

On Probabilistic Inference Through Parametric Tensor Decomposition in Base Tensor Networks

Sagad Hamid and [Tanya Braun](#)

SUM 2026



Factorised Probabilistic Models

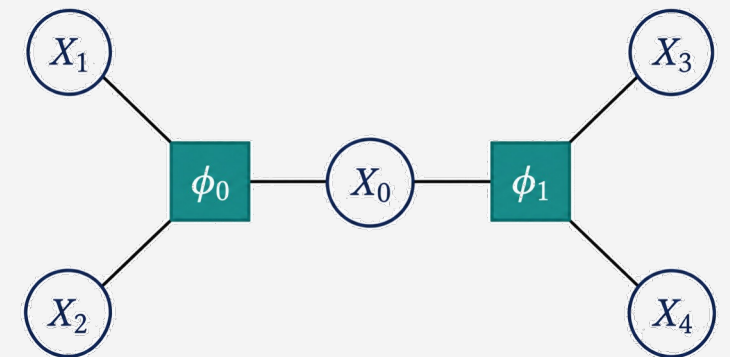
- Model $G =$
Set of m factors $\{\phi_i(\mathbf{X}_i)\}_{i=0}^{m-1}$ over random variables $\mathbf{X} = \bigcup_{i=0}^{m-1} \mathbf{X}_i$
 - Encodes a full joint probability distribution

$$P_G(\mathbf{X}) = \frac{1}{Z(G)} \prod_{i=0}^{m-1} \phi_i(\mathbf{X}_i)$$

- Z called *partition function*
- Inference problem
 - Compute answer to query $P(\mathbf{Q}|\mathbf{e})$, $\mathbf{Q}, \mathbf{e} \subset \mathbf{X}$
 - Sum out all variables not occurring in query (marginalisation):

$$P_G(\mathbf{Q}|\mathbf{e}) = \sum_{\mathbf{X}'} \frac{P_G(\mathbf{Q}, \mathbf{e}, \mathbf{X}')}{P_G(\mathbf{e}, \mathbf{X}')}$$

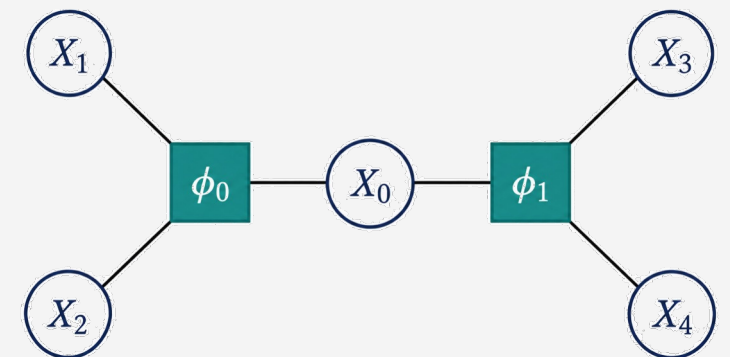
- Intractable* in general



Tractable Inference

- Exploit *parametric* structure
 - E.g., context-specific independence, determinism, symmetries
 - Encode in compact representations
 - E.g.,
parfactor graphs,
probabilistic circuits
 - Access during inference
 - E.g.,
lifted inference
→ tractability through exchangeability
[Niepert and Van den Broeck 2014]
- *What to do if not present?*

Introduce
parametric structure into model
→ Look at the problem through
the lens of tensors

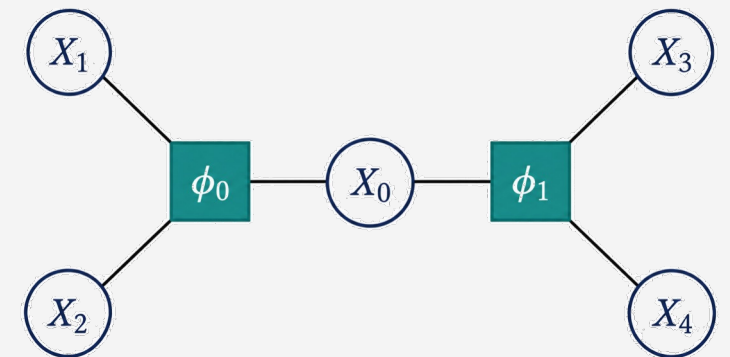


Tensors

- Tensor =
Array of numbers, addressed by indices
 - *Order* of a tensor n : number of indicesExamples:
 - $d = 0$: Scalar s (no index)
 - $d = 1$: Vector $\boldsymbol{v}$ (one index)
 - $d = 2$: Matrix $\boldsymbol{M}$ (two indices)
 - $d = n$: Tensor $\mathcal{T}$ (n indices)
- Tensor products
 - *Outer* product: $(Q \otimes R)_{i,j} = Q_i \otimes R_j$
 - *Hadamard* product: $(Q \odot R)_i = Q_i \odot R_i$

