Passing messages to lonely numbers

Heiko Bauke

数独

Sudoku

Marginal distributions and message passing

Solving Sudoku by message passing

Sudoku

Sudoku

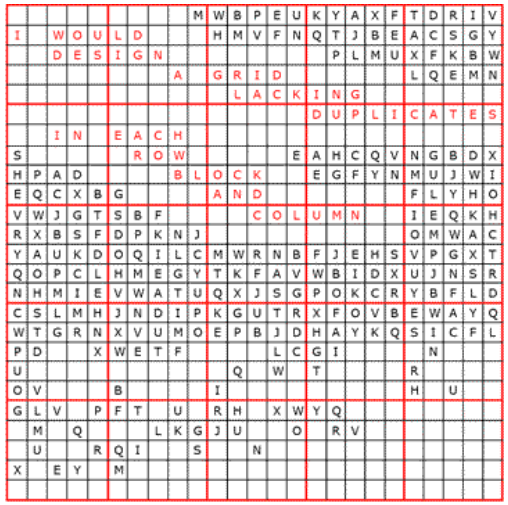
- popular Number Place puzzle (No math required!)
- first occurrence in “Dell Pencil Puzzles & Word Games” in 1979
- an example taken from “Die Zeit”, nr. 30, 2006

3	5	1	9	2	6	4	7	8
4	7	9	5	1	8	6	3	2
8	6	2	3	4	7	9	1	5
9	1	3	6	8	4	2	5	7
5	4	7	2	9	3	1	8	6
6	2	8	1	7	5	3	9	4
2	3	4	8	5	9	7	6	1
1	8	6	7	3	2	5	4	9
7	9	5	4	6	1	8	2	3

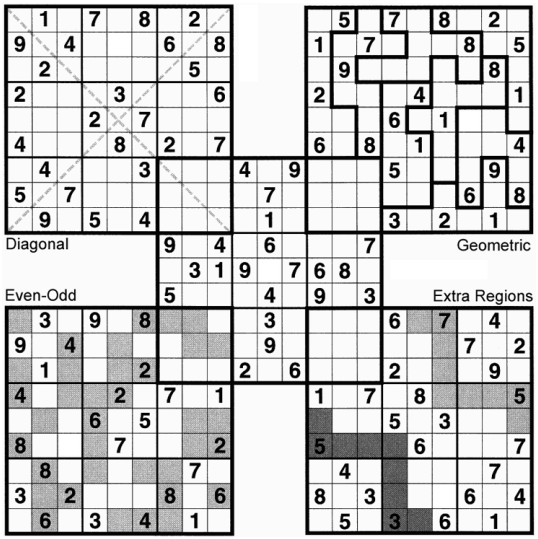
- 9×9 grid
- some cells already filled, called “givens”
- task: fill each column, row and sub-square with a permutation of $(1, 2, 3, 4, 5, 6, 7, 8, 9)$
- a “well formed” Sudoku has a unique solution

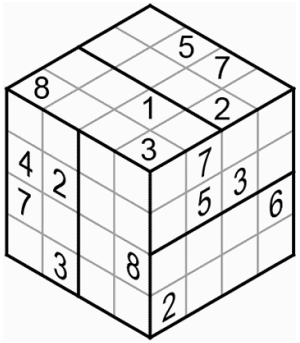
Sudoku variants

- larger Sudokus

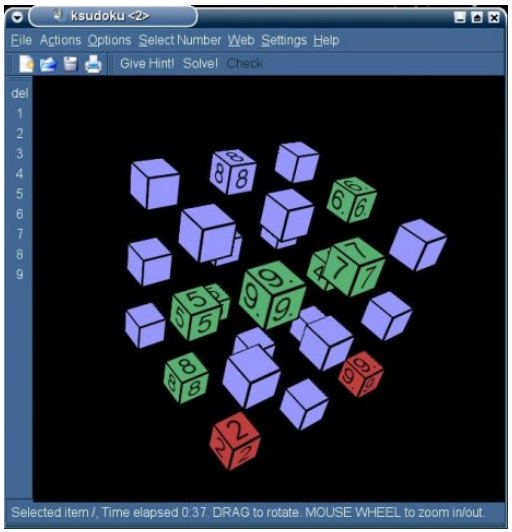


- other Sudoku variants





- three-dimensional Sudoku



Marginal distributions and message passing

Motivations and applications

- Message passing algorithms take techniques from statistical mechanics to calculate (or approximate)
  - marginal probability distributions or
  - partition functions
- and find applications in
  - statistical physics,
  - combinatorial optimization,
  - artificial intelligence,
  - signal processing, and
  - digital communications (coding), and …

