Probabilistic inference reformulated as tensor networks

Jin-Guo Liu (刘金国)
Quantum information & Scientific computing

香港科大(广州)
HKUST(GZ)

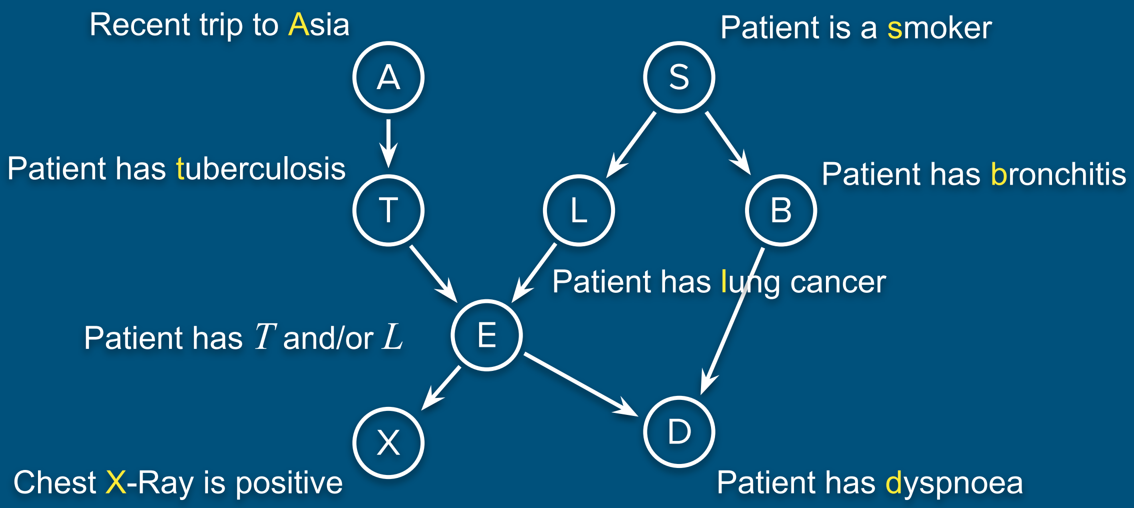
Contents

1. Reformulate probabilistic inference tasks in "**tensor**" language
2. A brief introduction to related **toolchains**

Probabilistic Modeling

Bayesian network: Asia

Each variable ~ {0, 1}, 0 for false, 1 for true.



