MESSAGE PASSING FOR EFFICIENT CONTRACTIONS

RESEARCH NOTES IN THE ENEXA AND QROM PROJECTS

December 15, 2025

Motivation of message passing: Efficiency of computing contractions

- Global contractions with local open variables: Decomposed into efficient local contractions
- Batchwise computation: Get multiple contractions with a single message passing scheme

Guaranteed behavior of the message passing algorithm:

- Tree hypergraphs
- Boolean propagation: Always sound, complete when doing exact logical reasoning schemes of 2-SAT and Horn-SAT (might put the latter in the propositional inference chapter)
- Basis propagation: Show with the data example, that single datapoint is exact, batchwise not. Can also think of this as formulas (e.g. if X then $\neg Y$, if $\neg X$ then Y)

For CompActNets: Expectation Propagation schemes.

- Reduce to contraction propagation in case of graphical models
- Flexibility to model freedom: Hard / soft / hybrid

1 Contraction Propagation

Generic algorithm:

- Variable are the hypergraph and the scheduler
- Fixed are the trivial initial messages, the message system by overlaping edges, variables by edge intersections, computation of message by contraction

Definition 1. *Given a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ the overlap graph $\mathcal{G}^{\text{overlap}}$ consists of $\mathcal{E}$ as nodes and an edges*

$$\mathcal{E}^{\text{overlap}} = \left\{ \{e_0, e_1\} \,:\, e_0, e_1 \in \mathcal{E},\, e_0 \neq e,\, e_0 \cap e_1 \neq \varnothing \right\}.$$

We say that $\mathcal{G}$ is a tree-hypergraph, if $\mathcal{G}^{\text{overlap}}$ is a tree.

Messages are sent between overlapping edges, i.e. along the edges of $\mathcal{G}^{\text{overlap}}$. We further distinguish between messages $e_0 \to e_1$ and $e_1 \to e_0$ and build the directed overlap graph

$$\mathcal{G}^{\to} = (\mathcal{E}, \mathcal{E}^{\to}) = \left(\mathcal{E}, \bigcup_{\{e_0,e_1\} \in \mathcal{E}^{\text{overlap}}} \{(e_0, e_1), (e_1, e_0)\} \right).$$

.

1.1 Message Scheduler

The scheduler S maintains a subset of $\mathcal{E}^{\to}$ of messages to be sent, which updated (possibly without change) after each message. Once the set of messages to be sent is empty, the algorithm terminates.

Algorithm 1 Generic Contraction Propagation

Require: Tensor Network $\tau^{\mathcal{G}}$ on a hypergraph $\mathcal{G}$, scheduler S, message conditions C (optional)

Initialize $S.init(\mathcal{G}, C)$
Initialize messages
while not $S.empty()$ **do**
    Take a (e_0, e_1) pair from S
    Update the message

$$\chi_{e_0 \to e_1}[X_{e_0 \cap e_1}] = \langle \{\tau^{e_0}[X_{e_0}]\} \cup \{\chi_{e_2 \to e_0}[X_{e_2 \cap e_0}] \,:\, (e_2, e_0) \in \mathcal{E}^{\to}, e_2 \neq e_1\} \rangle_{[X_{e_0 \cap e_1}]}$$

    $S.update((e_0, e_1), \chi_{e_0 \to e_1}[X_{e_0 \cap e_1}])$
end while
return Messages $\{\chi_{e_0 \to e_1}[X_{e_0 \cap e_1}] \,:\, (e_2, e_0) \in \mathcal{E}^{\to}\}$

- init: Initializes the message schedule, based on the graph
- pop: Determines the next message by poping from the message schedule
- update: Updates the message schedule
- empty: Checks whether the schedule is empty

We investigate two methods of implementing schedulers:

- Receive Scheduler: Messages are sent exactly once, when a condition on the received messages is met.
- Change Scheduler: Messages are resent, when a message received has been changed (given a criterion).

1.2 Receive Scheduler

Examples of using the Receive Scheduler (Algorithm 2) with differing message condition dictionaries are:

- Tree-based implementation: Any message is sent when all other message arrived (guarantee by Thm. 1)

$$C = \{(e_0, e_1) : \{(e_2, e_0) \,:\, e_2 \neq e_1,\, (e_2, e_0) \in \mathcal{E}^{\to}\} \,:\, (e_0, e_1) \in \mathcal{E}^{\to}\}$$

- Directed implementation: Restrict $\mathcal{E}^{\to}$ to those aligned in directionality (guarantee by Thm. 8, see Sect. 6)

$$\mathcal{E}^{\to} = \{(e_0, e_1) : e_0 \cap e_1 \subset e_0^{\text{out}},\, e_0 \cap e_1 \subset e_1^{\text{in}}\}$$

  Then choose C as above. Can avoid multiple message calculations using the Delta-transformed directed hypergraph.

- Forward chaining: Use the Delta-transformed directed hypergraph and message directions

  $\mathcal{E}^{\to} = \{(e, \{v\}) : v \text{ the positive literal in the definite clause } \tau^e\} \cup \{(\{v\}, e) : v \text{ a negative literal in the definite clause } \tau^e\}$

  Message conditions are more involved: Send $(e, \{v\})$ if to all negative literals a message arrived and to the positive literal no message arrived. Send $(\{v\}, e)$ if any message to v has arrived.

Termination is trivial: Since each message is sent at most once, the While loop is executed at most $|\mathcal{E}^{\to}|$ times.

1.3 Change Scheduler

An example for a change scheduler is the **Constraint-propagation implementation** (guarantee by Thm. 2 see Sect. 4)

- Start: All directions.
- Update: When the support of a message to e_1 changed all directions (e_1, e_2)
- Efficiency increase: Replace messages by their support, need only those with nontrivial support in the contraction.