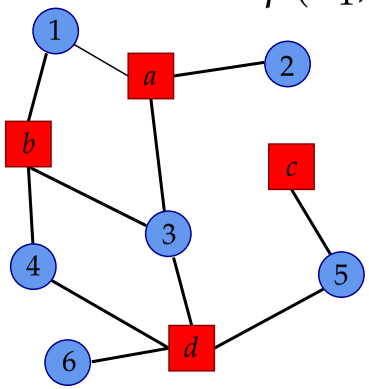
Factor graphs

- marginal distributions over finite sets $x_k \in \mathcal{X}$

$$p_i(x_i) = \sum_{\sim x_i} p(x_1, x_2, \ldots, x_N) = \sum_{\sim x_i} \prod_j f_j(\mathbf{X}_j)$$

- factor graph

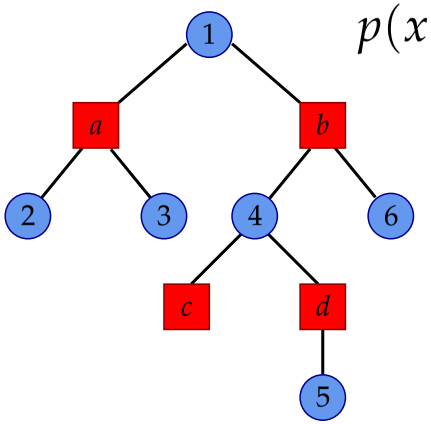
$$p(x_1, x_2, \ldots, x_6) = f_a(x_1, x_2, x_3) f_b(x_1, x_3, x_4) f_c(x_5) f_d(x_3, x_4, x_5, x_6)$$



$$\mathbf{X}_a = \{x_1, x_2, x_3\}$$
$$\mathbf{X}_b = \{x_1, x_3, x_4\}$$
$$\mathbf{X}_c = \{x_5\}$$
$$\mathbf{X}_d = \{x_3, x_4, x_5, x_6\}$$

- factor tree

$$p(x_1, x_2, \ldots, x_6) = f_a(x_1, x_2, x_3) f_b(x_1, x_4, x_6) f_c(x_4) f_d(x_4, x_5)$$



$$\mathbf{X}_a = \{x_1, x_2, x_3\}$$
$$\mathbf{X}_b = \{x_1, x_4, x_6\}$$
$$\mathbf{X}_c = \{x_4\}$$
$$\mathbf{X}_d = \{x_4, x_5\}$$

Distributive law on factor trees

- marginalization: difficult in general, but easy on trees

$$p(x_1) = \sum_{\sim x_1} f_1(x_1, x_2, x_3, x_4)\, f_2(x_1, x_5, x_6)\, f_3(x_3, x_7, x_8)\, f_4(x_4)\, f_5(x_5, x_9, x_{10}) \cdot f_6(x_5)\, f_7(x_6, x_{11}, x_{12})\, f_8(x_8)\, f_9(x_{10}, x_{13}, x_{14})\, f_{10}(x_{12}, x_{15}, x_{16})$$

- reorder terms

$$p(x_1) = \sum_{\sim x_1} \left[f_1(x_1, x_2, x_3, x_4)\, f_3(x_3, x_7, x_8)\, f_4(x_4)\, f_8(x_8) \right] \cdot \left[f_2(x_1, x_5, x_6)\, f_5(x_5, x_9, x_{10})\, f_6(x_5)\, f_7(x_6, x_{11}, x_{12})\, f_9(x_{10}, x_{13}, x_{14})\, f_{10}(x_{12}, x_{15}, x_{16}) \right]$$