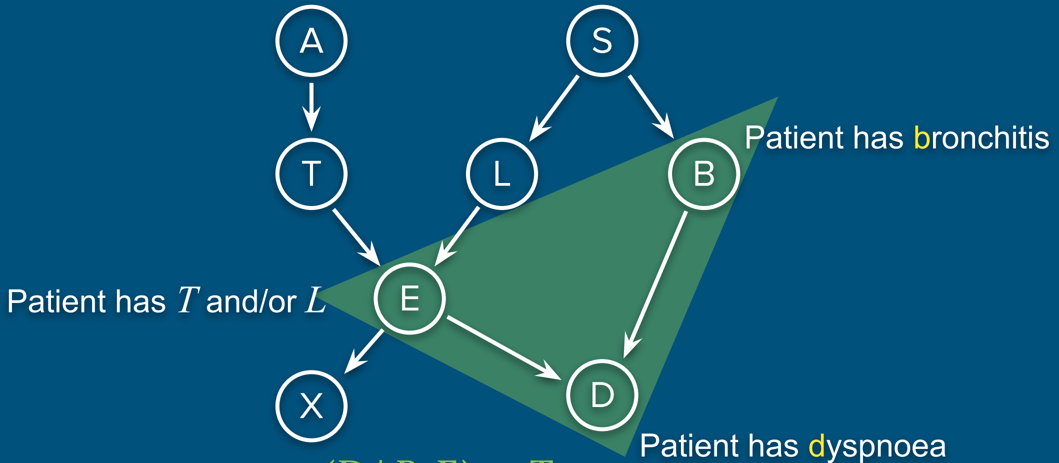
Probabilistic Inference Tasks

- Probability of evidence (PR): Calculates the total probability of the observed evidence across all possible states of the unobserved variables.
- Marginal inference (MAR): Computes the probability distribution of a subset of variables, ignoring the states of all other variables.
- Maximum a Posteriori Probability estimation (MAP): Finds the most probable state of a subset of unobserved variables given some observed evidence.
- ~~Marginal Maximum a Posteriori (MMAP): Finds the most probable state of a subset of variables, averaging out the uncertainty over the remaining ones.~~
- Sampling: Sample variables unbiasedly from the probability distribution.

Source: Uncertainty in Artificial Intelligence (UAI) 2022 competition

Mathematical Model	Tasks	Techniques	Software
Graphical models (Book: PRML)	Exact inference	Belief propagation	JunctionTrees.jl
	Approximate inference	Optimal tree decomposition (Althaus 2021)	Merlin, libDAI
			toulbar2
Tensor network (Orus 2013)	Generative Modeling (Han 2018)	Differential programming (Liao 2019)	Pytorch
	Quantum simulation (Pan 2021, Gao 2021)	Hyper-optimized contraction order (Gray 2021)	ITensor(s.jl)
	Combinatorial optimization (Liu 2023)	Generic tensor network (Liu 2023)	OMEinsum.jl

Tensor networks can computation the probability of a set of variables (PR)

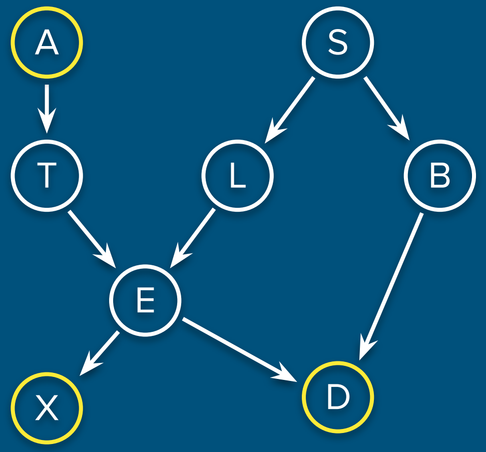


$p(D \mid B, E) \rightarrow T_{DBE}$

Tensors : $\{T_{XE}, T_{DBE}, T_{ETL}, T_{BS}, T_{LS}, T_{S}, T_{TA}, T_{A}\}$

Probabilistic Model

$$p(A, X, D) = \text{sum}(p(X \mid E) * p(D \mid B, E) \\ * p(E \mid T, L) \\ * p(B \mid S) * p(L \mid S) * p(S) \\ * p(T \mid A) * p(A), \{S, T, L, B, E\})$$

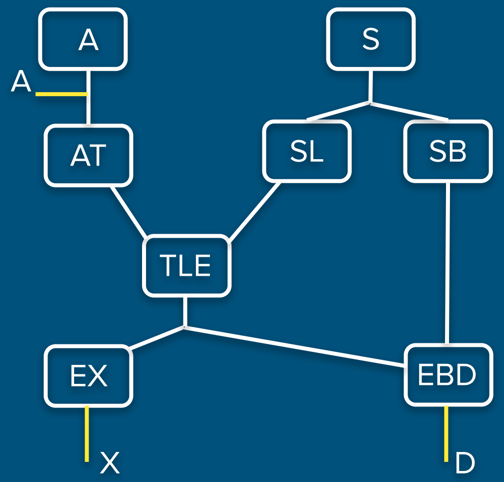


Tensor Network

Variables Λ: $\{A, S, T, L ,B, E, X, D\}$
Tensors T: $\{T_{XE}, T_{DBE}, T_{ETL}, T_{BS}, T_{LS}, T_S, T_{TA}, T_A\}$
Output σ_o: $\{A, X, D\}$

