Sudoku

An application of message passing

Heiko Bauke, Andrzej Görlich, Satish Korada,
Cyril Measson, Thierry Mora, Olivier Rivoire

数独

Classical Sudoku

Classical Sudoku

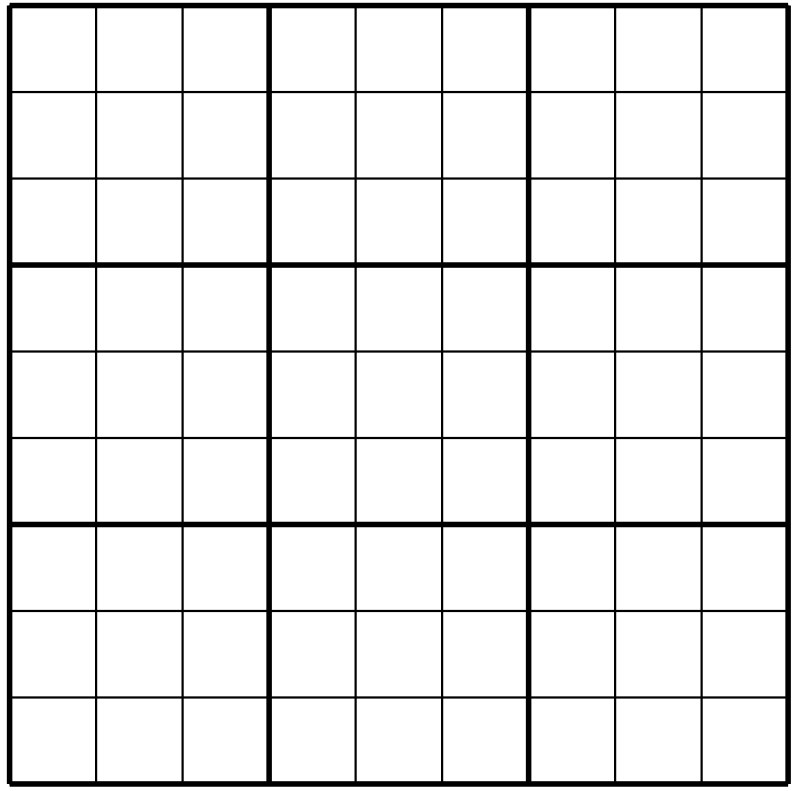
- popular Number Place puzzle (No math required!)

Classical Sudoku

- popular Number Place puzzle (No math required!)
- first occurrence in “Dell Pencil Puzzles & Word Games” in 1979

Classical 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



- 9×9 grid

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

- 9×9 grid
- some cells already filled, called “givens”

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

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

Classical Sudoku

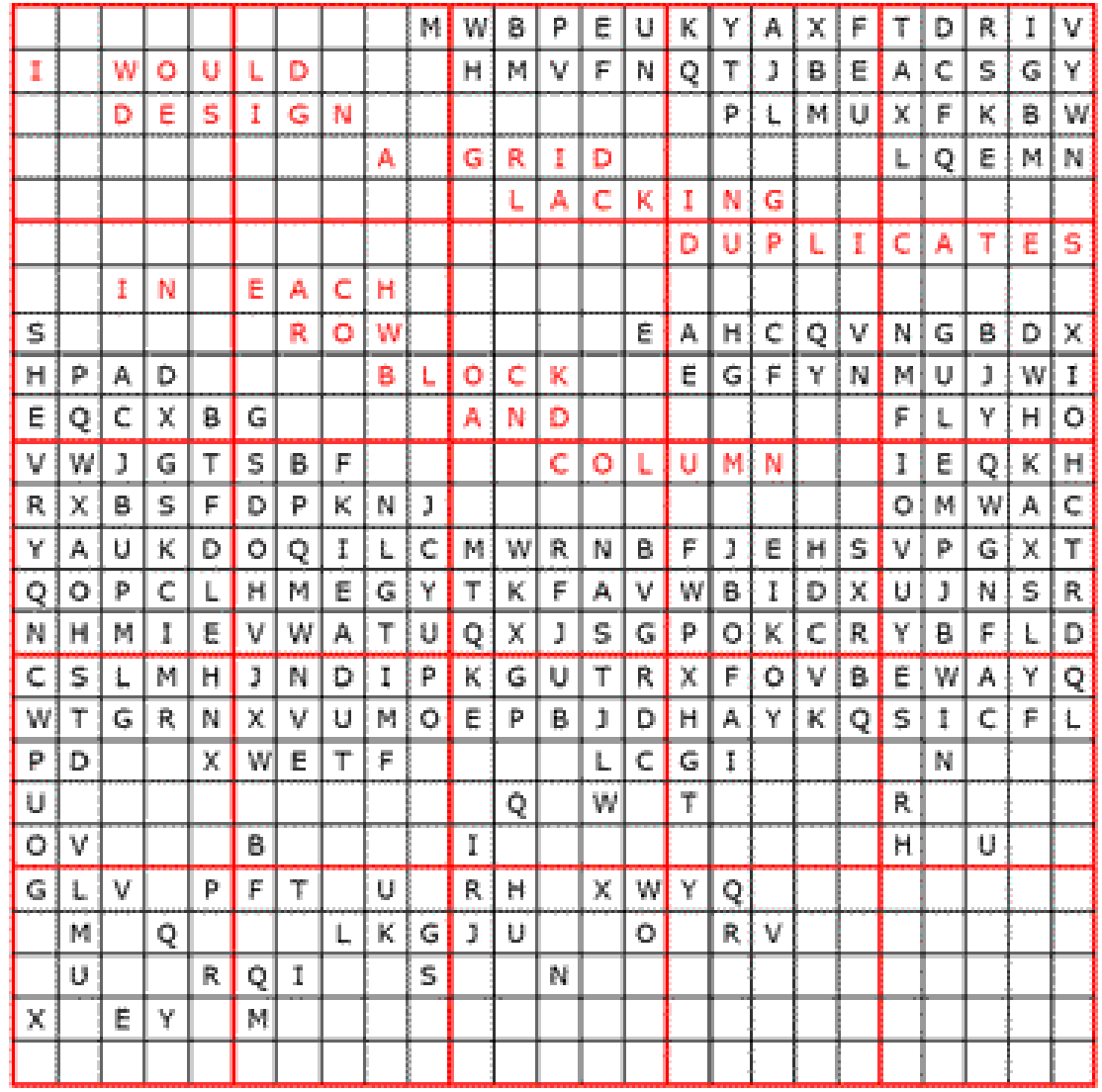
- popular Number Place puzzle (No math required!)
- first occurrence in “Dell Pencil Puzzles & Word Games” in 1979
- an example taken from “Die Zeit”, nr. 30, 2006