- apply distributive law

$$p(x_1) = \overbrace{\sum_{\sim x_1} f_1(x_1, x_2, x_3, x_4)\, f_3(x_3, x_7, x_8)\, f_4(x_4)\, f_8(x_8)}^{p'(x_1)} \cdot \underbrace{\sum_{\sim x_1} f_2(x_1, x_5, x_6)\, f_5(x_5, x_9, x_{10})\, f_6(x_5)\, f_7(x_6, x_{11}, x_{12})\, f_9(x_{10}, x_{13}, x_{14})\, f_{10}(x_{12}, x_{15}, x_{16})}_{p''(x_1)}$$

- marginal of product $\Rightarrow$ product of marginals

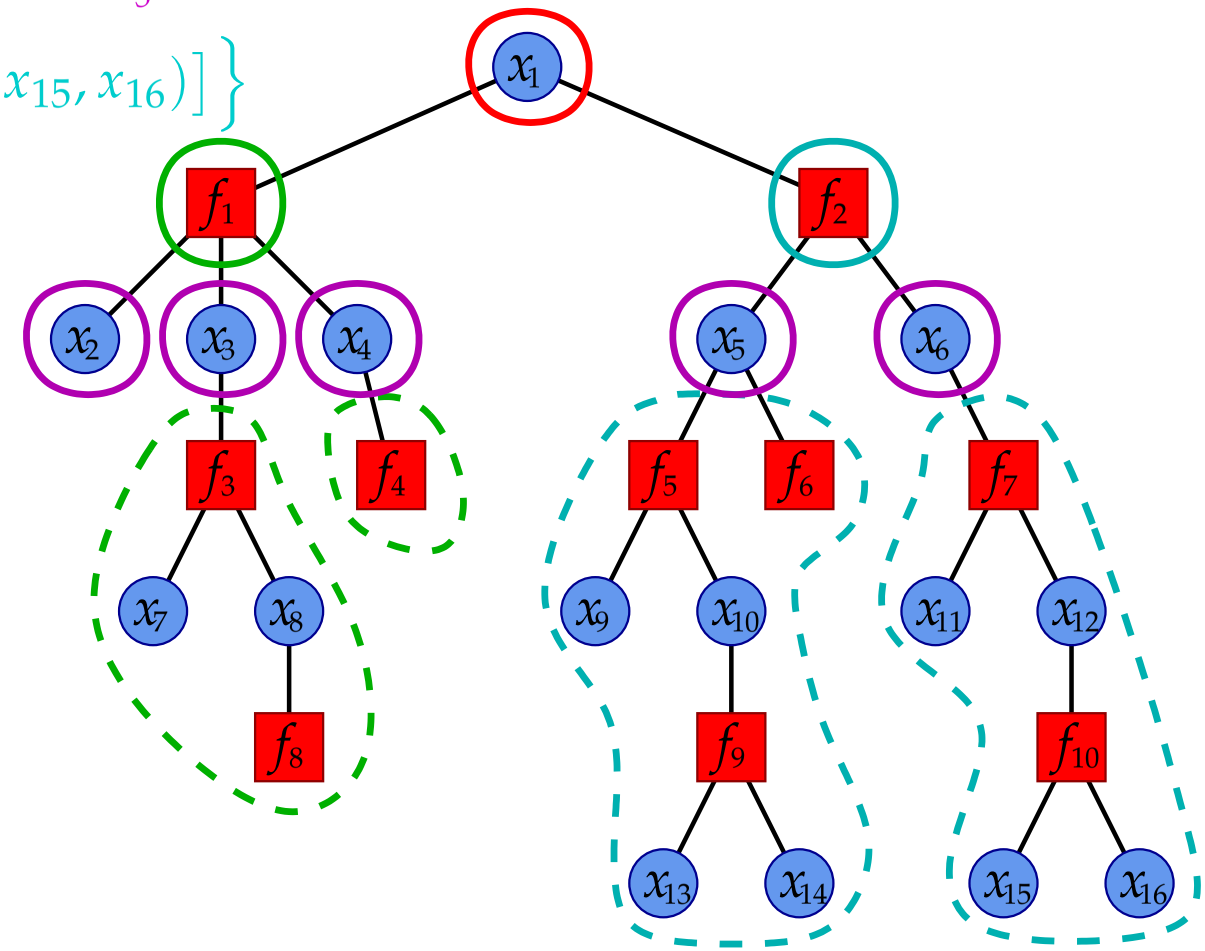
Distributive law on factor trees

- reorder and apply distributive law again

$$p'(x_1) = \sum_{x_2,x_3,x_4} \Big\{ f_1(x_1,x_2,x_3,x_4)\,[1]\,\Big[\sum_{\sim x_3} f_3(x_3,x_7,x_8)\,f_8(x_8)\Big]\,\Big[\sum_{\sim x_4} f_4(x_4)\Big]\Big\}$$

$$p''(x_1) = \sum_{x_5,x_6} \Big\{ f_2(x_1,x_5,x_6)\,\Big[\sum_{\sim x_5} f_5(x_5,x_9,x_{10})\,f_6(x_5)\,f_9(x_{10},x_{13},x_{14})\Big]\cdot$$

$$\Big[\sum_{\sim x_6} f_7(x_6,x_{11},x_{12})\,f_{10}(x_{12},x_{15},x_{16})\Big]\Big\}$$

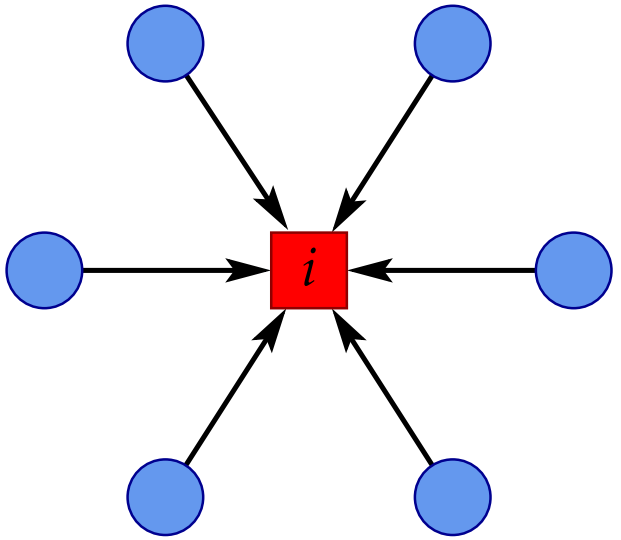


- $p'(x_1)$ and $p''(x_1)$ sums over *independent* child-marginals
