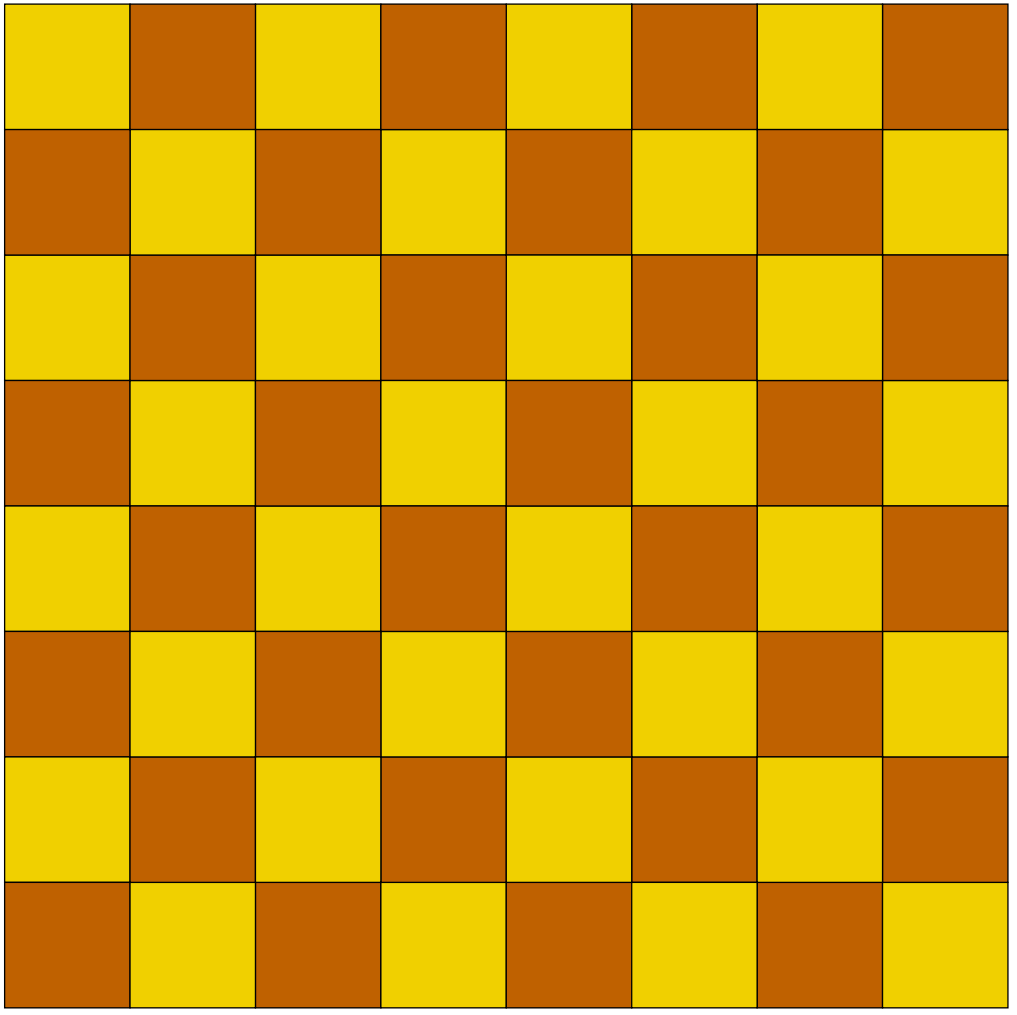
On counting arrays with certain properties

Pascal O. Vontobel

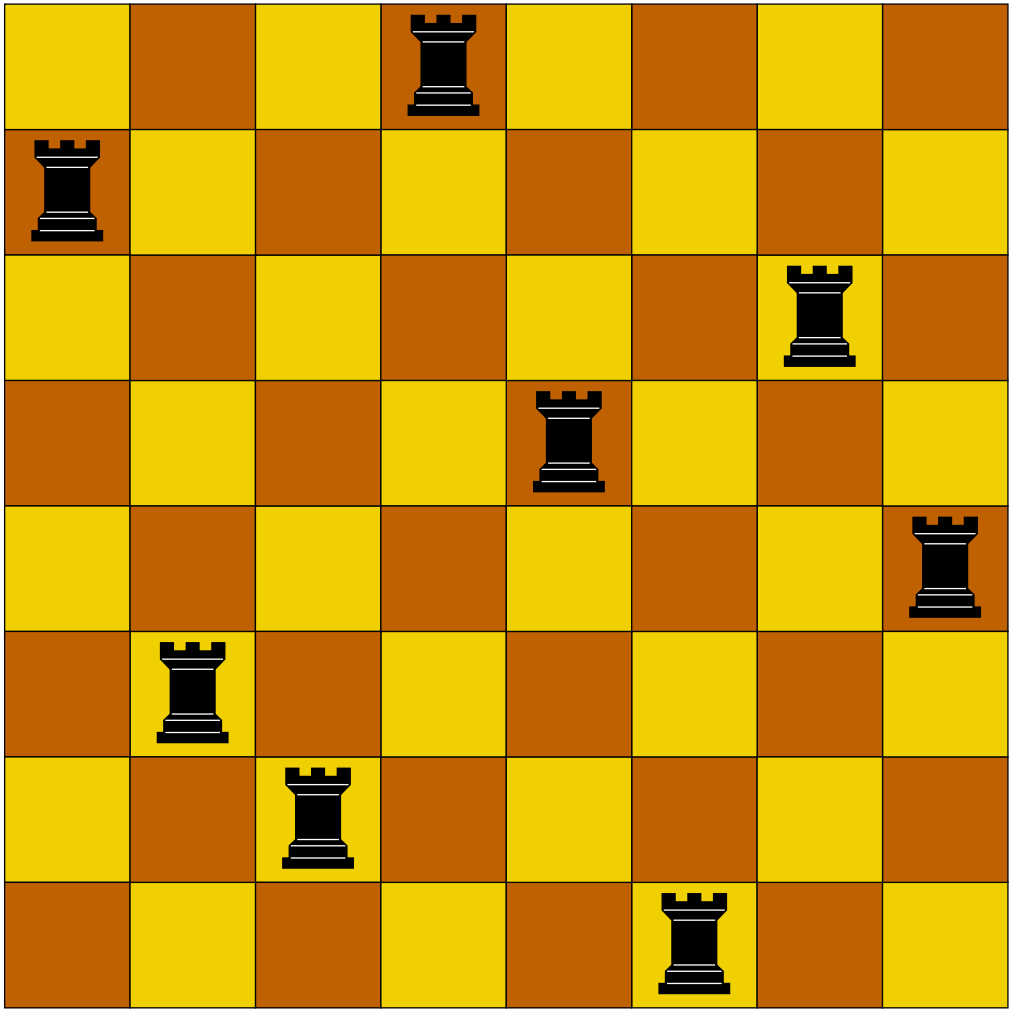
Department of Information Engineering
The Chinese University of Hong Kong

ACCESS Seminar **April 1, 2025**

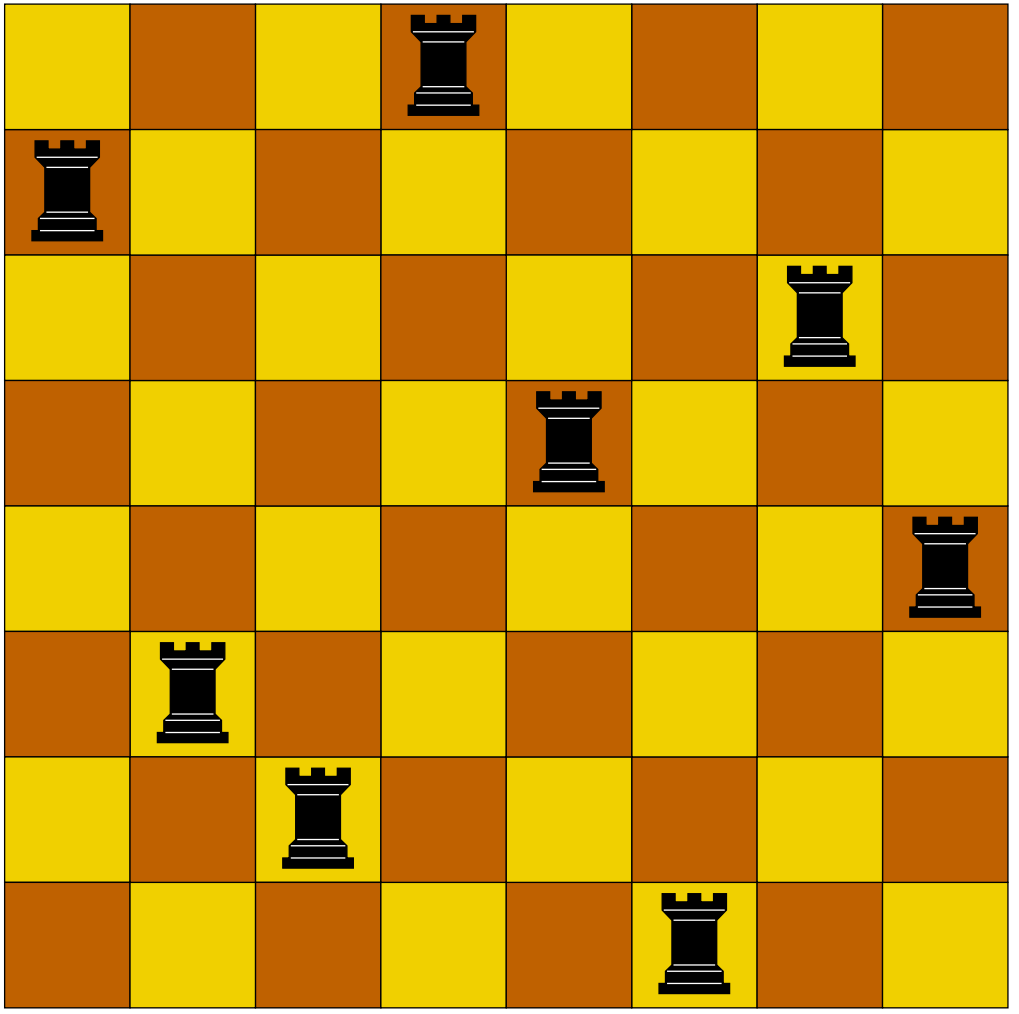
Chess Board



Chess Board

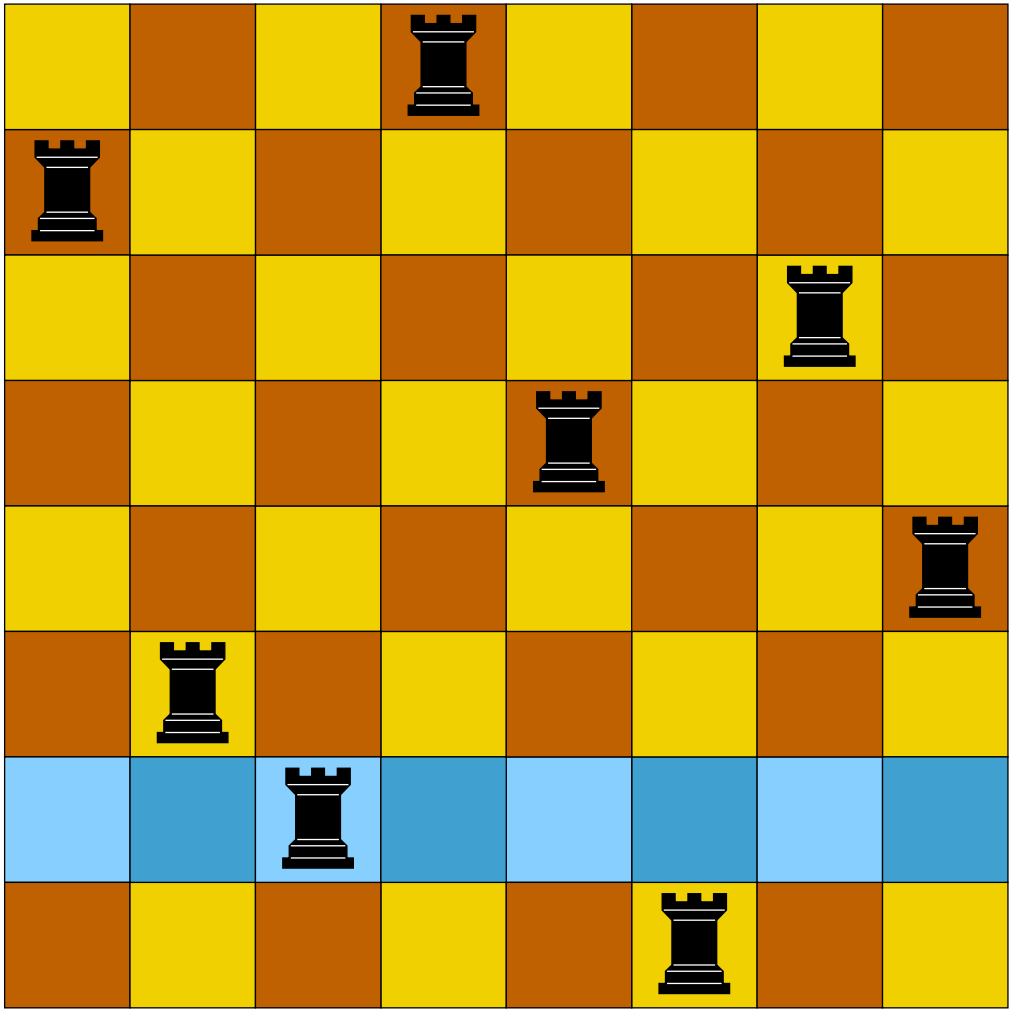


Chess Board



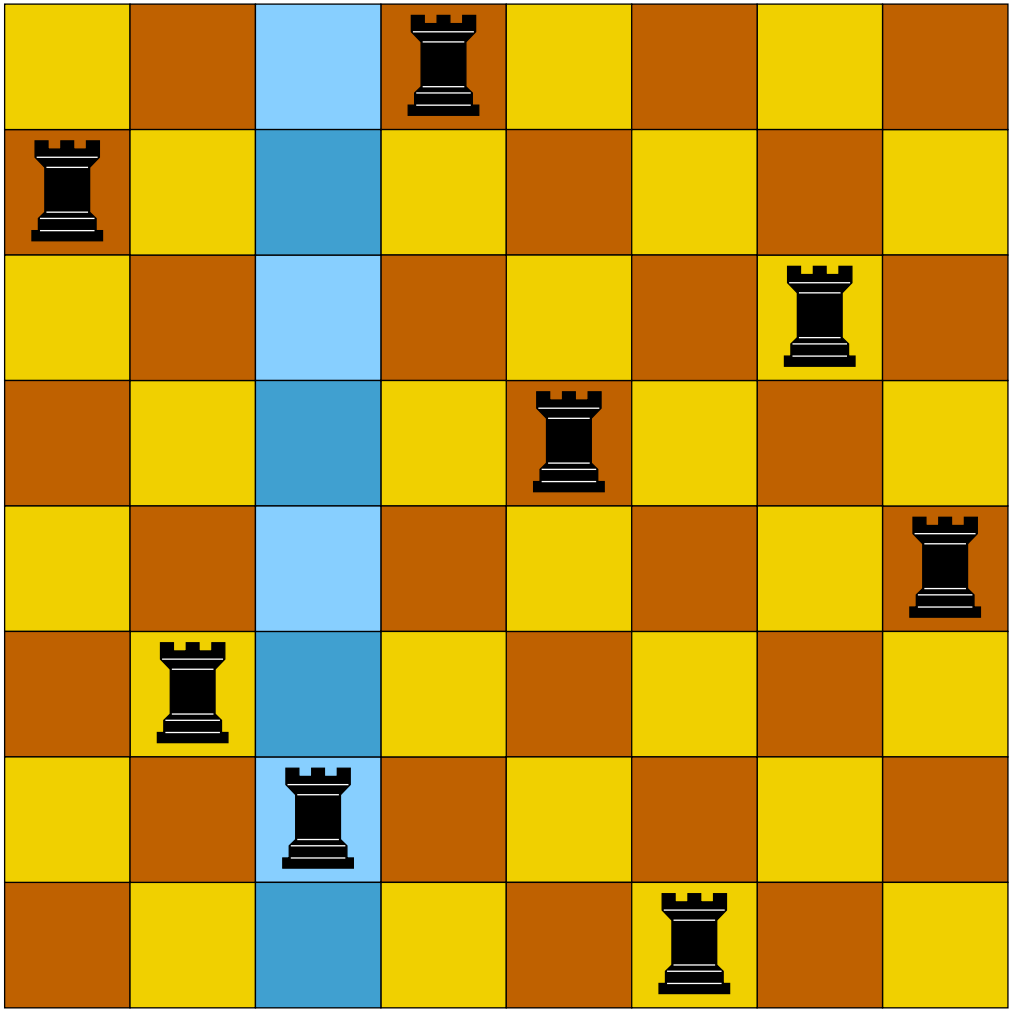
Question: in how many ways can we place 8 non-attacking rooks on a chess board?

Chess Board



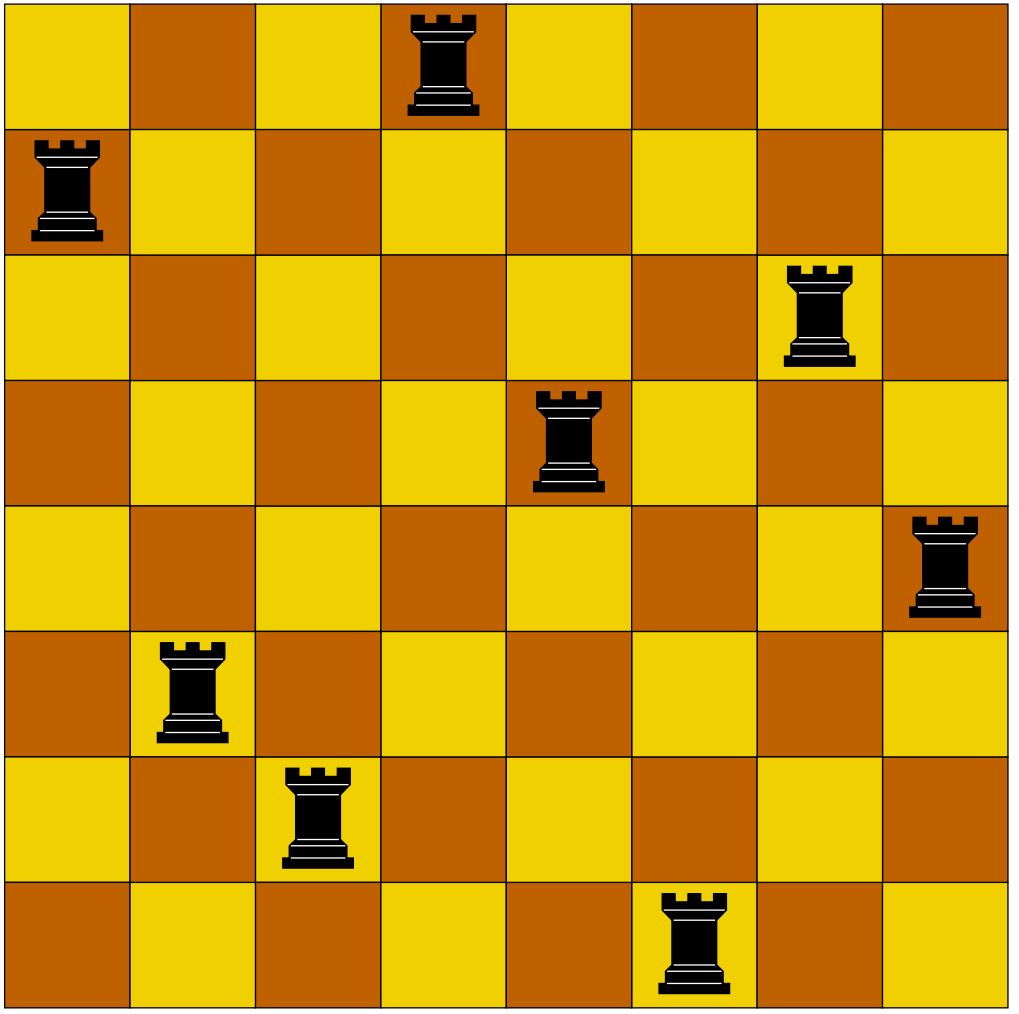
Row condition: exactly one rook per row.

Chess Board



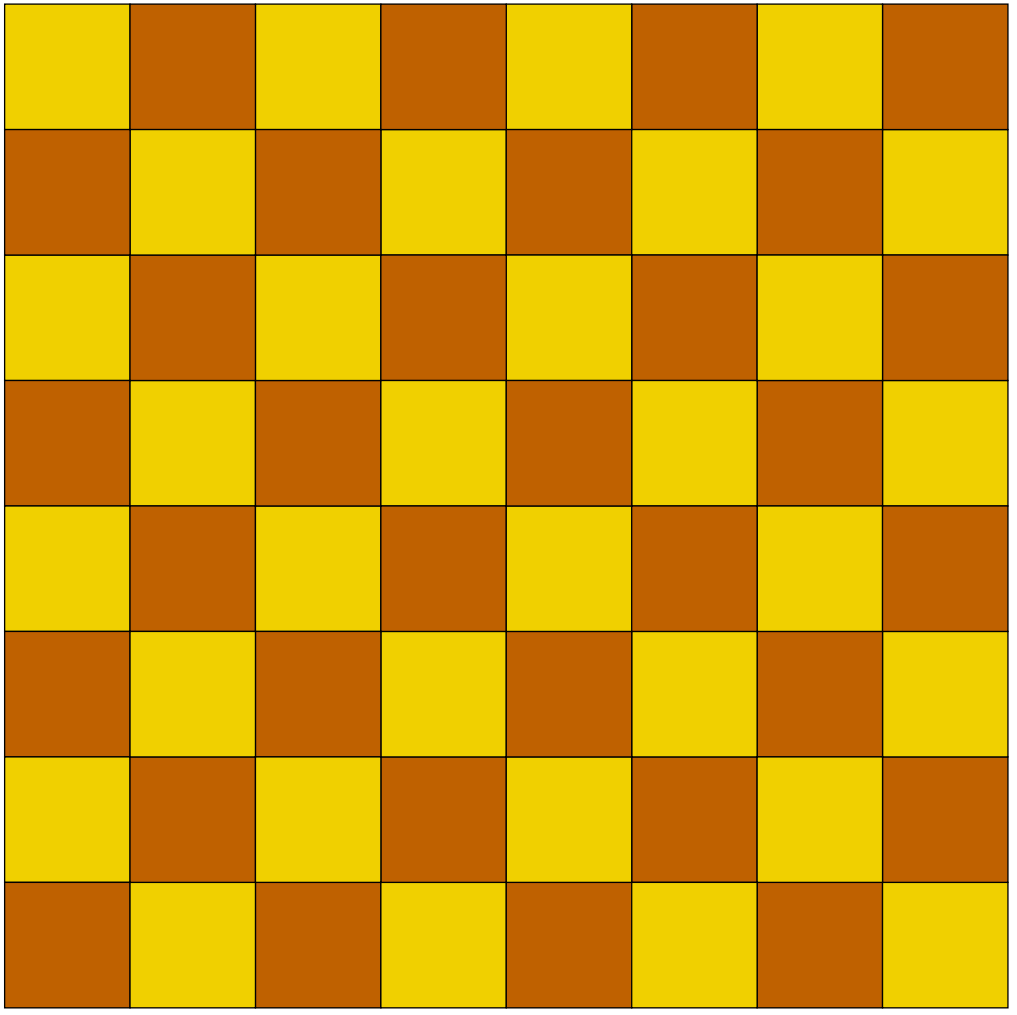
Column condition: exactly one rook per column.

Chess Board

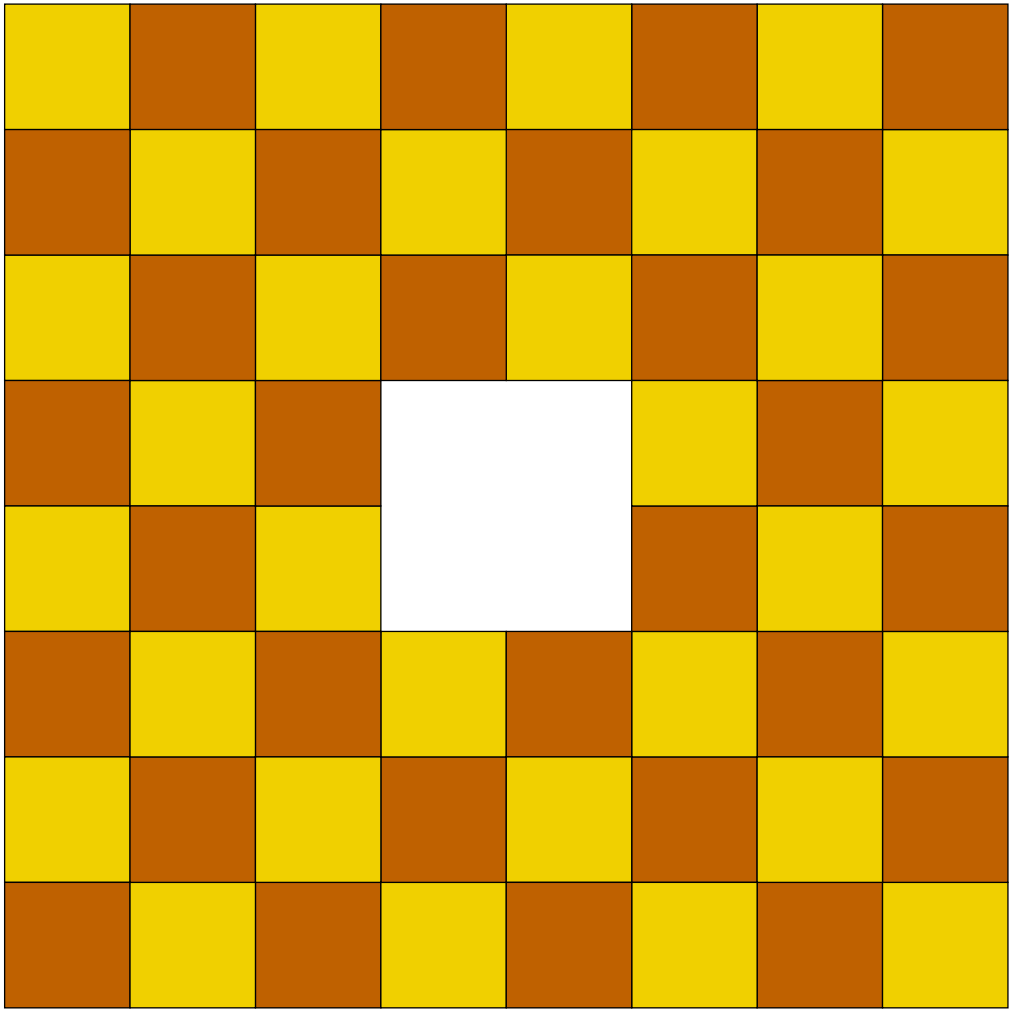


Question: in how many ways can we place 8 non-attacking rooks on a chess board?