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

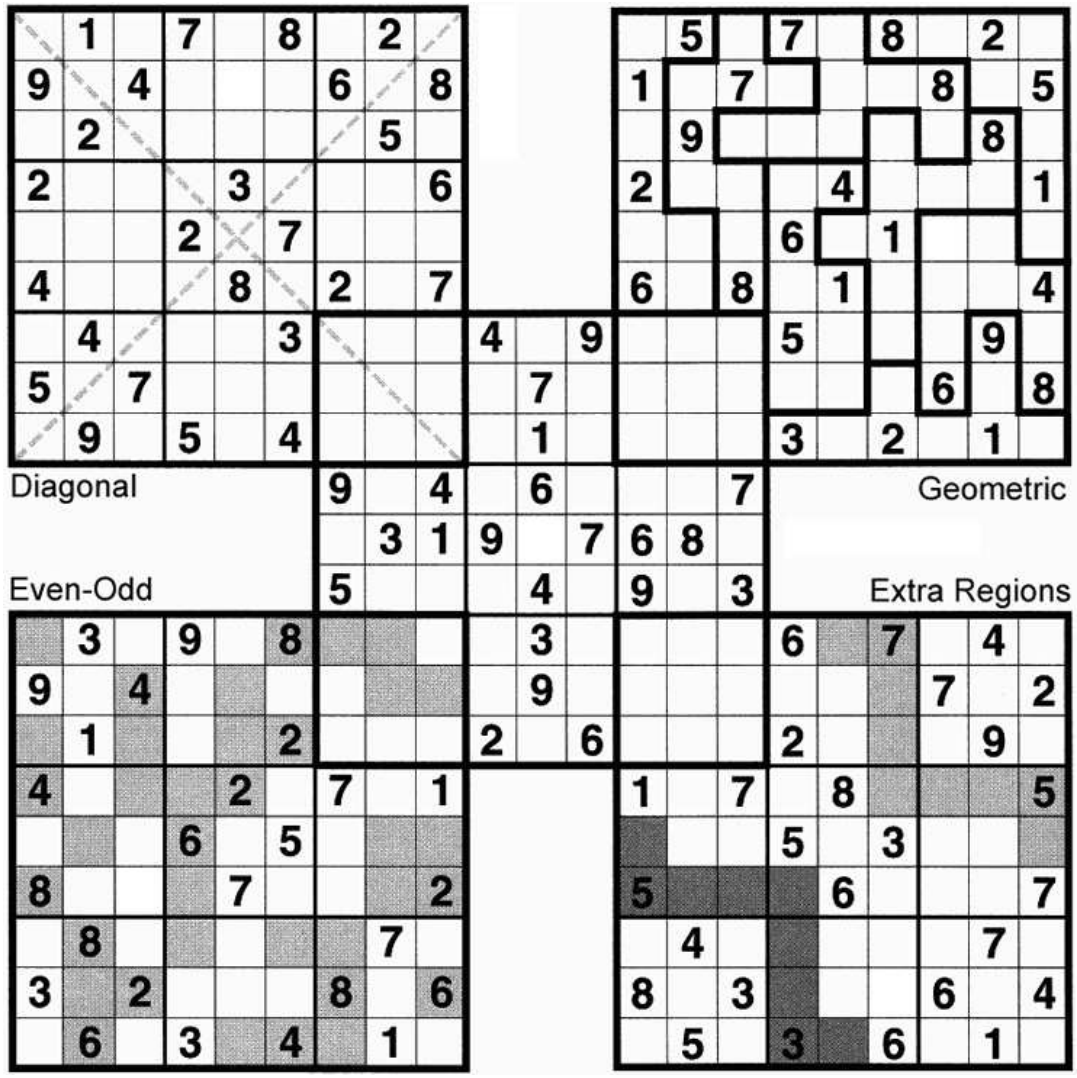
Sudoku variants

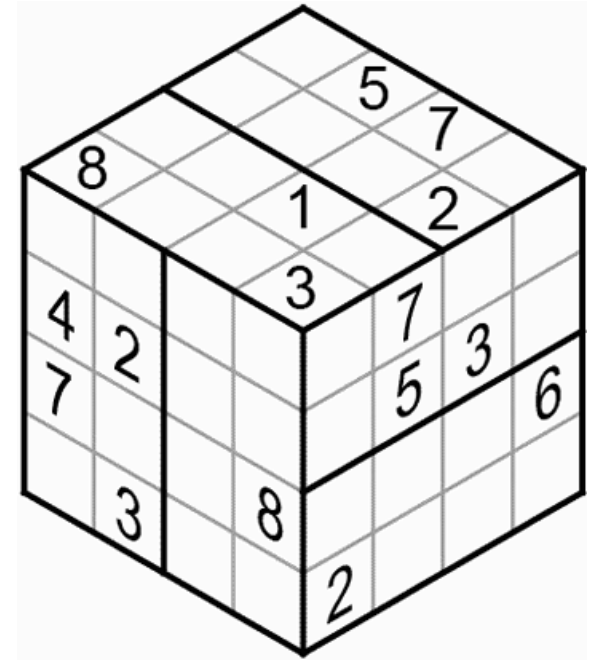
- larger Sudokus



Sudoku variants

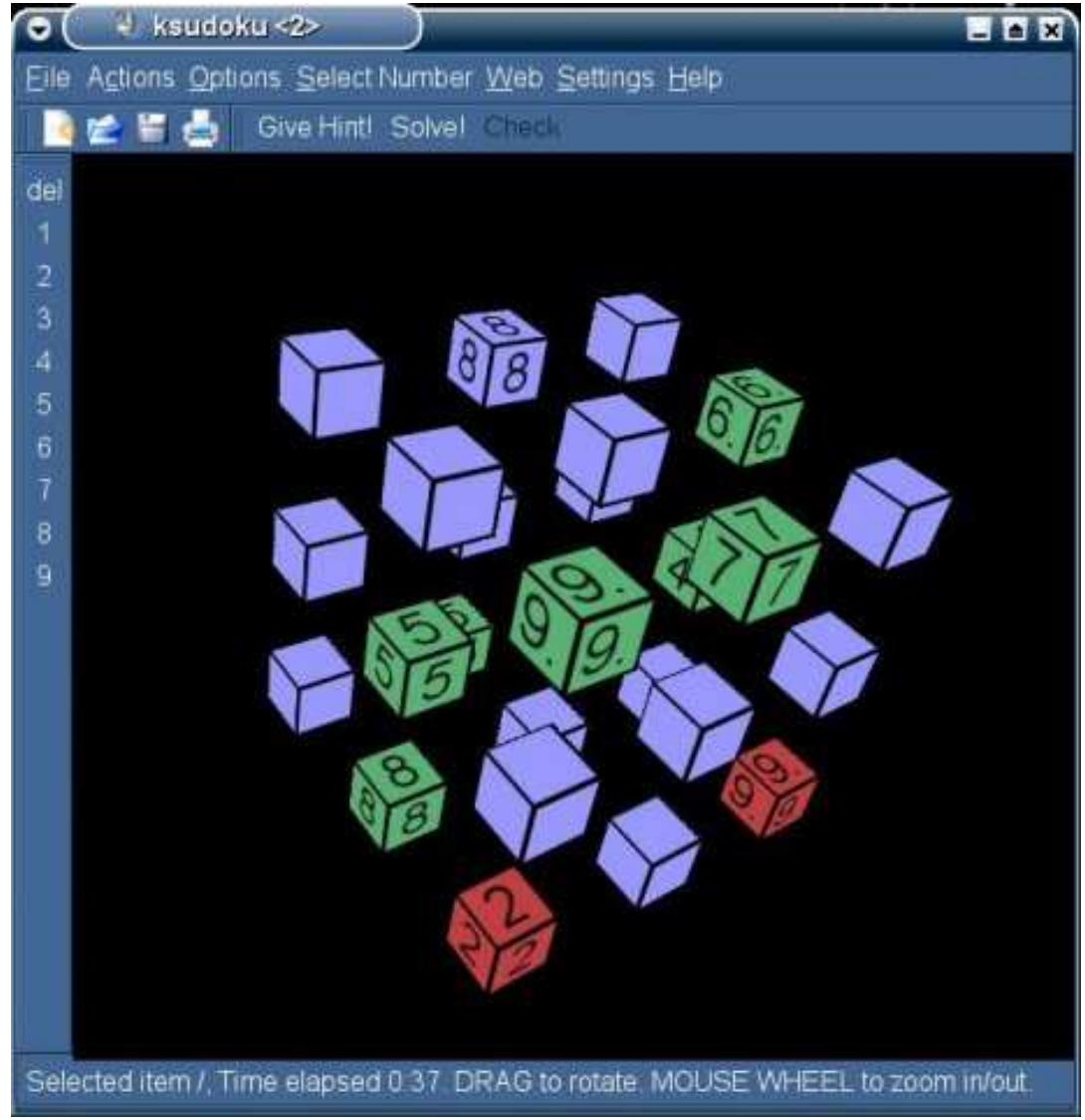
- other Sudokus variants





Sudoku variants

- three-dimensional Sudoku



Sudoku factor graph

- 81 fields $\widehat{=}$ variables x_i

Sudoku factor graph