Factors as Tensors

- Discrete factor over n variables = Tensor with n indices
 - Example: Factor $\phi(X_0, X_1, X_2)$
 - Tensor with three indices
 - Assignment x_0, x_1, x_2 mapping to potential c = Entry c at index values x_0, x_1, x_2 , i.e.,
$$\phi(x_0, x_1, x_2) \mapsto c \quad \triangleq \quad \mathcal{T}_{x_0, x_1, x_2} = c$$
- Factor product vs. tensor product
 - Outer product:
Product of factors with disjoint variable sets
 - Hadamard product:
Product of factors with identical variable sets

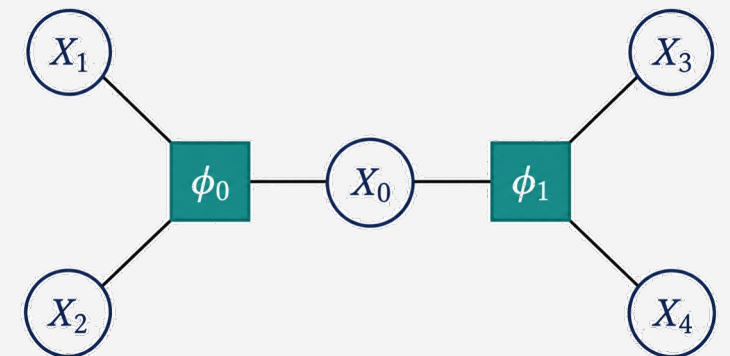


Tensor Networks and Contraction

- Tensor network =
Set of tensors with indices $T = (\mathcal{T}, I)$
 - Graphical representation
 - Nodes are tensors
 - Edges are indices
 - Index connected to two tensors called *shared* index
 - Index attached to single tensor called *free* index
- Encodes a contraction (result), denoted by $Z(\mathcal{T})$
 - *Contraction* =
Multiplication of tensors and sums over indices (= marginalisation)
 - Without any free indices, contraction results in a single number (= partition function)



$$C_{ij} = \sum_c A_{ic} B_{cj}$$



Base Tensor Networks for Efficient Inference

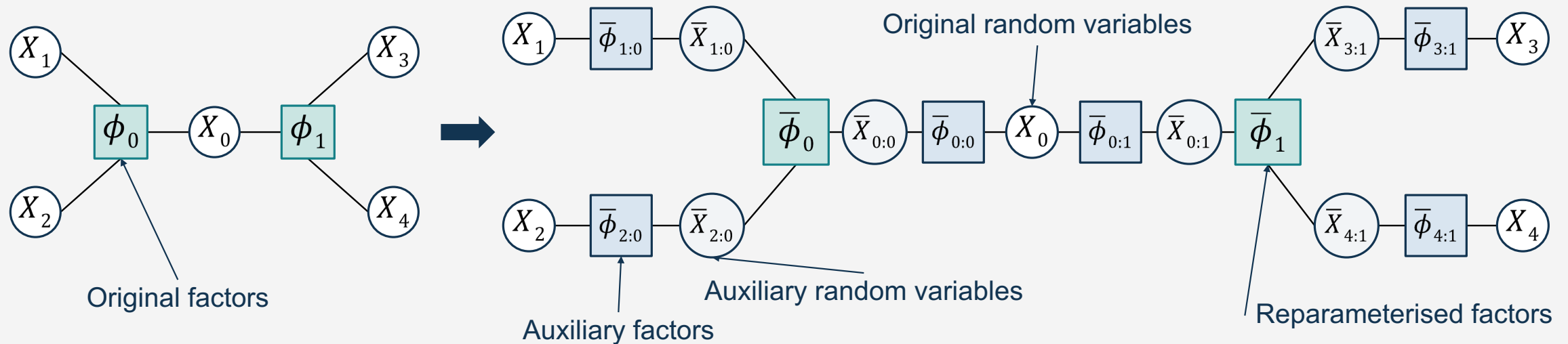
- Introduce structure into factor graphs to exploit for efficiency by turning them into tensor networks of a specific form, called base tensor network, which enable tractable inference
 - Compute partition function through contraction of a corresponding base tensor network
- Procedure
 1. Represent a set of factors G as base tensor network
 - Set of tensors with specific properties
 - Classes of *tractable tensors*
 2. Approximate base tensor as a weighted combination of tractable tensors
 - Choose class of tractable tensors
 - Compute weights through close-form solution to Frobenius norm
 3. Compute partition function as weighted combination of contraction results

