglement interactions. Computing expectation values for the PEPS network takes exponential time in the number of states of the network

[14]. On the graphical model's side, it is possible in principle to find expectation values of a PEPS state using the junction tree algorithm. Since the triangulated graph of the dual hypergraph of PEPS has a tree width that grows in the size of the network, the junction tree algorithm is exponential time.

In [12], the authors show that algorithms for computing expectation values are exponential in the treewidth of the tensor network. On the other hand, we have seen that the junction tree algorithm is exponential time in the treewidth of the dual graphical model. This indicates a similarity between the treewidth of a hypergraph and of its dual. A comparison of the treewidths of planar graphs and of their graph duals can be found in [17].

To avoid exponential running times, numerical approximations are used [13,14,21]. For graphical models, these are termed *loopy belief propagation* (see [20, Chapter 4] and references therein). A natural question is whether the algorithms for loopy belief propagation translate to known algorithms in the tensor network's community, e.g. for computing expectation values of PEPS, or whether they provide a new family of algorithms. In our opinion both answers to this question would be interesting.

## Acknowledgements

We would like to thank Jason Morton, Bill Huggins and Bernd Sturmfels for helpful discussions.

## Funding

National Science Foundation Mathematical Postdoctoral Sciences Research Fellowship (DMS1703821) to E.R.

## References

1. Bailly, R., Denis, F. & Rabusseau, G. (2015) Recognizable series on hypergraphs. *Language and Automata Theory and Applications, Lecture Notes in Computer Science, 8977*. Cham: Springer, pp. 639–651.
2. Banerjee, A., Char, A. & Mondal, B. (2017) Spectra of general hypergraphs. *Linear Algebra Appl.*, **518**, 14–30.
3. Berge, C. (1989) *Hypergraphs*, vol. 45 of Combinatorics of Finite sets. Amsterdam: North-Holland Publishing Co.
4. Borsuk, K. (1948) On the imbedding of systems of compacta in simplicial complexes. *Fund. Math.*, **35**, 217–234.
5. Chen, J., Cheng, S., Xie, H., Wang, L. & Xiang, T. (2017) On the equivalence of restricted boltzmann machines and tensor network states. *Phys. Rev. B*, **97**, 085104.
6. Critch, A. & Morton, J. (2014) Algebraic geometry of matrix product states. *SIGMA Symmetry Integrability Geom. Methods Appl.*, **10**.
7. Hackbusch, W. (2012) *Tensor Spaces and Numerical Tensor Calculus*, vol. 42. Springer Series in Computational Mathematics. Heidelberg: Springer.
8. Hatcher, A. (2002) *Algebraic Topology*. Cambridge: Cambridge University Press.
9. Kschischang, F., Frey, B. J. & Loeliger, H.-A. (2001) Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory*, **47**, 498–519.
10. Landsberg, J. M. (2017) Tensors and their uses in approximation theory, quantum information theory and geometry. Draft notes.
11. Loeliger, H.-A. & Vontobel, P. O. (2012) A factor-graph representation of probabilities in quantum mechanics. *IEEE Int. Symp. Inf. Theory,* Cambridge, MA, USA, pp. 656–660.
12. Markov, I. L. & Shi, Y. (2008) Simulating quantum computation by contracting tensor networks. *SIAM J. Comput.*, **38**, 963–981.

**13.** Nishino, T. & Okunishi, K. (1996) Corner transfer matrix renormalization group. *J. Phys. Soc. Jpn.*, **65**, 891–894.
**14.** Orús, R. (2014) A practical introduction to tensor networks: matrix product states and projected entangled pair state. *Ann. Physics*, **349**, 117–158.
**15.** Pejic, M. (2014) Quantum bayesian networks with application to games displaying parrondo's paradox. *Ph.D. Thesis*, Berkeley: University of California.
**16.** Porras, D., Verstraete, F. & Cirac, J. I. (2004) Density matrix renormalization group and periodic boundary conditions: a quantum information perspective. *Phys. Rev. Lett.*, **93** (227205).
**17.** Robertson, N. & Seymour, P. D. (1984) Graph minors. III. Planar tree-width. *J. Combin. Theory Ser. B*, **36**, 49–64.
**18.** Schollwöck, U. (2005) The density-matrix renormalization group. *Rev. Mod. Phys.*, **77**, 259–315.
**19.** Sullivant, S. (2017) *Algebraic Statistics*. Draft copy of book. http://www4.ncsu.edu/∼smsulli2/Pubs/asbook.pdfsmsulli2/Pubs/asbook.pdf
**20.** Wainwright, M. & Jordan, M. I. (2008) Graphical models, exponential families, and variational inference. *Foundation and Trends in Machine Learning*, **1**, 1–305.
**21.** Xie, Z. Y., Chen, J., Qin, M. P., Zhu, J. W., Yang, L. P. and Xiang, T. (2012) Coarse-graining renormalization by higher-order singular value decomposition. *Phys. Rev. B*, **86** (045139).