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

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

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

Sudoku factor graph

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

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

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

- deduce Sudoku solution from marginals, assuming unique solution

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

Sudoku factor graph

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

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

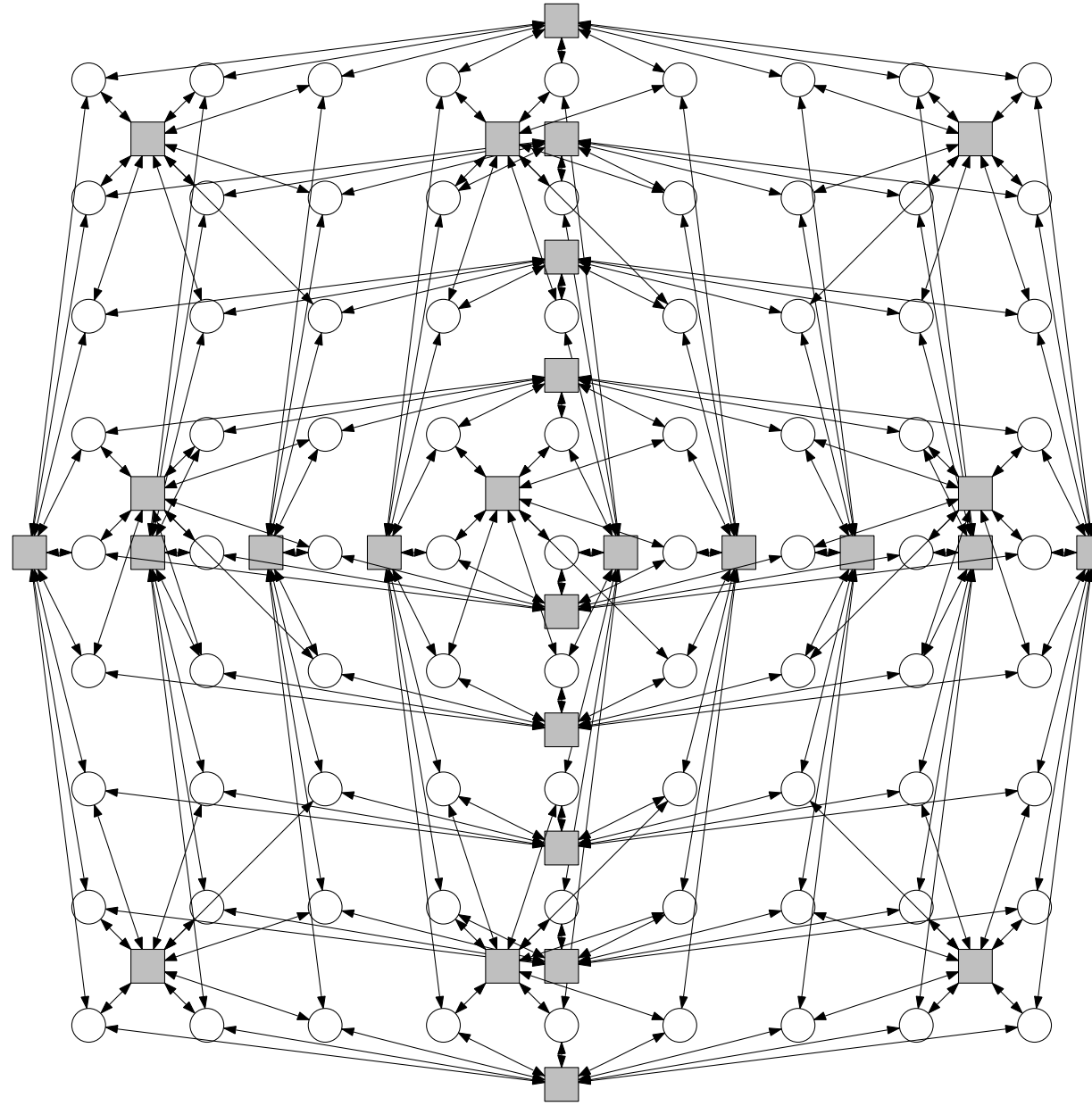
$$f(y_1, y_2, \ldots, y_9) = \mathbb{1}\left[(y_1, y_2, \ldots, y_9) \text{ is a perm. of } (1, 2, \ldots, 9)\right]$$