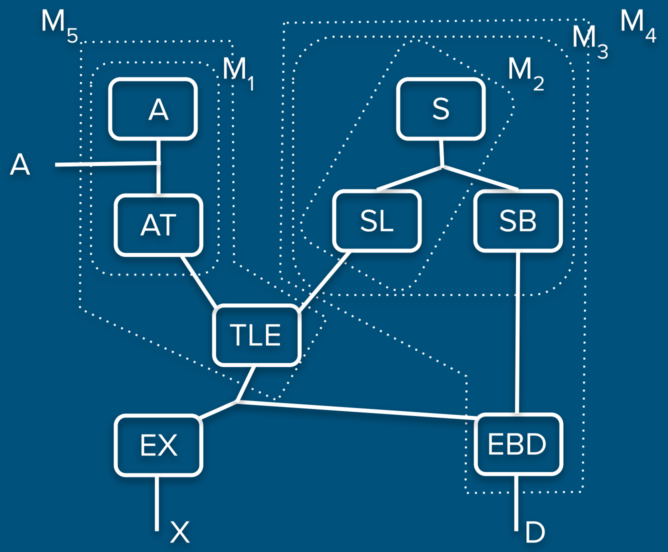
$p(A, X, D) = \texttt{contract}(\Lambda, T, \sigma_o)$

"`contract`" is the sum-product operation on tensor networks



(Robeva 2019)

Contracting a tensor network



M_1 = contract({A, T}, {T_A, T_{AT}}, {A, T})

M_2 = contract({S, L}, {T_S, T_{SL}}, {S, L})

M_3 = contract({S, L, B}, {M_2, T_{SB}}, {L, B})

M_4 = contract({E,L,B,D}, {M_3, T_{EBD}}, {E, D, L})

M_5 = contract({T, L, E, A}, {M_1, T_{TLE}}, {A, L, E})

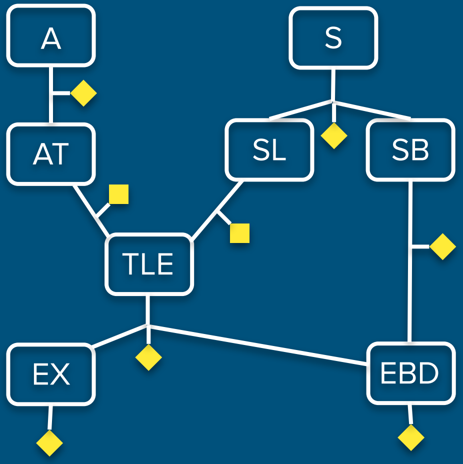
M_6 = contract({A, L, E, D}, {M_4, M_5}, {A, E, D})

p(A, X, D) = contract({A, E, D, L, E}, {M_5, M_6}, {A, X, D})

Time complexity = $O(n^{\#\text{ of variables}})$

(Markov 2008, Pan 2022, Kalachev 2022, Liu 2023)

Tensor networks can compute marginal probabilities (MAR) **efficiently**



—◆ id = (1, 1)

Probabilistic Model

p(A) = `sum`(p(A, S, T, L ,B, E, X, D), {S, T, L, B, E, X, D})
p(X) = `sum`(p(A, S, T, L ,B, E, X, D), {A, S, T, L, B, E, D})
...

