MESSAGE PASSING FOR EFFICIENT CONTRACTIONS

RESEARCH NOTES IN THE ENEXA AND QROM PROJECTS

December 3, 2025

Plan:

- Junction tree construction related to tree hypergraph construction
- Boolean propagation, extended by exact logical reasoning schemes (might put the latter in the propositional inference chapter)
- Basis propagation: Show with the data example, that single datapoint is exact, batchwise not.

1 Exact Contractions

We apply Thm. **??** to split a contraction into subcontractions, which are consecutively performed.

Contractions can be performed partially, and the result passed to the rest of the network as a message.

1.1 Construction of Cluster Graphs

Let us first introduce with the cluster graph a mechanism to coarse grain the hypergraph capturing a tensor network.

Definition 1 (Cluster Graph)**.** *Given a tensor network $\tau^{\mathcal{G}}$ a cluster partition is a partition of the tensor network into n clusters, by a function*

$$\alpha : \mathcal{E} \to [n] \,.$$

The clusters are with tensors decorated edge sets $C_i = \{e \,:\, \alpha(e) = i\}$ with variables $\mathcal{V}_i = \bigcup_{e \in C_i} e$.

We say, that the cluster graph satisfies the running intersection property, when for any clusters C_i and C_j and any $v \in \mathcal{V}_i \cup \mathcal{V}_j$ there is a path between C_i and C_j with $v \in \mathcal{V}_k$ for any cluster C_k along the path.

Given a cluster graph to a tensor network, we can execute any global contraction by a contraction of local contraction to each cluster.

Theorem 1. *Given a tensor network $\tau^{\mathcal{G}}$ and a cluster graph. We then define for each cluster the node set*

$$\tilde{\mathcal{V}}_i = \bigcup_{j \neq i} \mathcal{V}_j$$

and have

$$\left\langle \tau^{\mathcal{G}} \right\rangle_{[X_{\mathcal{U}}]} = \left\langle \left\{ \left\langle \tau^{C_i} \right\rangle_{\left[X_{\mathcal{V}_i \cap (\tilde{\mathcal{V}}_i \cup \mathcal{U})}\right]} : i \in [n] \right\} \right\rangle_{[X_{\mathcal{U}}]} .$$

Proof. By Thm. **??** applied for each cluster seen as a subgraph. □

1.2 Message Passing to calculate Contractions

Having a hypergraph $\mathcal{G}$, we iteratively apply Thm. **??** and call the $\mathcal{G}_2$ a cluster. When iterating until $\mathcal{G}$ is empty, we get a cluster graph, where all tensors are assigned to a cluster.

When the cluster are a polytree, that is a union of disjoint trees, we define messages between neighbored clusters C_i and C_j with $C_j \prec C_i$ by the contractions

$$\delta_{j\to i}\left[X_{\mathcal{V}^i\cap\mathcal{V}^j}\right] = \left\langle \{\delta_{\tilde{j}\to j}\left[X_{\mathcal{V}^{\tilde{j}}\cap\mathcal{V}^j}\right] \,:\, C_{\tilde{j}} \prec C_j\} \cup \tau^{C_j} \right\rangle_{\left[X_{\mathcal{V}^i\cap\mathcal{V}^j}\right]} .$$

We note, that the messages are well defined by these recursive equations, exactly when the cluster graph is a polytree.

When the cluster graph is a tree, we can choose a root cluster and order the clusters by the topological order $\prec$.

Lemma 1. *When the cluster graph is a tree satisfying the running intersection property, we have for neighbored clusters C_i and C_j with $C_j \prec C_i$*

$$\delta_{i\to\tilde{i}}\left[X_{\mathcal{V}^i\cap\mathcal{V}^{\tilde{i}}}\right] = \left\langle \{\tau^{C_j} \,:\, C_j \prec C_i\}\right\rangle_{\left[X_{\mathcal{V}^i\cap\mathcal{V}^{\tilde{i}}}\right]} .$$

Proof. By induction over the cardinality n of the preceding clusters. $n = 1$: For a single preceding cluster the statement holds trivial, since the preceding cluster is the cluster itself.

