

# Computer Assignment: Belief-Propagation for Sudoku

**Overview.** In this assignment you will model Sudoku as a constraint satisfaction problem (CSP) using a factor graph and implement a test solver with sum-product (belief propagation).

**Setup and notation.** Let  $x_{r,c} \in \{1, \dots, 9\}$  denote the value in row  $r$  and column  $c$  for  $r, c \in \{1, \dots, 9\}$ . Let  $x$  denote the vector of all 81 variables. A factor graph represents a nonnegative function  $f(x) = \prod_{a \in F} f_a(x_{\partial a})$ , and defines a distribution

$$\mu(x) = \frac{1}{Z} f(x), \quad Z = \sum_x f(x).$$

Observed clues  $\{y_{r,c}\}_{(r,c) \in O}$ , where  $O$  is the set of observed locations, are encoded as unary factors.

1. **Sudoku as a CSP with observations.** Let  $f_\sigma(x_1, \dots, x_9)$  be a function that equals 1 if  $(x_1, \dots, x_9)$  is a permutation of  $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$  and zero otherwise. Describe Sudoku as a CSP with three groups of constraints: rows, columns, and  $3 \times 3$  subblocks. Show how observed clues are defined by unary factors  $f_{r,c}(x_{r,c})$  which are 1 for the observed value and 0 otherwise. Write the full factorization of  $f(x)$  using permutation factors and unary factors.
2. **All-different constraints and pairwise factorization.** Each row/column/subgrid constraint  $f_\sigma: \mathcal{X}^9 \rightarrow \{0, 1\}$  is a *permutation* constraint on 9 variables taking values in  $\{1, 2, \dots, 9\}$  (i.e., it evaluates to 1 if pattern is a permutation of the alphabet and 0 otherwise). Since the number of variables equals the alphabet size, this is equivalent to an *all-different* constraint  $f_{\neq}: \mathcal{X}^9 \rightarrow \{0, 1\}$  on a set of 9 variables (i.e., it evaluates to 1 if all variables are different from each other and 0 otherwise).

Explain how an all-different constraint  $f_{\neq}$  on 9 variables can be factorized into 36 pairwise not-equal constraints  $f_{\neq}: \mathcal{X}^2 \rightarrow \{0, 1\}$  by using the complete graph on those 9 variables. Give an explicit formula for  $f_\sigma(x_1, \dots, x_9)$  in terms of the pairwise not-equal factors.

What is the total number of pairwise not-equal constraints required for Sudoku?

3. **Low-complexity BP update for not-equal factors.** For alphabet size  $q$  with pairwise not-equal factors  $f_{ij}(x_i, x_j) = f_{\neq}(x_i, x_j)$  enforcing  $x_i \neq x_j$ , use sum-product in the probability domain, derive the factor-to-variable update

$$\hat{m}_{ij \rightarrow i}(x_i) = \sum_{x_j=1}^q f_{\neq}(x_i, x_j) m_{j \rightarrow ij}(x_j).$$

Show this can be computed for all  $x_i$  in  $O(q)$  time by precomputing  $S = \sum_{x_j=1}^q m_{j \rightarrow \psi}(x_j)$  and using

$$\hat{m}_{ij \rightarrow i}(x_i) = S - m_{j \rightarrow \psi}(x_i).$$

Then write the variable-to-factor update for a Sudoku variable, including its unary factor and all incoming pairwise messages.

- Detecting a satisfying assignment and early termination.** Describe a procedure to detect a satisfying assignment during BP iterations. Your factor update procedure can check if the most likely symbol for  $x_i$  is different from the most-likely symbol for  $x_j$  for all  $i, j$ . Also discuss what to do if any belief becomes identically zero (e.g., indicating that a contradiction has arisen).
- Implementation details.** Please implement the basic BP update loop running 50 iterations with damping factor  $\alpha$  set to 0.5. Use damping to stabilize updates by combining the standard message update  $m'_{i \rightarrow ij}(x)$  with the previous message  $m_{i \rightarrow ij}^{(t)}(x)$ :

$$m_{i \rightarrow ij}^{(t+1)}(x) = (1 - \alpha)m_{i \rightarrow ij}^{(t)} + \alpha m'_{i \rightarrow ij}(x), \quad x \in \mathcal{X}.$$

After each update, one can normalize messages so they sum to one over the alphabet. Note that one can compute the reciprocal and multiply to reduce the number of divisions by  $q$ . This keeps numbers bounded and ensures comparability of beliefs and also forces the value of  $S$  to 1 in the shortcut above. Describe any other details in the algorithm that you implement.

- Test set.** Use the provided test set of Sudoku puzzles which contains 50 easyish puzzles. My BP solver is successful on 27/50 of these examples when I use 400 iterations and 26/50 when I use 40 iterations. Each puzzle is represented by 9 lines of 9 digits with 0 used to denote blanks. The following test example should decode successfully with BP:

```
003020600
900305001
001806400
008102900
700000008
006708200
002609500
800203009
005010300
```

- Bells and whistles.** Please submit the (written completely by hand) BP solver (say for the hard coded example above) as part of your assignment along with details requested above. If the code doesn't fully work, that is not preferred but it is ok.

After that, you are allowed to use generative AI coding tools to do other things like: debugging, adding command line arguments, having the solver read the sudoku.txt file and execute on all examples, reporting the success rate and timing, adding visualization of the decoding process, and adding guided decimation and backtracking.

You will also submit the final GenAI version along with a few paragraphs describing exactly what features it has and how easy (or hard) it was to implement these with GenAI.