- apply distributive law recursively until leaves reached

Marginalization by message passing

- recursive determination of marginals, sending messages bottom-up
- initialization at leaf node i

$$\mu_{i\to k}(x_i) = 1 \qquad\qquad \nu_{i\to k}(x_k) = f_i(x_k)$$



- variable/function node processing at node i

$$\mu_{i\to k}(x_i) = \prod_{j\in\{\text{n.n. } i\}\setminus\{k\}} \nu_{j\to i}(x_i)$$

$$\nu_{i\to k}(x_k) = \sum_{\mathbf{X}_i\setminus\{k\}} f_i(\mathbf{X}_i) \prod_{j\in\{\text{n.n. } i\}\setminus\{k\}} \mu_{j\to i}(x_j)$$

- marginalization at node i (calculate beliefs)

$$p_i(x_i) = \prod_{j\in\{\text{n.n. } i\}} \nu_{j\to i}(x_i)$$

$$p_{\mathbf{X}_i}(\mathbf{X}_i) = \prod_{j\in\{\text{n.n. } i\}} \mu_{j\to i}(x_j) = f_i(\mathbf{X}_i) \prod_{\substack{j\in\{\text{n.n. } i\}\\ l\in\{\text{n.n. } j\}\setminus\{i\}}} \nu_{l\to j}(x_j)$$

A detailed example

- Ising-spin energy function

$$E(s_1, s_2, \ldots, s_8) = 1 \cdot s_1 - 3 \cdot s_4 + 2 \cdot s_4 s_8 - 2 \cdot s_3 s_4 s_7 - 2 \cdot s_1 s_2 s_7 - 1 \cdot s_5 s_6 s_7$$