- deduce Sudoku solution from marginals, assuming unique solution

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

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

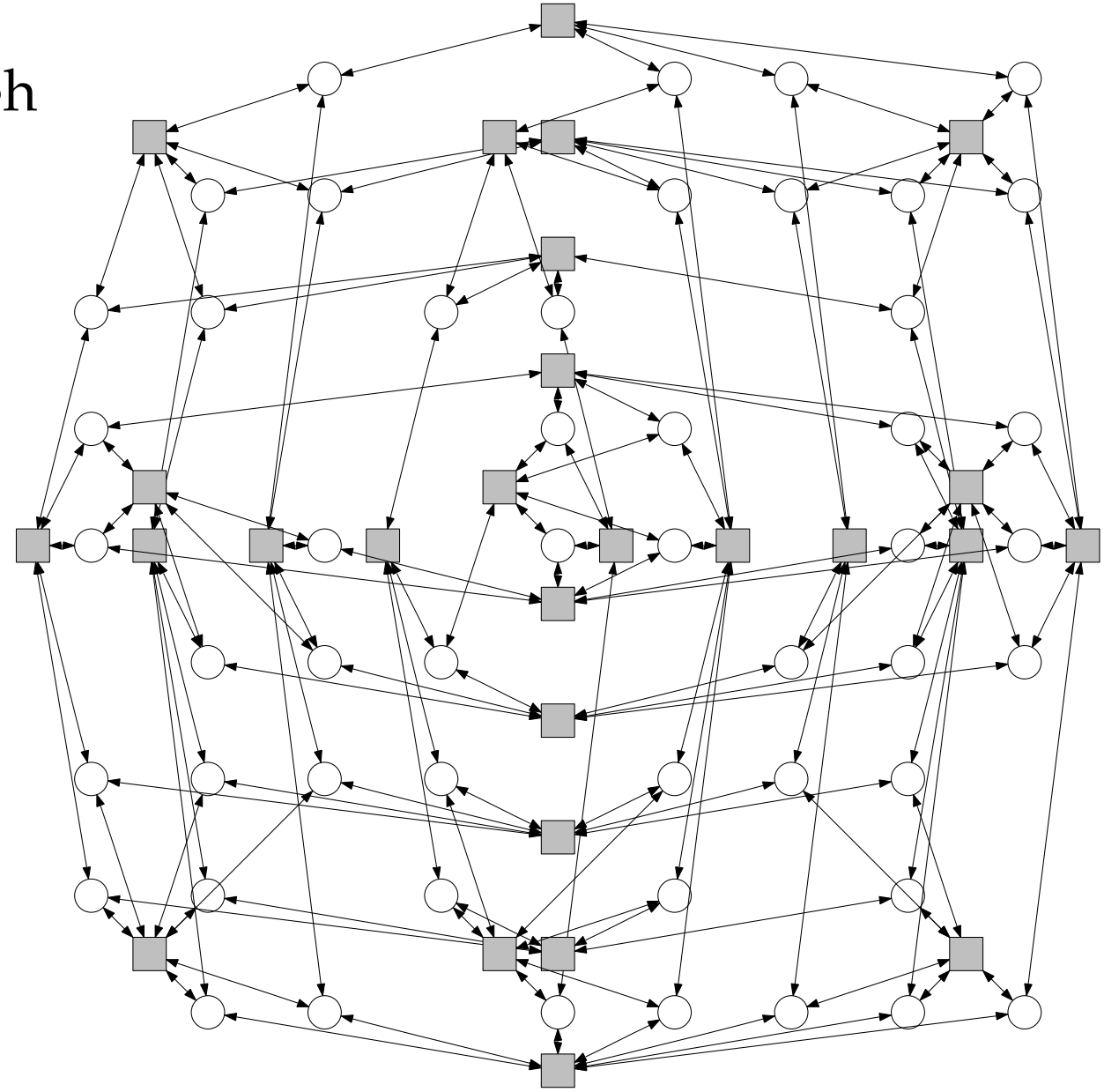