1. From Factor Model to Base Tensor Network

a. Reparameterise original factors

Replace each random variable X_j in a factor ϕ_i of model G with

- an auxiliary variable $\bar{X}_{j:i}$ and
- an auxiliary factor $\bar{\phi}_{j:i}(X_j, \bar{X}_{j:i})$



1. From Factor Model to Base Tensor Network

a. Reparameterise original factors

Replace each random variable X_j in a factor ϕ_i of model G with

- an auxiliary variable $\bar{X}_{j:i}$ and
- an auxiliary factor $\bar{\phi}_{j:i}(X_j, \bar{X}_j)$

Turns $\phi_i(\mathbf{X}_i)$ into $\bar{\phi}_i(\bar{\mathbf{X}}_i)$ with

$$\phi_i(\mathbf{X}_i) = \sum_{\bar{\mathbf{X}}_i} \bar{\phi}_i(\bar{\mathbf{X}}_i) \prod_j \bar{\phi}_{j:i}(X_j, \bar{X}_j)$$

- $\bar{\phi}_{j:i}$ needs to be invertible as a tensor to preserve the original factor
 - $\bar{\phi}_i(\bar{\mathbf{X}}_i)$ can map to the same numbers as $\phi_i(\mathbf{X}_i)$; then the $\bar{\phi}_{j:i}$ are diagonal matrices
 - Map to 1 if inputs have the same value
 - *Can choose any numbers as long as above equation holds*

1. From Factor Model to Base Tensor Network

b. Eliminate original variables

Sum out all original variables of model G

- Occur only in auxiliary factors together with an auxiliary variable
 - Number of auxiliary factors / variables = How often original variable occurred in any of the m factors
 - At most $m + 1$ variables involved
- Leaves the reparameterised original factors and the reduced auxiliary factors