- marginal probability distributions

$$\begin{aligned} p_i(s_i) &\sim \sum_{\sim s_i} \exp\left[-\beta E(s_1, s_2, \ldots, s_8)\right] \\ &\sim \sum_{\sim s_i} \mathrm{e}^{-1\beta s_1} \cdot \mathrm{e}^{3\beta s_4} \cdot \mathrm{e}^{-2\beta s_4 s_8} \cdot \mathrm{e}^{2\beta s_3 s_4 s_7} \cdot \mathrm{e}^{2\beta s_1 s_2 s_7} \cdot \mathrm{e}^{1\beta s_5 s_6 s_7} \end{aligned}$$

- $s_i \in \mathcal{X} = \{-1, 1\}$, sending messages $(\mu(-1), \mu(1))$ or $(\nu(-1), \nu(1))$
- parallel computation of *all* marginal distributions $p_i(x_i)$

Computational complexity

- naive computation

$$\mathcal{O}\left(N|\mathcal{X}|^N\right)$$

- message passing

$$\mathcal{O}\left(\sum_i d(f_i)|\mathcal{X}|^{d(f_i)}\right)$$

- if $\max_i d(f_i)$ independent of $N \Rightarrow$ polynomial complexity

Message passing in loopy graphs

The sum product algorithm:

- marginalization by message passing on trees
  $\Rightarrow$ exact
- marginalization by message passing on factor graphs with cycles
  $\Rightarrow$ approximation
  - choose initial messages (uniform, random, educated guess)
  - update messages iteratively (parallel, sequential, random)
  - normalize messages
  - stop if messages converged to a fixed point
  - convergence not guaranteed
  - but works pretty good
    if not too much short cycles, no frustration

Bethe-Peierls approximation

- magnetization of the Ising model $\widehat{=}$ probability that spin points up or down

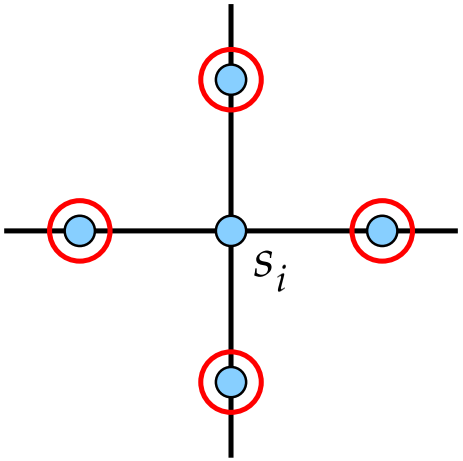
$$p(s_i) \sim \sum_{\substack{s_j \in \{-1,1\} \\ j \neq i}} \exp\left[-\beta H(s_1, \ldots, s_N)\right]$$

$$\sim \sum_{\substack{s_j \in \{-1,1\} \\ j \neq i}} \prod_{k \text{ and } l \text{ n.n.}} \exp\left(\beta s_k s_l\right)$$

- Bethe-Peierls approximation replaces lattice by effective fields

$$p(s_i) \sim \sum_{\substack{s_j \in \{-1,1\} \\ j \in \{\text{n.n. } i\}}} \exp\Big(\sum_{j \in \{\text{n.n. } i\}} \beta s_i s_j + \sum_{j \in \{\text{n.n. } i\}} \beta s_j h_{j \setminus i} \Big)$$

$$\sim \sum_{\substack{s_j \in \{-1,1\} \\ j \in \{\text{n.n. } i\}}} \exp\Big(\sum_{j \in \{\text{n.n. } i\}} \beta s_i s_j \Big) \prod_{j \in \{\text{n.n. } i\}} p_{j \setminus i}(s_j)$$



- self-consistency condition

$$p_{j \setminus i}(s_j) \sim \sum_{\substack{s_k \in \{-1,1\} \\ k \in \{\text{n.n. } j\} \setminus \{i\}}} \exp\Big(\sum_{k \in \{\text{n.n. } j\} \setminus \{i\}} \beta s_j s_k \Big) \prod_{k \in \{\text{n.n. } j\} \setminus \{i\}} p_{k \setminus j}(s_k)$$