Sudoku factor graph



Sudoku factor graph

- givens reduce effective factor graph

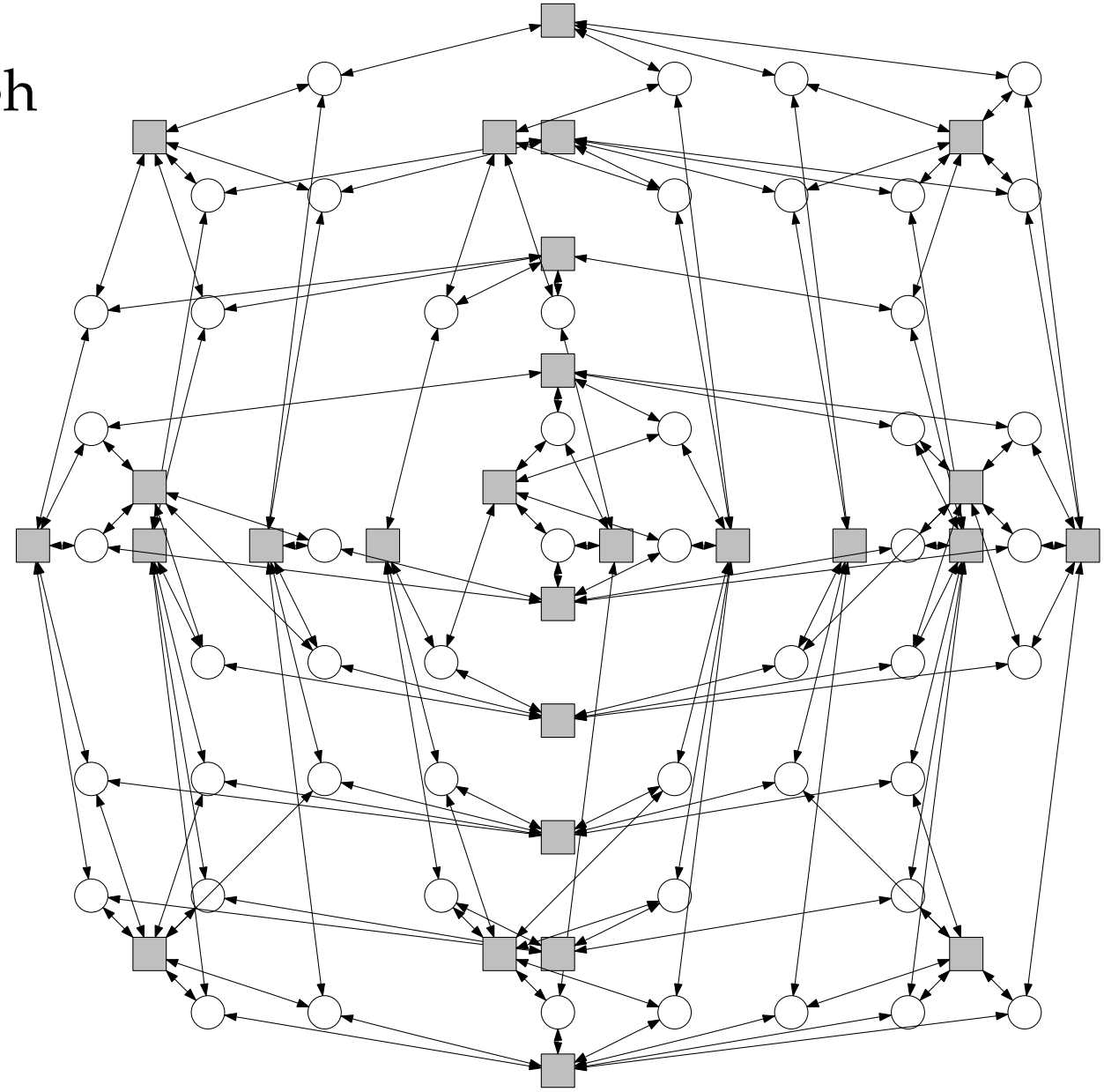
8	2		7	5			4	
9			8	6				
		1			3	8		7
	8	6	1			2		
	7		5			1		
1				2	9			
				1				8
		8		4		5		3
4			6					



