$\mathrm{tnreason}$ FOR CONSTRAINT SATISFACTION PROBLEMS

RESEARCH NOTES IN THE ENEXA AND QROM PROJECTS

January 21, 2026

We show, how Constraint Satisfaction Problems can be formalized by boolean tensor networks in the $\mathrm{tnreason}$ notation, and how their solution can be approached by tree search, possible in combination with monte carlo approaches.

1 Formalization

Definition 1. *Let there be a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $\tau^{\mathcal{G}}$ be a tensor network of boolean constraint tensors $\tau^{e}[X_e]$ to each $e \in \mathcal{E}$, that is*

$$\tau^{\mathcal{G}} = \{\tau^{e}[X_e] \,:\, e \in \mathcal{E}\} .$$

The Constraint Satisfaction Problem CSP to $\tau^{\mathcal{G}}$ is the decision whether there is a state $x_{\mathcal{V}}$ such that

$$\left\langle \tau^{\mathcal{G}} \right\rangle_{[X_{\mathcal{V}}=x_{\mathcal{V}}]} = 1 .$$

We say the CSP is satisfiable, when there is such a state, and unsatisfiable if not.

2 Solution

2.1 Global contraction

The most obvious solution is to contract the tensor network and decide the CSP based on

$$\left\langle \tau^{\mathcal{G}} \right\rangle_{[\varnothing]} > 0 .$$

This can be demanding, when having a large and densely connected tensor network.

2.2 Message passing

Local contractions instead of global provide a tradeoff between generality and efficiency. Their results can be passed to further contractions by message passing. Whenever a local contraction vanishes, the CSP is unsatisfiable. However, the converse is not true: In general, we cannot conclude that the CSP is satisfiable, when a message-passing scheme does converge without vanishing.

2.3 Guessing

To decide a CSP it is enough to construct a state $x_{\mathcal{V}}$ which satisfies all constraint tensors. One procedure is to consecutively guess the state of some variables $v \in \mathcal{V}$ and check, whether the guess can satisfy the constraint tensor.

2.4 Orchestration in a Backtracking Search Tree

We build a search tree for a constructive solution of the CSP, where each node represents a guess of a variables state. After a guess, a message passing scheme of varying complexity is performed to calculate the imediate consequences of the guess. Then an intelligent agent decides whether to

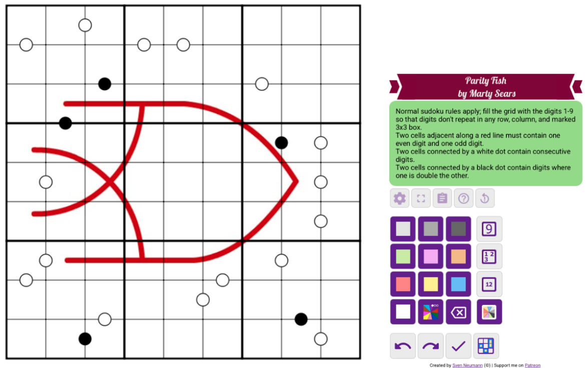

Figure 1: Sakana Fish Puzzle

- stop the message passing scheme
- increase the complexity of the scheme by extending the set of communicated variables, or the size of the contraction
- continue with a further guess of another variable
- undo the previous guess (e.g. necessary when reached a vanishing message indicating inconsistency)

3 Examples

3.1 Temporal Clue Game

- Nodes $\mathcal{V}$: To each characteristic of a murder (who, when, where, how etc) there is a categorical variable X_v with finite possibilities
- Edges $\mathcal{E}$: To each clue and accusation, there is a constraint tensor storing the possible states consistent with the clue

3.2 Graph Coloring

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ we design

- a variable to each nodes $v \in \mathcal{V}$ carrying the color
- a constraint to each edge $e \in \mathcal{E}$ by differing colors, i.e. $\tau^{(v_1,v_2)}\left[X_{v_1}, X_{v_2}\right] = \mathbb{I}^{x_{v_1} \neq x_{v_2}}\left[X_{v_1}, X_{v_2}\right]$

3.3 Knowledge Base Satisfiability

As described in the main report.

3.4 Sudoku

See the computation activation drafts in the summary.

3.4.1 Sakana Fish Puzzle

In addition to the standard Sudoku rules,let us consider the constraints depicted in Figure 1.