Message passing and statistical mechanics

- Bethe free energy $F_\mathrm{B} = U_\mathrm{B} - H_\mathrm{B}$ in terms of marginal distributions $p_i(x_i)$ and $p_{\boldsymbol{X}_i}(\boldsymbol{X}_i)$

$$U_\mathrm{B} = -\sum_i \sum_{\boldsymbol{X}_i} p_{\boldsymbol{X}_i}(\boldsymbol{X}_i) \ln f_i(\boldsymbol{X}_i)$$

$$H_\mathrm{B} = -\sum_i \sum_{\boldsymbol{X}_i} p_{\boldsymbol{X}_i}(\boldsymbol{X}_i) \ln p_i(\boldsymbol{X}_i) + \sum_i (d(x_i) - 1) p_i(x_i) \ln p_i(x_i)$$

- marginal distributions $p_i(x_i)$ and $p_{\boldsymbol{X}_i}(\boldsymbol{X}_i)$ are constrained variables, complex dependence on x_i
- fixed points of MP with positive beliefs $\widehat{=}$
  interior stationary points of constrained F_B
- $\forall$ factors soft $\Rightarrow$ $\forall$ local minima of F_B are fixed points of MP and at least one fixed point exists

$$f_i(\boldsymbol{X}_i) \text{ soft factor } \widehat{=}\ f_i(\boldsymbol{X}_i) > 0 \quad \forall \boldsymbol{X}_i$$

Solving Sudoku by message passing

or

Passing messages to lonely numbers

Sudoku factor graph

- 81 fields $\widehat{=}$ variables x_i
- uniform probability measure over set of solutions

$$p(x_1, x_2, \ldots, x_{81}) = \frac{1}{Z} \prod_{i \in \text{cols, rows, squares}} f(x_{i_1}, x_{i_2}, \ldots, x_{i_9})$$

with

$$f(y_1, y_2, \ldots, y_9) = \mathbb{1}\left[(y_1, y_2, \ldots, y_9) \text{ is a perm. of } (1, 2, \ldots, 9)\right]$$

(hard constrains)

- deduce Sudoku solution from marginals,
assuming unique solution $(v_1, v_2, \ldots, v_{81})$