Tensor Network

Variables: Λ
Tensors: augT = T $\cup$ $\{id_A, id_S, id_T, id_L, id_B, id_E, id_X, id_D\}$
Output: σ_o

{p(A), p(X), ...} = `gradient`(`contract`(Λ, augT, σ_o), $\{id_A, id_X, ...\}$)

"`gradient(f(args...), vars)`" computes the gradients $\{\partial f(args...)/\partial v \mid v \in vars\}$ with O(1) overhead

Tensor Network

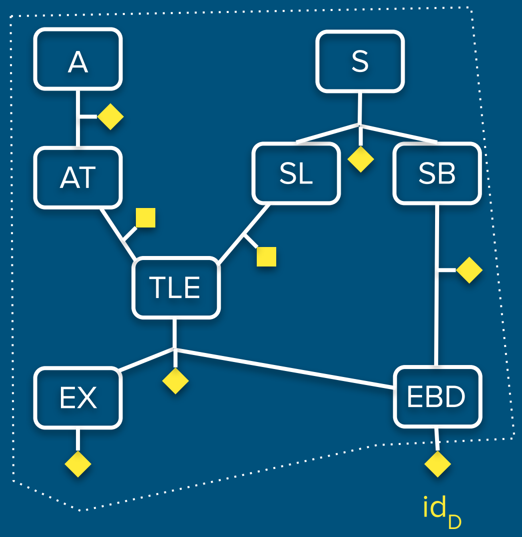
Variables: Λ
Tensors: $augT = T \cup \{id_A, id_S, id_T, id_L, id_B, id_E, id_X, id_D\}$
Output: σ_o

(1) $\texttt{contract}(\Lambda, augT, \sigma_o) = \texttt{contract}(\Lambda, T, \sigma_o)$

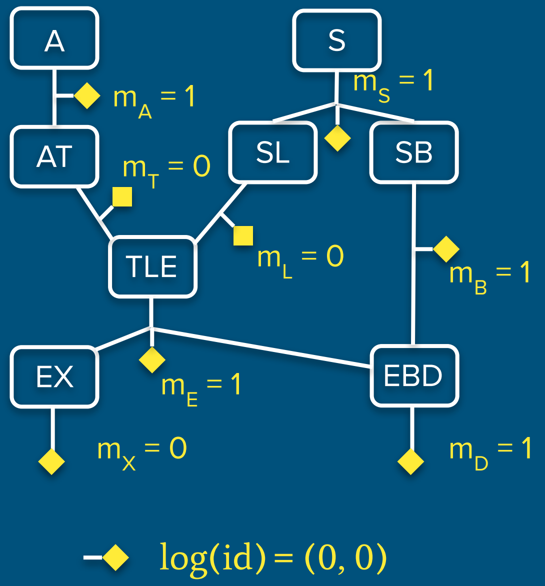
(2) $\texttt{contract}(\Lambda, augT, \sigma_o) = \texttt{sum}(p(v) * id_v(v), \{v\})$ for any $v \in \Lambda$

(3) $\texttt{gradient}(\texttt{contract}(\Lambda, augT, \sigma_o), id_v) = p(v)$

(4) key: gradients on all leaf variables can be computed with a single pass. The overhead is a constant: 3.



Tensor networks can find the most probable state (MAP) efficiently



(Liu 2021)

Probabilistic Model

$\log(p_m) = \mathtt{max}(\log(p(A, S, T, L, B, E, X, D)), \{A, S, T, L, B, E, X, D\})$
$m = \mathtt{argmax}(\log(p(A, S, T, L, B, E, X, D)), \{A, S, T, L, B, E, X, D\})$

Tropical Tensor Network

Variables : Λ
Tensors: $\log T = \{\log(t) \mid t \in \mathrm{aug}T\}$
Output: $\sigma_o = \{\}$

$$\begin{aligned}\log(p_m) &= \mathtt{tropical\_contract}(\Lambda, \log T, \sigma_o) \\ &= \max(\log(T_{XE}) + \log(T_{DBE}) + \log(T_{ETL}) + \log(T_{BS}) + \log(T_{LS}) \\ &\quad + \log(T_S) + \log(T_{TA}) + \log(T_A) + 0_A + 0_S + 0_T + 0_L \\ &\quad + 0_B + 0_E + 0_X + 0_D, \Lambda)\end{aligned}$$

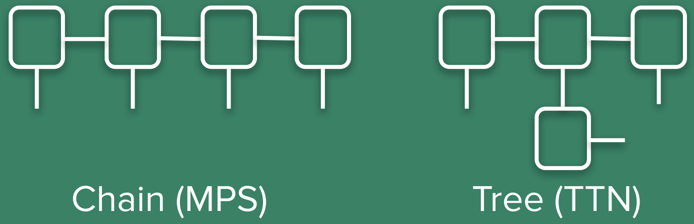
Tensor networks can sample variables unbiasedly from the probability distribution.