Algorithm 2 ReceiveScheduler

Data structure: ReceiveScheduler RS
  Q : queue
  $conditions$: dictionary (of send conditions)
function INIT($\mathcal{G}$, $conditions$)
  $RS.Q \leftarrow$ Queue()
  $RS.conditions \leftarrow conditions$
  for $(e_2, e_3) \in RS.conditions$ **do**
    if $empty(RS.conditions[(e_2, e_3)])$ **then**
      $RS.conditions.drop((e_2, e_3))$
      $RS.Q.append((e_2, e_3))$
    end if
  end for
  return RS
end function
function UPDATE(RS, e_0, e_1)
  for $(e_2, e_3) \in RS.conditions$ **do**
    if $(e_0, e_1) \in RS.conditions[(e_2, e_3)]$ **then**
      $RS.conditions[(e_2, e_3)].drop((e_0, e_1))$
      if $empty(RS.conditions[(e_2, e_3)])$ **then**
        $RS.conditions.drop((e_2, e_3))$
        $RS.Q.append((e_2, e_3))$
      end if
    end if
  end for
end function
function POP(RS)
  return $RS.Q.pop()$
end function
function EMPTY(RS)
  return $empty(RS.Q)$
end function

To show the termination of the algorithm, it is enough to bound the number of possible changes of each message. In the constraint-propagation implementation each direction (e_0, e_1) is loaded at most

$$\sum_{(e_2,e_0)\in\mathcal{G}^{\rightarrow}} \left(\prod_{v\in e_2\cap e_0} m_v \right)$$

many times onto S, the algorithm therefore terminates after at most

$$\sum_{(e_0,e_1)\in\mathcal{G}^{\rightarrow}} \sum_{(e_2,e_0)\in\mathcal{G}^{\rightarrow}} \left(\prod_{v\in e_2\cap e_0} m_v \right)$$

iterations.

Loopy BP: Resent messages, until a convergence citerion (e.g. norm bound of the message change, or limits of the numbers of messages) has been met.

2 Analysis in the tree-based implementation

We denote for each pair (e_0, e_1) the subset $\mathcal{E}^{\rightarrow(e_0,e_1)} \subset \mathcal{E}$ as the subset of edges $e \in \mathcal{E}$, for which each path to e_1 passes through e_0. Note, that by construction $e_0 \in \mathcal{E}^{\rightarrow(e_0,e_1)}$.

Theorem 1. *For any tensor network on a tree hypergraph, Algorithm 1 terminates in the tree-based implementation and returns final messages*

$$\chi_{e_0\rightarrow e_1}\left[X_{e_0\cap e_1}\right] = \left\langle \left\{ \tau^e\left[X_e\right] \,:\, e \in \mathcal{E}^{\rightarrow(e_0,e_1)} \right\} \right\rangle_{\left[X_{e_0\cap e_1}\right]}$$

Proof. We show this property by induction over the edge sets $\mathcal{E}^{\to(e_0,e_1)}$ to pairs $(e_0, e_1) \in \mathcal{E}^{\to}$, such that $\left|\mathcal{E}^{\to(e_0,e_1)}\right| \leq n$. Notice, that since always $e_0 \in \mathcal{E}^{\to(e_0,e_1)}$ we have $n \geq 1$.

$n = 1$: In this case we have $\mathcal{E}^{\to(e_0,e_1)} = \{e_0\}$ and e_0 is a leaf of the tree-hypergraph $\mathcal{G}$. The claimed message property holds thus by definition.

$n \to n+1$: Let us assume, that the message obeys the claimed property for edge sets with cardinality up to n. If there is no edge set with cardinality $n+1$, the property holds also for those with cardinality up to $n+1$. If there is an edge set $\mathcal{E}^{\to(e_0,e_1)}$ with size $n+1$, we have

$$\mathcal{E}^{\to(e_0,e_1)} = \{e_0\} \cup \left(\bigcup_{e_2 \in \mathcal{E}^{\to}} \mathcal{E}^{\to(e_2,e_0)} \right).$$

The message $\chi_{e_0 \to e_1}$ is sent, once all messages $\chi_{e_2 \to e_0}$ to $(e_2, e_0) \in \mathcal{E}^{\to}\{(e_1, e_0)\}$ arrived. By definition we have

$$\chi_{e_0 \to e_1}\left[X_{e_0 \cap e_1}\right] = \left\langle \{\tau^{e_0}\left[X_{e_0}\right]\} \cup \{\chi_{e_2 \to e_0}\left[X_{e_2 \cap e_0}\right] \,:\, (e_2, e_0) \in \mathcal{E}^{\to},\, e_2 \neq e_1\} \right\rangle_{\left[X_{e_0 \cap e_1}\right]}$$

Now we use the induction assumption on $\mathcal{E}^{\to(e_2,e_0)}$ (since its cardinality is at most n) and get

$$\begin{aligned}
\chi_{e_0 \to e_1}\left[X_{e_0 \cap e_1}\right] &= \left\langle \{\tau^{e_0}\left[X_{e_0}\right]\} \cup \left(\bigcup_{(e_2,e_0) \in \mathcal{E}^{\to},\, e_2 \neq e_1} \left\langle \{\tau^{e_3}\left[X_{e_3}\right] \,:\, e_3 \in \mathcal{E}^{\to(e_2,e_0)}\} \right\rangle_{\left[X_{e_2 \cap e_0}\right]} \right) \right\rangle_{\left[X_{e_0 \cap e_1}\right]} \\
&= \left\langle \{\tau^{e_0}\left[X_{e_0}\right]\} \cup \left(\bigcup_{(e_2,e_0) \in \mathcal{E}^{\to},\, e_2 \neq e_1} \{\tau^{e_3}\left[X_{e_3}\right] \,:\, e_3 \in \mathcal{E}^{\to(e_2,e_0)}\} \right) \right\rangle_{\left[X_{e_0 \cap e_1}\right]} \\
&= \left\langle \{\tau^{e}\left[X_{e}\right] \,:\, e \in \mathcal{E}^{\to(e_0,e_1)}\} \right\rangle_{\left[X_{e_0 \cap e_1}\right]}
\end{aligned}$$

Here we used the commutation of contraction property in the second equation, which is justified by the assumed tree property of the hypergraph. Thus, the message property holds also for any edge sets of size $n+1$.

By induction, the claimed message property therefore holds for all final messages. □

3 Graph Manipulation

The graph $\mathcal{G}$ can result from manipulations: e.g. Bethe-Clustering ("deltafication").