Odd-even indicating hidden variables The central hidden variable to be added is whether there is an even or odd digit at each position, denoted by $Y_{r0,r1,c0,c1}$. We define the even checker as the function mapping even numbers to 1 and odd to 0, i.e.

$$E : [n^2] \to [2] .$$

Now the even constraint is implemented

$$\begin{aligned}\beta^E\left[Y_{r0,r1,c0,c1}, I_{r0,r1,c0,c1,:}\right] =&\epsilon_1\left[Y_{r0,r1,c0,c1}\right] \otimes \left(\sum_{i\in[n^2]\,:\,i \text{ is even}} \epsilon_i\left[I_{r0,r1,c0,c1,:}\right]\right) \\ &+ \epsilon_0\left[Y_{r0,r1,c0,c1}\right] \otimes \left(\sum_{i\in[n^2]\,:\,i \text{ is odd}} \epsilon_i\left[I_{r0,r1,c0,c1,:}\right]\right) .\end{aligned}$$

Implementing red line constraints Now along all pairs $(r0, r1, c0, c1), (\tilde{r0}, \tilde{r1}, \tilde{c0}, \tilde{c1})$ the red lines we add constraint cores

$$\oplus\left[Y_{r0,r1,c0,c1}, Y_{\tilde{r0},\tilde{r1},\tilde{c0},\tilde{c1}}\right]$$

implementing the constraint, that odd-even digits are alternating.

Implementing white dot constraints These constraints can be implemented by cores to positions $(r0, r1, c0, c1), (\tilde{r0}, \tilde{r1}, \tilde{c0}, \tilde{c1})$ connected via a white dot. The cores have coordinates

$$C^{\text{white}}_{(r0,r1,c0,c1),(\tilde{r0},\tilde{r1},\tilde{c0},\tilde{c1})}\left[I_{r0,r1,c0,c1} = i_{r0,r1,c0,c1}, I_{\tilde{r0},\tilde{r1},\tilde{c0},\tilde{c1}} = i_{\tilde{r0},\tilde{r1},\tilde{c0},\tilde{c1}}\right] = \begin{cases} 1 & \text{if} \quad \left|i_{r0,r1,c0,c1} - i_{\tilde{r0},\tilde{r1},\tilde{c0},\tilde{c1}}\right| = 1 \\ 0 & \text{else}\end{cases} .$$

Note that they entail the constraints of red lines:

$$\oplus\left[Y_{r0,r1,c0,c1}, Y_{\tilde{r0},\tilde{r1},\tilde{c0},\tilde{c1}}\right]$$

Implementing black dot constraints Analogously, the black dots can be implemented by

$$C^{\text{black}}_{(r0,r1,c0,c1),(\tilde{r0},\tilde{r1},\tilde{c0},\tilde{c1})}\left[I_{r0,r1,c0,c1} = i_{r0,r1,c0,c1}, I_{\tilde{r0},\tilde{r1},\tilde{c0},\tilde{c1}} = i_{\tilde{r0},\tilde{r1},\tilde{c0},\tilde{c1}}\right] = \begin{cases} 1 & \text{if} \quad i_{r0,r1,c0,c1} = 2 * i_{\tilde{r0},\tilde{r1},\tilde{c0},\tilde{c1}} \\ 1 & \text{if} \quad i_{r0,r1,c0,c1} = 0.5 * i_{\tilde{r0},\tilde{r1},\tilde{c0},\tilde{c1}} \\ 0 & \text{else}\end{cases} .$$

Note that they entail the constraints (since at least one of the neighbors needs is even)

$$\vee\left[Y_{r0,r1,c0,c1}, Y_{\tilde{r0},\tilde{r1},\tilde{c0},\tilde{c1}}\right] .$$

Solution approach Can we start the reasoning process by focusing only on the Y. variables? E.g. guess one, propagate on the rest and see if there is a contradiction.

4 Toy Example for MCTS

As a toy example, let there be a knowledge base of atoms $A1$ and $A2$ representing two accounts to be used in an accounting proposal, and E represents incoming invoices. The constraints are that exactly one of $A1$ or $A2$ is booked, that is $X_{A1} \oplus X_{A2}$.

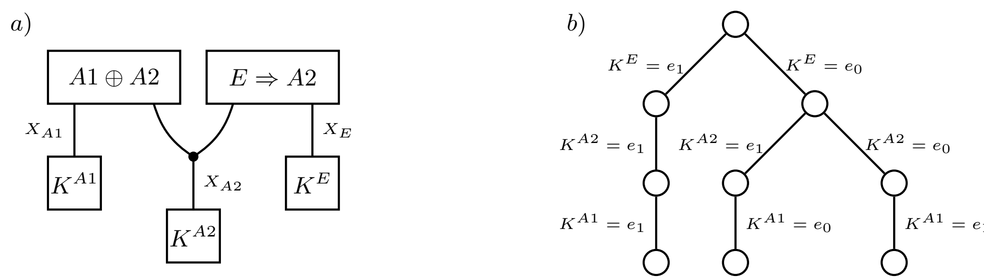

Figure 2: a) Tensor Network Representation of a toy CSP, in which two formulas $A1 \oplus A2$ and $E \Rightarrow A2$ have to be satisfied. The Knowledge Cores K contain the possible choices after a constraint propagation and are taken from $e_0, e_1, \underline{1}, \underline{0}$. b) The corresponding search tree, where the assignments to variables are guessed in the order $E, A2, A1$. At each parent with a single child, the other choice has been ruled out by constraint propagation. In our toy example with minimally connected constraints, single-core constraint propagation ensures that the guessed reasoning path remains consistent.