Goal: $s \sim p(\Lambda)$

Algorithm:

1. Compute `sum`$(p(\Lambda), \Lambda)$, cache intermediate results.
2. Initialize a sample over empty set $s \sim p(\{\})$.
3. **Back propagate** the sample s to leaves and obtain $s \sim p(\Lambda)$.

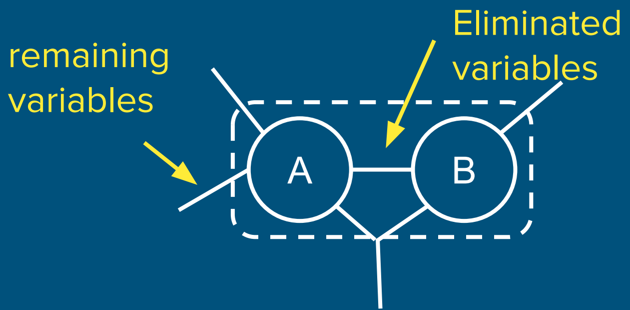
Known samping algorithms



(Han 2018, Cheng 2019)

Samples can be "back-propagated"

1. Each variable is eliminated at most once



2. Given a sample for the remaining variables

   $s \sim p(\text{remaining variables})$

   the value of eliminated variable can be sampled unbiasedly

$t \sim p(\text{eliminated variables} \mid \text{remaining variables})$

Tool: TensorInference

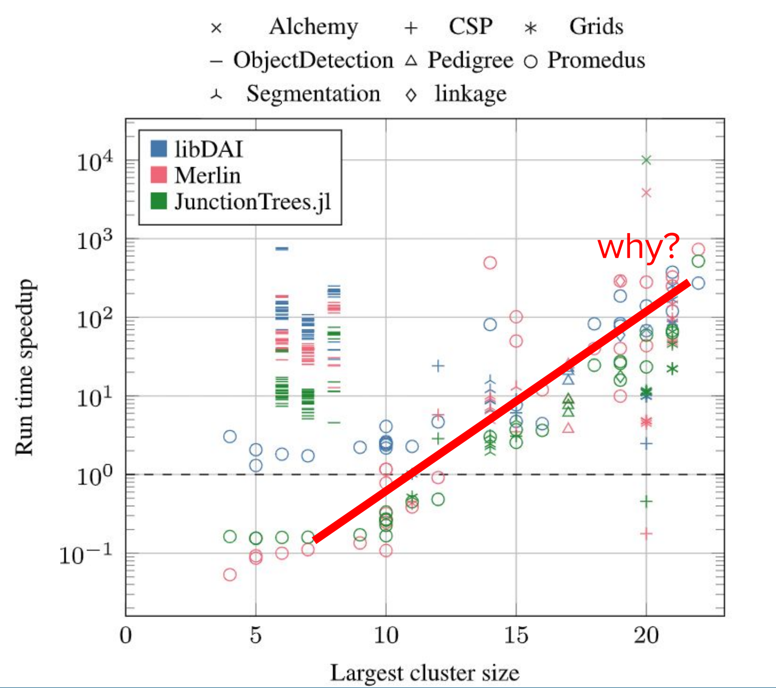
TensorInference.jl

- **OMEinsum** (Tensor network -> binary contraction -> BLAS or CuBLAS)
    - Greedy
    - Local Search (with slicing)
    - Min-Cut
- **TropicalGEMM** and **CuTropicalGEMM**