Sudoku factor graph

- givens reduce effective factor graph
- but still very loopy

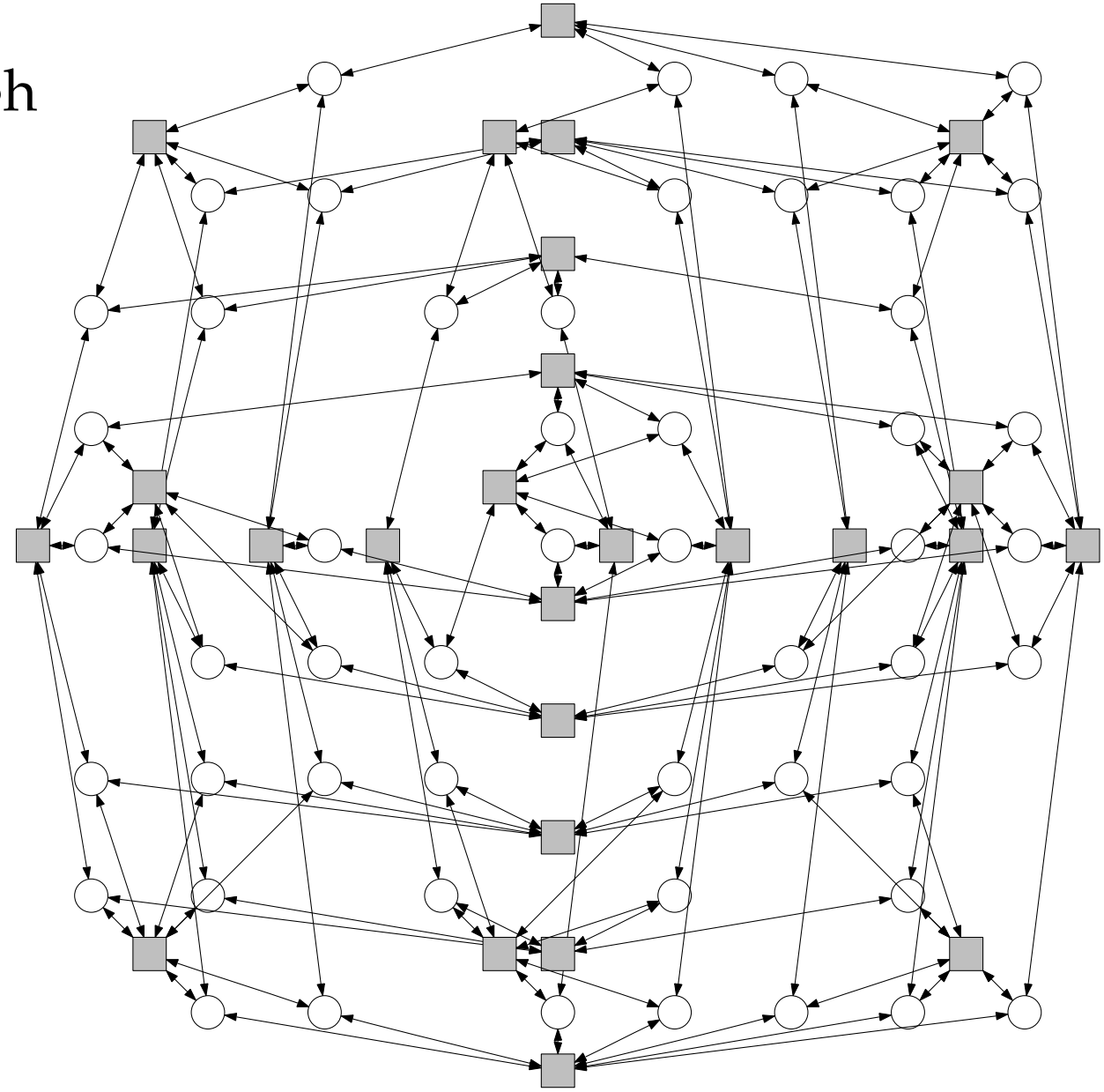
8	2		7	5			4	
9			8	6				
		1			3	8		7
	8	6	1			2		
	7		5			1		
1				2	9			
				1				8
		8		4		5		3
4			6					



Sudoku factor graph

- givens reduce effective factor graph
- but 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)

- messages $\mu_{i\to k}(x)$: normalised probability distributions

Message passing (sum product algorithm)

- messages $\mu_{i\to k}(x)$: normalised 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 …”

Message passing (sum product algorithm)

- messages $\mu_{i\to k}(x)$: normalised 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:

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

Message passing (sum product algorithm)

- messages $\mu_{i\to k}(x)$: normalised 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:

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

- function node processing: marginalisation with respect to x

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

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

Message initialisation / update / marginalisation

- Message initialisation

Message initialisation / update / marginalisation

- Message initialisation
  - If neither i nor k a given variable node:

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

Message initialisation / update / marginalisation

- Message initialisation
  - If neither i nor k a given variable node:

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

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

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

Message initialisation / update / marginalisation

- Message initialisation
  - If neither i nor k a given variable node:

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

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

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

- Update policies:

Message initialisation / update / marginalisation

- Message initialisation
  - If neither i nor k a given variable node:

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

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

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

- Update policies:
  - parallel sequential
    (update either *all* messages from function nodes to variable nodes *or* from variable nodes to function nodes)

Message initialisation / update / marginalisation

- Message initialisation
  - If neither i nor k a given variable node:

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

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

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

- Update policies:
  - 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)

Message initialisation / update / marginalisation

- Message initialisation
  - If neither i nor k a given variable node:
    $$\mu_{i\to k}(x) = \tfrac{1}{9}\,, \qquad \text{for } \forall x$$
  - If i or k is a given variable node with value v_i:
    $$\mu_{i\to k}(x) = \mathbb{1}\left[x = v_i\right]$$
- Update policies:
  - 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)
- Marginalisation:
  $$\mu_i(x) \sim \prod_{j\in\{\text{n.n. of } i\}} \mu_{j\to i}(x)$$

Results for sum product algorithm

- Sudoku problems of low and average difficulty ($\gtrapprox$ 25 givens)