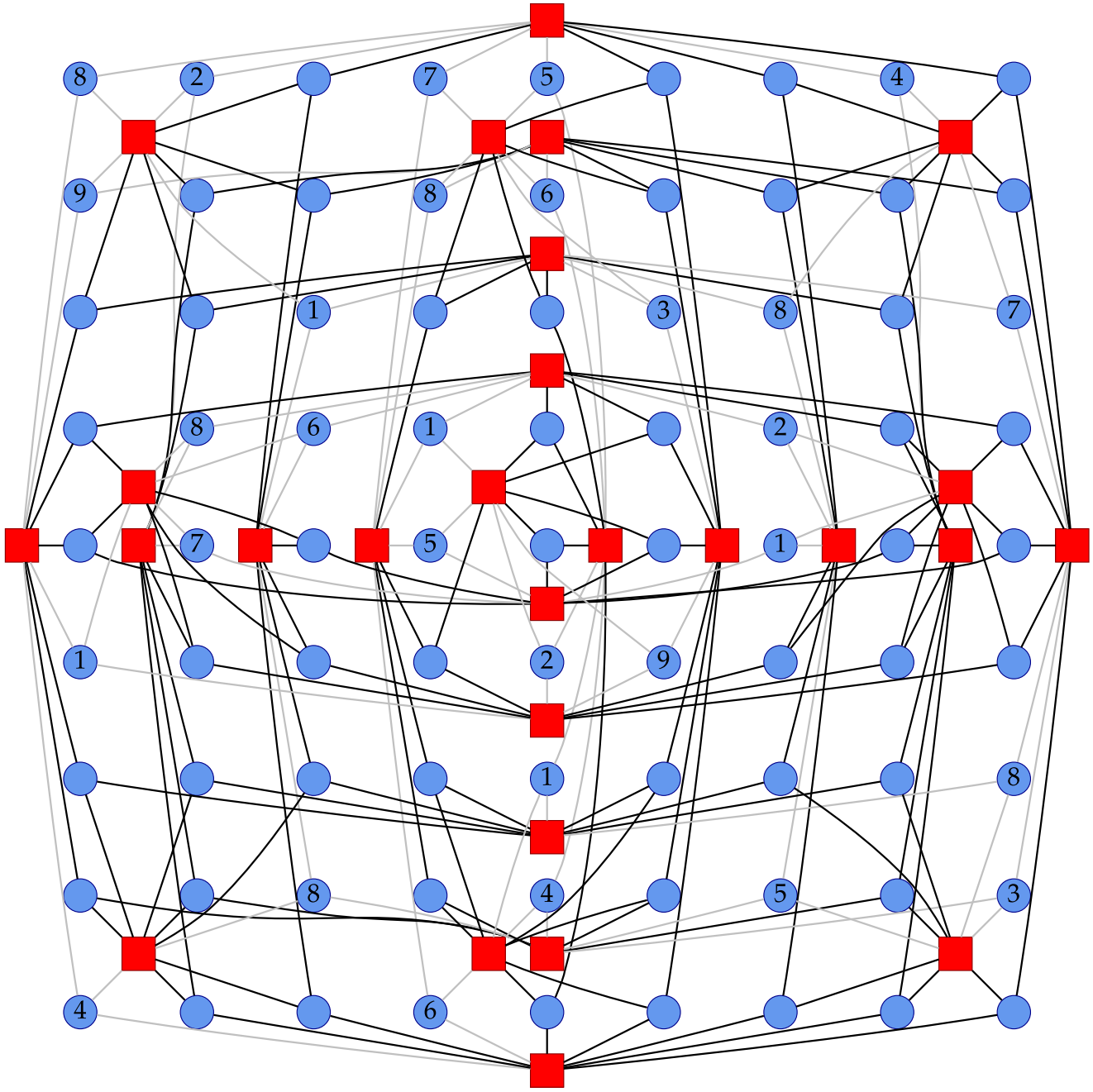
$$p_i(x_i) = \sum_{\substack{x_j \\ j \in \{1,2,\ldots,81\} \setminus \{i\}}} p(x_1, x_2, \ldots, x_{81}) = \mathbb{1}\left[x_i = v_i\right]$$

- probability distribution can be expressed in terms of a factor graph

Sudoku factor graph

- givens reduce factor graph and determine constrains (function nodes)
- but factor graph still very loopy
- Will message passing work?

8	2		7	5			4	
9			8	6				
		1			3	8		7
	8	6	1			2		
	7		5			1		
1				2	9			
				1				8
		8		4		5		3
4			6					



Message passing (sum product algorithm)

- $\mu_{i\to k}(x_i)$ and $\nu_{i\to k}(x_k)$: normalized probability distributions
- meaning:
  - If i a variable and k a function node: "I think, I am …"
  - If i a function and k a variable node: "I think, you are …"
- variable node processing:

$$\mu_{i\to k}(x_i) \sim \prod_{j\in\{\text{n.n. } i\}\setminus\{k\}} \nu_{j\to i}(x_i)$$

- function node processing: marginalization with respect to x_k

$$\nu_{i\to k}(x_k) \sim \sum_{\substack{x_k,x_l\in\{1,2,\ldots,9\},\\ l\in\{\text{n.n. } i\}\setminus\{k\}}} f(x_k,\ldots,x_l,\ldots) \prod_{j\in\{\text{n.n. } i\}\setminus\{k\}} \mu_{j\to i}(x_j)$$

$$f(y_1,y_2,\ldots,y_9) = \mathbb{1}\left[(y_1,y_2,\ldots,y_9) \text{ is a perm. of } (1,2,\ldots,9)\right]$$