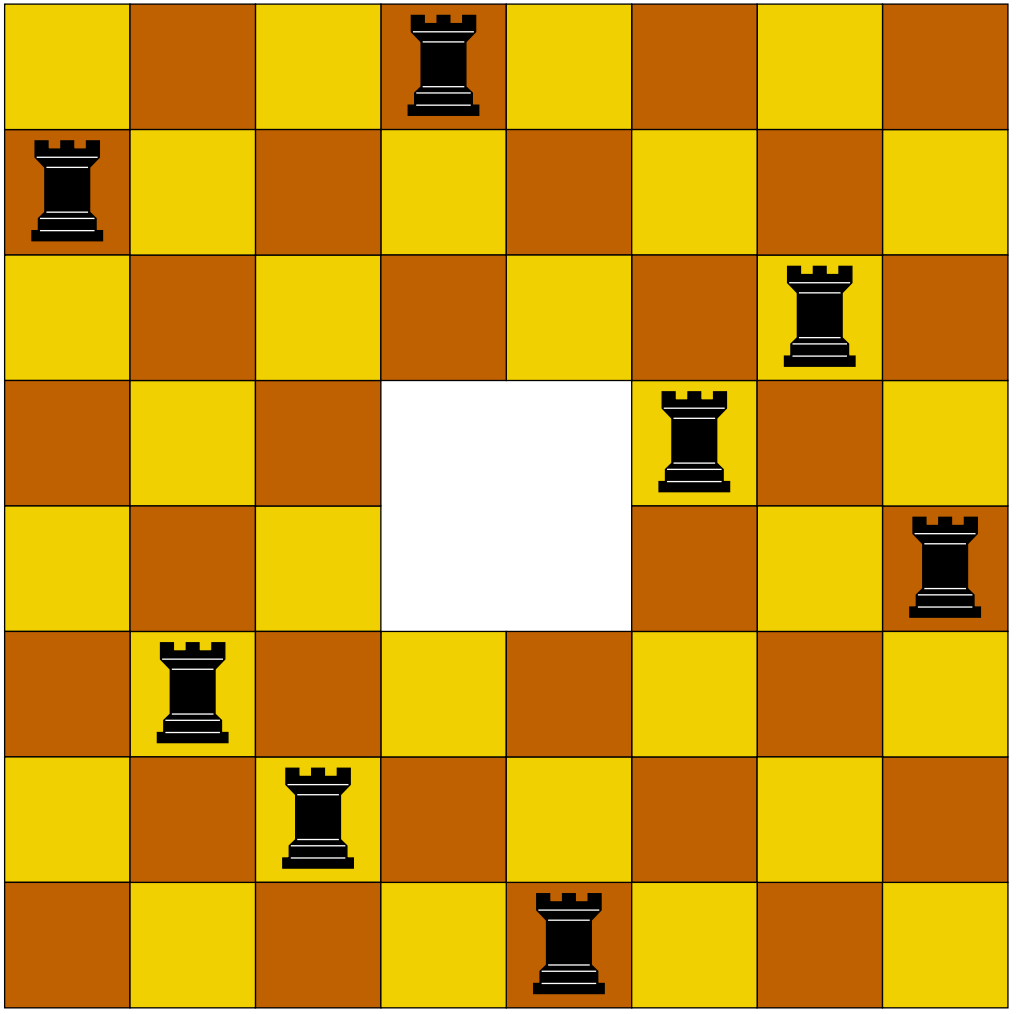
Chess Board



Chess Board

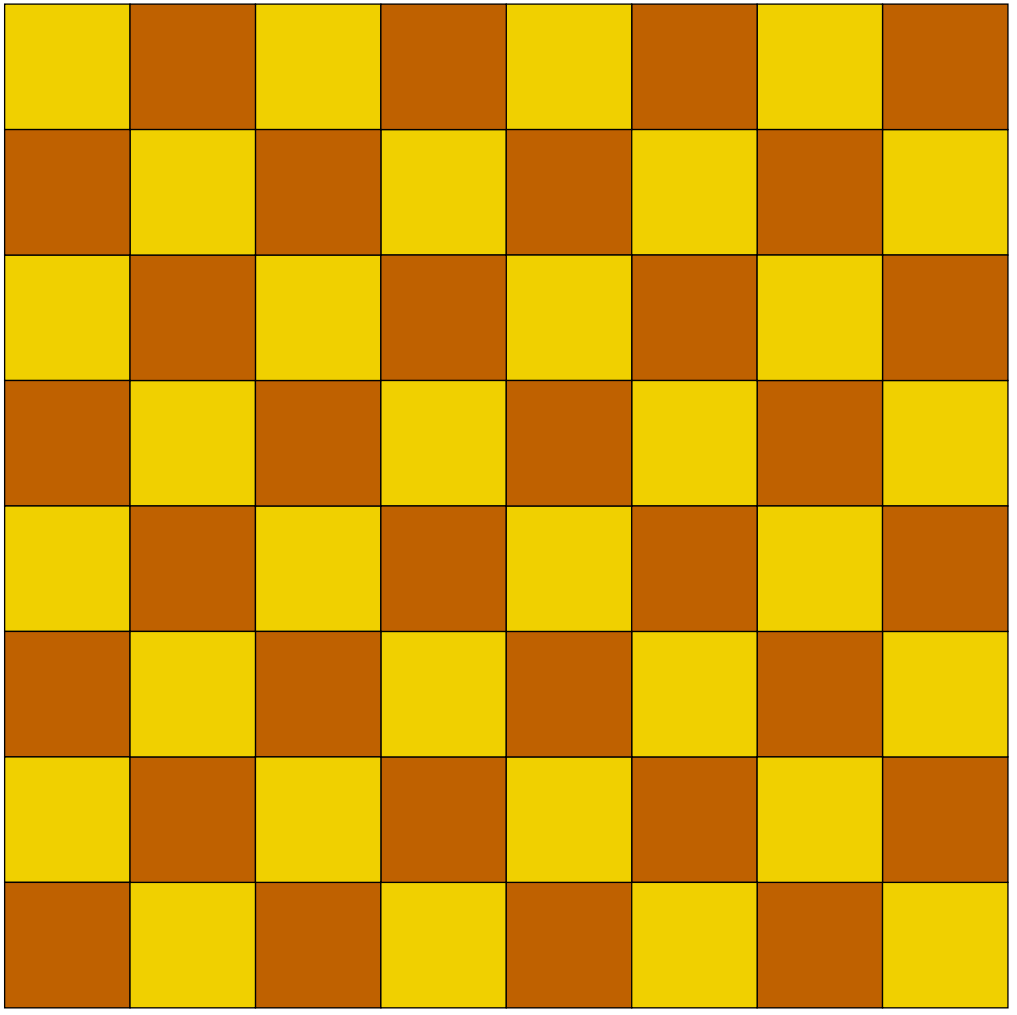


Chess Board

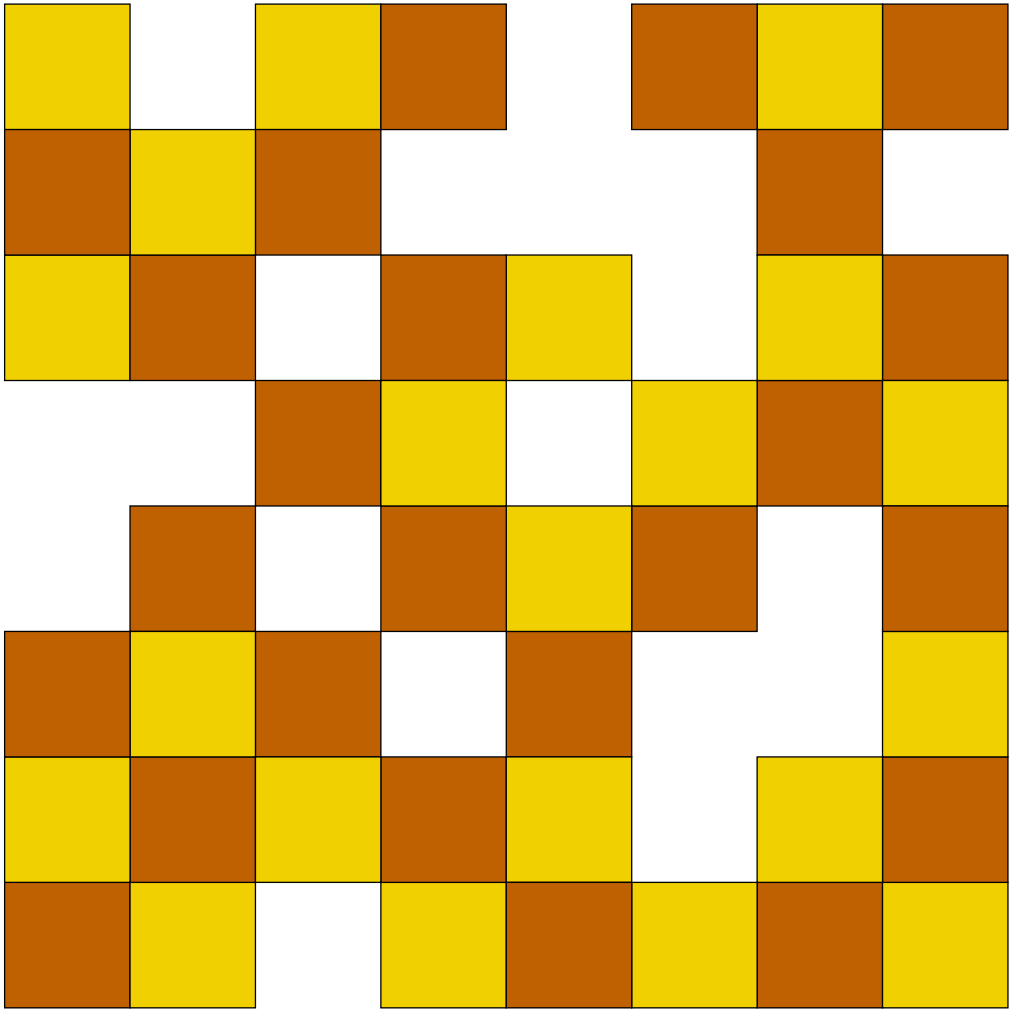


Question: in how many ways can we place 8 non-attacking rooks on this modified chess board?

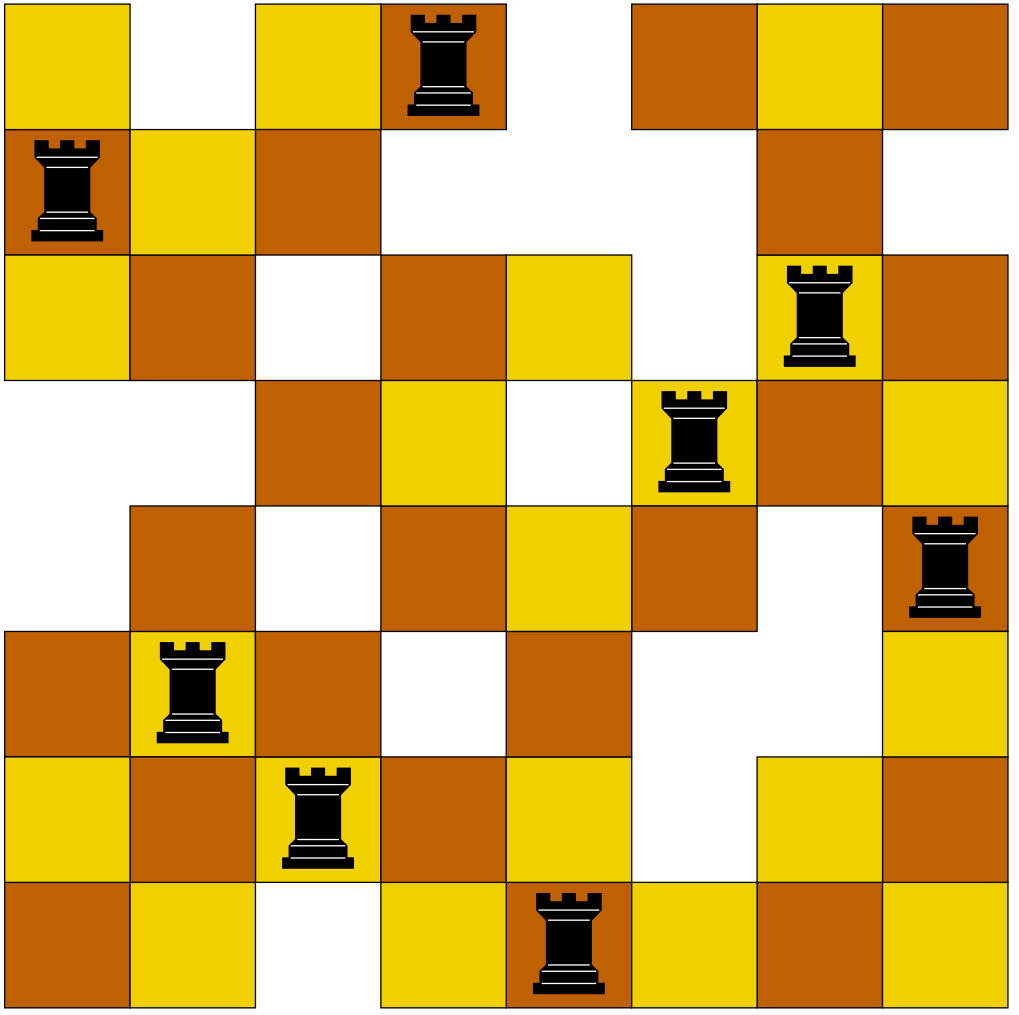
Chess Board



Chess Board

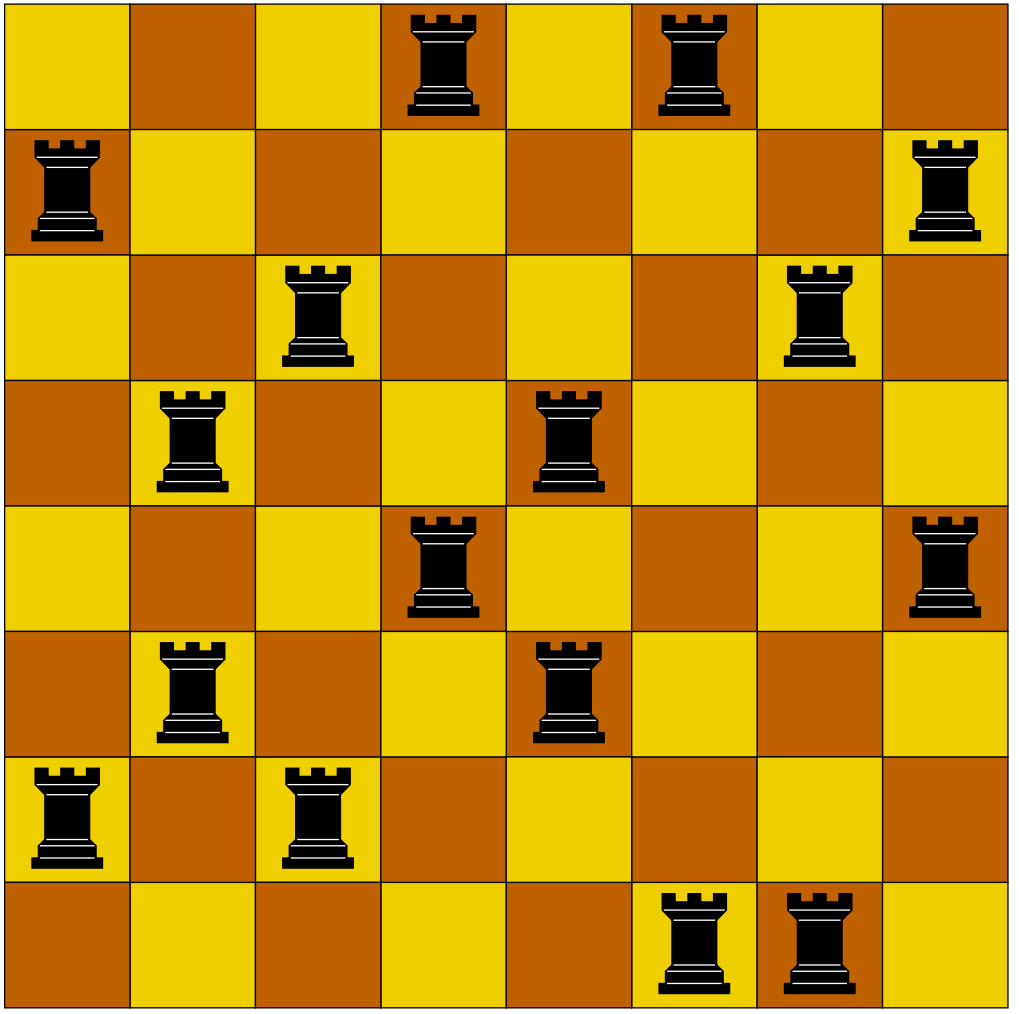


Chess Board



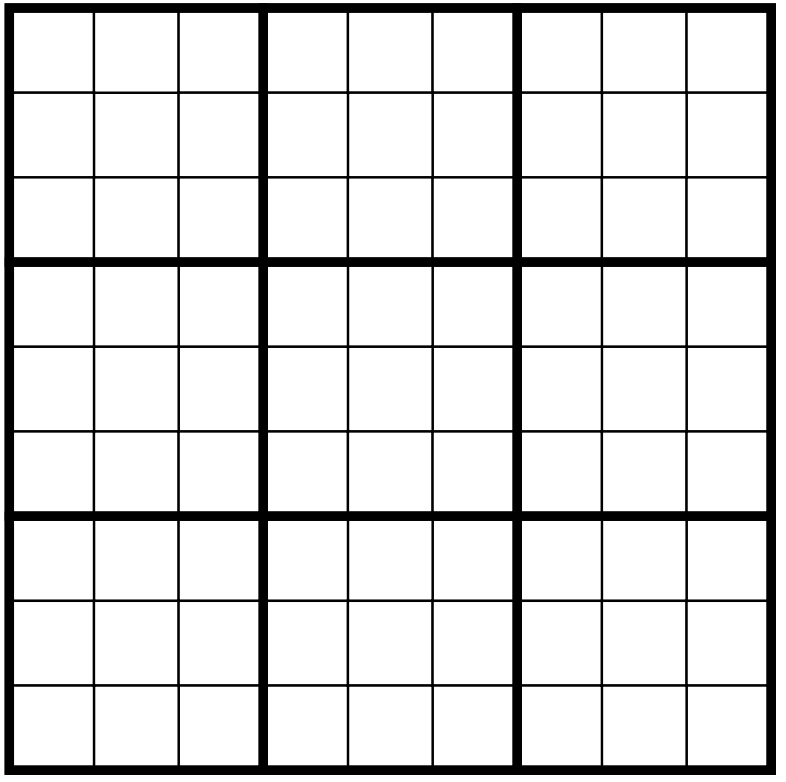
Question: in how many ways can we place 8 non-attacking rooks on this modified chess board?

Chess Board



Question: in how many ways can we place 16 rooks so that there are exactly two rooks per row and column on this chess board?

Sudoku



Sudoku

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

Sudoku

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

Question: how many Sudoku arrays are there?

(More technically: how many valid configurations are there?)

Sudoku

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

Row condition: numbers $1, \ldots, 9$ appear exactly once.

Sudoku

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

Column condition: numbers $1, \ldots, 9$ appear exactly once.

Sudoku

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

Sub-block condition: numbers $1, \ldots, 9$ appear exactly once.

Sudoku

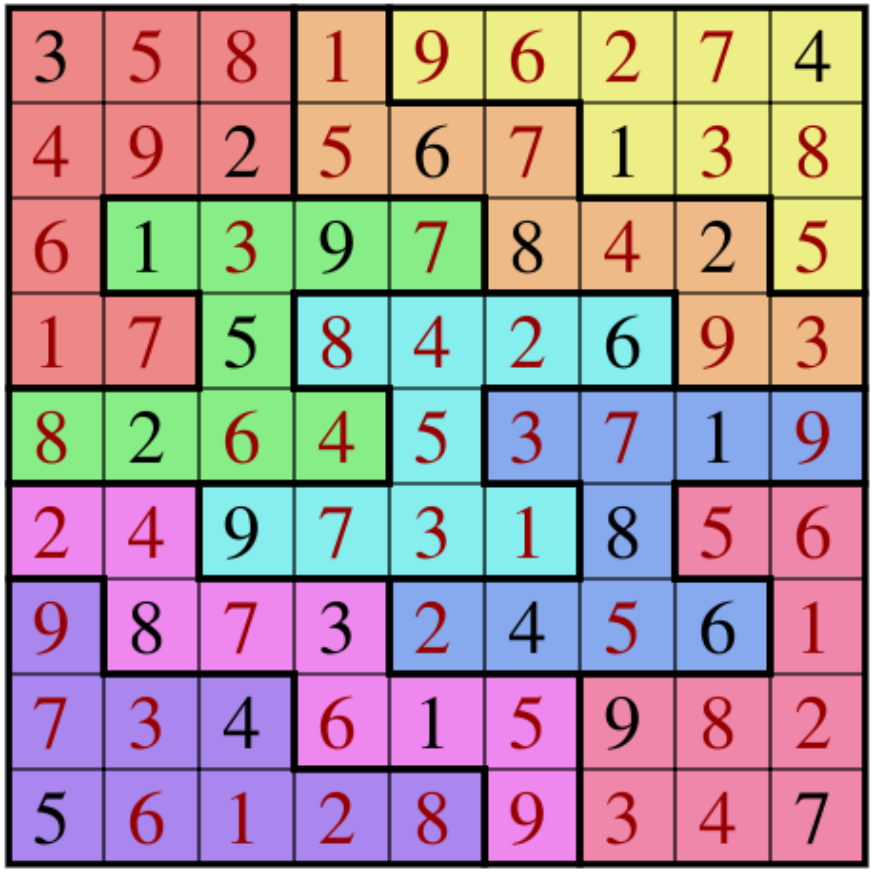
1	2	5	3	9	6	8	7	4
4	6	3	8	7	5	2	9	1
7	9	8	2	4	1	5	3	6
5	4	7	6	1	2	9	8	3
2	3	9	5	8	4	1	6	7
8	1	6	9	3	7	4	2	5
6	8	1	7	5	9	3	4	2
3	7	4	1	2	8	6	5	9
9	5	2	4	6	3	7	1	8

Question: how many Sudoku arrays are there?

(More technically: how many valid configurations are there?)

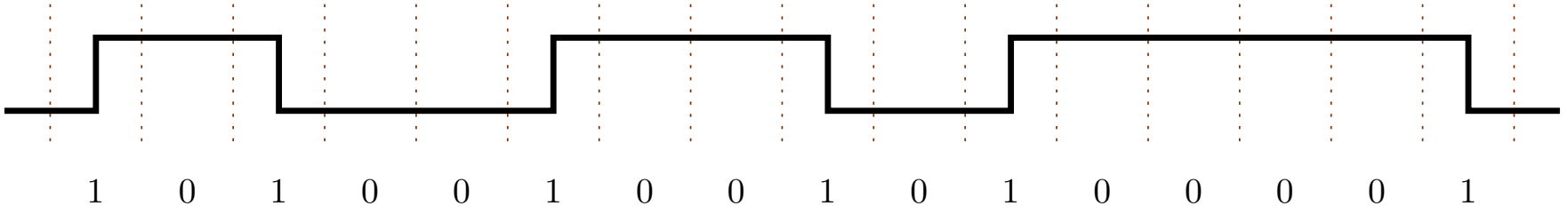
Other Sudoku Setups

Other Sudoku Setups



1D constraints in communications

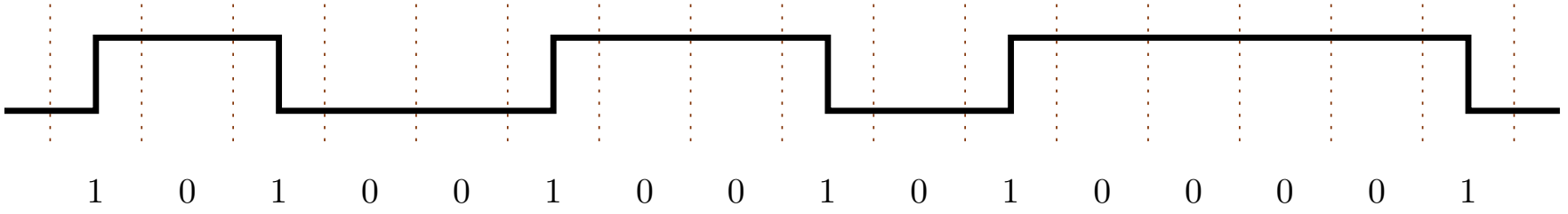
RLL Constraints



A (d, k) RLL constraint imposes:

- At least d zero symbols between two ones.
- At most k zero symbols between two ones.

RLL Constraints

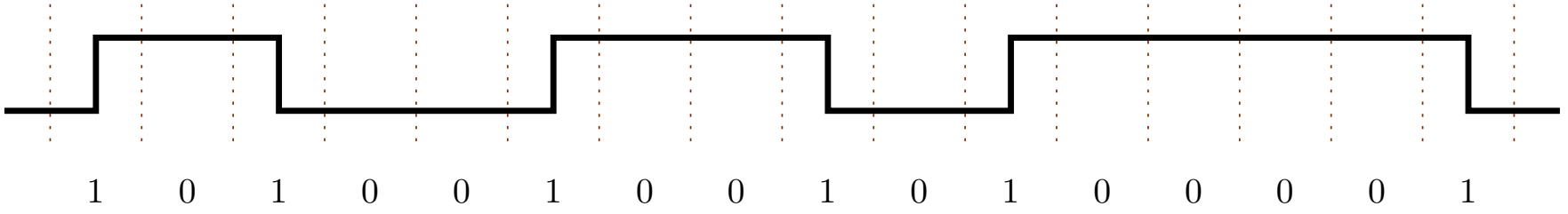


A (d, k) RLL constraint imposes:

- At least d zero symbols between two ones.
- At most k zero symbols between two ones.

Question: how many sequences of length T fulfill these constraints?

RLL Constraints



A (d, k) RLL constraint imposes:

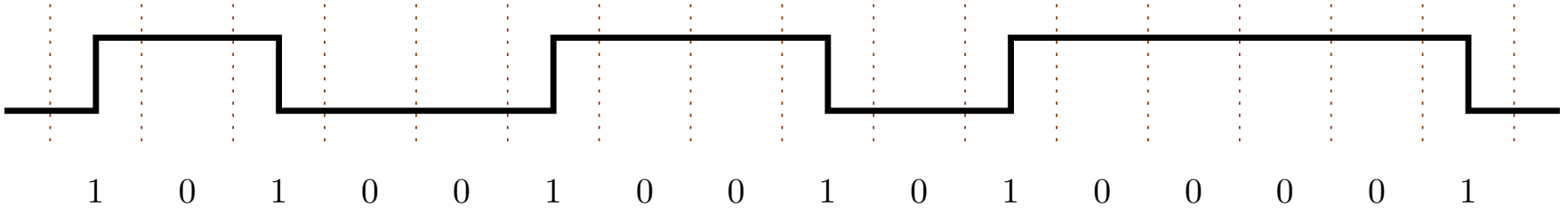
- At least d zero symbols between two ones.
- At most k zero symbols between two ones.