We have two manipulation strategies, which we show to preserve contractions:

- Bethe construction of factor graphs
- Coarse graining

3.1 Factor graphs

Definition 2. *Given a tensor network $\tau^{\mathcal{G}}$ on a decorated hypergraph $\mathcal{G}$, we define the bethe hypergraph $\tilde{\mathcal{G}}$ as $(\mathcal{U}, \tilde{\mathcal{E}} \cup \Delta^{\mathcal{G}})$ where we have*

- *Recolored Edges $\tilde{\mathcal{E}} = \{\tilde{e} \,:\, e \in \mathcal{E}\}$ where $\tilde{e} = \{v^e \,:\, v \in e\}$, which decoration tensor has same coordinates as τ^e*
- *Nodes $\mathcal{U} = \mathcal{V} \cup (\bigcup_{e \in \mathcal{E}} \tilde{e})$*
- *Delta Edges $\Delta^{\mathcal{G}} = \big\{\{v\} \cup \{v^e \,:\, e \ni v\} \,:\, v \in \mathcal{V}\big\}$, each of which decorated by a delta tensor $\delta^{\{v^e \,:\, e \ni v\}}$*

The resulting overlap graph is bipartite, since any variable appears exactly in one cluster in $\tilde{\mathcal{E}}$ and in one cluster of $\Delta^{\mathcal{G}}$. This further makes the dual of the Bethe Cluster Hypergraph a proper graph (i.e. edges consistent of node pairs).

We effectively copy each node v by v^e. Let us now show, that when assigning dirac delta tensors the contraction reproduces the original contraction.

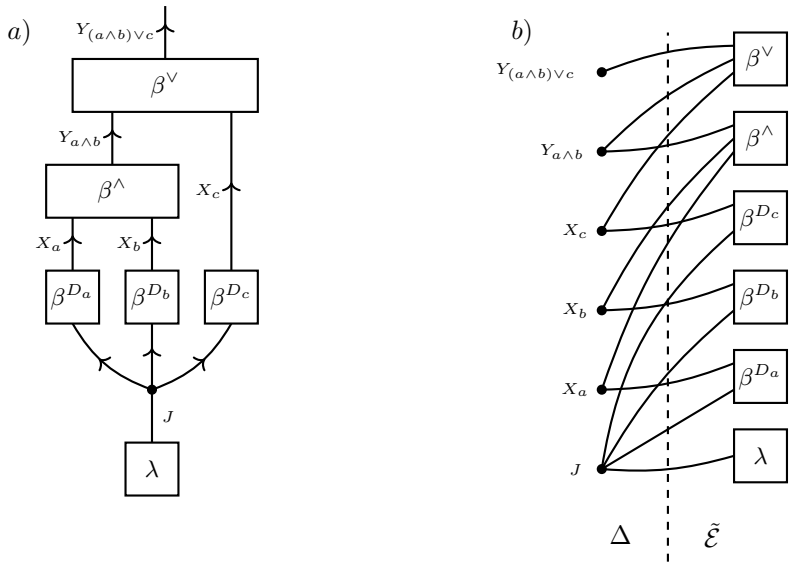

Figure 1: Example of a Bethe Cluster Graph. a) Example of a Tensor Network $\tau^{\mathcal{G}}$, which represents the by λ averaged evaluation of the formula $(a \wedge b) \vee c$ on data D. b) Corresponding Bethe Cluster Hypergraph, which dual is bipartite by the sets Δ and $\tilde{\mathcal{E}}$.

By adding delta tensors to each node $v \in \mathcal{V}$ and defining its leg variables by v^e for $e \in \mathcal{E}$. We mark each such delta tensor by a cluster in $\Delta^{\mathcal{G}}$, as defined in the following (see also Figure 1).

By Lem. **??** this construction does not change contractions.

Lemma 1 (Bethe Invariance)**.** *Given a tensor network $\tau^{\mathcal{G}}\left[X_{[d]}\right]$ on a hypergraph $\mathcal{G}$. We define a tensor network on its bethe hypergraph $\tilde{\mathcal{G}}$ by the tensors for $e \in \mathcal{E}$*

$$\tilde{\tau}^{\tilde{e}}\left[X_{\tilde{e}}\right] = \left\langle \tau^{e}\left[X_e\right], \delta\left[X_e, X_{\tilde{e}}\right]\right\rangle_{[X_{\tilde{e}}]}$$

and the tensors for $\tilde{e} \in \Delta^{\mathcal{G}}$

$$\tilde{\tau}^{\tilde{e}}\left[X_{\tilde{e}}, X_v\right] = \delta\left[X_{\tilde{e}}, X_v\right] .$$

Then we have

$$\tau^{\mathcal{G}}\left[X_{[d]}\right] = \left\langle \{\tilde{\tau}^{\tilde{e}}\left[X_{\tilde{e}}\right] \,:\, \tilde{e} \in \tilde{\mathcal{E}} \cup \Delta^{\mathcal{G}}\}\right\rangle_{[X_{\mathcal{V}}]}$$

Proof. By invariance of adding dirac deltas. □

Equal constructions:

- BP on factor graphs in Mézard (2009)
- Bethe Cluster graphs in Koller and Friedman (2009)

3.2 Coarse graining

Definition 3. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\tilde{\mathcal{G}} = (v, \tilde{\mathcal{E}})$ be two hypergraphs sharing their node set and $c : \mathcal{E} \to \tilde{\mathcal{E}}$ a map. We say that c is a coarse graining map and $\tilde{\mathcal{G}}$, if for all $e \in \mathcal{E}$*

$$e \subset c(e) .$$

For any tensor network $\tau^{\mathcal{G}}\left[X_{\mathcal{V}}\right]$ on $\mathcal{G}$ we further define a coarse grained tensor network $\tilde{\tau}^{\tilde{\mathcal{G}}}\left[X_{\mathcal{V}}\right]$ on $\tilde{\mathcal{G}}$ with hypercores to $\tilde{e} \in \tilde{\mathcal{E}}$ by

$$\tilde{\tau}^{\tilde{e}}\left[X_{\tilde{e}}\right] = \left\langle \{\tau^{e}\left[X_e\right] \,:\, \tilde{e} = c(e)\}\right\rangle_{[X_{\tilde{e}}]} .$$

In the literature such coarse graining procedures appear as Cluster Graphs.