$n \to n+1$: Let us now assume, that the statement holds for up to n preceding clusters, and let there be $n+1$ preceding clusters. We build another cluster graph for the cores different from C_i, by assigning each cluster $C_{\tilde{j}}$ to the neighbor C_j where $j \in \mathrm{N}(i)$, for which

$$C_{\tilde{j}} \prec C_j .$$

We use Thm. 1 on this constructed cluster graph and get

$$\left\langle \{\tau^{C_{\tilde{j}}}\left[X_{\mathcal{V}^{\tilde{j}}}\right] \,:\, \tilde{j} \neq i\}\right\rangle_{\left[X_{\mathcal{V}^i}\right]} = \left\langle \left\{ \left\langle \{\tau^{C_{\tilde{j}}}\left[X_{\mathcal{V}^{\tilde{j}}}\right] \,:\, \tilde{j} \prec j\}\right\rangle_{\left[X_{\tilde{\mathcal{V}}^j}\right]} \,:\, j \in \mathrm{N}(i)\right\}\right\rangle_{\left[X_{\mathcal{V}^i}\right]}$$

Here by $\tilde{\mathcal{V}}^j$ we denote the intersection of

$$\tilde{\mathcal{V}}^j = \left(\bigcup_{\tilde{j}\prec j} \mathcal{V}^{\tilde{j}}\right) \cap \left(\bigcup_{\tilde{j}\not\prec j} \mathcal{V}^{\tilde{j}}\right)$$

By the running intersection property, we have $\mathcal{V}^j \cap \mathcal{V}^i = \tilde{\mathcal{V}}^j$.

We further have for any $j \in \mathrm{N}(i)$ that

$$\left|\{\tilde{j} \prec j\}\right| \leq n .$$

We can therefore apply the assumption of the induction and get

$$\left\langle \{\tau^{C_{\tilde{j}}}\left[X_{\mathcal{V}^{\tilde{j}}}\right] \,:\, \tilde{j} \prec j\}\right\rangle_{\left[X_{\tilde{\mathcal{V}}^j}\right]} = \delta_{j\to i}\left[X_{\mathcal{V}^j\cap\mathcal{V}^i}\right]$$

With the above, we arrive at

$$\delta_{i\to\tilde{i}}\left[X_{\mathcal{V}^i\cap\mathcal{V}^{\tilde{i}}}\right] = \left\langle \{\tau^{C_j} \,:\, C_j \prec C_i\}\right\rangle_{\left[X_{\mathcal{V}^i\cap\mathcal{V}^{\tilde{i}}}\right]} . \qquad \square$$

Theorem 2. *When the cluster graph is a tree satisfying the running intersection property, then we have for each cluster C_i with neighbors $\mathrm{N}(i)$*

$$\left\langle \tau^{\mathcal{G}}\right\rangle_{\left[X_{\mathcal{V}_i}\right]} = \left\langle \{\delta_{j\to i}\left[X_{\mathcal{V}^i\cap\mathcal{V}^j}\right] \,:\, j \in N(i)\} \cup \{\tau^{C_i}\}\right\rangle_{\left[X_{\mathcal{V}_i}\right]} . \tag{1}$$

Proof. We use the topological order $\prec$ of the clusters by the tree, when choosing a root by cluster C_i.

The claim then follows from Thm. 1 and Lem. 1.

$\square$

While we have defined message passing along the topological order of a graph, we can also define messages against the topological order, that is

$$\delta_{j\leftarrow i} = \left\langle \{\delta_{i\leftarrow\tilde{j}} \,:\, C_i \prec C_{\tilde{j}}\} \cup \tau^{C_i}\right\rangle_{\left[X_{\mathcal{V}_i\cap\mathcal{V}_j}\right]}$$

To this end, we can get a similar statement for nodes, which are not the rotes of the cluster tree. The constractions at each cluster can then be computed batchwise, based on message passed along a topological order and against.

These message passing schemes can be derived from Lagrangian parameters given a local consistency polytope **?**.