Question: how many sequences of length T fulfill these constraints?

Answer: typically, the answer to such questions looks like

$$N(T) = \exp\big(C \cdot T + o(T)\big).$$

RLL Constraints



A (d, k) RLL constraint imposes:

- At least d zero symbols between two ones.
- At most k zero symbols between two ones.

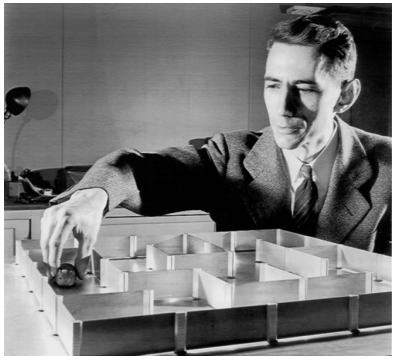
Question: how many sequences of length T fulfill these constraints?

Answer: typically, the answer to such questions looks like

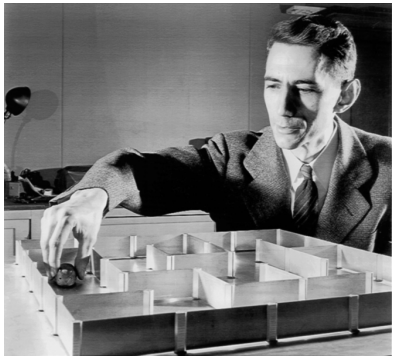
$$N(T) = \exp\big(C \cdot T + o(T)\big).$$

C: "capacity" or "entropy."

Shannon (1948), Figure 2



Shannon (1948), Figure 2



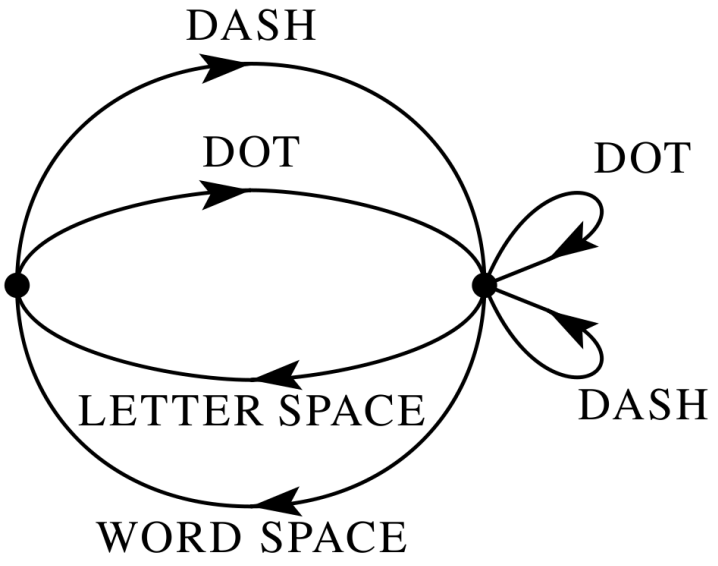
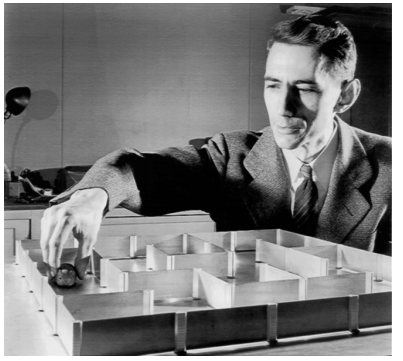


Fig. 2—Graphical representation of the constraints on telegraph symbols.

Shannon (1948)

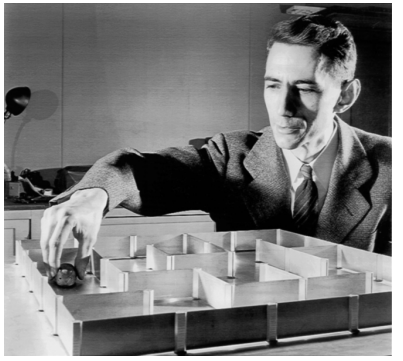


Definition: The capacity C of a discrete channel is given by

$$C = \operatorname*{Lim}_{T\to\infty} \frac{\log N(T)}{T}$$

where $N(T)$ is the number of allowed signals of duration T.

Shannon (1948)



Definition: The capacity C of a discrete channel is given by

$$C = \operatorname*{Lim}_{T\to\infty} \frac{\log N(T)}{T}$$

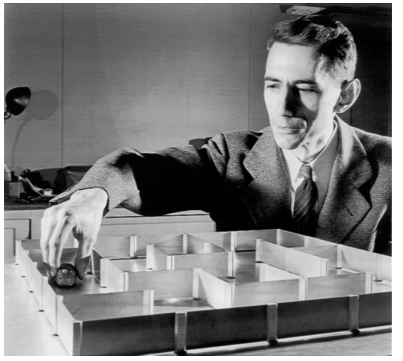
where $N(T)$ is the number of allowed signals of duration T.

Theorem 1: Let $b_{ij}^{(s)}$ be the duration of the s^{th} symbol which is allowable in state i and leads to state j. Then the channel capacity C is equal to $\log W$ where W is the largest real root of the determinant equation:

$$\left|\sum_s W^{-b_{ij}^{(s)}} - \delta_{ij}\right| = 0$$

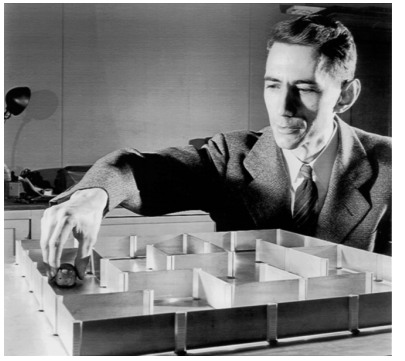
where $\delta_{ij} = 1$ if $i = j$ and is zero otherwise.

Shannon (1948)



$$C = \lim_{T \to \infty} \frac{\log N(T)}{T}$$

Shannon (1948)



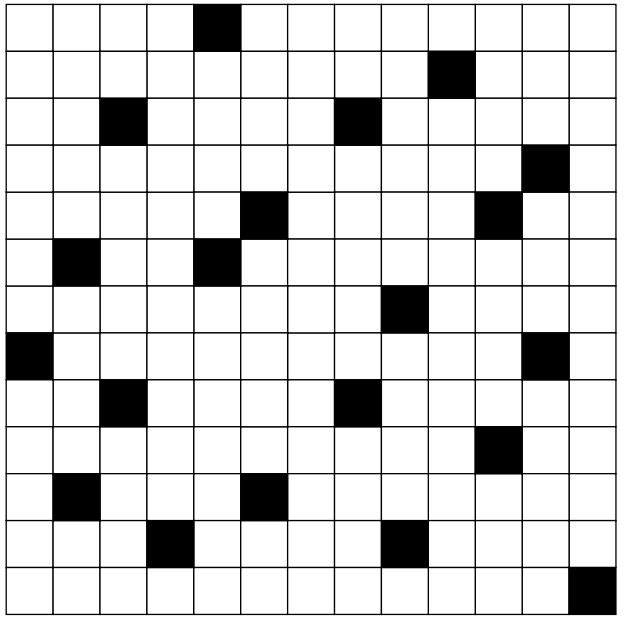
$$C = \lim_{T \to \infty} \frac{\log N(T)}{T}$$

i.e.,

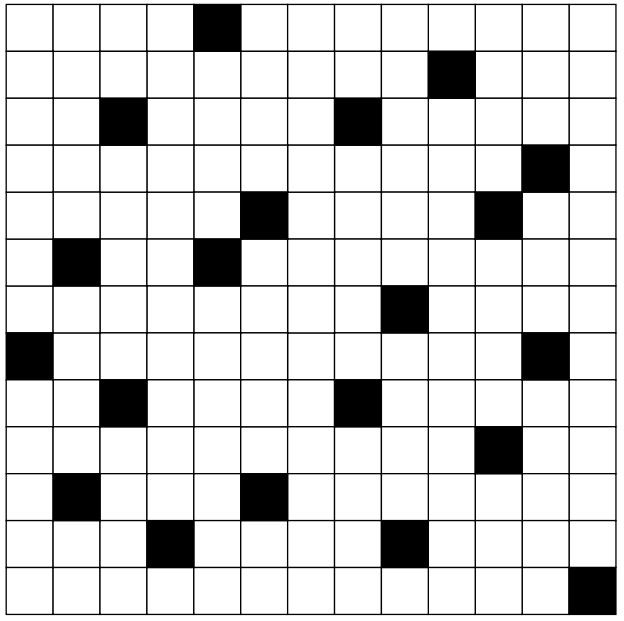
$$N(T) = 2^{C \cdot T + o(T)}$$

2D constraints in communications

Two-Dimensional RLL Constraints



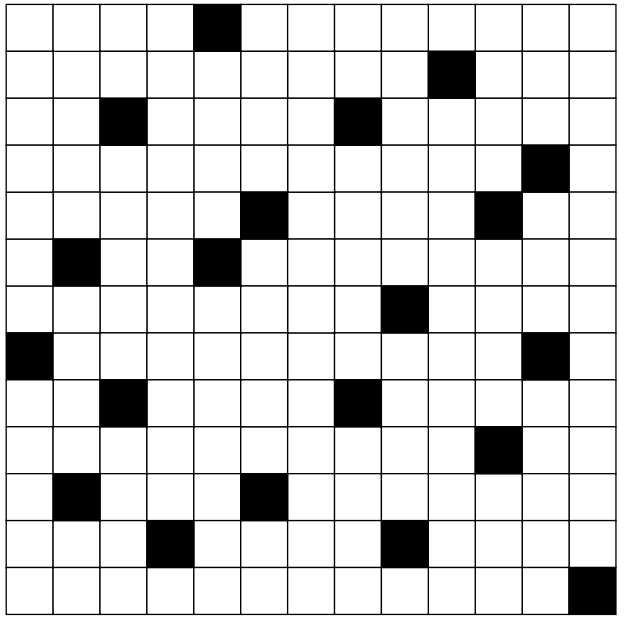
Two-Dimensional RLL Constraints



A $(d_1, k; d_2, k_2)$ RLL constraint imposes:

- ...
- ...

Two-Dimensional RLL Constraints

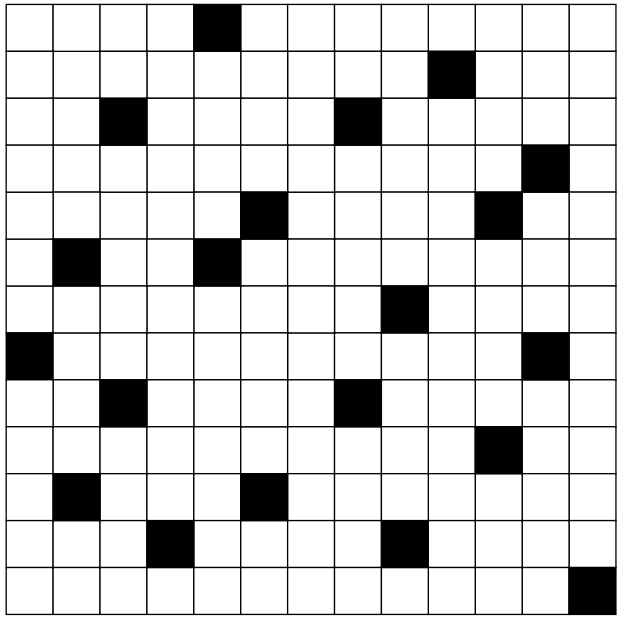


A $(d_1, k; d_2, k_2)$ RLL constraint imposes:

- ...
- ...

Question: How many arrays of size $m \times n$ fulfill these constraints?

Two-Dimensional RLL Constraints



A $(d_1, k; d_2, k_2)$ RLL constraint imposes:

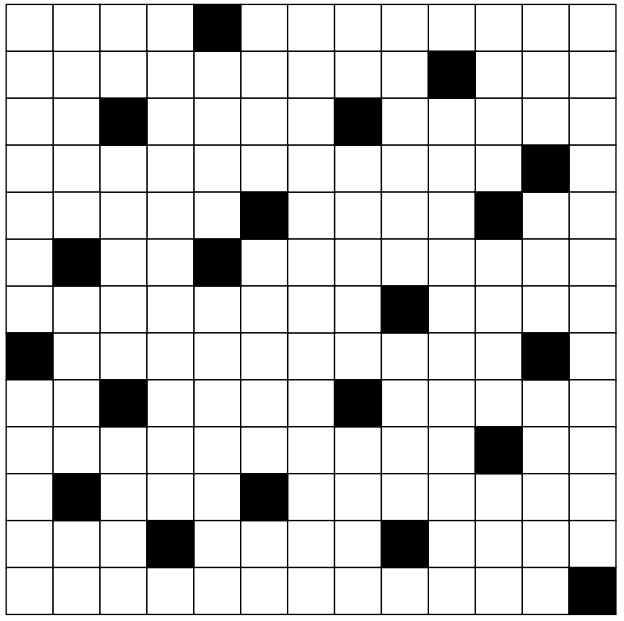
- ...
- ...

Question: How many arrays of size $m \times n$ fulfill these constraints?

Answer: Typically, the answer to such questions looks like

$$N(m,n) = \exp\big(C \cdot mn + o(mn)\big).$$

Two-Dimensional RLL Constraints



A $(d_1, k; d_2, k_2)$ RLL constraint imposes:

- ...
- ...

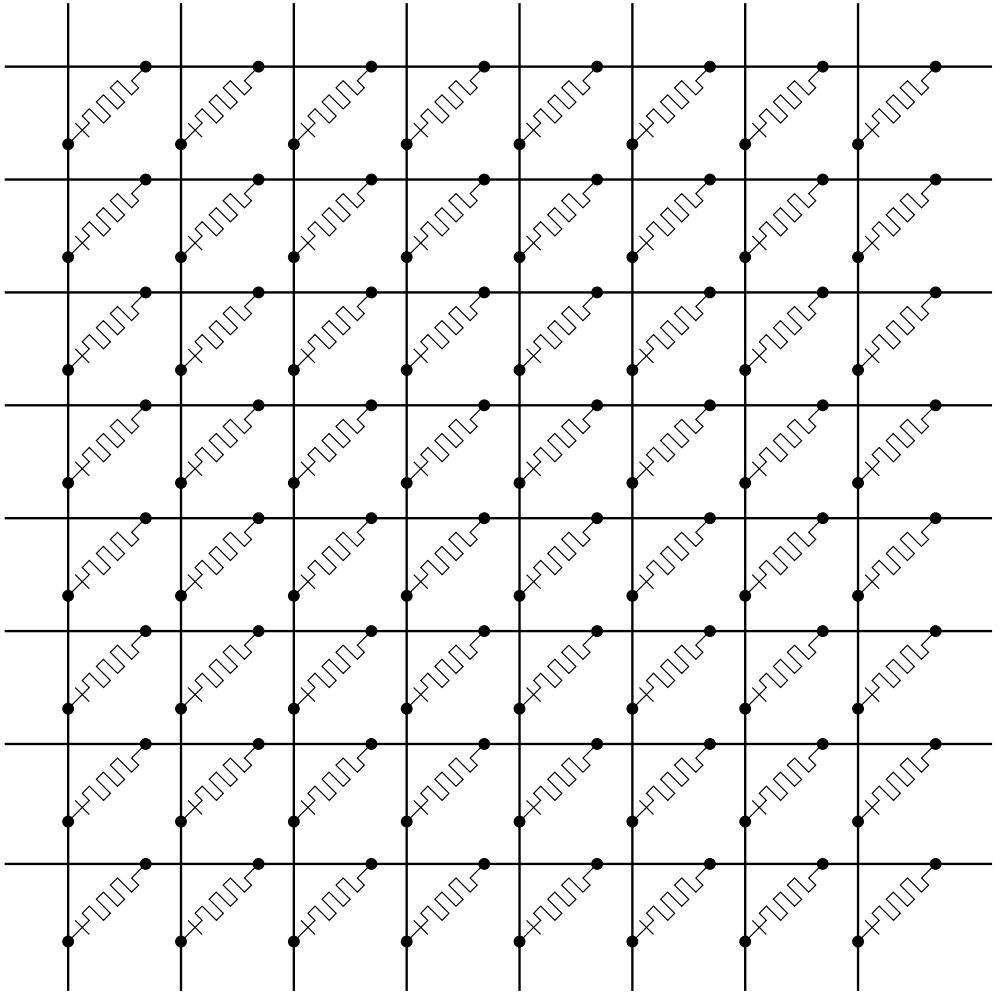
Question: How many arrays of size $m \times n$ fulfill these constraints?

Answer: Typically, the answer to such questions looks like

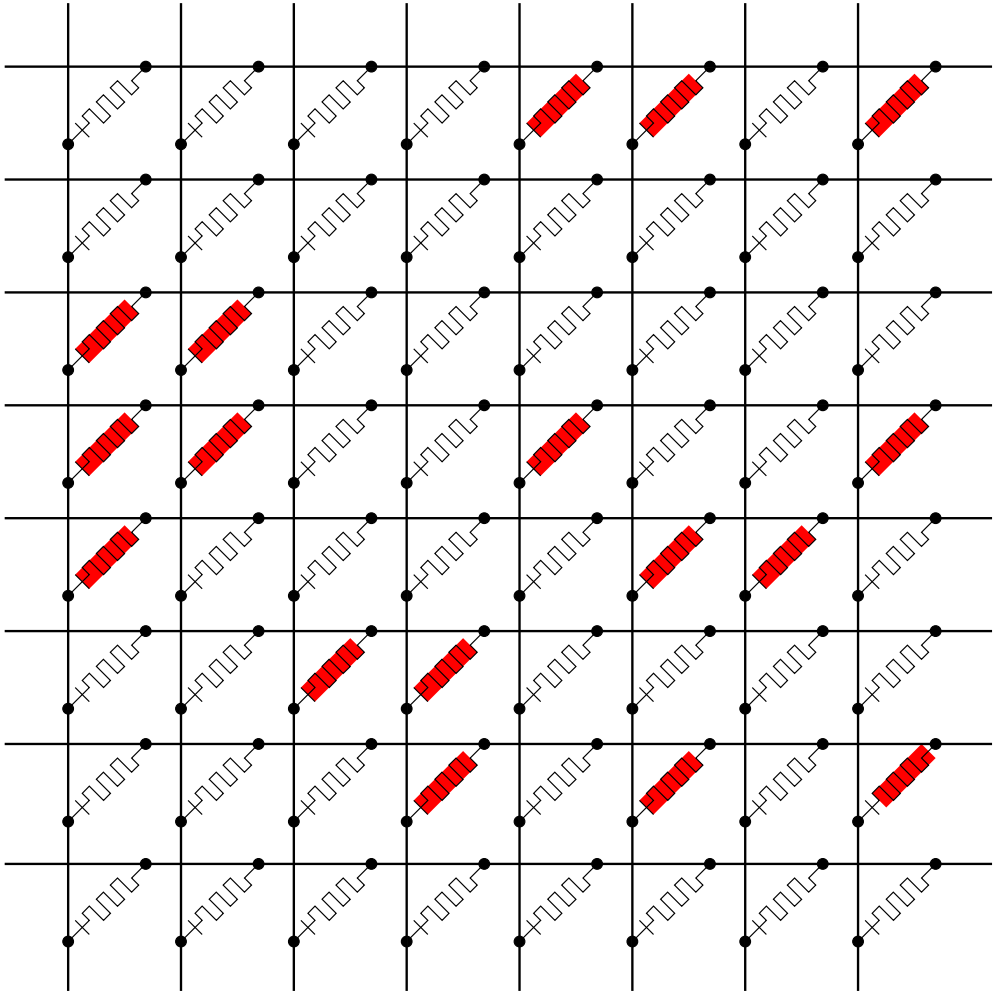
$$N(m,n) = \exp\big(C \cdot mn + o(mn)\big).$$

C: "capacity" or "entropy."

Memristor Array

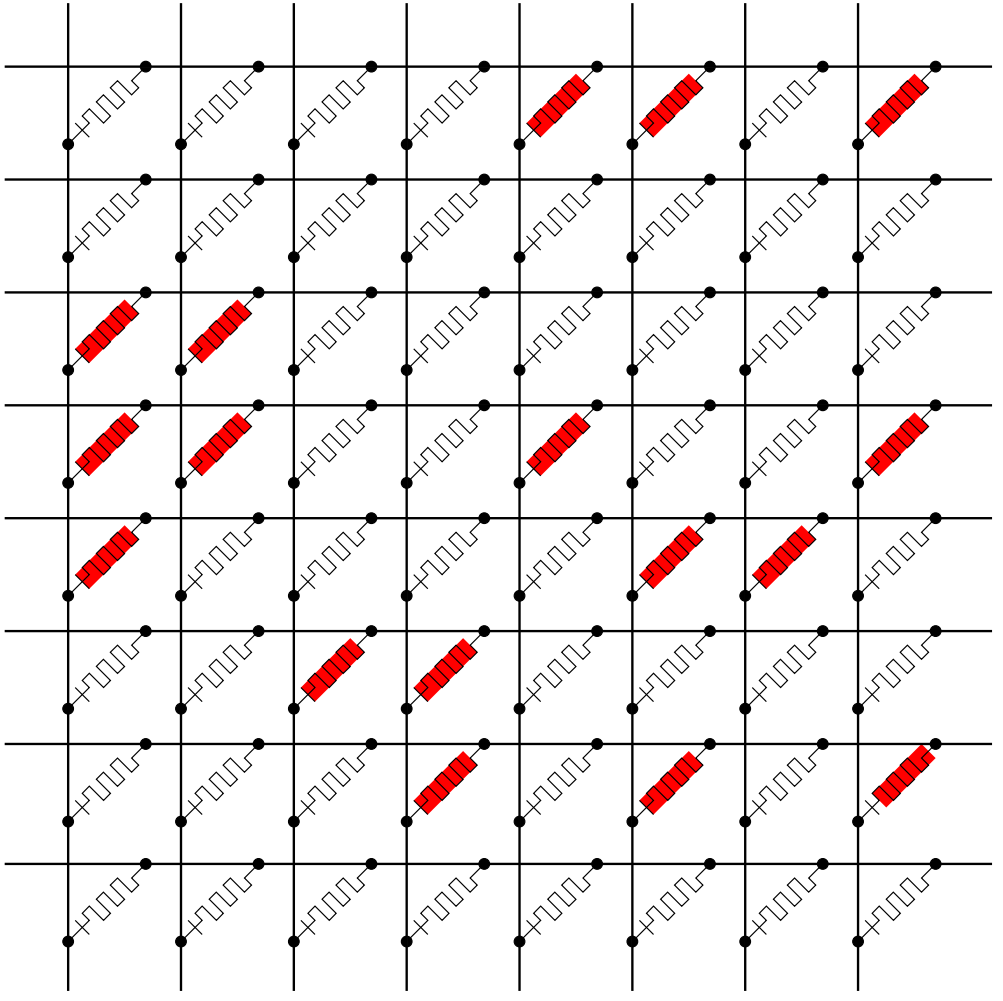


Memristor Array



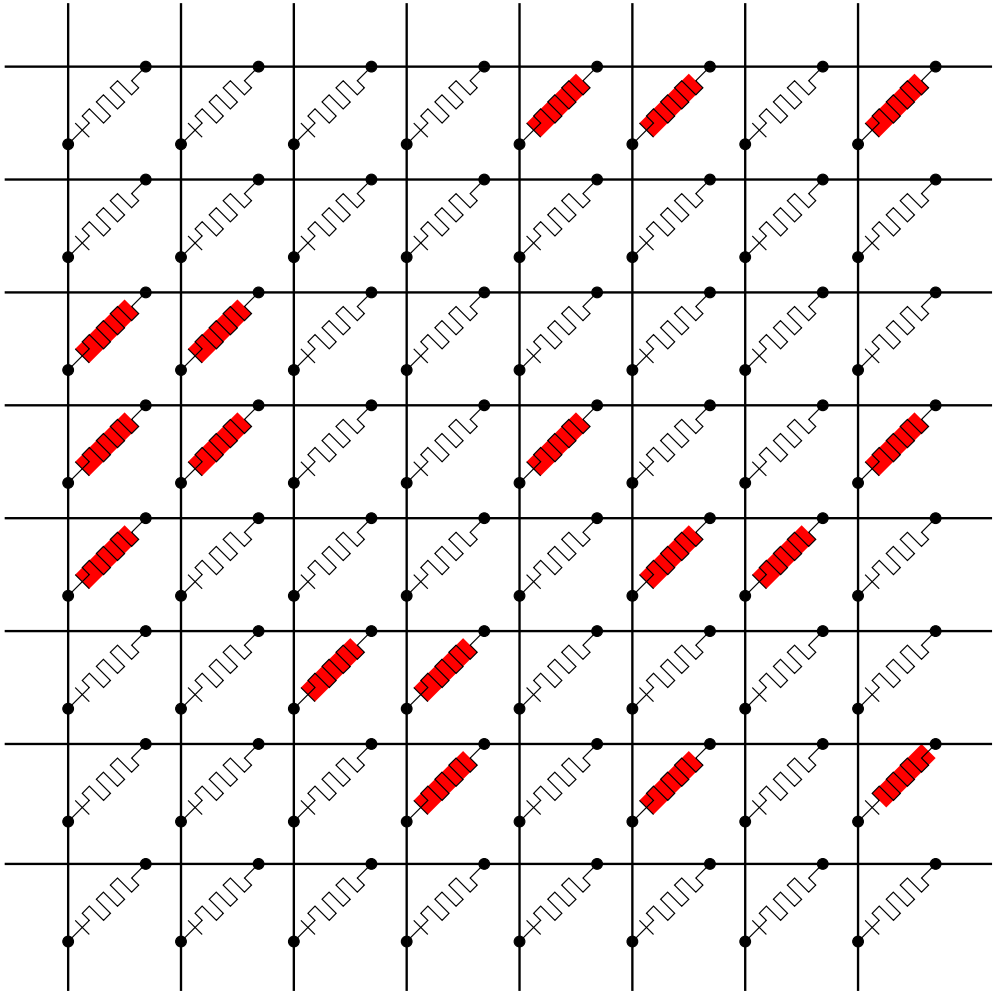
Information is stored by putting some memristors into the high-conductance state.

Memristor Array



It is desirable to limit the current in the driving circuitry and the wires.