We notice that coarse graining of tensor networks preserves their contractions, since

$$\begin{aligned}
\tau^{\mathcal{G}}\left[X_{\mathcal{V}}\right] &= \langle \{\tau^{e}\left[X_{e}\right] \,:\, e \in \mathcal{E}\} \rangle_{[X_{\mathcal{V}}]} \\
&= \left\langle \bigcup_{\tilde{e}\in\tilde{\mathcal{E}}} \{\tau^{e}\left[X_{e}\right] \,:\, c(e) = \tilde{e}\} \right\rangle_{[X_{\mathcal{V}}]} \\
&= \left\langle \{\langle \{\tau^{e}\left[X_{e}\right] \,:\, c(e) = \tilde{e}\} \rangle_{[X_{e}]} \,:\, \tilde{e} \in \tilde{\mathcal{E}}\} \right\rangle_{[X_{\mathcal{V}}]} \\
&= \left\langle \{\tilde{\tau}^{\tilde{e}}\left[X_{\tilde{e}}\right] \,:\, \tilde{e} \in \tilde{\mathcal{E}}\} \right\rangle_{[X_{\mathcal{V}}]} \\
&= \tilde{\tau}^{\tilde{\mathcal{G}}}\left[X_{\mathcal{V}}\right] .
\end{aligned}$$

3.3 Coarse graining to Tree Hypergraphs

Construction of coarse grained tree hypergraphs, by the junction tree algorithm

- Amounts to triangulation of underlying "extended graph" (i.e. node pairs contained in hypergraphs)
- Use the cliques of the triangulated graph as hyperedges (guaranteed to be a tree, a so called "clique tree").

3.3.1 Junction Tree Algorithm

Classically the junction tree algorithm is formulated on the extended graph (we define extended graphs here by two nodes adjacent if and only if they are contained in a hypergraph) Lauritzen and Spiegelhalter (1988), where one exploits that a graph has a junction tree if and only if it is triangulated Lauritzen (1996). The junction tree algorithm therefore constructs a triangulation of the graph by adding edges (also called chordal graph Koller and Friedman (2009)), and gets a tree-hypergraph by its cliques. Note that finding a minimal triangulation is NP-hard.

We can equally formulate the junction tree algorithm on hypergraphs. Equivalent to searching for a triangulation of an extended graph, we here search for a coarse grained hypergraph, which is a tree. Then the cliques of the triangulated graphs correspond with the hyperedges of the coarse grained hypergraph.

3.3.2 Variable Elimination

Based on disjoint subsets of $\mathcal{V}$ we can construct a coarse graining procedure of a hypergraph: Construct iteratively $\tilde{\mathcal{E}}$ by iterating through the node subsets and

- Add those $e \in \mathcal{E}$ which have not been assigned before and contain variables in the subset.
- Choose the new edge by the union of the old one with the assigned edges in $\mathcal{E}$ and dropping those nodes, which are not appearing in the rest of $\mathcal{E}$ any more.
- Return the intersection of the new edge with the rest of $\mathcal{E}$ back to $\mathcal{E}$.

By construction this gives a tree hypergraph.

4 Boolean Propagation

Sound in the sense that entailment decisions made during the propagation are always true.

Instead of the exact calculation of a contraction, let us now investigate schemes to sparsify the tensors before a contraction. To this end, we first show underlying properties of contractions enabling these schemes.

To state the next theorem we use the nonzero function $\mathbb{I}_{\neq 0} : \mathbb{R} \to [2]$ by $\mathbb{I}_{\neq 0}(x) = 1$ if $x \neq 0$ and $\mathbb{I}_{\neq 0}(x) = 0$ else. Applied coordinatewise on tensors $\tau\left[X_{[d]}\right]$ it marks the nonzero coordinates by 1. The resulting boolean tensor is denoted by $\mathbb{I}_{\neq 0}\left(\tau\left[X_{[d]}\right]\right)$.

Further, we use the partial order $\prec$ of boolean tensors, which is defined by

$$\tau\left[X_{[d]}\right] \prec \tilde{\tau}\left[X_{[d]}\right] \quad \Leftrightarrow \quad \forall x_{[d]} \in \bigtimes_{k\in[d]} [m_k] : \tau\left[X_{[d]} = x_{[d]}\right] \leq \tilde{\tau}\left[X_{[d]} = x_{[d]}\right]$$

Theorem 2. *For any implementation of Algorithm 1 we have at any stage of the algorithm for any message channel* (e_0, e_1)

$$\mathbb{I}_{\neq 0}\left(\left\langle \tau^{\mathcal{G}}\left[X_{[d]}\right]\right\rangle_{\left[X_{e_0\cap e_1}\right]}\right) \prec \mathbb{I}_{\neq 0}\left(\chi_{e_0\to e_1}\left[X_{e_0\cap e_1}\right]\right) .$$

To show this theorem we in the following proof the monotonicity of tensor contractions and the invariance of adding supports of subcontractions.

4.1 Monotonicity of Tensor Contraction

We show that adding boolean tensor cores to a contraction orders the results by the partial ordering introduced in Def. **??**.

Theorem 3 (Monotonicity of Tensor Contractions)**.** *Let* $\tau^{\mathcal{G}}, \tau^{\tilde{\mathcal{G}}}$ *be tensor network of non-negative tensors and* $X_{\mathcal{U}}$ *an arbitrary set of random variables. Then we have*

$$\mathbb{I}_{\neq 0}\left(\left\langle \tau^{\mathcal{G}} \cup \tau^{\tilde{\mathcal{G}}}\right\rangle_{[X_{\mathcal{U}}]}\right) \prec \mathbb{I}_{\neq 0}\left(\left\langle \tau^{\mathcal{G}}\right\rangle_{[X_{\mathcal{U}}]}\right) .$$

Proof. It suffices to show that for any $x_{\mathcal{U}}$ with

$$\mathbb{I}_{\neq 0}\left(\left\langle \tau^{\mathcal{G}} \cup \tau^{\tilde{\mathcal{G}}}\right\rangle_{[X_{\mathcal{U}}=x_{\mathcal{U}}]}\right) = 1$$

we also have

$$\mathbb{I}_{\neq 0}\left(\left\langle \tau^{\mathcal{G}}\right\rangle_{[X_{\mathcal{U}}=x_{\mathcal{U}}]}\right) = 1 .$$

For any $x_{\mathcal{U}}$ satisfying the first equation we find an extension $x_{\mathcal{V}}$ to all variables of the tensor networks such that

$$\left\langle \tau^{\mathcal{G}} \cup \tau^{\tilde{\mathcal{G}}}\right\rangle_{[X_{\mathcal{V}}=x_{\mathcal{V}}]} > 0$$

and it follows that

$$\left\langle \tau^{\mathcal{G}}\right\rangle_{[X_{\mathcal{V}}=x_{\mathcal{V}}]} > 0 \quad \text{and} \quad \left\langle \tau^{\tilde{\mathcal{G}}}\right\rangle_{[X_{\mathcal{V}}=x_{\mathcal{V}}]} > 0 .$$