Message initialization / update / marginalization

- message initialization
  - If neither i nor k a given variable node:

$$\mu_{i\to k}(x_i) = \tfrac{1}{9}\,, \qquad \text{for } \forall x_i$$

  - If i or k is a given variable node with value v_i:

$$\mu_{i\to k}(x_i) = \mathbb{1}\left[x_i = v_i\right]$$

- update schedules
  - parallel sequential: update either *all* messages
    - from function nodes to variable nodes *or*
    - from variable nodes to function nodes
  - random: choose a random pair i and k
- marginalization

$$p_i(x_i) \sim \prod_{j\in\{\text{n.n. of } i\}} \nu_{j\to i}(x_i)$$

Results for sum product algorithm

- Sudoku problems of low and average difficulty ($\gtrapprox 25$ givens)
  - message passing solves Sudoku almost always,
    marginals converge to $p_i(x_i) = (0, \ldots, 1, \ldots, 0)$
- Sudoku problems of maximal difficulty (only 17 givens)
  - message passing solves Sudoku often,
    55 % parallel sequential update, 63 % random update
  - short loops and numerical instabilities drive messages into contradictions
- If there are a lot of givens, message passing is equivalent to “logical deduction”.
- Sudoku problems with more than one solution
  - parallel sequential update: no convergence
  - random update: converges to some solution

Message passing (max product algorithm)

- try to avoid numerical instabilities and rounding errors
- message passing works on every commutative semiring $\mathbb{K}$
- for a commutative semiring we need
  - set of elements with a "$+$" with a "0", and a "$\cdot$" with a "1"
  - commutative and associative laws for "$+$" and "$\cdot$"
  - distributive law

$\mathbb{K}$	"$(+,0)$"	"$(\cdot,1)$"	algorithm
$\mathbb{R}_{\geq 0}$	$(+,0)$	$(\cdot,1)$	sum product
$\mathbb{R}_{\geq 0} \cup \{\infty\}$	$(\min,\infty)$	$(\cdot,1)$	min product
$\mathbb{R}_{\geq 0}$	$(\max,0)$	$(\cdot,1)$	max product
$\mathbb{R} \cup \{\infty\}$	$(\min,\infty)$	$(+,0)$	min sum
$\mathbb{R} \cup \{-\infty\}$	$(\max,-\infty)$	$(+,0)$	max sum

Message passing (max product algorithm)

- max product algorithm over $\mathbb{K} = (\{0,1\}, (\max, 0), (\cdot, 1))$
- messages $\mu_{i \to k}(x_i)$ and $\nu_{i \to k}(x_k)$: 9-tuples of 0 and 1 indicating possible field assignments, e. g. $(1,0,0,0,0,1,0,1,0)$
- meaning:
  - If i a variable and k a function node:
    "I know, I can be at most ..."
  - If i a function and k a variable node:
    "I know, you can be at most ..."
- variable node and function node processing as in sum product algorithm
- no numerical instabilities, outperforms sum product algorithm slightly (solves 71 % of Sudokus with 17 givens)
- combination of max product algorithm and sum product algorithm improves performace

Is MP a practical algorithm for Sudoku?

- No!
  - convergence not guaranteed
  - classical backtracking much faster
  - MP does not scale, function node update $O(N!)$

$$\nu_{i\to k}(x_k) \sim \sum_{\substack{x_k, x_l \in \{1,2,\ldots,N\}, \\ l \in \{\text{n.n. } i\}\setminus\{k\}}} f(x_k, \ldots, x_l, \ldots) \prod_{j \in \{\text{n.n. } i\}\setminus\{k\}} \mu_{j\to i}(x_j)$$

$$f(y_1, y_2, \ldots, y_N) = \mathbb{1}\left[(y_1, y_2, \ldots, y_N) \text{ is a perm. of } (1, 2, \ldots, N)\right]$$

  - message updates
    - NP-hard for sum product algorithm
    - polynomial complexity for max product algorithm (mapping to “maximum weight bipartite matching”)
  - combination of max product algorithm with sum product algorithm gives best performance

Conclusions

Message passing

- meta-algorithm in many forms
  (e. g. sum product, max product)
- messages are sent along edges of a factor graph
- exact on trees
- powerful approximation scheme, if graph not too loopy and no frustration
- if messages converge, sum product algorithm equivalent to Bethe approximation
- only useful if degree of functions
  nodes does not grow with system size
- message passing can solve Sudoku
- but not the best algorithm for Sudoku