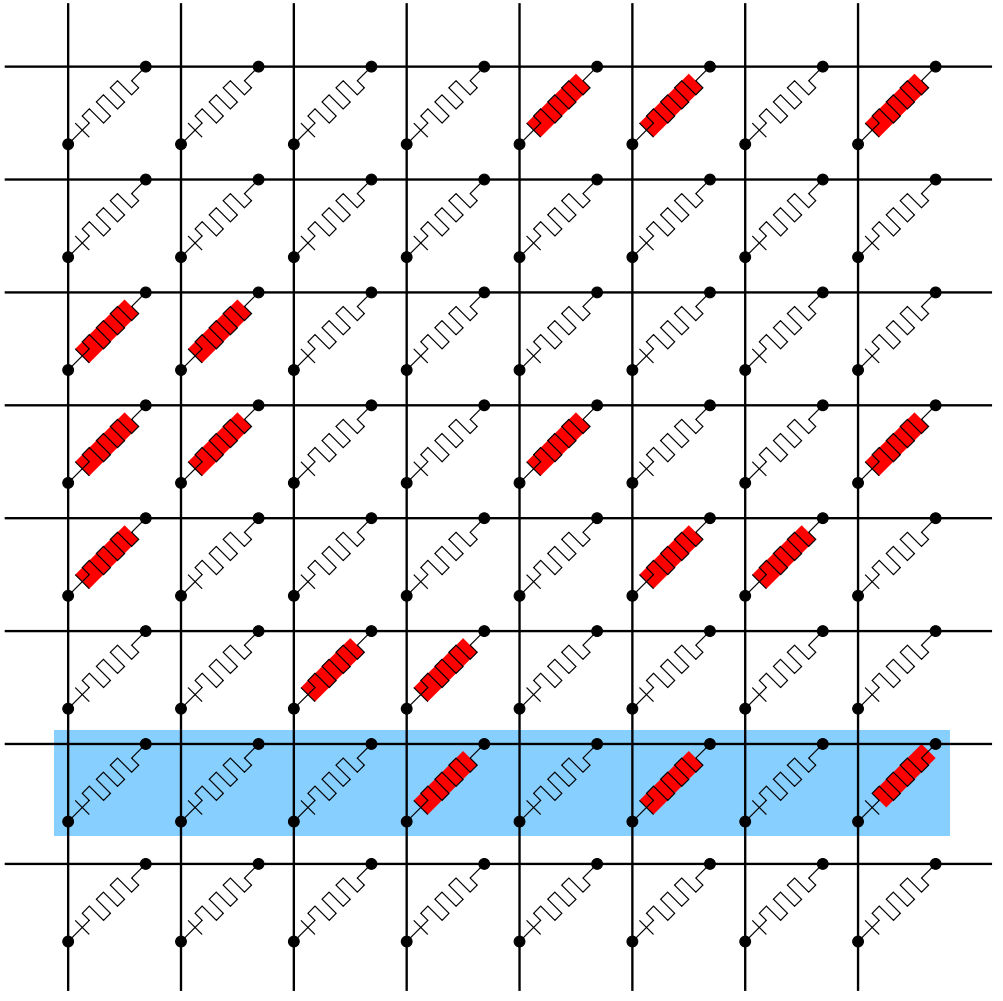
Memristor Array



It is desirable to limit the current in the driving circuitry and the wires.

$\Rightarrow$ Reduction of current, energy, electromigration, half-selection, etc.

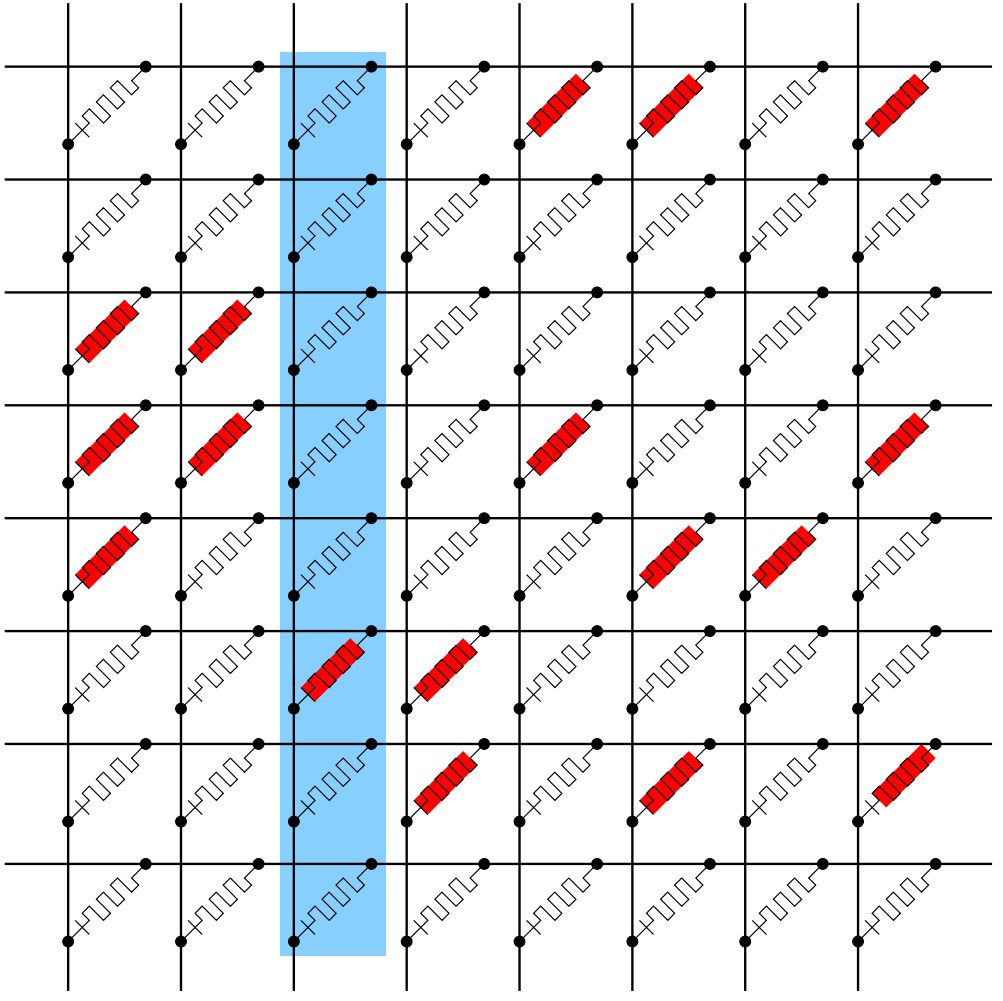
Memristor Array



⇒ Use coding schemes

that limit the number of ON-state memristors per row.

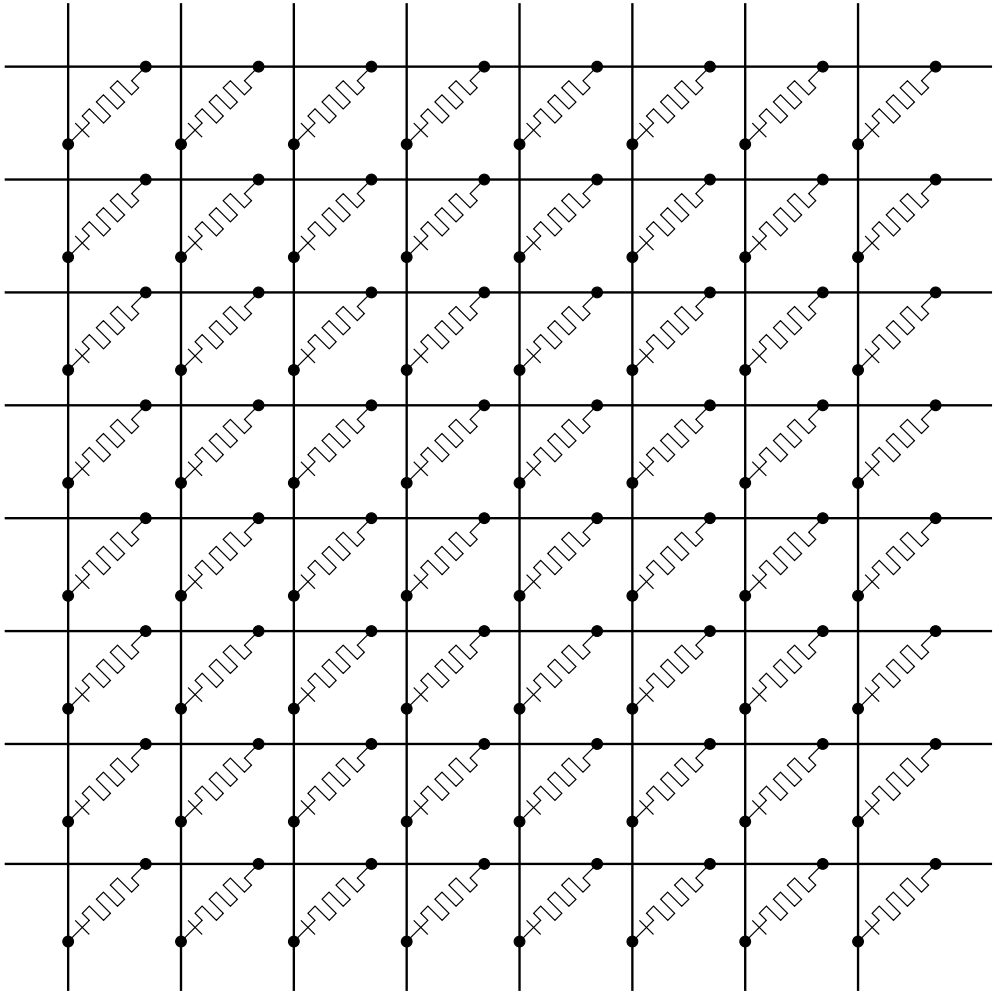
Memristor Array



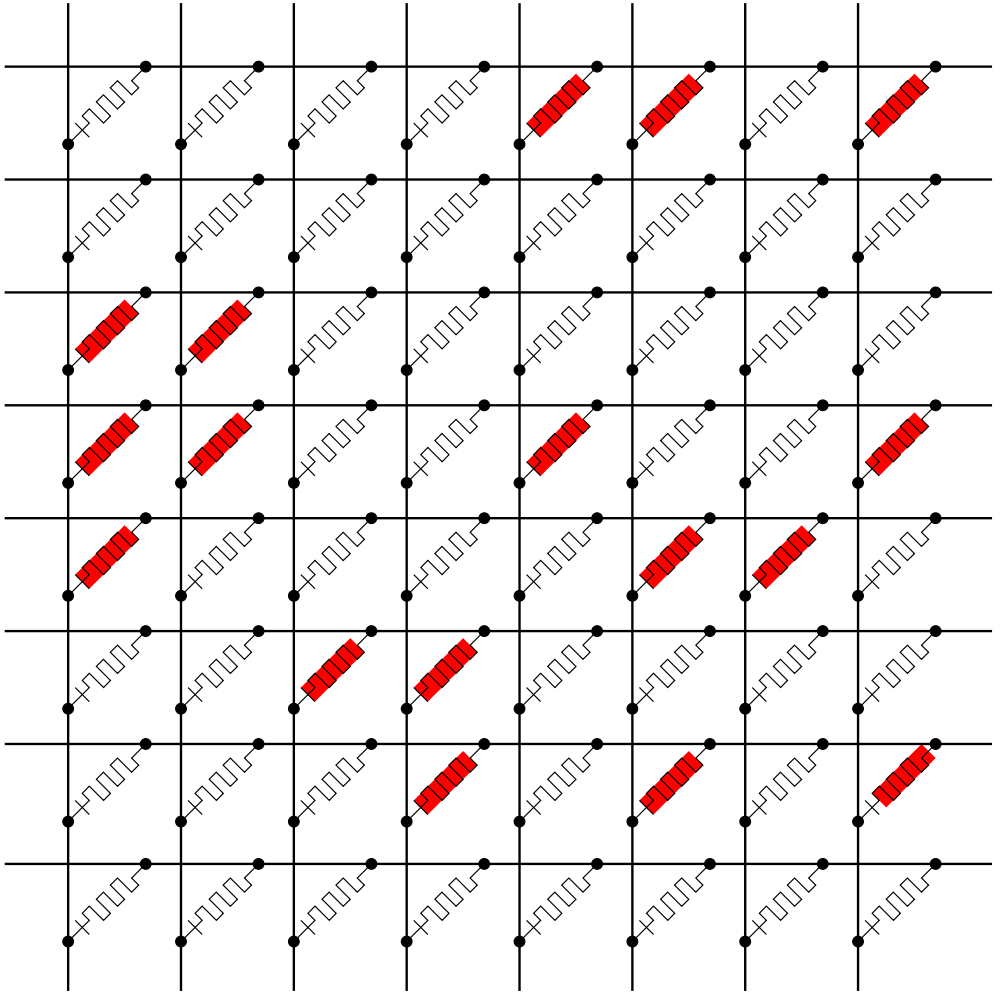
⇒ Use coding schemes

that limit the number of ON-state memristors per column.

Counting Constrained Arrays

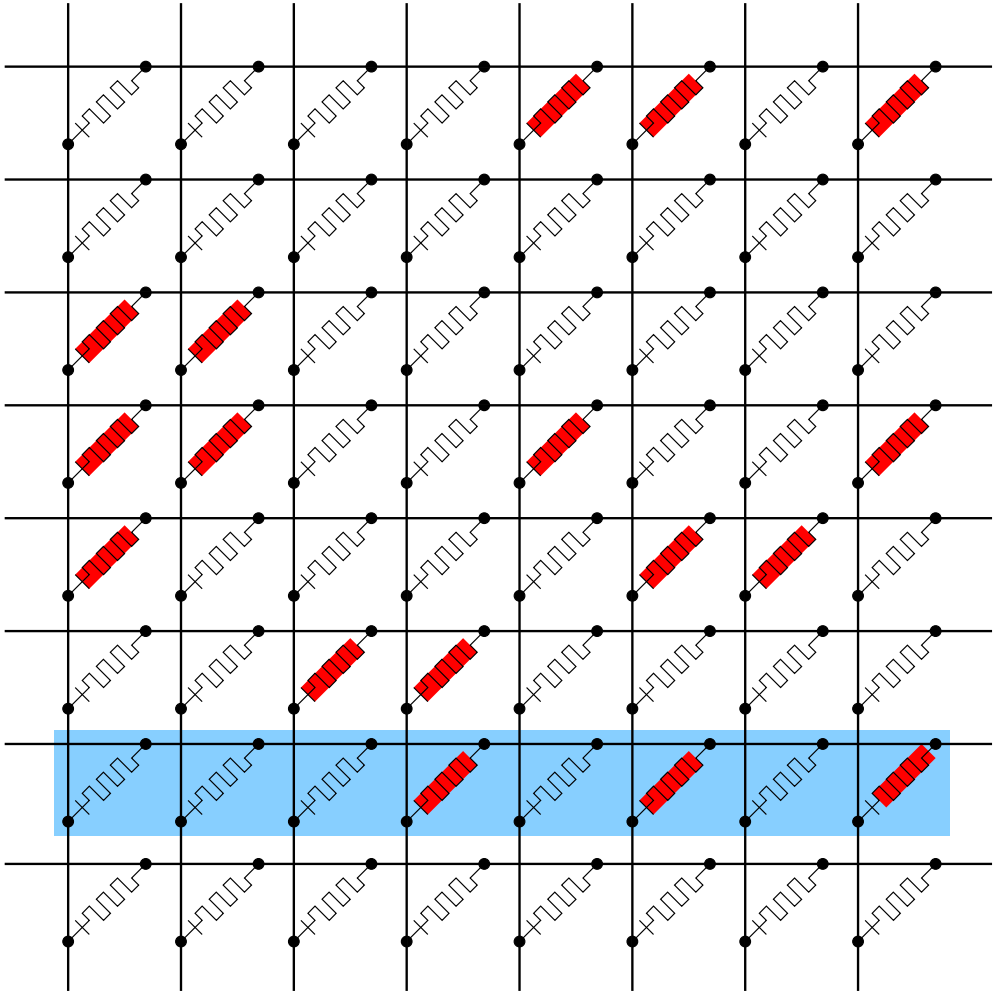


Counting Constrained Arrays



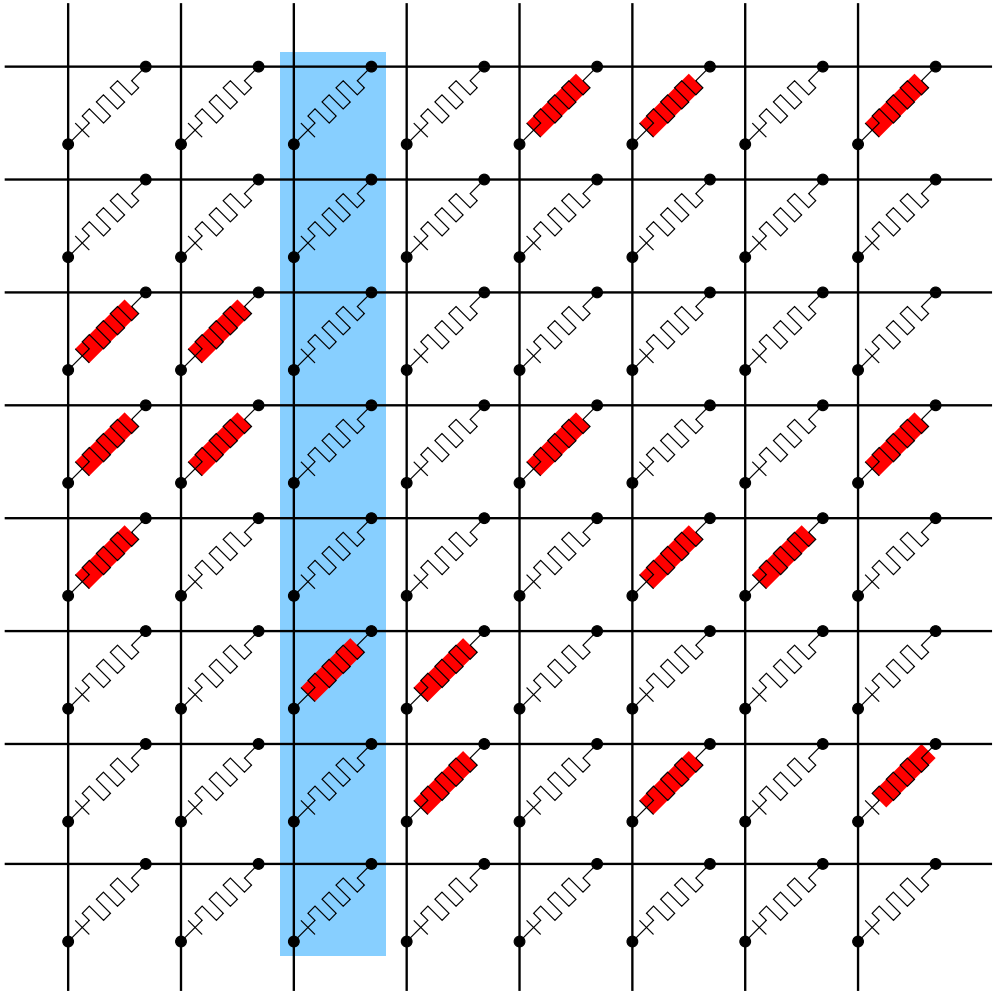
Question: In how many ways can we have at most 4 memristors in the ON-state per row and per column?

Counting Constrained Arrays



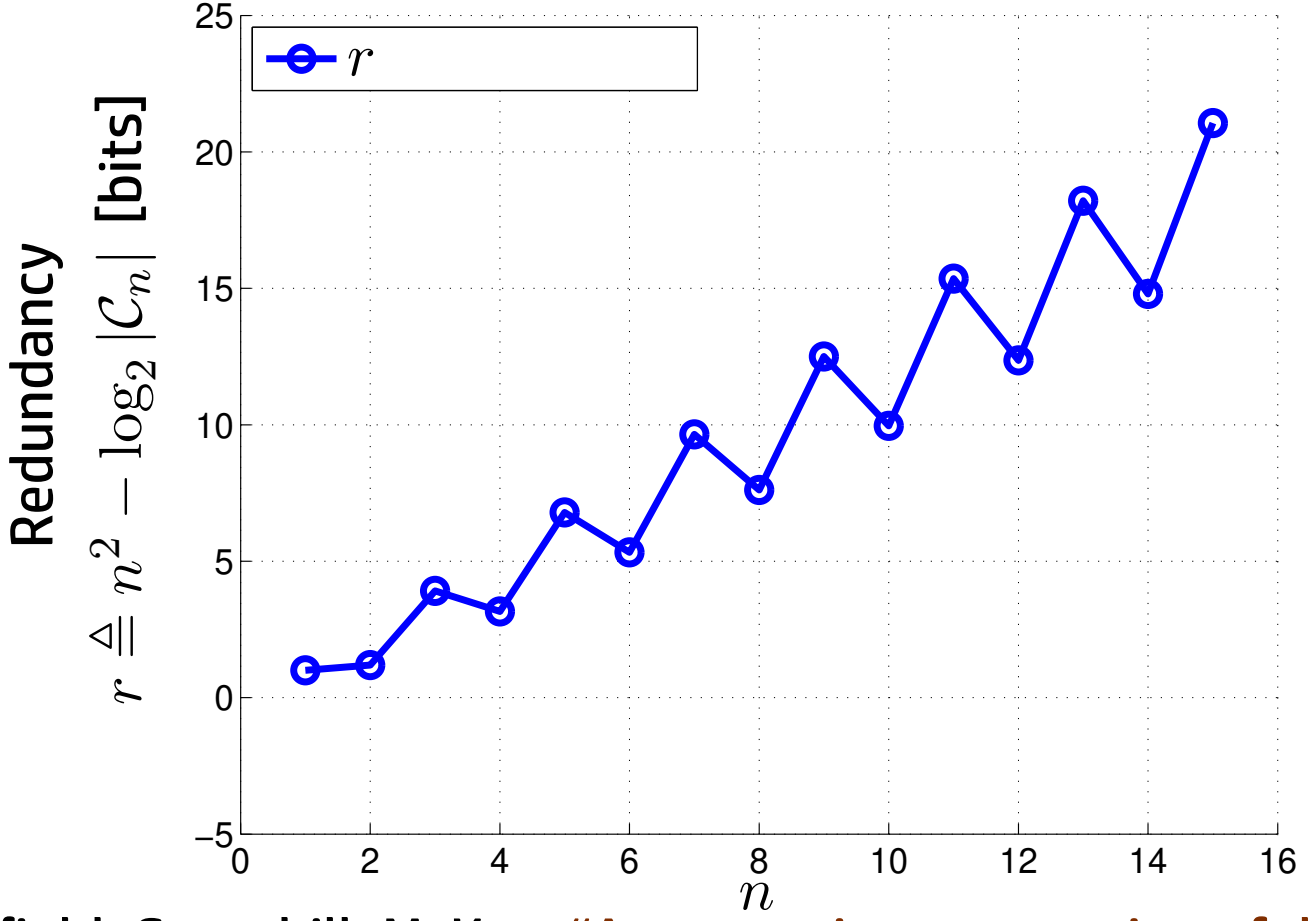
Row condition: at most 4 memristors in ON-state.

Counting Constrained Arrays



Column condition: at most 4 memristors in ON-state.

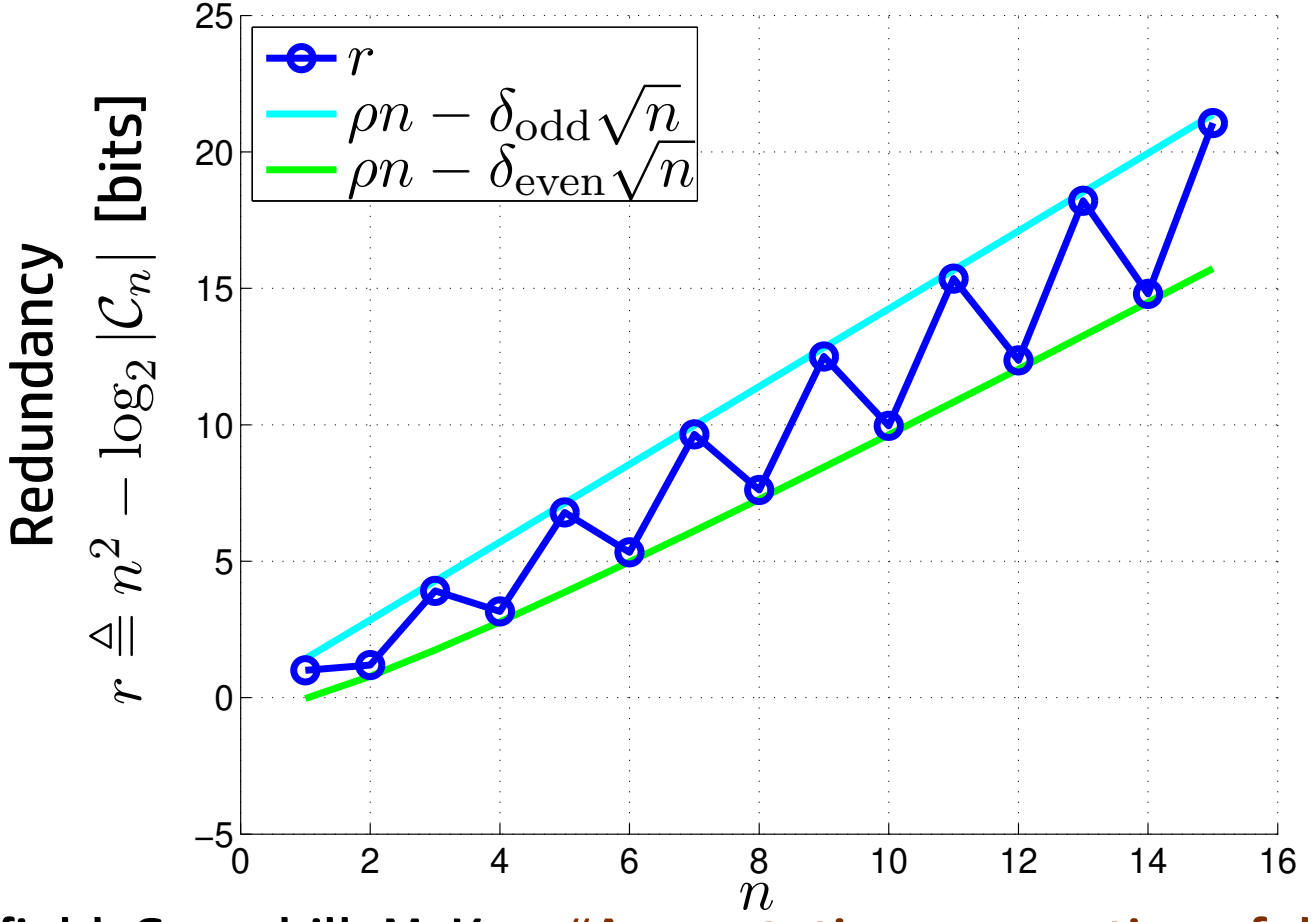
Redundancy for $n \times n$ Binary Arrays Row and Column Weight at most $n/2$



r-values:
courtesy of R. Roth

- Canfield, Greenhill, McKay, "Asymptotic enumeration of dense 0–1 matrices with specified line sums," *J. Comb. Theory A*, 2008.
- Ordentlich, Parvaresh, Roth, "Asymptotic enumeration of binary matrices with bounded row and column weights," *SIAM J. Disc. Math.*, 2011.