But this already implies, that

$$\mathbb{I}_{\neq 0}\left(\left\langle \tau^{\mathcal{G}}\right\rangle_{[X_{\mathcal{U}}=x_{\mathcal{U}}]}\right) = 1 . \qquad \square$$

Lemma 2. *Let* $\tau^{\mathcal{G}}$ *and* $\tau^{\tilde{\mathcal{G}}}$ *be non-negative tensor networks on the same hypergraph, and let us assume that for all* $e \in \mathcal{E}$

$$\mathbb{I}_{\neq 0}\left(\tau^{e}\left[X_e\right]\right) \prec \mathbb{I}_{\neq 0}\left(\tilde{\tau}^{e}\left[X_e\right]\right) .$$

Then we have for any subset $\mathcal{U}$

$$\mathbb{I}_{\neq 0}\left(\left\langle \tau^{\mathcal{G}}\right\rangle_{[X_{\mathcal{U}}]}\right) \prec \mathbb{I}_{\neq 0}\left(\left\langle \tau^{\tilde{\mathcal{G}}}\right\rangle_{[X_{\mathcal{U}}]}\right) .$$

Proof. It is sufficient to show that for all $y_{\mathcal{U}} \in \bigtimes_{v\in\mathcal{U}}[m_v]$ we have

$$\left(\mathbb{I}_{\neq 0}\left(\left\langle \tau^{\mathcal{G}}\right\rangle_{[X_{\mathcal{U}}=y_{\mathcal{U}}]}\right) = 1\right) \Rightarrow \left(\mathbb{I}_{\neq 0}\left(\left\langle \tau^{\tilde{\mathcal{G}}}\right\rangle_{[X_{\mathcal{U}}=y_{\mathcal{U}}]}\right) = 1\right) .$$

If and only if for there is an index $x_{\mathcal{V}}$ with $x_{\mathcal{V}}|_{\mathcal{U}} = y_{\mathcal{U}}$ with

$$\forall e \in \mathcal{E} : \tau^{e}\left[X_e = x_e\right] = 1$$

then $\mathbb{I}_{\neq 0}\left(\left\langle \tau^{\mathcal{G}}\right\rangle_{[X_{\mathcal{U}}=y_{\mathcal{U}}]}\right) = 1$. Note, that by assumption we have for this index also

$$\forall e \in \mathcal{E} : \tilde{\tau}^{e}\left[X_e = x_e\right] = 1$$

and thus $\mathbb{I}_{\neq 0}\left(\left\langle \tau^{\tilde{\mathcal{G}}}\right\rangle_{[X_{\mathcal{U}}=y_{\mathcal{U}}]}\right) = 1$. $\square$

4.2 Invariance of Adding Subcontractions

Let us now state an equivalence of the contraction, when we add the result of the same contraction. This property was used in the proof of Thm. **??**.

Theorem 4 (Invariance under adding subcontractions)**.** *Let $\tau^{\mathcal{G}}$ be a tensor network of non-negative tensors with variables $X_{\mathcal{V}}$ and let $\tau^{\tilde{\mathcal{G}}}$ be a subset. Then we have for any subset $X_{\mathcal{U}}$ of $X_{\mathcal{V}}$*

$$\left\langle \tau^{\mathcal{G}} \cup \left\{ \mathbb{I}_{\neq 0} \left(\left\langle \tau^{\tilde{\mathcal{G}}} \right\rangle_{[X_{\mathcal{U}}]} \right) \right\} \right\rangle_{[X_{\mathcal{V}}]} = \left\langle \tau^{\mathcal{G}} \right\rangle_{[X_{\mathcal{V}}]} .$$

Proof. For any $x_{\mathcal{V}}$ with

$$\left\langle \tau^{\mathcal{G}} \right\rangle_{[X_{\mathcal{V}}=x_{\mathcal{V}}]} = 0$$

we also have

$$\left\langle \tau^{\mathcal{G}} \cup \{ \mathbb{I}_{\neq 0} \left(\left\langle \tau^{\tilde{\mathcal{G}}} \right\rangle_{[X_{\mathcal{U}}]} \right) \} \right\rangle_{[X_{\mathcal{V}}=x_{\mathcal{V}}]} = 0 .$$

For any $x_{\mathcal{V}}$ with

$$\left\langle \tau^{\mathcal{G}} \right\rangle_{[X_{\mathcal{V}}=x_{\mathcal{V}}]} \neq 0$$

we have for the reduction $x_{\mathcal{U}}$ of the index $x_{\mathcal{V}}$ that

$$\left\langle \tau^{\tilde{\mathcal{G}}} \right\rangle_{[X_{\mathcal{U}}=x_{\mathcal{U}}]} \neq 0$$

and thus

$$\begin{aligned} \left\langle \tau^{\mathcal{G}} \cup \{ \mathbb{I}_{\neq 0} \left(\left\langle \tau^{\tilde{\mathcal{G}}} \right\rangle_{[X_{\mathcal{U}}]} \right) \} \right\rangle_{[X_{\mathcal{V}}=x_{\mathcal{V}}]} &= \left\langle \tau^{\mathcal{G}} \right\rangle_{[X_{\mathcal{V}}=x_{\mathcal{V}}]} \cdot \mathbb{I}_{\neq 0} \left(\left\langle \tau^{\tilde{\mathcal{G}}} \right\rangle_{[X_{\mathcal{U}}]} \right) [X_{\mathcal{U}} = x_{\mathcal{U}}] \\ &= \left\langle \tau^{\mathcal{G}} \right\rangle_{[X_{\mathcal{V}}=x_{\mathcal{V}}]} . \end{aligned}$$

□

4.3 Consequences for boolean Contraction Propagation

Lemma 3. *In any implementation of Algorithm 1 and at any iteration of the* While *loop, we have the old message $\chi_{e_2 \to e_1}[X_{e_2 \cap e_1}]$ and the updated message $\hat{\chi}_{e_2 \to e_1}[X_{e_2 \cap e_1}]$ obey*

$$\chi_{e_2 \to e_1}[X_{e_2 \cap e_1}] \prec \hat{\chi}_{e_2 \to e_1}[X_{e_2 \cap e_1}] .$$

Proof. By induction over the iterations of the While loop, and using monotonicity of contractions. □

We are now ready to show the above guarantee on the soundness of message passing.