c. Interpret result as tensor network

Model G of m factors turned into a so-called *base tensor network* T

- Contains only shared indices
- It holds $Z(G) = Z(T)$

1. From Factor Model to Base Tensor Network

a. After reparameterisation

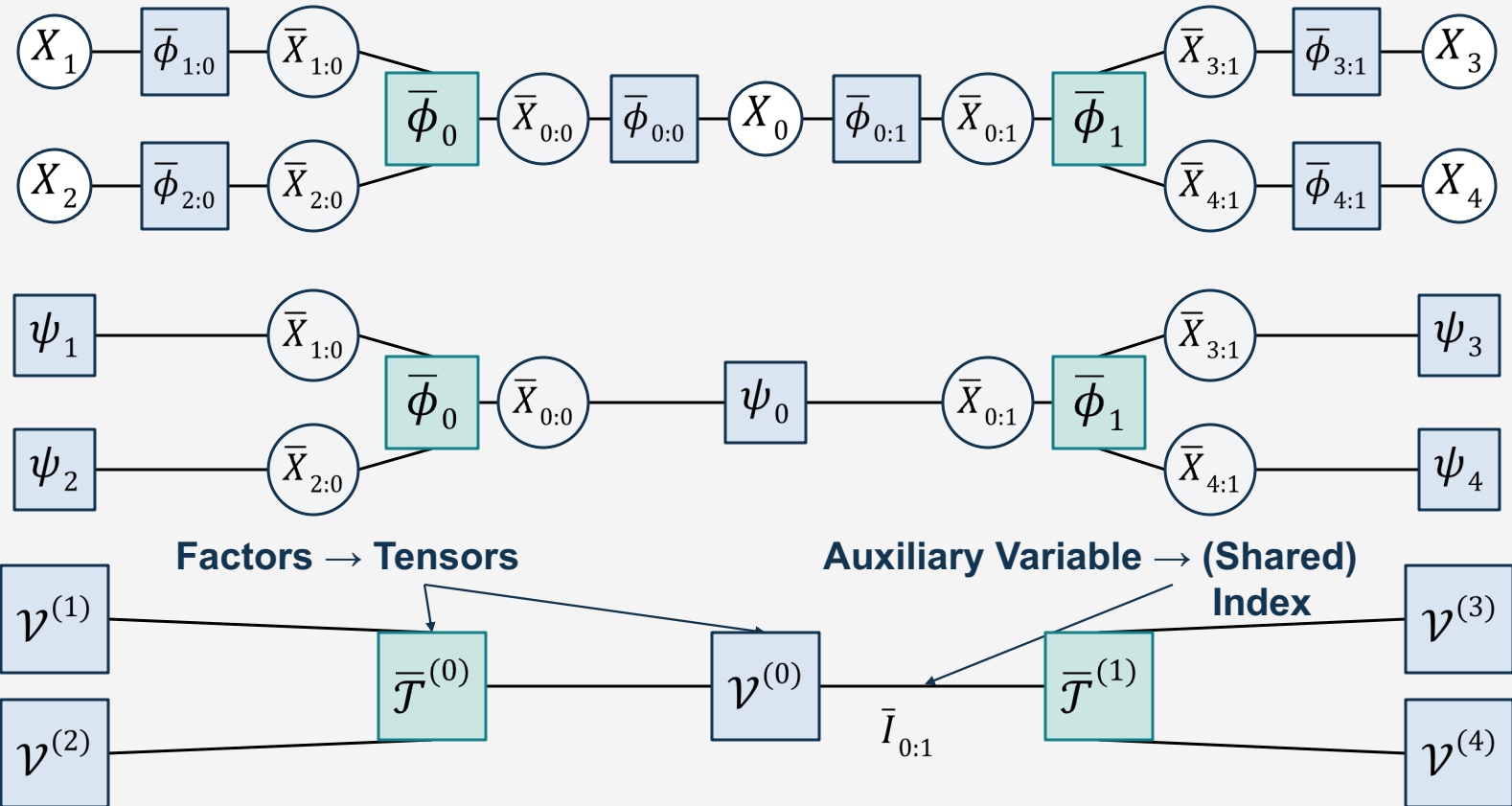


b. Sum out random variables



c. Interpret as tensor network

$$Z(G) = Z(T)$$



Base Tensor Network

- Base tensor network =
Tensor network with no free indices, consisting of
 - So-called *core tensors* $\mathcal{C} = \{\mathcal{C}^{(0)}, \dots, \mathcal{C}^{(q-1)}\}$
 - Disjoint index sets among core tensors that together cover all network indices
 - I.e., the reparameterised original factors
 - Outer product of q core tensors forms the *base tensor*

$$\mathcal{B} = \bigotimes_{j=0}^{q-1} \mathcal{C}^{(j)} = \bigotimes_{j=0}^{m-1} \bar{T}^{(j)}$$

- So called *linked tensors* $\mathcal{L}$
 - Disjoint index sets among linked tensors, connected to core tensors
 - I.e., the reduced auxiliary factors with a single shared index

Tractable Base Tensors

- n number of indices, each with d values
- k number of linked tensors
- n_{max} largest number of indices on any linked tensor
- r number of decomposition components

$$c_1 = \binom{n_{max} + d - 1}{d - 1}$$

$$c_2 = \binom{n - n_{max} + d - 1}{d - 1}$$

Base Tensor	Whole BTN Contraction Cost	Brief Explanation
Rank-one Tensor	$\mathcal{O}(kd^{n_{max}})$	One vector per index separates global index dependence
CP Decomposition	$\mathcal{O}(rkd^{n_{max}})$	Adds r rank-one tensors
Symmetric Tensor	$\mathcal{O}(k(d^{n_{max}} + c_1c_2))$	Reordering indices leaves tensor entry unchanged
Symmetry-CP Structure	$\mathcal{O}(rk(d^{n_{max}} + c_1c_2))$	Combine local rank-one computations with count spaces