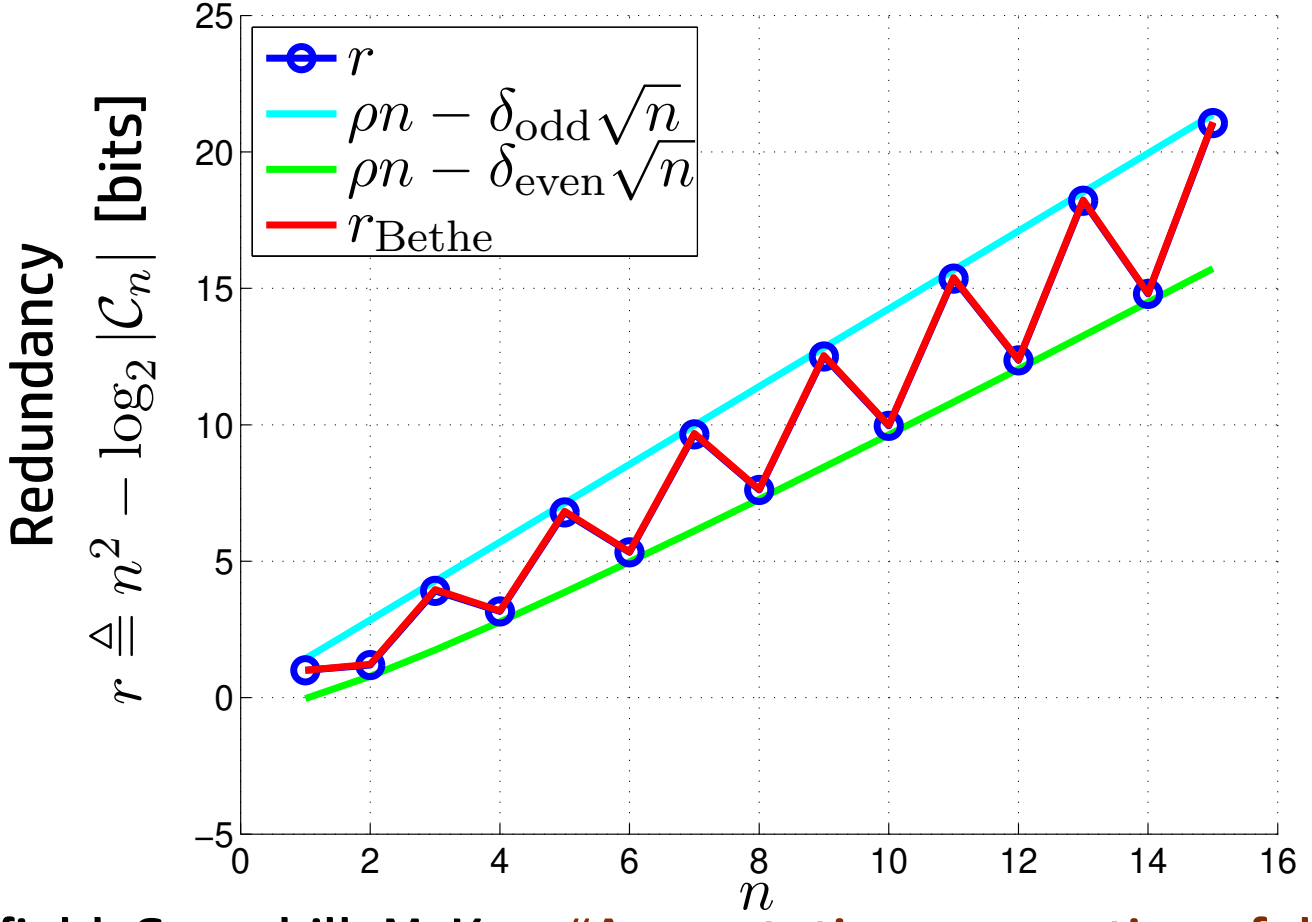
Redundancy for $n \times n$ Binary Arrays Row and Column Weight at most $n/2$



r-values:
courtesy of R. Roth

- Canfield, Greenhill, McKay, "Asymptotic enumeration of dense 0–1 matrices with specified line sums," *J. Comb. Theory A*, 2008.
- Ordentlich, Parvaresh, Roth, "Asymptotic enumeration of binary matrices with bounded row and column weights," *SIAM J. Disc. Math.*, 2011.

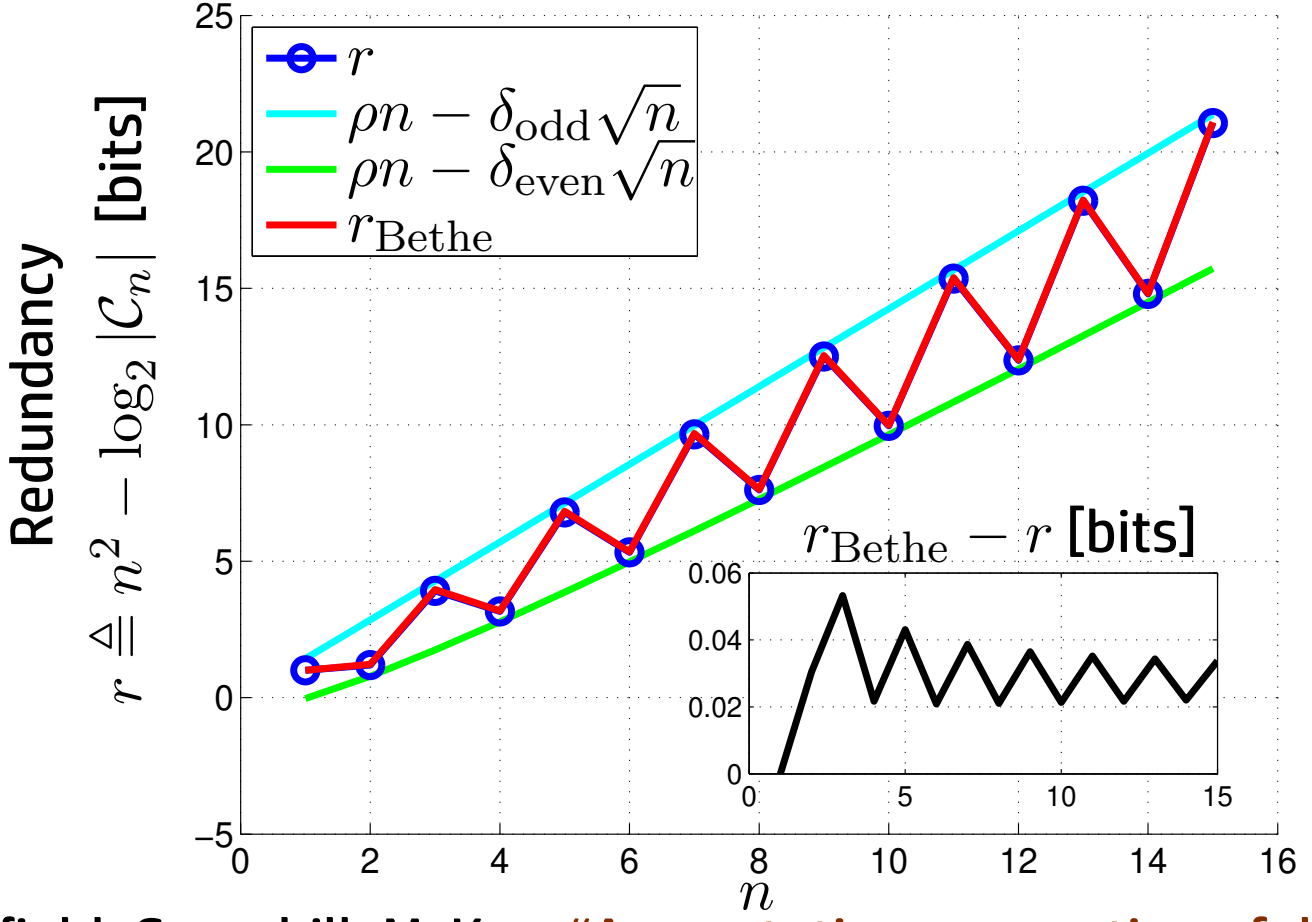
Redundancy for $n \times n$ Binary Arrays Row and Column Weight at most $n/2$



r-values:
courtesy of R. Roth

- Canfield, Greenhill, McKay, "Asymptotic enumeration of dense 0–1 matrices with specified line sums," *J. Comb. Theory A*, 2008.
- Ordentlich, Parvaresh, Roth, "Asymptotic enumeration of binary matrices with bounded row and column weights," *SIAM J. Disc. Math.*, 2011.

Redundancy for $n \times n$ Binary Arrays Row and Column Weight at most $n/2$



r-values:
courtesy of R. Roth

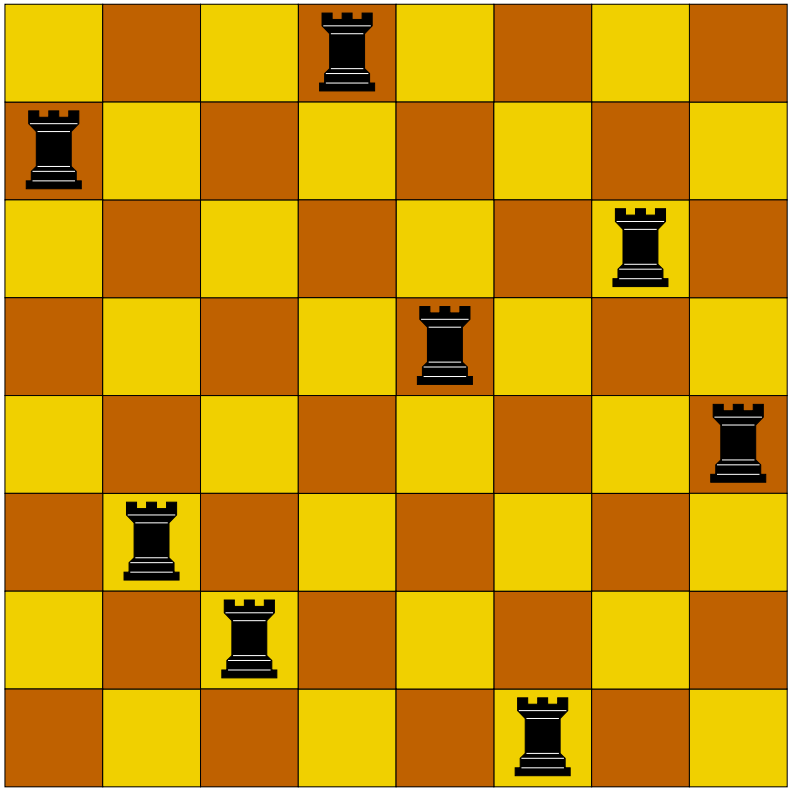
- Canfield, Greenhill, McKay, "Asymptotic enumeration of dense 0–1 matrices with specified line sums," *J. Comb. Theory A*, 2008.
- Ordentlich, Parvaresh, Roth, "Asymptotic enumeration of binary matrices with bounded row and column weights," *SIAM J. Disc. Math.*, 2011.

Overview

- Setting up a graphical model
- Permanent of a matrix
- Factor graphs and the sum-product algorithm
- The total sum of a factor graph and its Bethe approximation
- A combinatorial interpretation of the Bethe approximation
- Further comments
- Conclusions

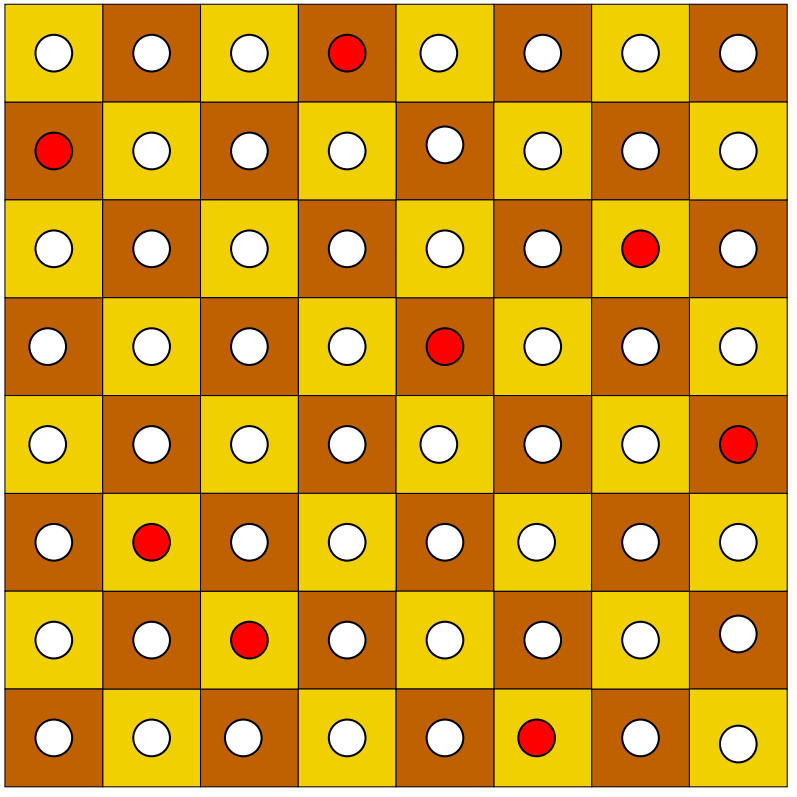
Towards a graphical model

Towards a Graphical Model

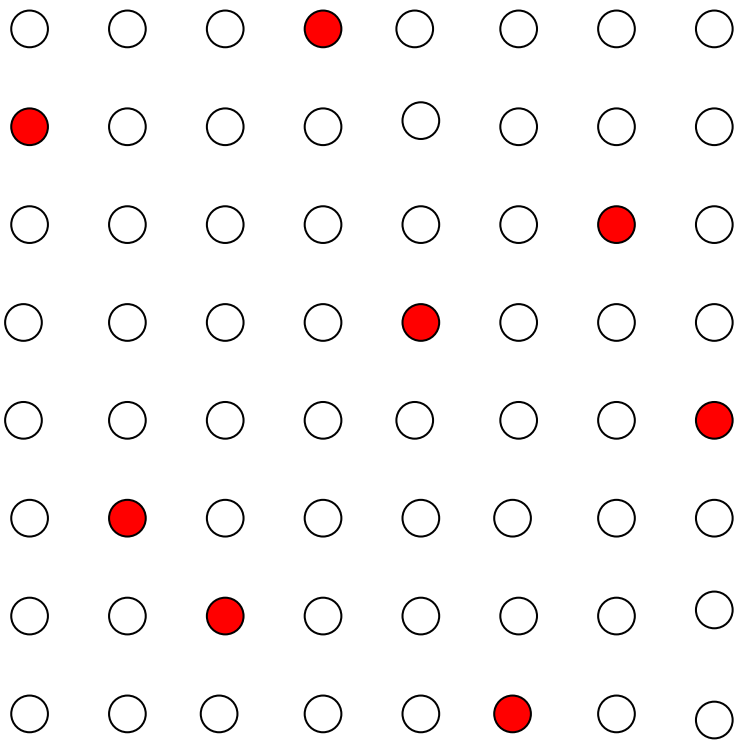


Question: in how many ways can we place 8 non-attacking rooks on a chess board?

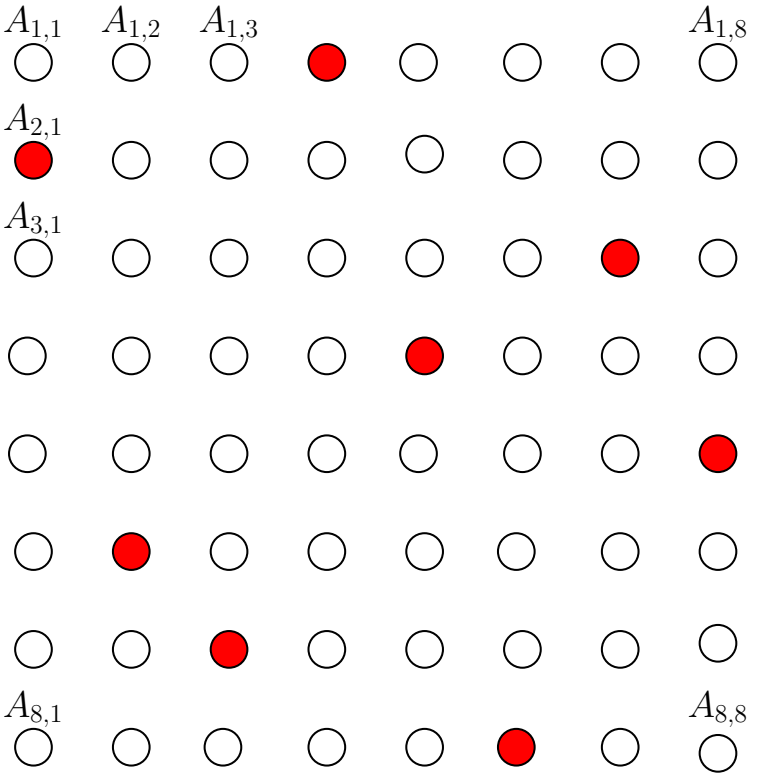
Towards a Graphical Model



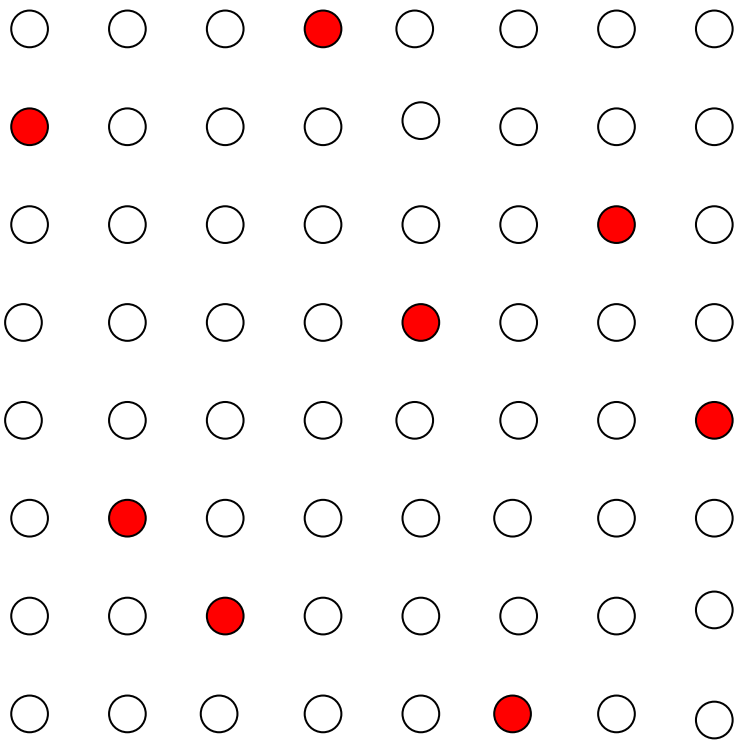
Towards a Graphical Model



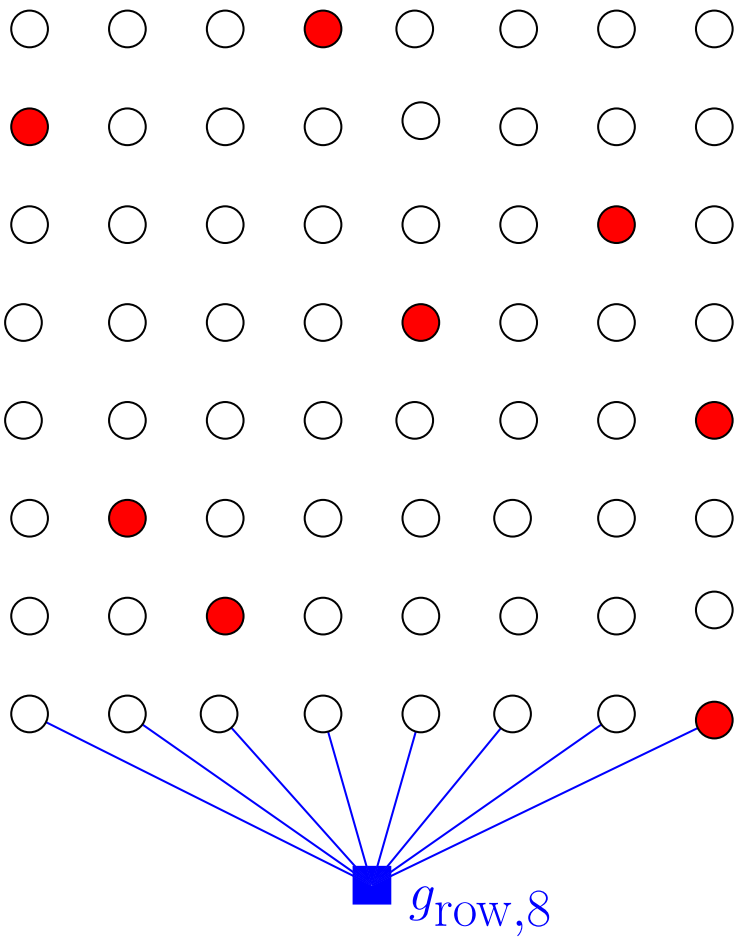
Towards a Graphical Model



Towards a Graphical Model

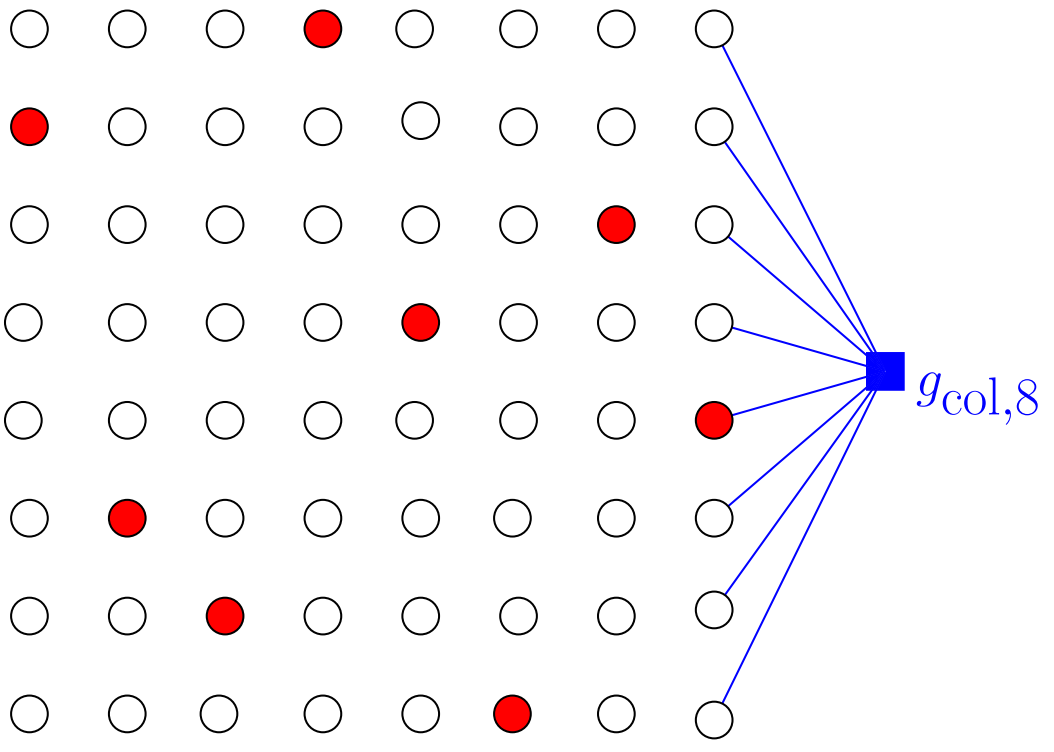


Towards a Graphical Model



$$g_{\mathrm{row},8}(a_{8,1},\ldots,a_{8,8}) \triangleq \begin{cases} 1 & \text{exactly one rook} \\ 0 & \text{otherwise} \end{cases}$$

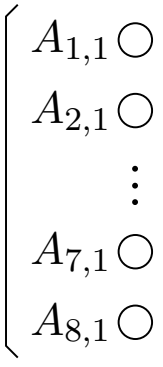
Towards a Graphical Model



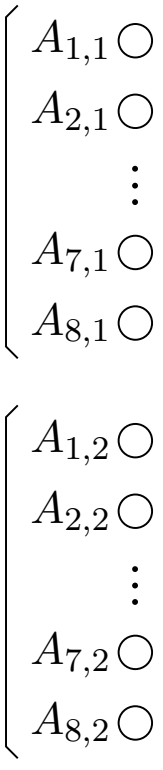
$$g_{\mathrm{col},8}(a_{1,8},\ldots,a_{8,8}) \triangleq \begin{cases} 1 & \text{exactly one rook} \\ 0 & \text{otherwise} \end{cases}$$