Proof of Thm. 2. We first show via induction on the message updates that at any stage of Algorithm 1 we have

$$\left\langle \tau^{\mathcal{G}}[X_{\mathcal{V}}] \right\rangle_{[\varnothing]} X_{\mathcal{V}} = \left\langle \tau^{\mathcal{G}}[X_{\mathcal{V}}] \cup \{ \mathbb{I}_{\neq 0} (\chi_{e_0 \to e_1}[X_{e_0 \cap e_1}]) : (e_0, e_1) \in \mathcal{G}^{\to} \} \right\rangle_{[X_{\mathcal{V}}]} . \tag{1}$$

Since the messages are initialized by trivial tensors, this is true after the initialization of Algorithm 1. Now we assume that this equation holds at an arbitrary state of the algorithm at the start of the While loop, and let (e_0, e_1) be the channel chosen by the scheduler. Then we choose

$$\tau^{\tilde{\mathcal{G}}} = \{ \tau^{e_0}[e_1] \} \cup \{ \chi_{e_2 \to e_0}[X_{e_2 \cap e_0}] : (e_2, e_0) \in \mathcal{G}^{\to}, e_2 \neq e_1 \}$$

and apply Thm. 4 to get

$$\begin{aligned} \left\langle \tau^{\mathcal{G}}[X_{\mathcal{V}}] \right\rangle_{[\varnothing]} X_{\mathcal{V}} &= \left\langle \tau^{\mathcal{G}}[X_{\mathcal{V}}] \cup \{ \mathbb{I}_{\neq 0} (\chi_{e_3 \to e_2}[X_{e_3 \cap e_2}]) : (e_3, e_2) \in \mathcal{G}^{\to} \} \right\rangle_{[X_{\mathcal{V}}]} \\ &= \left\langle \tau^{\mathcal{G}}[X_{\mathcal{V}}] \cup \{ \mathbb{I}_{\neq 0} (\chi_{e_3 \to e_2}[X_{e_3 \cap e_2}]) : (e_3, e_2) \in \mathcal{G}^{\to} \} \cup \{ \mathbb{I}_{\neq 0} \left(\left\langle \tau^{\tilde{\mathcal{G}}} \right\rangle_{[X_{e_0 \cap e_1}]} \right) \} \right\rangle_{[X_{\mathcal{V}}]} \\ &= \left\langle \tau^{\mathcal{G}}[X_{\mathcal{V}}] \cup \{ \mathbb{I}_{\neq 0} (\chi_{e_3 \to e_2}[X_{e_3 \cap e_2}]) : (e_3, e_2) \in \mathcal{G}^{\to}, (e_3, e_2) \neq (e_0, e_1) \} \right. \\ &\qquad \left. \cup \{ \mathbb{I}_{\neq 0} (\chi_{e_0 \to e_1}[X_{e_0 \cap e_1}]), \mathbb{I}_{\neq 0} (\hat{\chi}_{e_0 \to e_1}[X_{e_0 \cap e_1}]) \} \right\rangle_{[X_{\mathcal{V}}]} \\ &= \left\langle \tau^{\mathcal{G}}[X_{\mathcal{V}}] \cup \{ \mathbb{I}_{\neq 0} (\chi_{e_3 \to e_2}[X_{e_3 \cap e_2}]) : (e_3, e_2) \in \mathcal{G}^{\to}, (e_3, e_2) \neq (e_0, e_1) \} \right. \\ &\qquad \left. \cup \{ \mathbb{I}_{\neq 0} (\hat{\chi}_{e_0 \to e_1}[X_{e_0 \cap e_1}]) \} \right\rangle_{[X_{\mathcal{V}}]} . \end{aligned}$$

Here we used that by Lem. 3 we have

$$\langle \mathbb{I}_{\neq 0}\left(\chi_{e_0\to e_1}\left[X_{e_0\cap e_1}\right]\right), \mathbb{I}_{\neq 0}\left(\hat{\chi}_{e_0\to e_1}\left[X_{e_0\cap e_1}\right]\right)\rangle_{\left[X_{e_0\cap e_1}\right]} = \mathbb{I}_{\neq 0}\left(\hat{\chi}_{e_0\to e_1}\left[X_{e_0\cap e_1}\right]\right) .$$

We therefore have by induction (1) during the algorithm.

The claim now follows from Thm. 3, since

$$\mathbb{I}_{\neq 0}\left(\left\langle \tau^{\mathcal{G}}\left[X_{\mathcal{V}}\right] \cup \left\{\mathbb{I}_{\neq 0}\left(\chi_{e_0\to e_1}\left[X_{e_0\cap e_1}\right]\right) : (e_0,e_1)\in \mathcal{G}^{\to}\right\}\right\rangle_{\left[X_{e_0\cap e_1}\right]}\right)$$
$$\prec \mathbb{I}_{\neq 0}\left(\mathbb{I}_{\neq 0}\left(\chi_{e_0\to e_1}\left[X_{e_0\cap e_1}\right]\right)\right) = \mathbb{I}_{\neq 0}\left(\chi_{e_0\to e_1}\left[X_{e_0\cap e_1}\right]\right) . \qquad \square$$

5 Complete Unit Clause Propagation (UCP)

Investigate here three conditions, where constraint propagation is also complete. (tree condition holds more general!)

Unit clauses are assignments to single variables. They are propagated between clusters of formulas

UCP is Algorithm 1 in case of bethe hypergraphs (having bipartite overlap graphs with nodes to previous hyperedges and variable nodes), and using the constraint propagation implementation, that is

- Hypergraph with hyperedges to single variables and to previous edges
- Initial queue by those edges containing single nodes

$$\mathcal{Q} = \left\{\{v\} : \{v\}\in\mathcal{E}\right\}$$

After termination of the Knowledge Propagation Algorithm we return "$UNSAT$", if any knowledge core vanishes. In that case we have an unsatisfiable CSP. If no knowledge core vanishes, we define A as the subset of variables with nontrivial knowledge cores and output x_A where for $v\in A$

$$x_v = \epsilon^{-1}(\kappa\left[\{v\}\right]) .$$

We always have, that UCP is sound (since KP is sound).