2. Approximate Base Tensor through Weighted Tractable Decomposition

a. Decompose the base tensor into tractable components (keep linked tensors fixed)

- Exact additive decomposition

$$\mathcal{B} = \sum_{j=0}^{q-1} \alpha_j \mathcal{B}^{(j)}$$

- With $Z(\mathcal{B}) = \sum_{j=0}^{q-1} \alpha_j Z(\mathcal{B}^{(j)})$

b. Choose tractable components $\tilde{\mathcal{B}}^{(j)}$ for $\mathcal{B}^{(j)}$

- Base tensor $\mathcal{B}$ is product of core tensors but the core tensors are not necessarily tractable

c. Compute weights α such that

$$\mathcal{B} \approx \tilde{\mathcal{B}} = \sum_{j=0}^{q-1} \alpha_j \tilde{\mathcal{B}}^{(j)}$$

2. Approximate Base Tensor through Weighted Tractable Decomposition

c. Compute weights α such that

$$\mathcal{B} \approx \tilde{\mathcal{B}} = \sum_{j=0}^{q-1} \alpha_j \tilde{\mathcal{B}}^{(j)}$$

Closed form solution

$$\min_{\alpha} (\|\mathcal{B}\|_F^2 + \alpha^T G \alpha - 2 \alpha^T b)$$

Minimise $\|\mathcal{B} - \tilde{\mathcal{B}}\|_F^2$

$$\|\mathcal{B} - \tilde{\mathcal{B}}\|_F^2 = \left\| \mathcal{B} - \sum_{j=0}^{q-1} \alpha_j \tilde{\mathcal{B}}^{(j)} \right\|_F^2 = \underbrace{\|\mathcal{B}\|_F^2}_{\text{Norm of original tensor}} + \sum_{j_1, j_2=0}^{q-1} \alpha_{j_1} \alpha_{j_2} \underbrace{\langle \tilde{\mathcal{B}}^{(j_1)}, \tilde{\mathcal{B}}^{(j_2)} \rangle_F}_{\text{Distance between components}} - 2 \sum_{j=0}^{q-1} \alpha_j \underbrace{\langle \mathcal{B}, \tilde{\mathcal{B}}^{(j)} \rangle_F}_{\text{Distance to original tensor}}$$

Norm of original tensor

$$\|\mathcal{B}\|_F^2 = \prod_{k=0}^{q-1} \|\mathcal{B}^{(k)}\|_F^2$$

Distance between components

$$G_{ij} = \langle \tilde{\mathcal{B}}^{(j_1)}, \tilde{\mathcal{B}}^{(j_2)} \rangle_F$$

Distance to original tensor

$$b_i = \langle \mathcal{B}, \tilde{\mathcal{B}}^{(j)} \rangle_F$$

3. Compute Partition Function by Contraction

- Contract the component networks and add the results
 - For base tensor $\mathcal{B}$ decomposed into chosen tractable components $\tilde{\mathcal{B}} = (\tilde{\mathcal{B}}^{(0)}, \dots, \tilde{\mathcal{B}}^{(q-1)})$ with learned weights α , it holds:

$$Z(\mathcal{B}) \approx Z(\tilde{\mathcal{B}}) = \sum_{j=0}^{q-1} \alpha_j Z(\tilde{\mathcal{B}}^{(j)})$$

- To compute partition function of a model G with a corresponding base tensor network approximated as above, include linked tensors $\mathcal{L}$ in contraction above; then, it holds

$$Z(G) = Z(\mathcal{B} \cup \mathcal{L}) \approx Z(\tilde{\mathcal{B}} \cup \mathcal{L})$$

Summary

- Introduce structure into factor graphs to exploit for efficiency by turning them into tensor networks of a specific form, called *base tensor network*, which enable tractable inference
- Procedure
 1. Represent a set of factors G as base tensor network
 - Set of tensors with shared indices only: *core* tensors and *linked* tensors
 - Classes of *tractable tensors*
 2. Approximate base tensor as a weighted combination of tractable tensors
 - Choose class of tractable tensors
 - Compute weights through close-form solution to Frobenius norm
 3. Compute partition function as weighted combination of contraction results
- *Future work*: Look at other ways of learning base tensor representations, find further tractable class, extend the query language, ...



<https://arxiv.org/abs/2609.23774>