Towards a Graphical Model

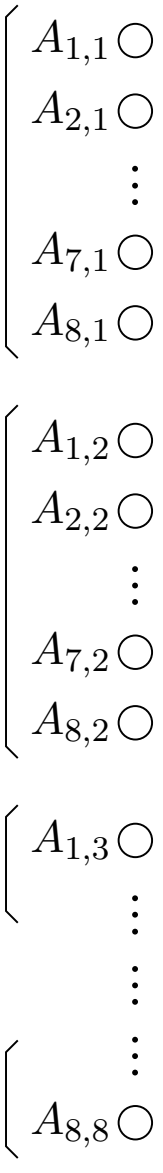
Towards a Graphical Model



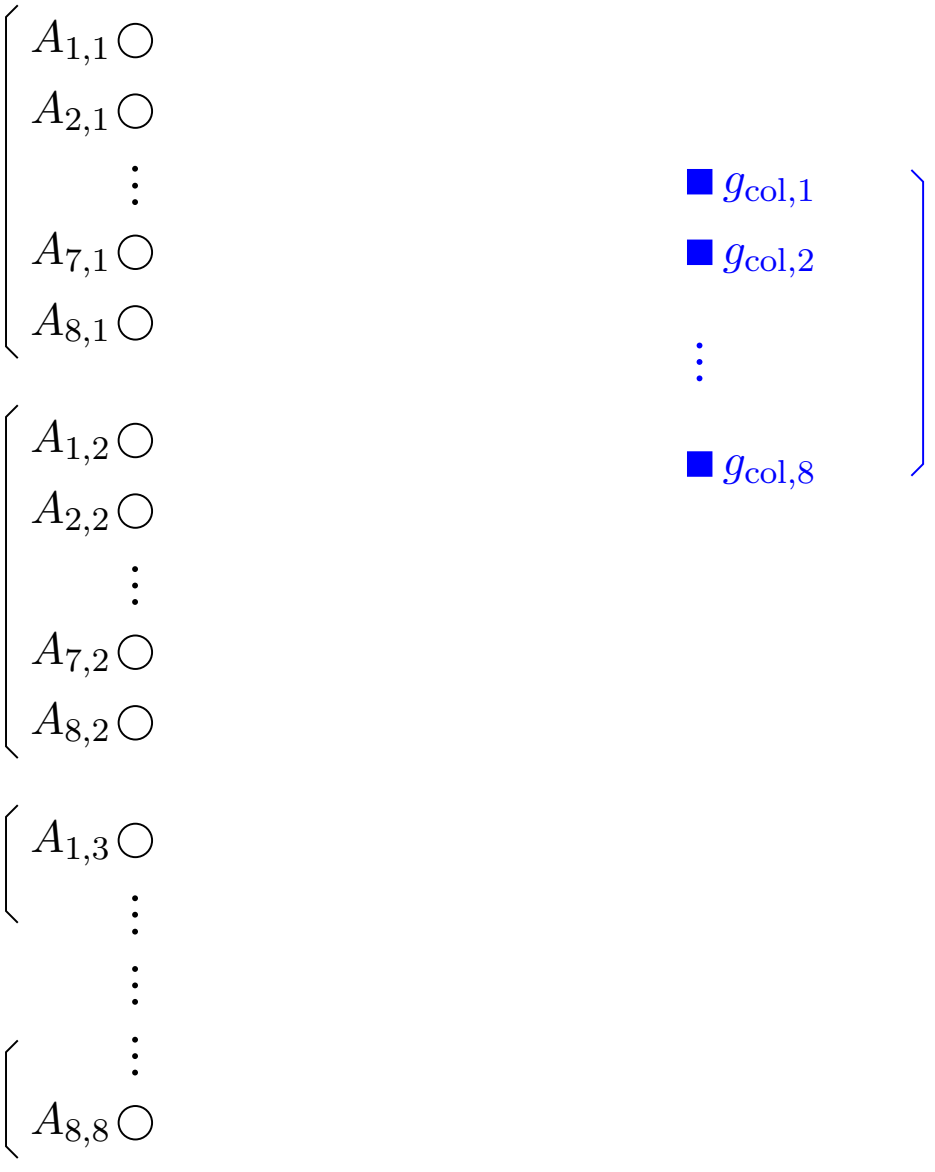
Towards a Graphical Model



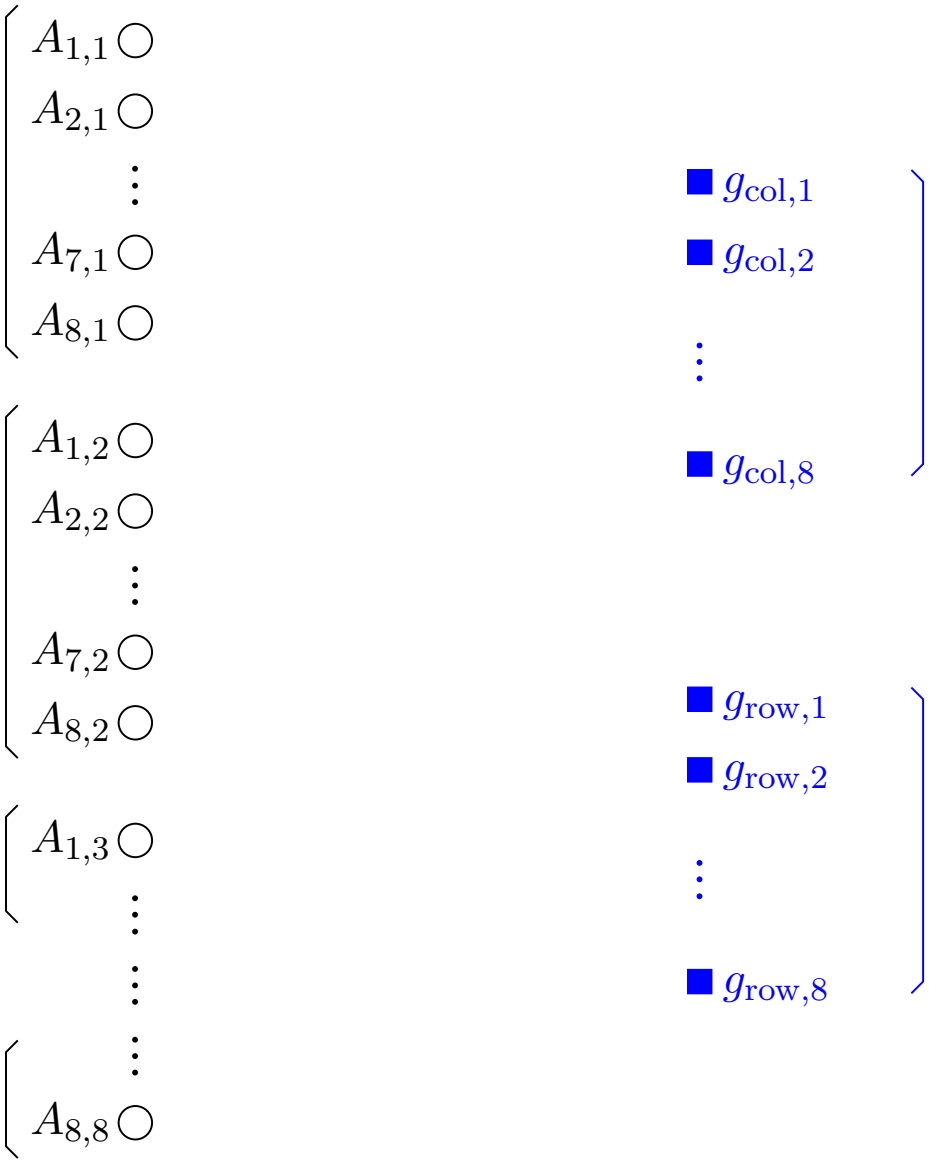
Towards a Graphical Model



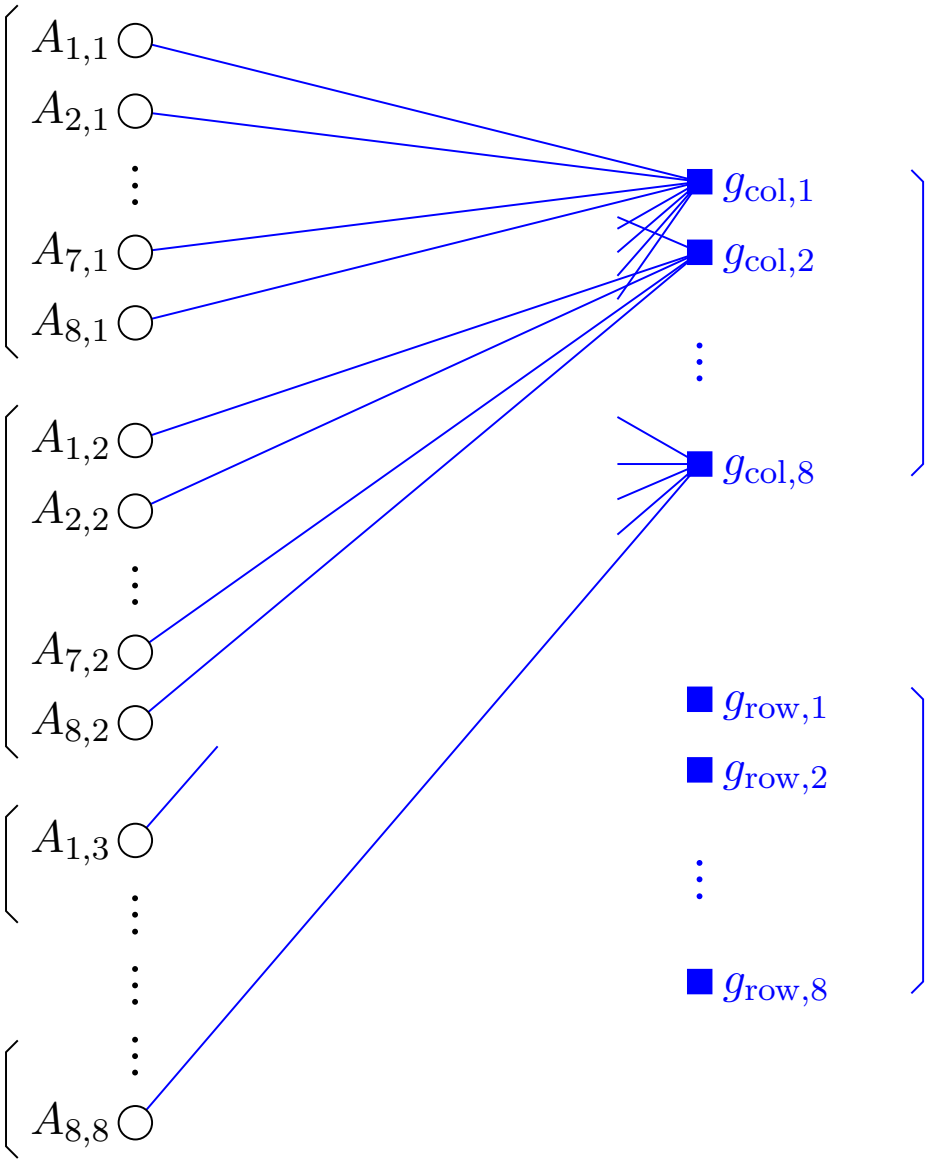
Towards a Graphical Model



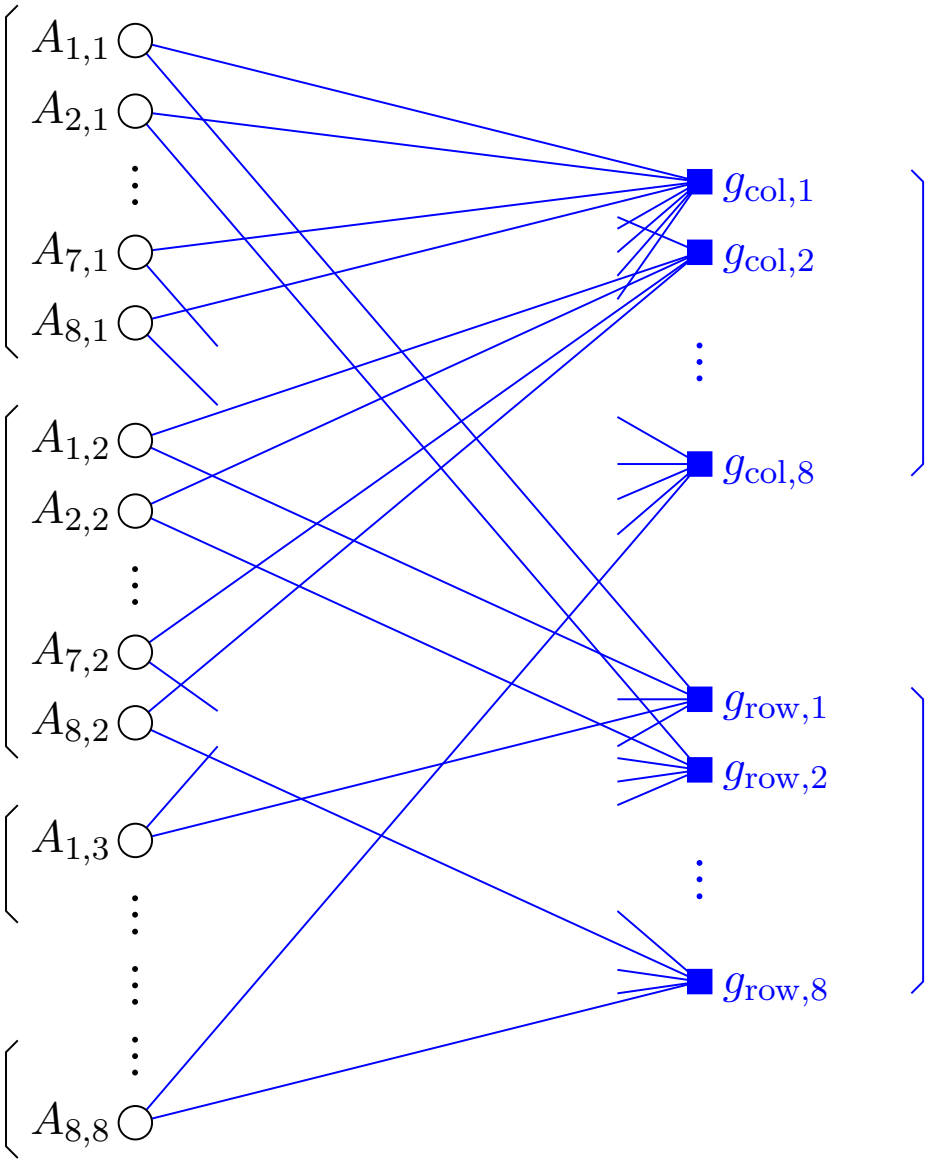
Towards a Graphical Model



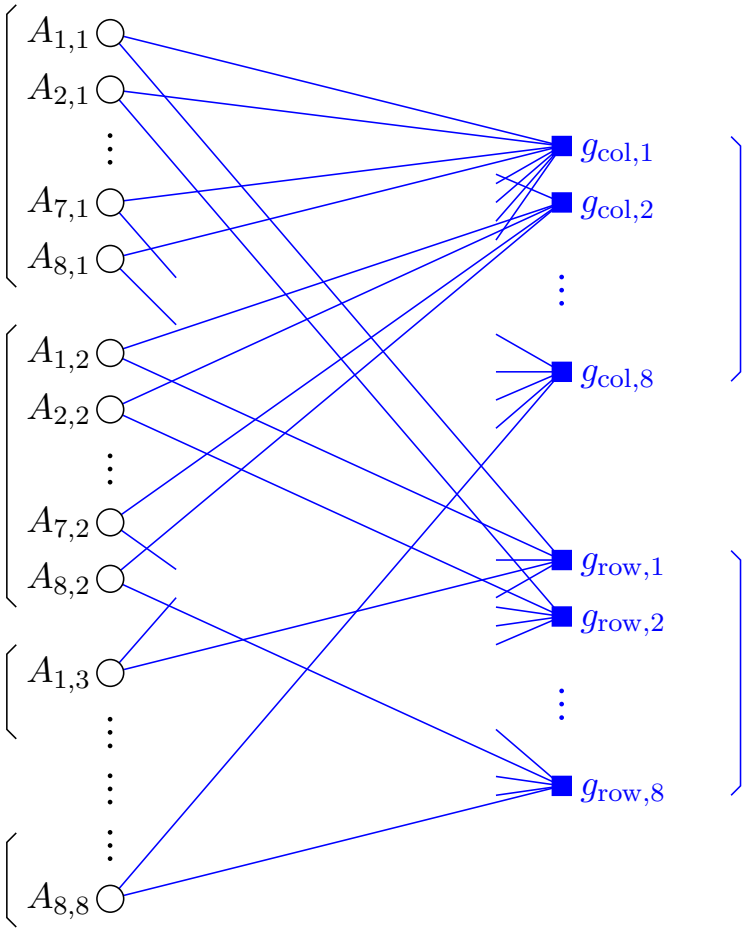
Towards a Graphical Model



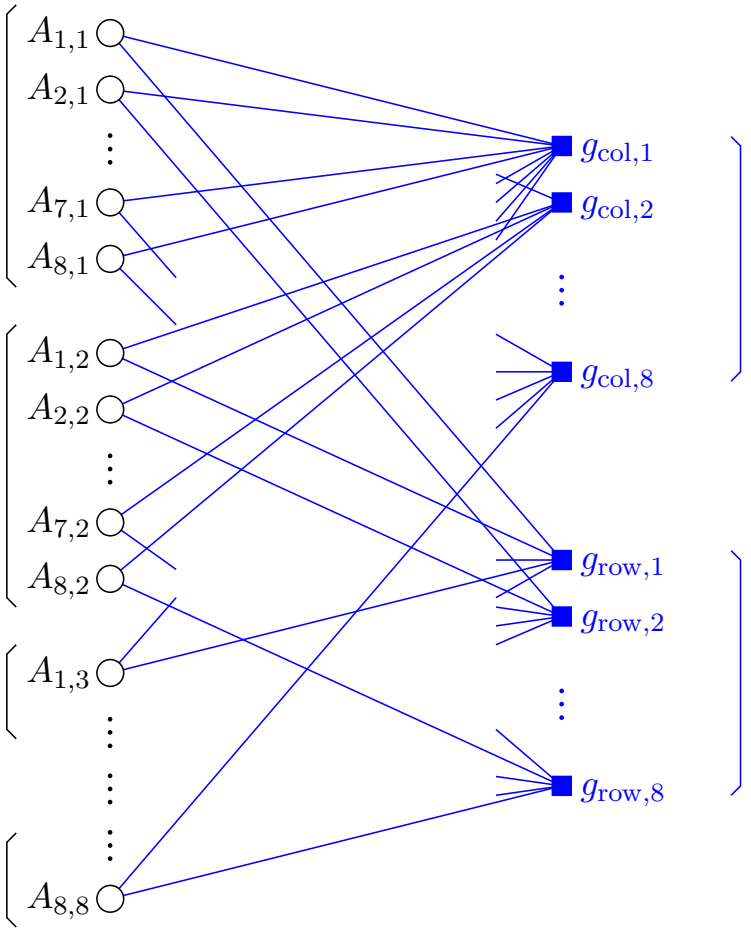
Towards a Graphical Model



Towards a Graphical Model



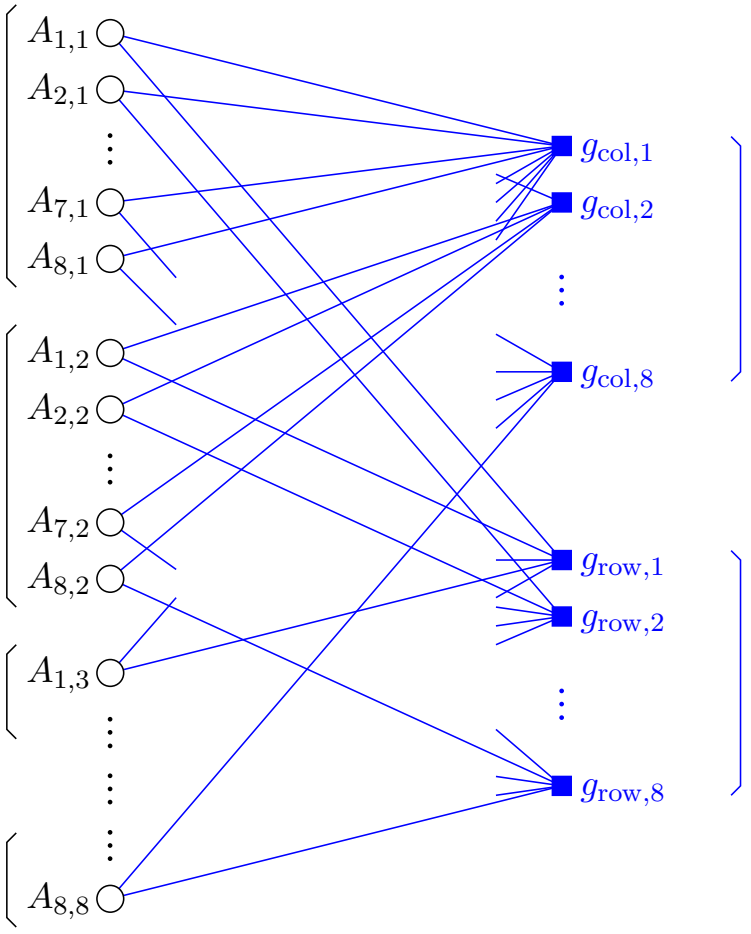
Towards a Graphical Model



Global function:

$$g(a_{1,1}, \ldots, a_{8,8}) = \prod_j g_{\text{col},j}(a_{1,j}, \ldots, a_{8,j}) \times \prod_i g_{\text{row},i}(a_{i,1}, \ldots, a_{i,8})$$

Towards a Graphical Model



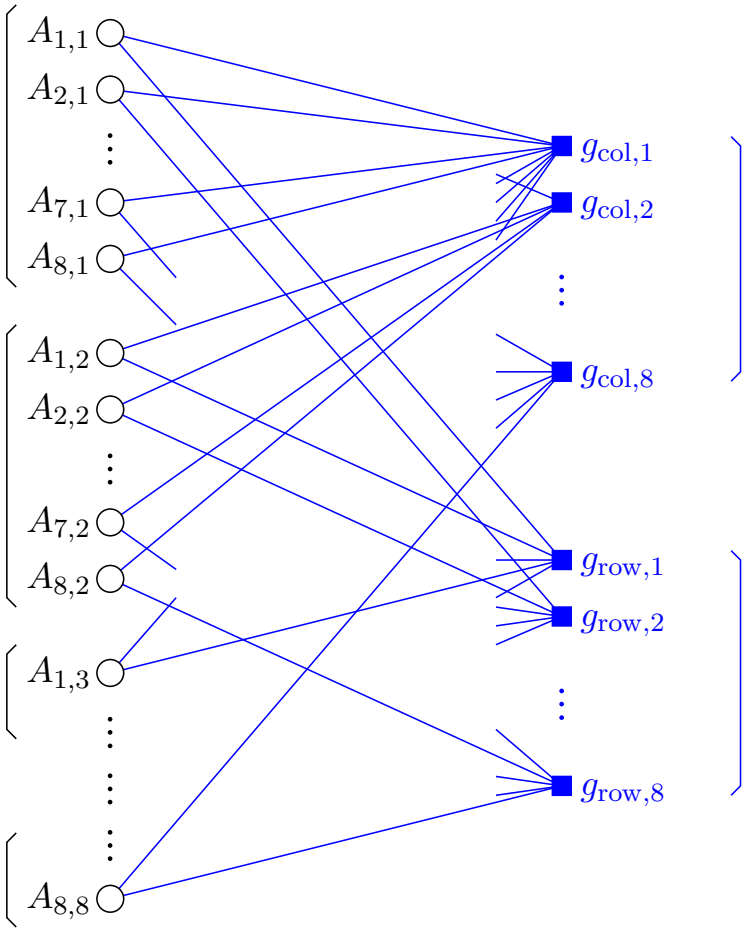
Global function:

$$g(a_{1,1}, \ldots, a_{8,8}) = \prod_j g_{\text{col},j}(a_{1,j}, \ldots, a_{8,j}) \times \prod_i g_{\text{row},i}(a_{i,1}, \ldots, a_{i,8})$$

Total sum:

$$Z = \sum_{a_{1,1},\ldots,a_{8,8}} g(a_{1,1}, \ldots, a_{8,8})$$

Towards a Graphical Model



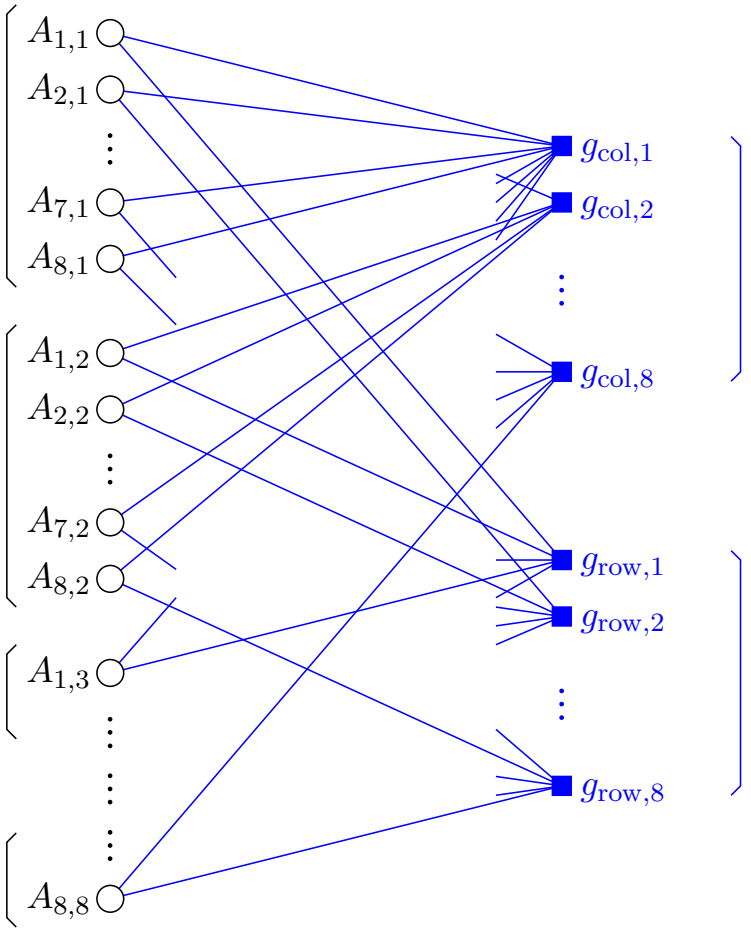
Global function:

$$g(a_{1,1}, \ldots, a_{8,8}) = \prod_j g_{\text{col},j}(a_{1,j}, \ldots, a_{8,j}) \times \prod_i g_{\text{row},i}(a_{i,1}, \ldots, a_{i,8})$$

Total sum (partition function):

$$Z = \sum_{a_{1,1},\ldots,a_{8,8}} g(a_{1,1}, \ldots, a_{8,8})$$

Towards a Graphical Model



Global function:

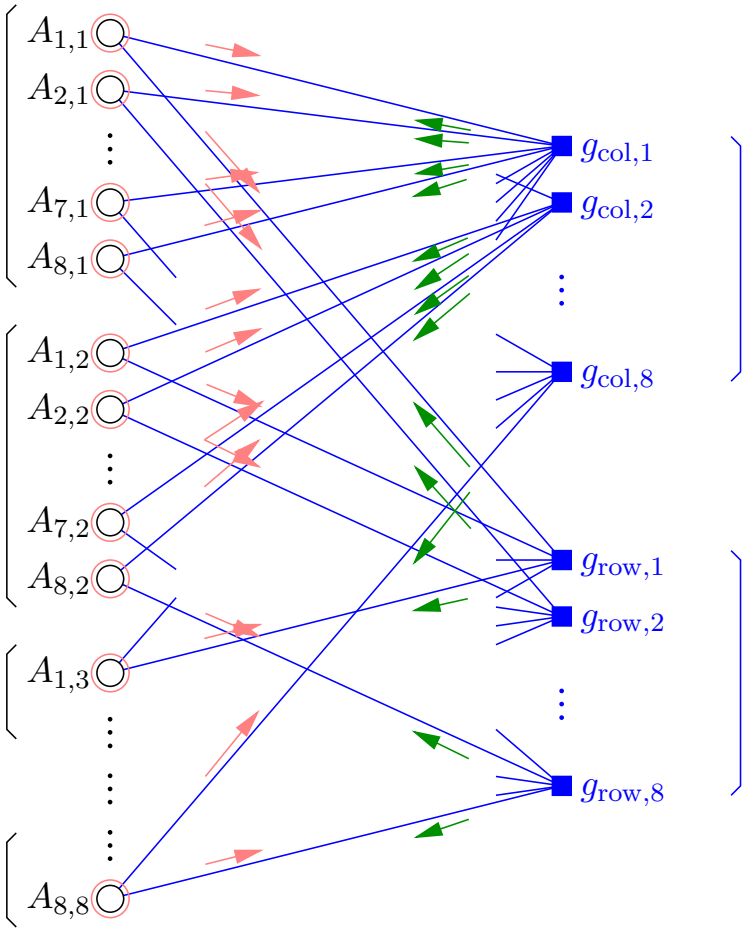
$$g(a_{1,1}, \ldots, a_{8,8}) = \prod_j g_{\text{col},j}(a_{1,j}, \ldots, a_{8,j}) \times \prod_i g_{\text{row},i}(a_{i,1}, \ldots, a_{i,8})$$

Total sum (partition function):

$$Z = \sum_{a_{1,1},\ldots,a_{8,8}} g(a_{1,1}, \ldots, a_{8,8})$$

Use of loopy belief propagation for approximating Z?