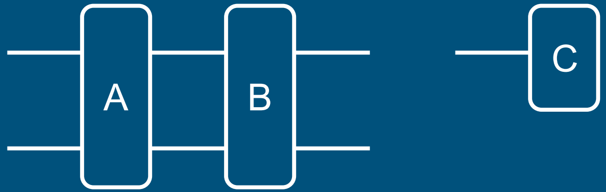
↘example↙

TensorBFS/TensorInference.jl

Benchmarks



Reason: Tensor network contraction optimizer also optimizes the time complexity

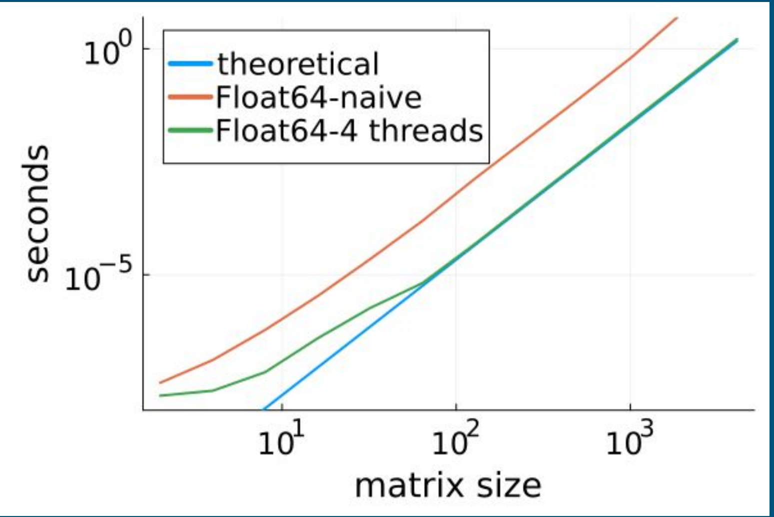


Space ~ $O(n^4)$, Time ~ $O(n^6)$

Space ~ $O(n^4)$, Time ~ $O(n^5)$

Faster Tropical tensor operations (CPU)

Collaborator: Chris Elrod
JuliaHub Inc.

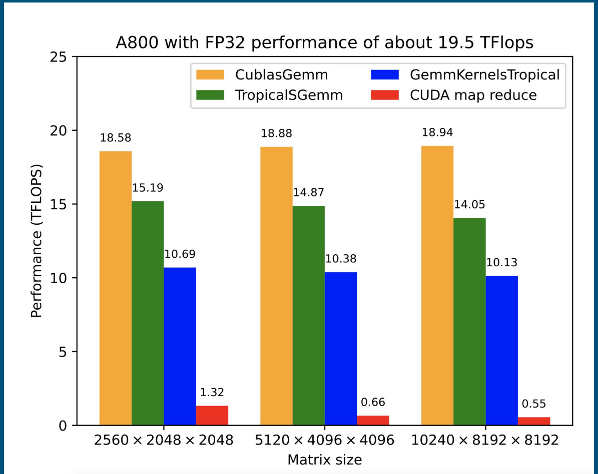


Github repo:
TensorBFS/TropicalGEMM.jl

Faster Tropical tensor operations (GPU)

Open Source Promotion Plan 2023
TropicalGEMM on GPU

Student: Xuan-Zhao Gao
Hong Kong University of Science and Technology (Guangzhou)



Github repo (work in progress):
ArrogantGao/CuTropicalGEMM.jl

Summarize

- Tensor network language simplifies the probabilistic inference tasks
  - No “message”
  - Back-propagation is implicit
- Tensor network language improves performance
  - Contraction order optimization with reduced time complexity
  - Make use of GEMM packages
  - GPU acceleration, slicing technique, et al.

Collaborator: Martin Roa Villescas
Eindhoven University of Technology

Code: [TensorBFS/TensorInference.jl]

JOSS paper under review:
[openjournals/joss-reviews/issues/5684]