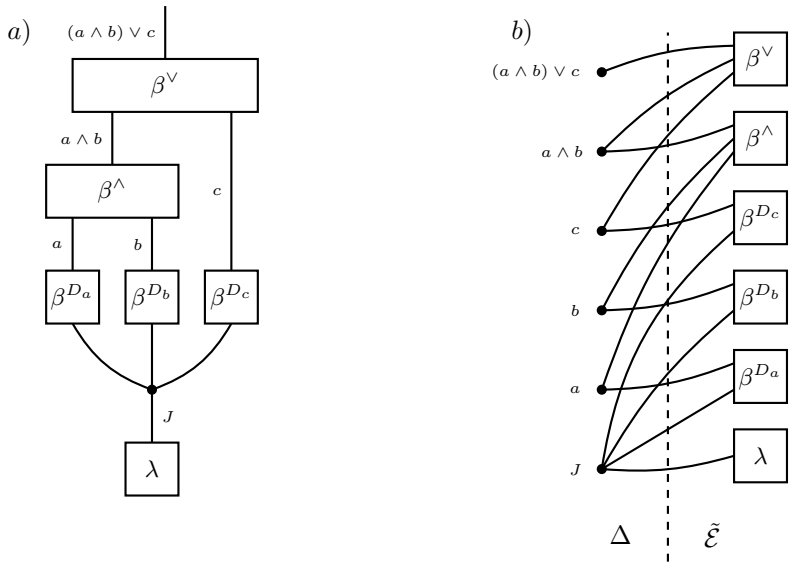

Figure 1: Example of a Bethe Cluster Graph. a) Example of a Tensor Network $\tau^{\mathcal{G}}$, which represents the by λ averaged evaluation of the formula $(a \wedge b) \vee c$ on data D. b) Corresponding Bethe Cluster Hypergraph, which dual is bipartite by the sets Δ and $\tilde{\mathcal{E}}$.

1.3 Variable Elimination Cluster Graphs

Remark 1 (Construction of Cluster Graphs by Variable Elimination)**.** *Following an elimination order of the colors, mark those tensors containing the colors, which have not been marked before, as the cluster. A clique tree can be constructed by these cluster, when iterating through the clusters and either connect them to previous disconnected clusters or leave the current cluster disconnected. Add the disconnected clusters with the current cluster in case there are overlaps of their open colors. If the disconnected cluster added has more open colors,*

1.4 Bethe Cluster Graphs

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).

Definition 2. *Given a tensor network $\tau^{\mathcal{G}}$ on a decorated hypergraph $\mathcal{G}$, we define the Bethe Cluster 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 $\mathcal{U} = \bigcup_{e \in \mathcal{E}} \tilde{e}$*
- *Delta Edges $\Delta^{\mathcal{G}} = \big\{\{v^e : e \ni v\} : v \in \mathcal{V}\big\}$, each of which decorated by a delta tensor $\delta^{\{v^e : e \ni v\}}$*

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

The dual 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).

2 Sound 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.

2.1 Monotonicity of Tensor Contraction

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 it marks the nonzero coordinates by 1.

We show that adding boolean tensor cores to an 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

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

□

2.2 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 $X_{\mathcal{V}}$*

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

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

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

□

Remark 2. *Similar statements hold, when dropping the non-negativity assumption on the, but demanding that all variables are left open.*

3 Complete Unit Clause Propagation (UCP)

Investigate three conditions, where constraint propagation is also complete.

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

UCP is Knowledge Propagation (see Algorithm 1 in the report) in case of

- Domain edges by single variable nodes, i.e.

$$\mathcal{E}^k = \{\{v\} : v \in \mathcal{V}\}$$

- Initial queue by those edges containing single nodes

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

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 $v \in A$

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

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

Definition 3. *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.*

3.1 Forward Chaining for Horn-SAT

Definition 4 (Definite Horn-SAT)**.** *Let $\{\tau^e[X_e] : e \in \mathcal{E}\}$ 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 2. *Let $\tau^{\mathcal{G}} = \{\tau^e[X_e] : e \in \mathcal{E}\}$ 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. □

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. □

3.2 UCP for 2-SAT

Definition 5 (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 3. *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 $x_v^{s_c} = 0$). □

3.3 UCP for Tree-SAT

Definition 6 (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. □

4 Outlook

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 **?**).

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.

5 Basis Propagation

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.

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.

Passing a message ϵ_i in direction thus gives the message $\epsilon_{q(i)}$.

Note, that this is exact, whenever the graph is directed and acyclic. We do not need acyclicity of the underlying undirected graph. Make a theorem!

Distinguish:

- Single inference: Basis propagation
- Batchwise inference: "Tensor Parallelism", but in the most interesting cases not exact.

Remark 3 (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*

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