Towards a Graphical Model



Global function:

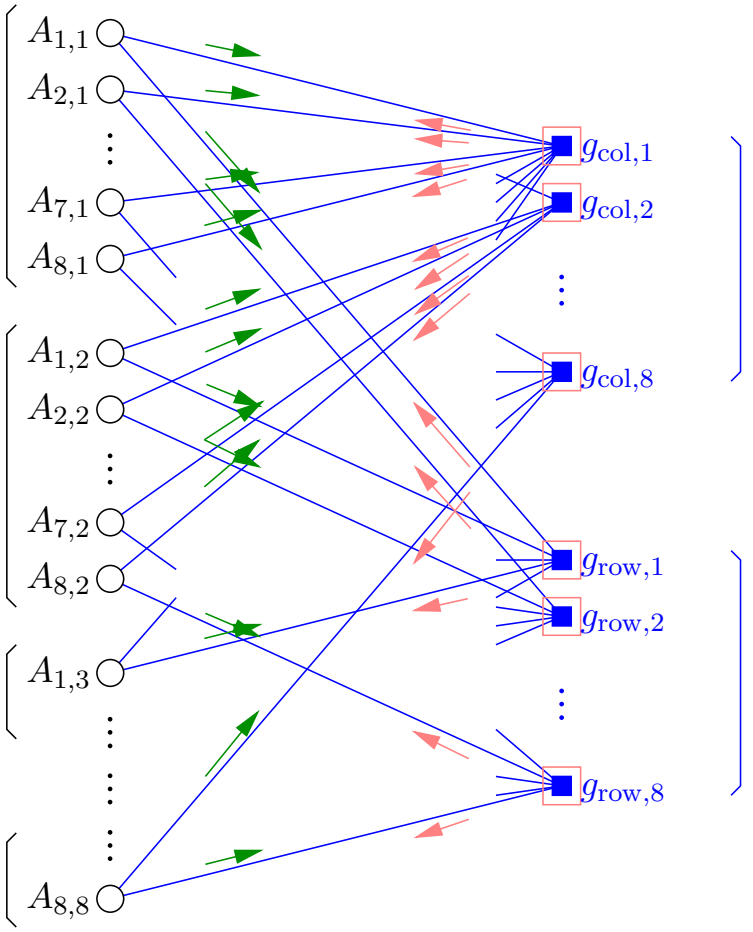
$$g(a_{1,1}, \ldots, a_{8,8}) = \prod_j g_{\text{col},j}(a_{1,j}, \ldots, a_{8,j}) \times \prod_i g_{\text{row},i}(a_{i,1}, \ldots, a_{i,8})$$

Total sum (partition function):

$$Z = \sum_{a_{1,1},\ldots,a_{8,8}} g(a_{1,1}, \ldots, a_{8,8})$$

Use of loopy belief propagation for approximating Z?

Towards a Graphical Model



Global function:

$$g(a_{1,1}, \ldots, a_{8,8}) = \prod_j g_{\text{col},j}(a_{1,j}, \ldots, a_{8,j}) \times \prod_i g_{\text{row},i}(a_{i,1}, \ldots, a_{i,8})$$

Total sum (partition function):

$$Z = \sum_{a_{1,1},\ldots,a_{8,8}} g(a_{1,1}, \ldots, a_{8,8})$$

Use of loopy belief propagation for approximating Z?

The permanent of a matrix

Determinant vs. Permanent of a Matrix

Consider the matrix $\boldsymbol{\theta} = \begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \\ \theta_{31} & \theta_{32} & \theta_{33} \end{pmatrix}$.

Determinant vs. Permanent of a Matrix

Consider the matrix $\boldsymbol{\theta} = \begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \\ \theta_{31} & \theta_{32} & \theta_{33} \end{pmatrix}$.

The determinant of $\boldsymbol{\theta}$:

$$\begin{aligned} \det(\boldsymbol{\theta}) = &+\theta_{11}\theta_{22}\theta_{33} + \theta_{12}\theta_{23}\theta_{31} + \theta_{13}\theta_{21}\theta_{32} \\ &- \theta_{11}\theta_{23}\theta_{32} - \theta_{12}\theta_{21}\theta_{33} - \theta_{13}\theta_{22}\theta_{31}. \end{aligned}$$

Determinant vs. Permanent of a Matrix

Consider the matrix $\boldsymbol{\theta} = \begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \\ \theta_{31} & \theta_{32} & \theta_{33} \end{pmatrix}$.

The determinant of $\boldsymbol{\theta}$:

$$\begin{aligned} \det(\boldsymbol{\theta}) = &+\theta_{11}\theta_{22}\theta_{33} + \theta_{12}\theta_{23}\theta_{31} + \theta_{13}\theta_{21}\theta_{32} \\ &- \theta_{11}\theta_{23}\theta_{32} - \theta_{12}\theta_{21}\theta_{33} - \theta_{13}\theta_{22}\theta_{31}. \end{aligned}$$

The permanent of $\boldsymbol{\theta}$:

$$\begin{aligned} \operatorname{perm}(\boldsymbol{\theta}) = &+ \theta_{11}\theta_{22}\theta_{33} + \theta_{12}\theta_{23}\theta_{31} + \theta_{13}\theta_{21}\theta_{32} \\ &+ \theta_{11}\theta_{23}\theta_{32} + \theta_{12}\theta_{21}\theta_{33} + \theta_{13}\theta_{22}\theta_{31}. \end{aligned}$$

Determinant vs. Permanent of a Matrix

The determinant of an $n \times n$-matrix $\boldsymbol{\theta}$

$$\det(\boldsymbol{\theta}) = \sum_{\sigma} \operatorname{sgn}(\sigma) \prod_{i \in [n]} \theta_{i,\sigma(i)}.$$

where the sum is over all $n!$ permutations of the set $[n] \triangleq \{1, \ldots, n\}$.

Determinant vs. Permanent of a Matrix

The determinant of an $n \times n$-matrix $\boldsymbol{\theta}$

$$\det(\boldsymbol{\theta}) = \sum_{\sigma} \operatorname{sgn}(\sigma) \prod_{i \in [n]} \theta_{i,\sigma(i)}.$$

where the sum is over all $n!$ permutations of the set $[n] \triangleq \{1, \ldots, n\}$.

The permanent of an $n \times n$-matrix $\boldsymbol{\theta}$:

$$\operatorname{perm}(\boldsymbol{\theta}) = \sum_{\sigma} \prod_{i \in [n]} \theta_{i,\sigma(i)}.$$

Determinant vs. Permanent of a Matrix

The determinant of an $n \times n$-matrix $\boldsymbol{\theta}$

$$\det(\boldsymbol{\theta}) = \sum_{\sigma} \operatorname{sgn}(\sigma) \prod_{i \in [n]} \theta_{i,\sigma(i)}.$$

where the sum is over all $n!$ permutations of the set $[n] \triangleq \{1, \ldots, n\}$.

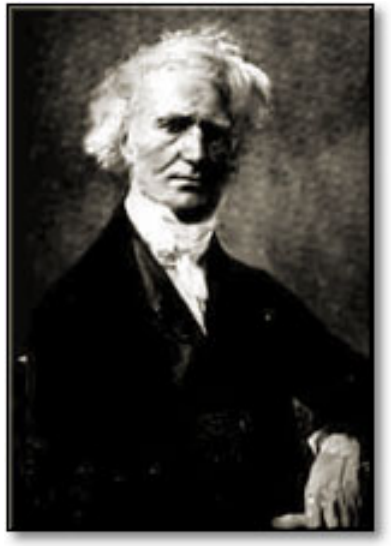
The permanent of an $n \times n$-matrix $\boldsymbol{\theta}$:

$$\operatorname{perm}(\boldsymbol{\theta}) = \sum_{\sigma} \prod_{i \in [n]} \theta_{i,\sigma(i)}.$$

The permanent turns up in a variety of contexts, especially in combinatorial problems, statistical physics (partition function), ...

Historical Remarks

In 1812, Binet and Cauchy **independently** introduced functions that are nowadays called permanents.



G. P. M. Binet, "Mémoire sur un système de formules analytiques, et leur application à des considrations géométriques," Journal de l'École Polytechnique, Paris 9, pp. 280–302, 1812.

L. A. Cauchy, "Mémoire sur les fonctions qui ne peuvent obtenir que deux valeurs égales et de signes contraires par suite des transpositions opérées entre les variables qu'elles renferment," Journal de l'École Polytechnique, Paris 10, pp. 29–112, 1812.

Exactly Computing the Permanent

Exactly Computing the Permanent

- Brute-force computation:

$$O(n \cdot n!) = O\left(n^{3/2} \cdot (n/e)^n\right) \quad \text{arithmetic operations.}$$

Exactly Computing the Permanent

- Brute-force computation:

$$O(n \cdot n!) = O\left(n^{3/2} \cdot (n/e)^n\right) \quad \text{arithmetic operations.}$$

- Ryser's algorithm:

$$\Theta(n \cdot 2^n) \quad \text{arithmetic operations.}$$

Exactly Computing the Permanent

- Brute-force computation:

$$O(n \cdot n!) = O\left(n^{3/2} \cdot (n/e)^n\right) \quad \text{arithmetic operations.}$$

- Ryser's algorithm:

$$\Theta(n \cdot 2^n) \quad \text{arithmetic operations.}$$

- Complexity class [Valiant, 1979]:

$$\#\mathrm{P} \quad (\text{“sharp P” or “number P”}),$$

where #P is the set of the counting problems associated with the decision problems in the set NP. (Note that even the computation of the permanent of zero-one matrices is #P-complete.)

Estimating the Permanent

More efficient algorithms are possible if one does not want to compute the permanent of a matrix exactly.

Estimating the Permanent

More efficient algorithms are possible if one does not want to compute the permanent of a matrix exactly.

- For a matrix that contains positive and negative entries:

  $\rightarrow$ "constructive and destructive interference of terms in the summation."

Estimating the Permanent

More efficient algorithms are possible if one does not want to compute the permanent of a matrix exactly.

- For a matrix that contains positive and negative entries:

  $\rightarrow$ "constructive and destructive interference of terms in the summation."

- For a matrix that contains only non-negative entries:

  $\rightarrow$ "constructive interference of terms in the summation."

Estimating the Permanent

FROM NOW ON: we focus on the case where all entries of the matrix are non-negative, i.e.

$$\theta_{ij} \geq 0 \quad \forall i, j.$$

Estimating the Permanent

FROM NOW ON: we focus on the case where all entries of the matrix are non-negative, i.e.

$$\theta_{ij} \geq 0 \quad \forall i, j.$$

- Markov chain Monte Carlo based methods: [Broder, 1986], ...

Estimating the Permanent

FROM NOW ON: we focus on the case where all entries of the matrix are non-negative, i.e.

$$\theta_{ij} \geq 0 \quad \forall i, j.$$

- Markov chain Monte Carlo based methods: [Broder, 1986], ...
- Godsil-Gutman formula based methods: [Karmarkar et al., 1993], [Barvinok, 1997ff.], [Chien, Rasmussen, Sinclair, 2004], ...

Estimating the Permanent

FROM NOW ON: we focus on the case where all entries of the matrix are non-negative, i.e.

$$\theta_{ij} \geq 0 \quad \forall i, j.$$

- Markov chain Monte Carlo based methods: [Broder, 1986], ...
- Godsil-Gutman formula based methods: [Karmarkar et al., 1993], [Barvinok, 1997ff.], [Chien, Rasmussen, Sinclair, 2004], ...
- Fully polynomial-time randomized approximation schemes (FPRAS): [Jerrum, Sinclair, Vigoda, 2004], ...

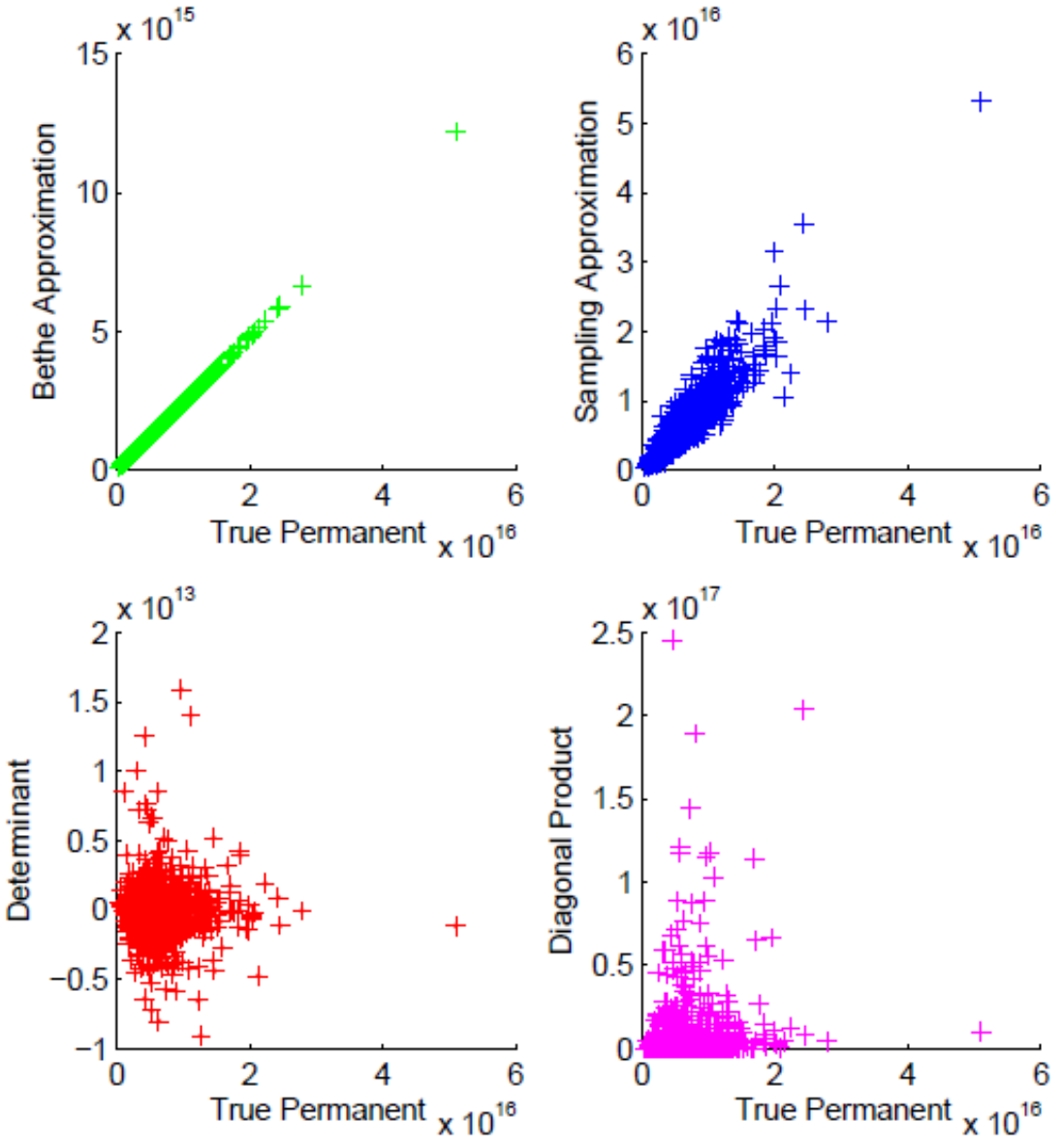
Estimating the Permanent

FROM NOW ON: we focus on the case where all entries of the matrix are non-negative, i.e.

$$\theta_{ij} \geq 0 \quad \forall i, j.$$

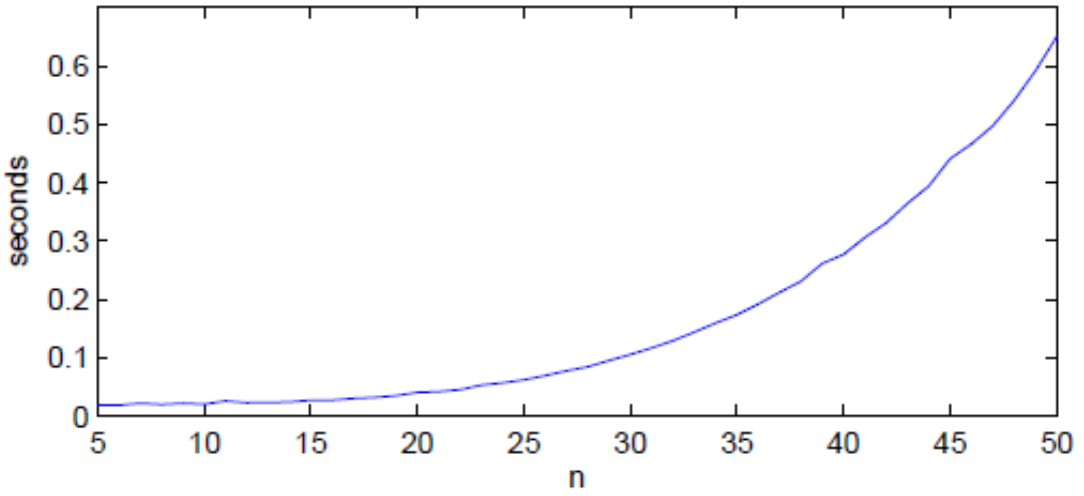
- Markov chain Monte Carlo based methods: [Broder, 1986], ...
- Godsil-Gutman formula based methods: [Karmarkar et al., 1993], [Barvinok, 1997ff.], [Chien, Rasmussen, Sinclair, 2004], ...
- Fully polynomial-time randomized approximation schemes (FPRAS): [Jerrum, Sinclair, Vigoda, 2004], ...
- Bethe-approximation-based / sum-product-algorithm-based methods: [Chertkov et al., 2008], [Huang and Jebara, 2009], ...

Estimating the Permanent

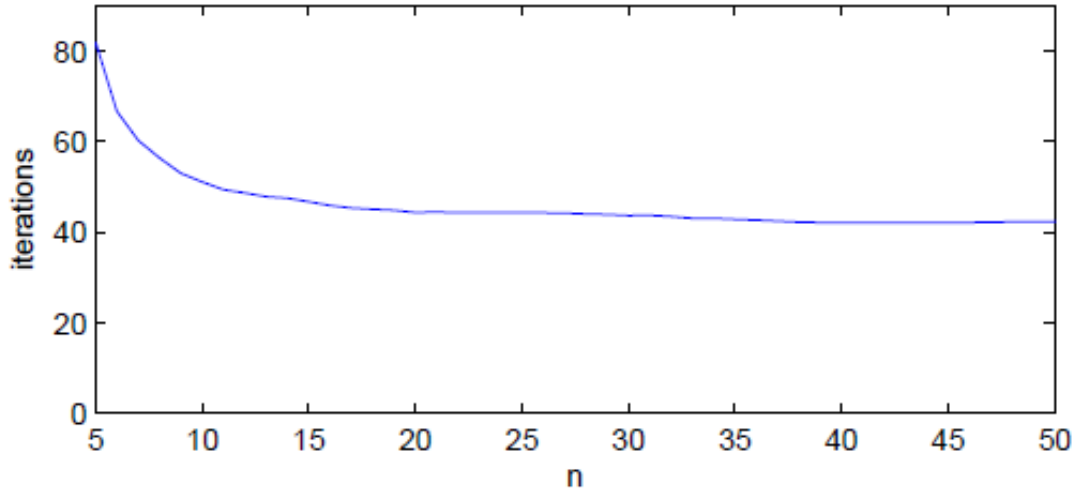


From [Huang/Jebara, 2009].

Estimating the Permanent



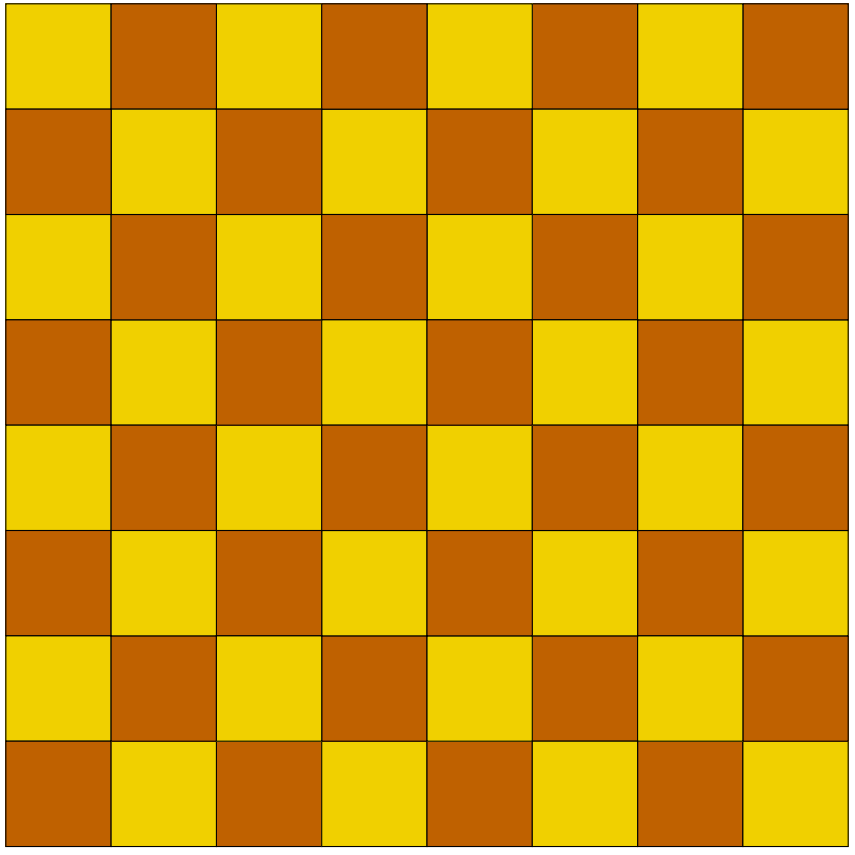
(a) Running time



(b) Iterations

From [Huang/Jebara, 2009].

Valid Rook Configs. and Permanents

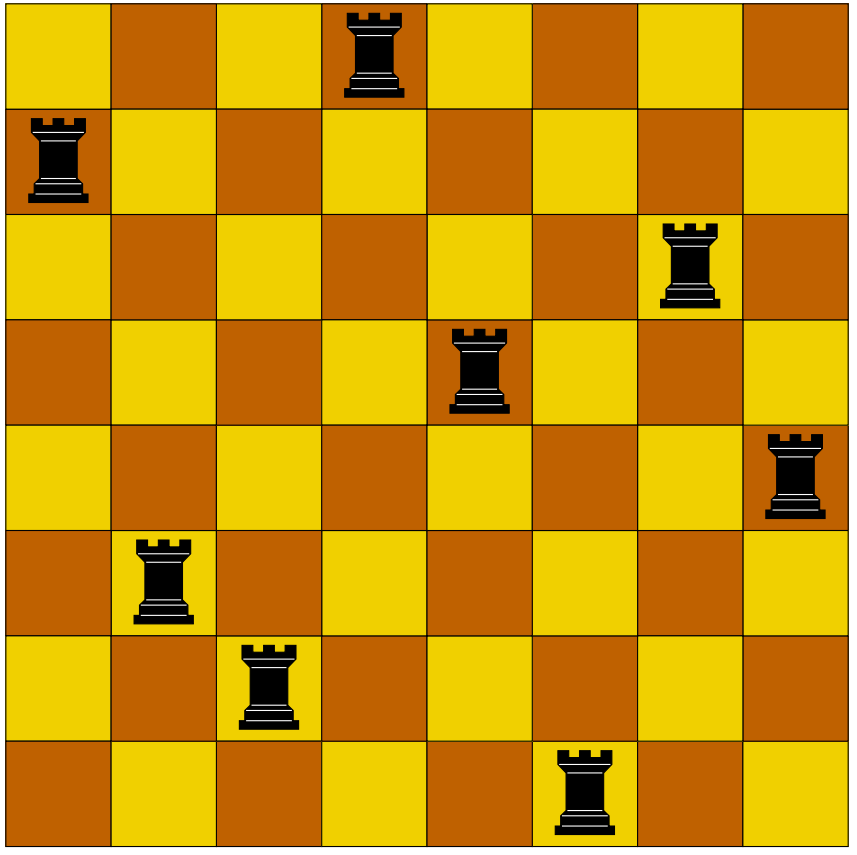


Number of valid rook configurations

$$= \operatorname{perm} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$= \sum_{\sigma} \prod_{i \in [8]} \theta_{i,\sigma(i)}$$