Definition 4. *We say that UCP is complete for a CSP, if it outputs "$UNSAT$" or for no node $v\in A$ we have that at all solutions of the CSP have the same index at v.*

5.1 Forward Chaining for Horn-SAT

Definition 5 (Definite Horn-SAT)**.** *Let $\{\tau^e\left[X_e\right] : e\in\mathcal{E}\}$ be a CSP. We say it is a definite Horn-SAT problem, if each τ^e is a clause, which has exactly one positive literal.*

Forward chaining is a linear time and complete satisfiability checker on Horn Logic (a subset of Propositional Logic where each clause has at most one positive literal).

- Messages passed are the one-hot encodings of assignment to single variables (unit clauses).
- Clusters are the Horn clauses, which receive messages for their negative literals. A Cluster sends a nontrivial message to a variable, if the variable is the only unassigned literal in the clause and the clause has not been satisfied yet.

Lemma 4. *Let $\tau^{\mathcal{G}} = \{\tau^e\left[X_e\right] : e\in\mathcal{E}\}$ be a Definite Horn-SAT problem, then UCP does never output "$UNSAT$". Let further x_A be the output of UCP. Then $x_v = 1$ for each $v\in A$ and*

$$1_A\times 0_{\mathcal{V}/A} \quad \textit{and} \quad 1_A\times 1_{\mathcal{V}/A}$$

are solutions of $\tau^{\mathcal{G}}$.

Proof. First of all, when all knowledge cores are trivial, vanishing or ϵ_1, then the nonzero transformation of any contraction is trivial, vanishing or ϵ_1. Thus, since this assumption is met at the start all knowledge cores remain such during the algorithm.

At the termination of UCP we simplify the definite clauses by erasing the literals in A. This erasure results either in empty clauses or in definite clauses with at least 2 literals, since otherwise the algorithm would not have terminated. Since at least one positive and one negative literal remain, these are satisfied if all atoms are 1 and if all atoms are 0. $\square$

Theorem 5. *UCP for Definite Horn-SAT is complete.*

Proof. From the above lemma we know that the remaining variables are in at least one solution 0 and in at least one 1. Thus they are neither entailed nor contradicted. □

5.2 UCP for 2-SAT

Definition 6 (2-SAT)**.** *Let $\{\tau^e [X_e] : e \in \mathcal{E}\}$ be a CSP. We say it is an 2-SAT problem, if each τ^e has order at most 2 and is a clause.*

2-SAT is in P and can be solved by the message passing algorithm Unit Clause Propagation (UCP).

- At each connected component of the problems factor graph, choose on variable an do the message passing scheme below with initialization by the variable on 0 and on 1.
- Messages passed are the one-hot encodings of assignment to single variables (unit clauses).
- Any clause that receives a message has only a single literal left and either gets directly trivial (if the message coincides with the literal) or assigns the remaining variable and passes further.

Lemma 5. *Let $\{\tau^e [X_e] : e \in \mathcal{E}\}$ be a 2-SAT instance. Given the outputs $x_c^{s_c}$, $s_c \in [n_c]$ and $n_c \in \{0, 1, 2\}$ of UCP for each connected component $c \subset \mathcal{V}$ we have*

$$\langle \{\tau^e [X_e] : e \in \mathcal{E}\} \rangle_{[X_{\mathcal{V}}]} = \bigotimes_c \left(\sum_{s_c \in [n_c]} \epsilon_{x_c^{s_c}} [X_c] \right) .$$

Proof. For each component c of $\mathcal{G}$ we choose a start variable and choose a value $x_v \in [2]$. We then have

$$\langle \tau^c \cup \{\epsilon_{x_v} [X_v]\} \rangle_{[X_c]} = \begin{cases} 0 [X_c] & \text{if UCP returns "UNSAT"} \\ \epsilon_{x_c^{s_c}} [X_c] & \text{if UCP returns } x_c^{s_c} \end{cases} .$$

□

We use UCP for entailment/contradiction decision by checking whether for each $k \in c$ $x_k^{s_c}$ is constant. Exception: When one component is not sat, the whole 2-SAT instance is unsatisfiable and all entailment and contradiction properties hold.

Theorem 6. *UCP for 2-SAT is complete.*

Proof. We assume that 2-SAT at hand is satisfiable. Exactly when $x_v^{s_c}$ is constant for $s_c \in [n_c]$ we can write, using the above Lemma

$$\langle \{\tau^e [X_e] : e \in \mathcal{E}\} \rangle_{[X_{\mathcal{V}}]} = \epsilon_{x_k^{s_c}} [X_k] \otimes \langle \{\tau^e [X_e] : e \in \mathcal{E}\} \rangle_{[X_{\mathcal{V}/\{v\}}]} .$$

In case of $x_v^{s_c} = 1$ this is an equivalent criterion for entailment (respectively contradiction in case of $x_v^{s_c} = 0$). □

5.3 UCP for Tree-SAT

Definition 7 (Tree-SAT)**.** *Let $\{\tau^e [X_e] : e \in \mathcal{E}\}$ be a CSP. We say it is an Tree-SAT problem, if the factor graph is minimal connected.*

Note that we do not demand the constraint cores to be clauses in the Tree-SAT definition.

We modify UCP slightly: If any constraint tensor is decomposed into a tensor product of a basis vector of one variable and an arbitrary rest tensor, the constraint tensor is added to the queue at initialization.

Theorem 7. *The modified UCP for Tree-SAT is complete.*

Proof. Since the message-passing provides exact contractions and the messages in UCP communicate the support. □

5.4 Outlook

5.4.1 With backtracking: DPLL for generic SAT

DPLL combines backtracking search with unit clause propagation (UCP). A form of message passing is applied to reduce the clauses given the current partial assignment: When guessed an assignment to a variable, the variable is removed from all clauses, either making the clause trivial (coinciding assignment) or smaller. If only one literal remains in a clause, the variable would be assigned accordingly (unit propagation). This can be directly done in the intermediate message passing scheme, or understood as the next backtracking step ("Find-Unit-Clause" in Figure 7.17 in Russell and Norvig (2021)).

5.4.2 With randomization: WalkSAT

WalkSAT is a stochastic local search algorithm for SAT. It starts with a random assignment and iteratively flips variables to reduce the number of unsatisfied clauses. This can be understood as a (modified) Gibbs sampling algorithm, where the number of unsatisfied clauses is the energy function to be minimized. The modifications are:

- Selection of variable to be resampled: Typically chosen by looking at unsatisfied clauses and picking a variable that minimizes the number of newly unsatisfied clauses (whereas in Gibbs sampling one follows a fixed variable order).
- Marginal probability: Typically fixed by a mixing parameter, whereas in Gibbs sampling would be sensitive to the energy differences.