Results for sum product algorithm

- Sudoku problems of low and average difficulty ($\gtrapprox 25$ givens)
  - message passing solves Sudoku almost always,

    marginals converge to $\mu_i(x) = (0, \ldots, 1, \ldots, 0)$

Results for sum product algorithm

- Sudoku problems of low and average difficulty ($\gtrapprox 25$ givens)
  - message passing solves Sudoku almost always,

    marginals converge to $\mu_i(x) = (0, \ldots, 1, \ldots, 0)$
- Sudoku problems of maximal difficulty (only 17 givens)

Results for sum product algorithm

- Sudoku problems of low and average difficulty ($\gtrapprox 25$ givens)
  - message passing solves Sudoku almost always,
    marginals converge to $\mu_i(x) = (0, \ldots, 1, \ldots, 0)$
- Sudoku problems of maximal difficulty (only 17 givens)
  - message passing solves Sudoku often,
    55 % parallel sequential update, 63 % random update

Results for sum product algorithm

- Sudoku problems of low and average difficulty ($\gtrapprox$ 25 givens)
  - message passing solves Sudoku almost always,

    marginals converge to $\mu_i(x) = (0, \ldots, 1, \ldots, 0)$
- 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

Results for sum product algorithm

- Sudoku problems of low and average difficulty ($\gtrapprox 25$ givens)
  - message passing solves Sudoku almost always,

    marginals converge to $\mu_i(x) = (0, \ldots, 1, \ldots, 0)$
- 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".

Results for sum product algorithm

- Sudoku problems of low and average difficulty ($\gtrapprox 25$ givens)
  - message passing solves Sudoku almost always,

    marginals converge to $\mu_i(x) = (0, \ldots, 1, \ldots, 0)$
- 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

Results for sum product algorithm

- Sudoku problems of low and average difficulty ($\gtrapprox 25$ givens)
  - message passing solves Sudoku almost always,

    marginals converge to $\mu_i(x) = (0, \ldots, 1, \ldots, 0)$
- 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

Results for sum product algorithm

- Sudoku problems of low and average difficulty ($\gtrapprox$ 25 givens)
  - message passing solves Sudoku almost always,

    marginals converge to $\mu_i(x) = (0, \ldots, 1, \ldots, 0)$
- 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 (max product algorithm)

- try to avoid numerical instabilities and rounding errors
- message passing works on every commutative semiring $\mathbb{K}$

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
  - a set of elements with a "$+$" with a "0" and a "$\cdot$" with a "1"
  - commutative and associative laws for "$+$" and "$\cdot$"
  - distributive law

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
  - a 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 $\mathbb{K} = (\{0,1\}, (\max, 0), (\cdot, 1))$

Message passing (max product algorithm)

- max product algorithm over $\mathbb{K} = (\{0,1\}, (\max, 0), (\cdot, 1))$
- messages $\mu_{i\to k}(x)$: 9-tuples of 0 and 1 indicating possible field assignments, e. g. $(1,0,0,0,0,1,0,1,0)$

Message passing (max product algorithm)

- max product algorithm over $\mathbb{K} = (\{0,1\}, (\max, 0), (\cdot, 1))$
- messages $\mu_{i\to k}(x)$: 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 ..."

Message passing (max product algorithm)

- max product algorithm over $\mathbb{K} = (\{0,1\}, (\max, 0), (\cdot, 1))$
- messages $\mu_{i\to k}(x)$: 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 like sum product algorithm

Message passing (max product algorithm)

- max product algorithm over $\mathbb{K} = (\{0,1\}, (\max, 0), (\cdot, 1))$
- messages $\mu_{i\to k}(x)$: 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 like sum product algorithm
- no numerical instabilities, outperforms sum product algorithm slightly (solves 71 % of Sudokus with 17 givens)

Open problems

- What distinguishes Sudokus for these message passing converges from those, for which message passing does not converge?

Open problems

- What distinguishes Sudokus for these message passing converges from those, for which message passing does not converge?
- Get a better understanding of numerical instabilities of sum product algorithm.

Open problems

- What distinguishes Sudokus for these message passing converges from those, for which message passing does not converge?
- Get a better understanding of numerical instabilities of sum product algorithm.
- Combine max product algorithm with sum product algorithm.

Open problems

- What distinguishes Sudokus for these message passing converges from those, for which message passing does not converge?
- Get a better understanding of numerical instabilities of sum product algorithm.
- Combine max product algorithm with sum product algorithm.
- Combine max product algorithm with back tracking.

Open problems

- What distinguishes Sudokus for these message passing converges from those, for which message passing does not converge?
- Get a better understanding of numerical instabilities of sum product algorithm.
- Combine max product algorithm with sum product algorithm.
- Combine max product algorithm with back tracking.
- What about larger Sudoku variants?

Open problems

- What distinguishes Sudokus for these message passing converges from those, for which message passing does not converge?
- Get a better understanding of numerical instabilities of sum product algorithm.
- Combine max product algorithm with sum product algorithm.
- Combine max product algorithm with back tracking.
- What about larger Sudoku variants?
- What about other message passing schemes, e. g. formulations with binary variables?