6 Basis Propagation

Theorem 8. *Given a directed acyclic hypergraph and a tensor network of directed boolean tensors respecting the hypergraph. Then the final messages of Algorithm 1 in the directed implementation are exact.*

Proof. Just show inductively that the messages are one-hot encodings of the function evaluation. □

We have an interpretation of the messages by one-hot encodings of function evaluation at each node. The basis vectors at the leafs (i.e. edges with no incoming nodes) are understood as one-hot encodings of inputs. Passing a message ϵ_i in direction thus gives the message $\epsilon_{q(i)}$.

Note, that the assumptions of Thm. 8 are met, whenever the graph is directed and acyclic. We do not need acyclicity of the underlying undirected graph.

This is because any basis encoding of a function, the decomposition

$$\beta^q = \sum_{y \in \mathrm{im}(q)} \Big(\sum_{i : q(i) = y} \epsilon_i \Big) \otimes \epsilon_y$$

is a SVD of the matrification of β^q with respect to incoming and outgoing legs.

Example 1 (Function evaluation on a dataset)**.** *Let the only non-basis vectors be the incoming ones, e.g. by a dataset (i.e. averaging by $\mathbb{I}[I]$). Message Passing of directed and boolean message by basis encoding of functions can be interpreted as function evaluation. Each subfunction evaluation is passed in its one-hot encoding. Distinguish:*

- *Single inference: Basis propagation (see Figure 2)*
- *Batchwise inference: "Tensor Parallelism", but in the most interesting cases not exact. If the graph is minimally connected, we are guaranteed that the averages are exact. Also we can apply boolean theory to have sound messages: If in a message a state is not supported, this state cannot be reached by any data point*

Remark 1 (Basis Calculus as Message Passing)**.** *Given a tensor network of directed and binary tensor cores, each representing a function q_e depending on variables e^{in}. When there are not directed cycles, we define the compositions of q_e to be the function q from the nodes $\mathcal{V}^1$ not appearing as incoming nodes to the nodes $\mathcal{V}^2$ not appearing as outgoing nodes in an edge. Choosing arbitrary $x_v \in [m_v]$ for $v \in \mathcal{V}^1$ we have*

$$\left\langle \{ \beta^{q_e}\left[X_{e^{\mathrm{out}}}, X_{e^{\mathrm{in}}}\right] : e = (e^{\mathrm{out}}, e^{\mathrm{in}}) \in \mathcal{E} \} \right\rangle_{[\mathcal{V}^2]} = \epsilon_{q(x_v \,:\, v \in \mathcal{V}^1)} \,.$$

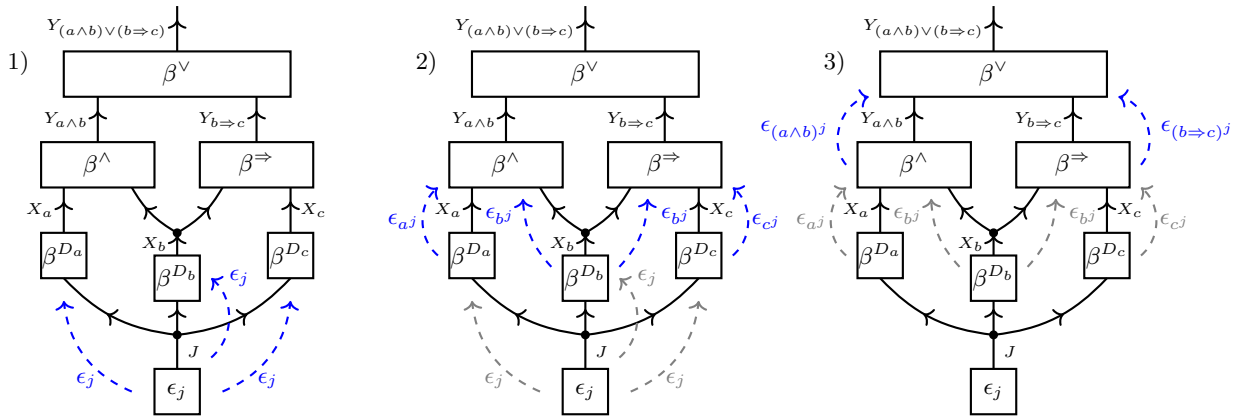

Figure 2: Behavior of the contraction propagation algorithm Algorithm 1 for the evaluation of the formula $(X_a \wedge X_b) \vee (X_b \Rightarrow X_c)$ on a datapoint (a^j, b^j, c^j), which is selected from a dataset D. The algorithm is executed in the directed implementation, in which each message is sent exactly once. We sketch the 9 iterations of the While loop until the algorithm terminates in the epochs 1) (selection of the datapoint from the dataset) 2) (computation of finest formula components) and 3) (computation of coarser formula components) . In each epoch we sketch in gray the received messages at the previous epoch and in blue the sent messages. Since the hypercores are directed and boolean, the messages are one-hot encodings interpreted as evaluations of associated functions.

7 Generalizations

There are two generalization directions:

- Kikuchi message passing schemes: Messages are sent between layers of hyperedges (where so far have the hyperedges and their intersections in two layers).
- Expectation-propagation on generalized features: Hyperedges are understood as sets of features. Messages are computed by forward map in the whole inference cluster, and backward map in the message cluster. So far: Used only the hybrid indicator features on the edges, for which forward maps are simple contractions.

References

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, Cambridge, Mass., 1. edition edition, July 2009. ISBN 978-0-262-01319-2.

S. L. Lauritzen and D. J. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194, January 1988. ISSN 0035-9246. doi: 10.1111/j.2517-6161.1988.tb01721.x. URL https://doi.org/10.1111/j.2517-6161.1988.tb01721.x.

Steffen L. Lauritzen. *Graphical Models*. Clarendon Press, Oxford, July 1996. ISBN 978-0-19-852219-5.

Marc Mézard. *Information, Physics, and Computation*. ACADEMIC, Oxford ; New York, March 2009. ISBN 978-0-19-857083-7.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach, Global Edition: A Modern Approach, Global Edition*. Pearson, Boston, 4 edition, May 2021. ISBN 978-1-292-40113-3.

Martin J. Wainwright and Michael Irwin Jordan. *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc, 2008. ISBN 978-1-60198-184-4.