Valid Rook Configs. and Permanents

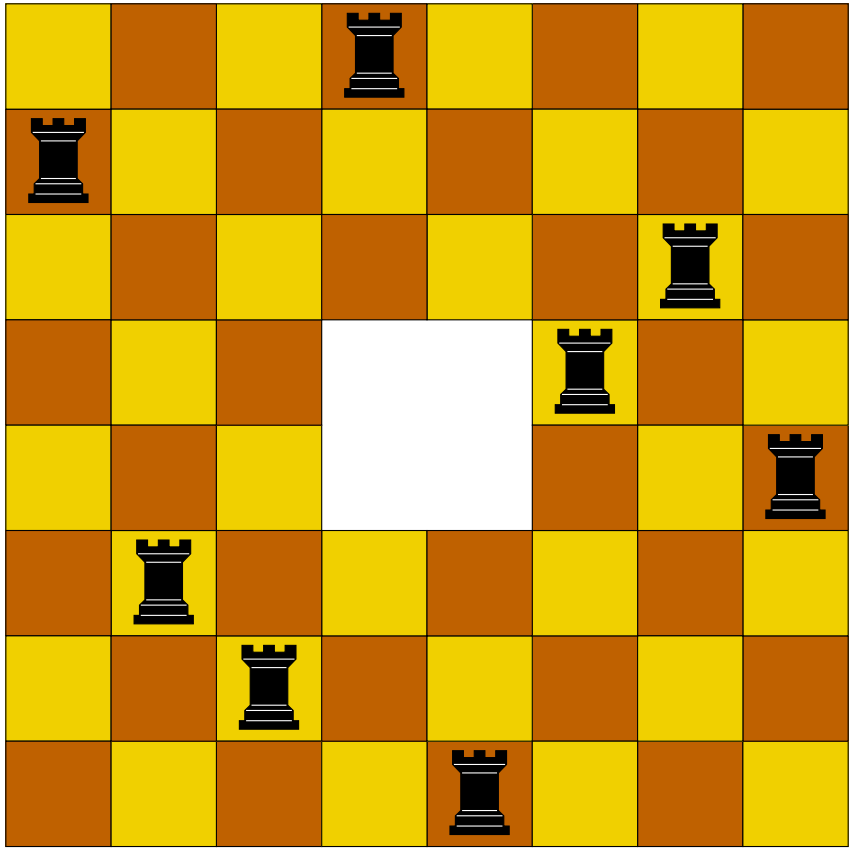


Number of valid rook configurations

$$= \operatorname{perm} \begin{pmatrix} 1 & 1 & 1 & \mathbf{1} & 1 & 1 & 1 & 1 \\ \mathbf{1} & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \mathbf{1} & 1 \\ 1 & 1 & 1 & 1 & \mathbf{1} & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & \mathbf{1} \\ 1 & \mathbf{1} & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & \mathbf{1} & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & \mathbf{1} & 1 & 1 \end{pmatrix}$$

$$= \sum_{\sigma} \prod_{i \in [8]} \theta_{i,\sigma(i)}$$

Valid Rook Configs. and Permanents

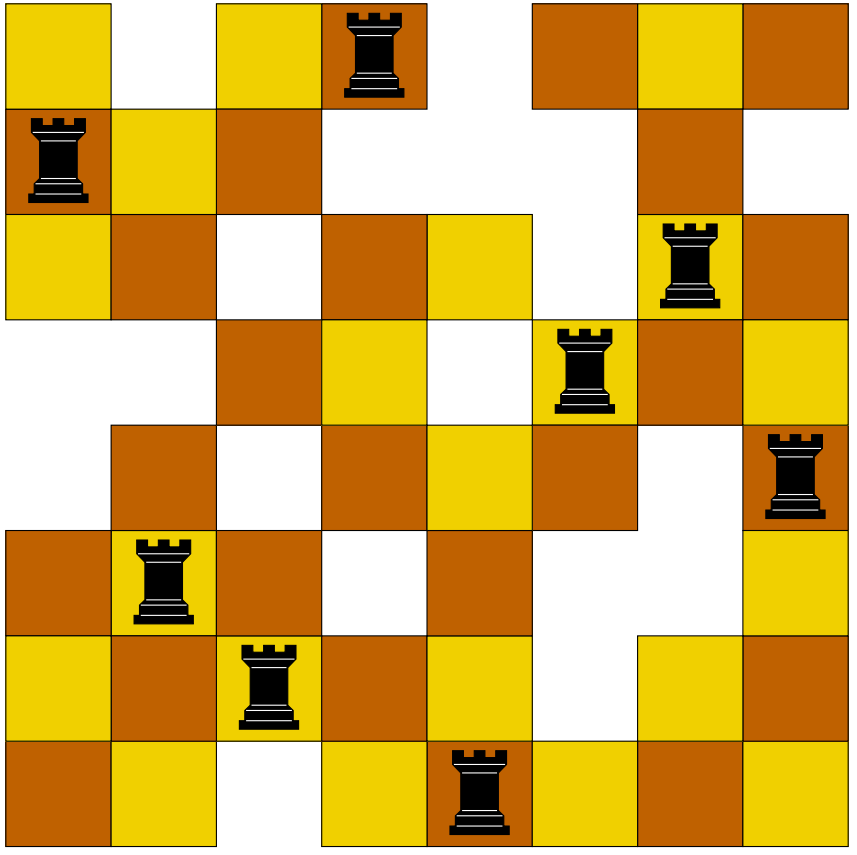


Number of valid rook configurations

$$= \operatorname{perm} \begin{pmatrix} 1 & 1 & 1 & \mathbf{1} & 1 & 1 & 1 & 1 \\ \mathbf{1} & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \mathbf{1} & 1 \\ 1 & 1 & 1 & 0 & 0 & \mathbf{1} & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & \mathbf{1} \\ 1 & \mathbf{1} & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & \mathbf{1} & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & \mathbf{1} & 1 & 1 & 1 \end{pmatrix}$$

$$= \sum_{\sigma} \prod_{i \in [8]} \theta_{i,\sigma(i)}$$

Valid Rook Configs. and Permanents

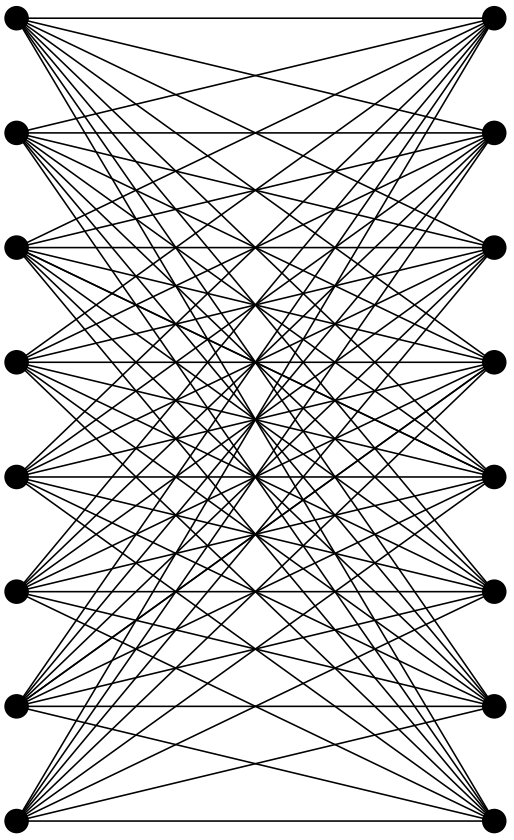


Number of valid rook configurations

$$= \operatorname{perm} \begin{pmatrix} 1 & 0 & 1 & \mathbf{1} & 0 & 1 & 1 & 1 \\ \mathbf{1} & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & \mathbf{1} & 1 \\ 0 & 0 & 1 & 1 & 0 & \mathbf{1} & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & \mathbf{1} \\ 1 & \mathbf{1} & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & \mathbf{1} & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & \mathbf{1} & 1 & 1 & 1 \end{pmatrix}$$

$$= \sum_{\sigma} \prod_{i \in [8]} \theta_{i,\sigma(i)}$$

Perfect Matchings and Permanents

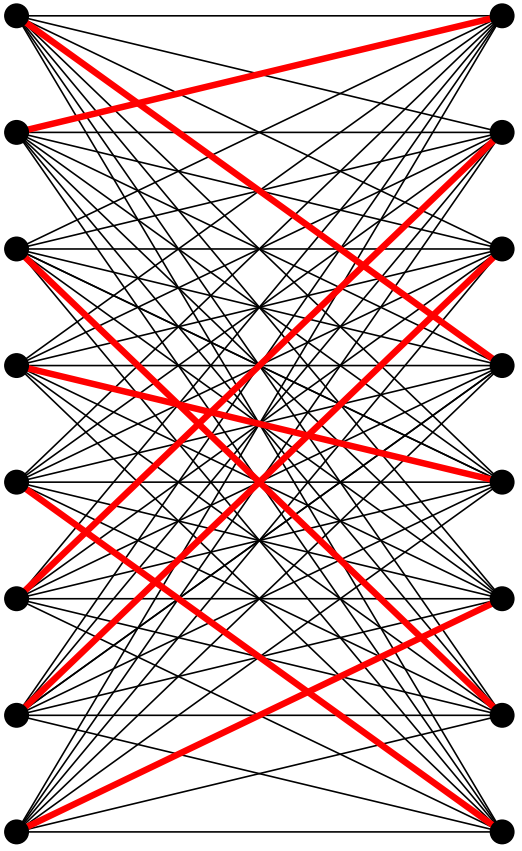


Number of perfect matchings

$$= \operatorname{perm} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$= \sum_{\sigma} \prod_{i \in [8]} \theta_{i,\sigma(i)}$$

Perfect Matchings and Permanents

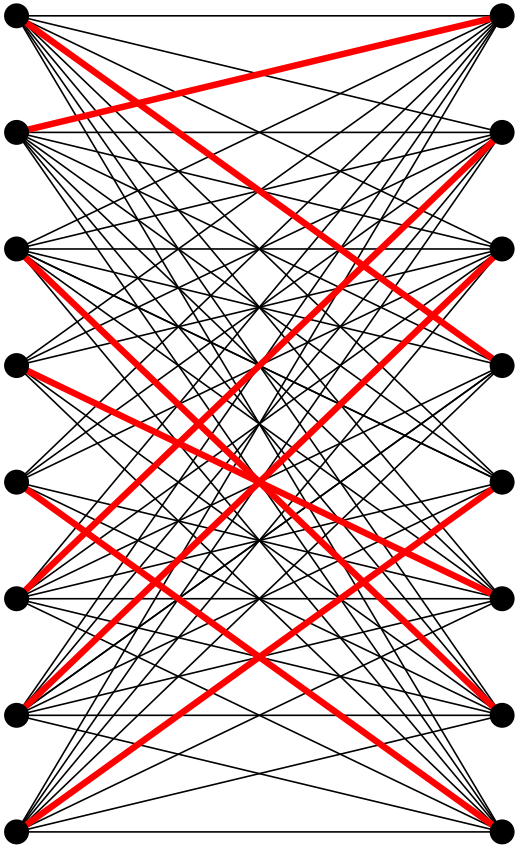


Number of perfect matchings

$$= \operatorname{perm} \begin{pmatrix} 1 & 1 & 1 & \mathbf{1} & 1 & 1 & 1 & 1 \\ \mathbf{1} & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \mathbf{1} & 1 \\ 1 & 1 & 1 & 1 & \mathbf{1} & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & \mathbf{1} \\ 1 & \mathbf{1} & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & \mathbf{1} & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & \mathbf{1} & 1 & 1 \end{pmatrix}$$

$$= \sum_{\sigma} \prod_{i \in [8]} \theta_{i,\sigma(i)}$$

Perfect Matchings and Permanents

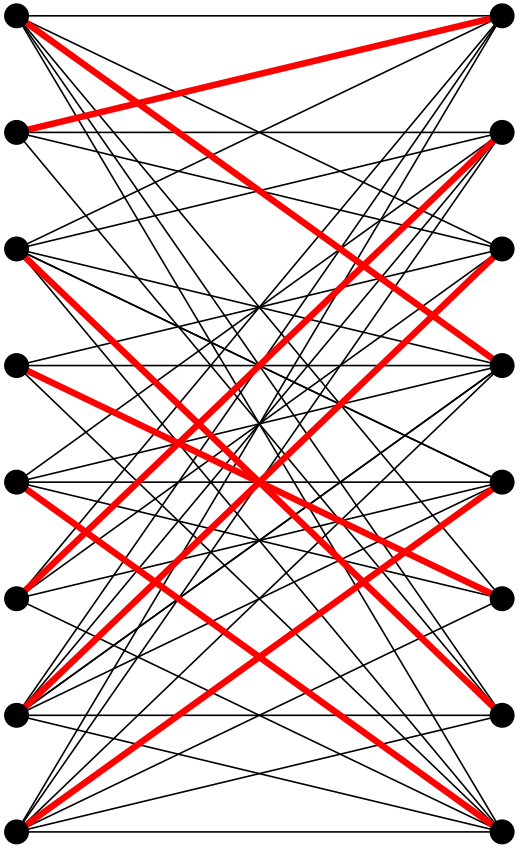


Number of perfect matchings

$$= \operatorname{perm} \begin{pmatrix} 1 & 1 & 1 & \mathbf{1} & 1 & 1 & 1 & 1 \\ \mathbf{1} & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & \mathbf{1} & 1 \\ 1 & 1 & 1 & 0 & 0 & \mathbf{1} & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & \mathbf{1} \\ 1 & \mathbf{1} & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & \mathbf{1} & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & \mathbf{1} & 1 & 1 & 1 \end{pmatrix}$$

$$= \sum_{\sigma} \prod_{i \in [8]} \theta_{i,\sigma(i)}$$

Perfect Matchings and Permanents

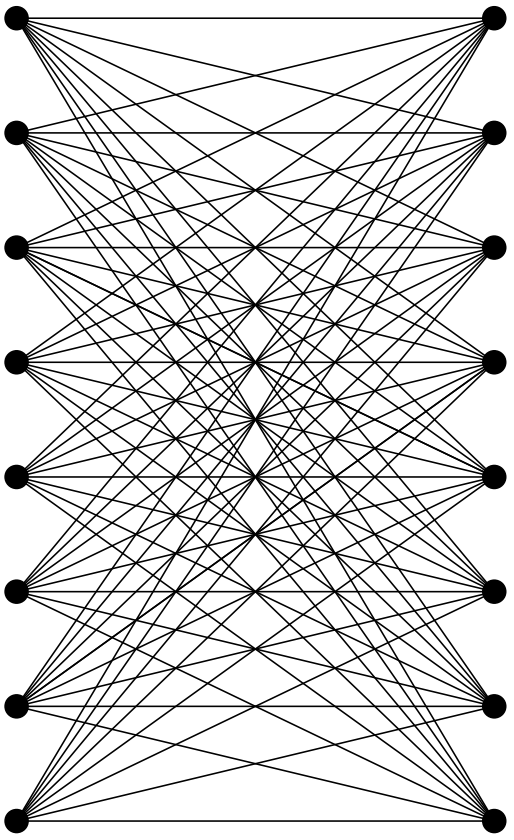


Number of perfect matchings

$$= \operatorname{perm} \begin{pmatrix} 1 & 0 & 1 & \mathbf{1} & 0 & 1 & 1 & 1 \\ \mathbf{1} & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & \mathbf{1} & 1 \\ 0 & 0 & 1 & 1 & 0 & \mathbf{1} & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & \mathbf{1} \\ 1 & \mathbf{1} & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & \mathbf{1} & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & \mathbf{1} & 1 & 1 & 1 \end{pmatrix}$$

$$= \sum_{\sigma} \prod_{i \in [8]} \theta_{i,\sigma(i)}$$

Perfect Matchings and Permanents

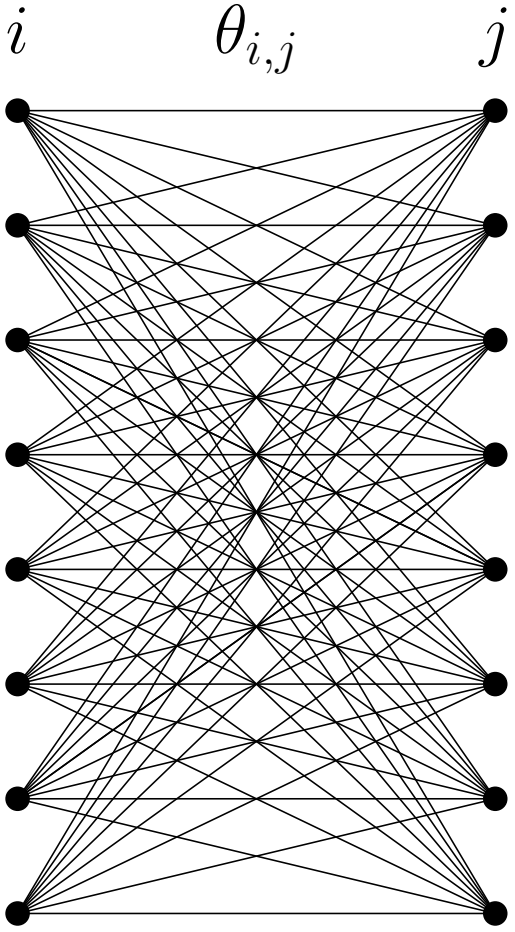


Number of perfect matchings

$$= \operatorname{perm} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$= \sum_{\sigma} \prod_{i \in [8]} \theta_{i,\sigma(i)}$$

Perfect Matchings and Permanents

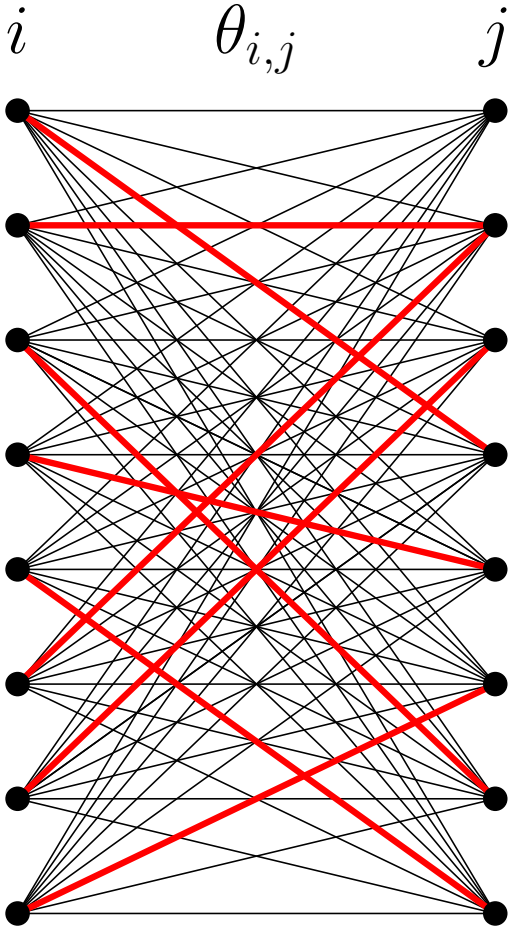


Total sum of weighted perf. matchings

$$= \operatorname{perm} \begin{pmatrix} \theta_{11}\,\theta_{12}\,\theta_{13}\,\theta_{14}\,\theta_{15}\,\theta_{16}\,\theta_{17}\,\theta_{18} \\ \theta_{21}\,\theta_{22}\,\theta_{23}\,\theta_{24}\,\theta_{25}\,\theta_{26}\,\theta_{27}\,\theta_{28} \\ \theta_{31}\,\theta_{32}\,\theta_{33}\,\theta_{34}\,\theta_{35}\,\theta_{36}\,\theta_{37}\,\theta_{38} \\ \theta_{41}\,\theta_{42}\,\theta_{43}\,\theta_{44}\,\theta_{45}\,\theta_{46}\,\theta_{47}\,\theta_{48} \\ \theta_{51}\,\theta_{52}\,\theta_{53}\,\theta_{54}\,\theta_{55}\,\theta_{56}\,\theta_{57}\,\theta_{58} \\ \theta_{61}\,\theta_{62}\,\theta_{63}\,\theta_{64}\,\theta_{65}\,\theta_{66}\,\theta_{67}\,\theta_{68} \\ \theta_{71}\,\theta_{72}\,\theta_{73}\,\theta_{74}\,\theta_{75}\,\theta_{76}\,\theta_{77}\,\theta_{78} \\ \theta_{81}\,\theta_{82}\,\theta_{83}\,\theta_{84}\,\theta_{85}\,\theta_{86}\,\theta_{87}\,\theta_{88} \end{pmatrix}$$

$$= \sum_{\sigma} \prod_{i \in [8]} \theta_{i,\sigma(i)}$$

Perfect Matchings and Permanents

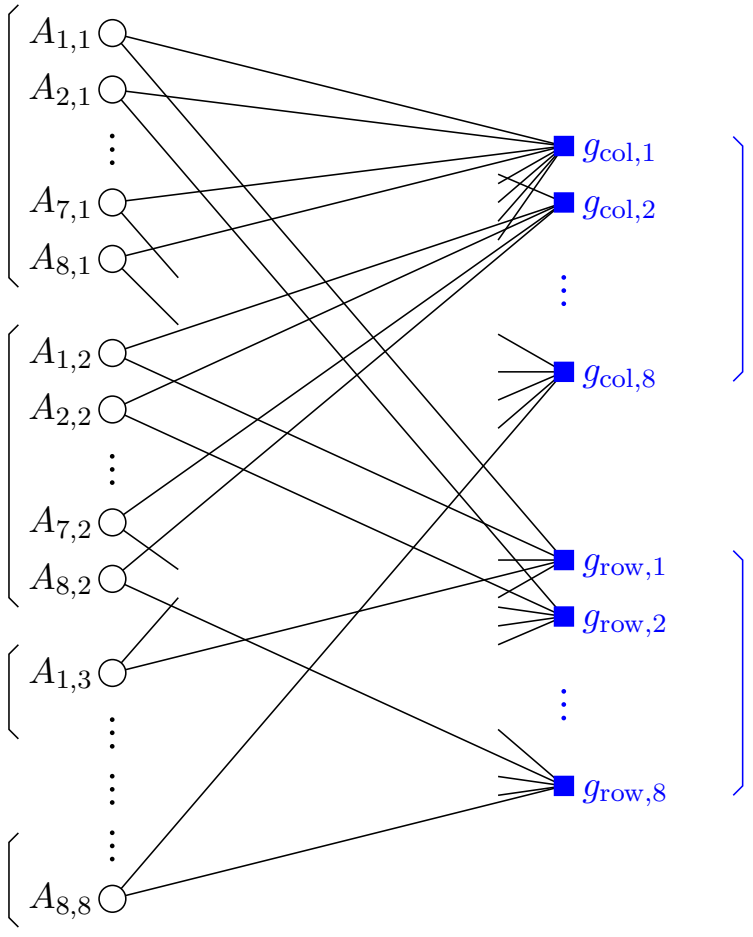


Total sum of weighted perf. matchings

$$= \operatorname{perm} \begin{pmatrix} \theta_{11} & \theta_{12} & \theta_{13} & \theta_{14} & \theta_{15} & \theta_{16} & \theta_{17} & \theta_{18} \\ \theta_{21} & \theta_{22} & \theta_{23} & \theta_{24} & \theta_{25} & \theta_{26} & \theta_{27} & \theta_{28} \\ \theta_{31} & \theta_{32} & \theta_{33} & \theta_{34} & \theta_{35} & \theta_{36} & \theta_{37} & \theta_{38} \\ \theta_{41} & \theta_{42} & \theta_{43} & \theta_{44} & \theta_{45} & \theta_{46} & \theta_{47} & \theta_{48} \\ \theta_{51} & \theta_{52} & \theta_{53} & \theta_{54} & \theta_{55} & \theta_{56} & \theta_{57} & \theta_{58} \\ \theta_{61} & \theta_{62} & \theta_{63} & \theta_{64} & \theta_{65} & \theta_{66} & \theta_{67} & \theta_{68} \\ \theta_{71} & \theta_{72} & \theta_{73} & \theta_{74} & \theta_{75} & \theta_{76} & \theta_{77} & \theta_{78} \\ \theta_{81} & \theta_{82} & \theta_{83} & \theta_{84} & \theta_{85} & \theta_{86} & \theta_{87} & \theta_{88} \end{pmatrix}$$

$$= \sum_{\sigma} \prod_{i \in [8]} \theta_{i,\sigma(i)}$$

Graphical Model for Permanent



(function nodes are suitably defined based on $\boldsymbol{\theta}$)

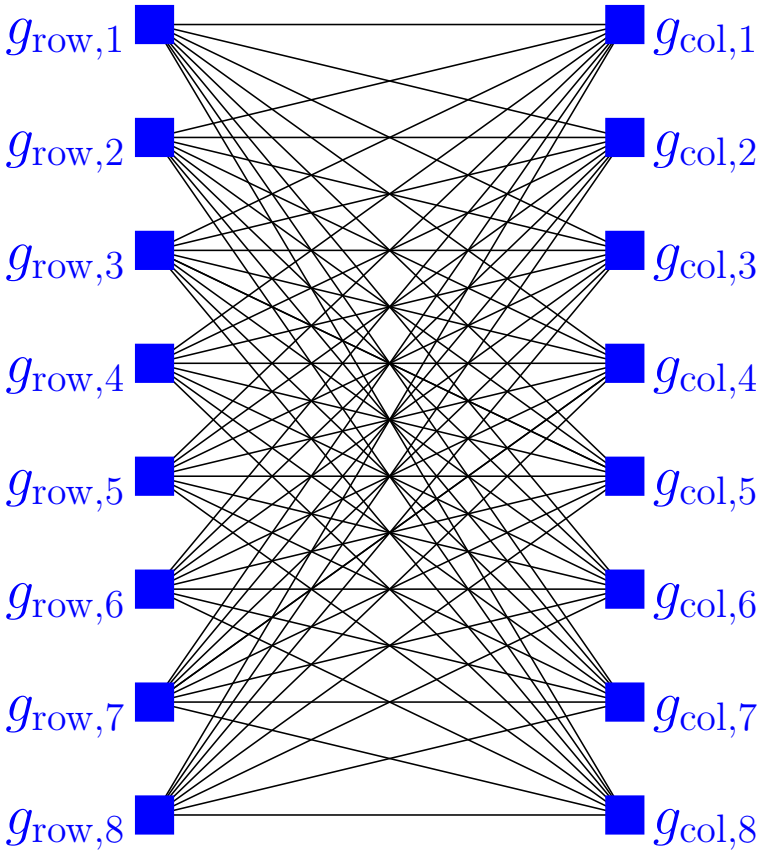
Global function:

$$g(a_{1,1}, \ldots, a_{8,8}) = \prod_j g_{\text{col},j}(a_{1,j}, \ldots, a_{8,j}) \times \prod_i g_{\text{row},i}(a_{i,1}, \ldots, a_{i,8})$$

Permanent:

$$\text{perm}(\boldsymbol{\theta}) = Z = \sum_{a_{1,1},\ldots,a_{8,8}} g(a_{1,1}, \ldots, a_{8,8})$$

Graphical Model for Permanent



(function nodes are suitably defined based on $\boldsymbol{\theta}$)

(variable nodes have been omitted)

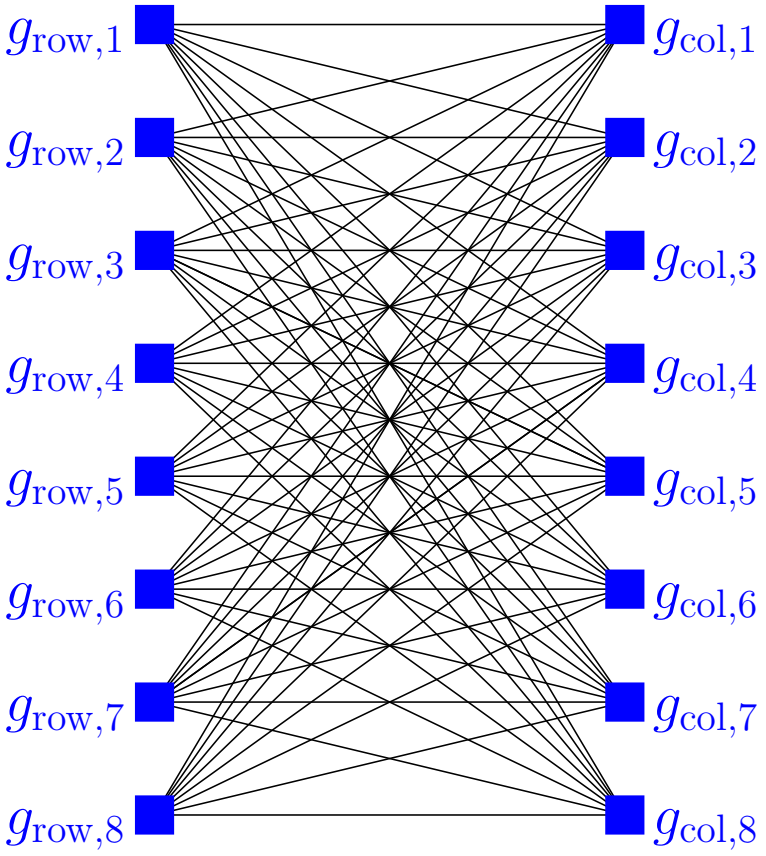
Global function:

$$g(a_{1,1}, \ldots, a_{8,8}) = \prod_j g_{\text{col},j}(a_{1,j}, \ldots, a_{8,j}) \times \prod_i g_{\text{row},i}(a_{i,1}, \ldots, a_{i,8})$$

Permanent:

$$\text{perm}(\boldsymbol{\theta}) = Z = \sum_{a_{1,1},\ldots,a_{8,8}} g(a_{1,1}, \ldots, a_{8,8})$$

Graphical Model for Permanent

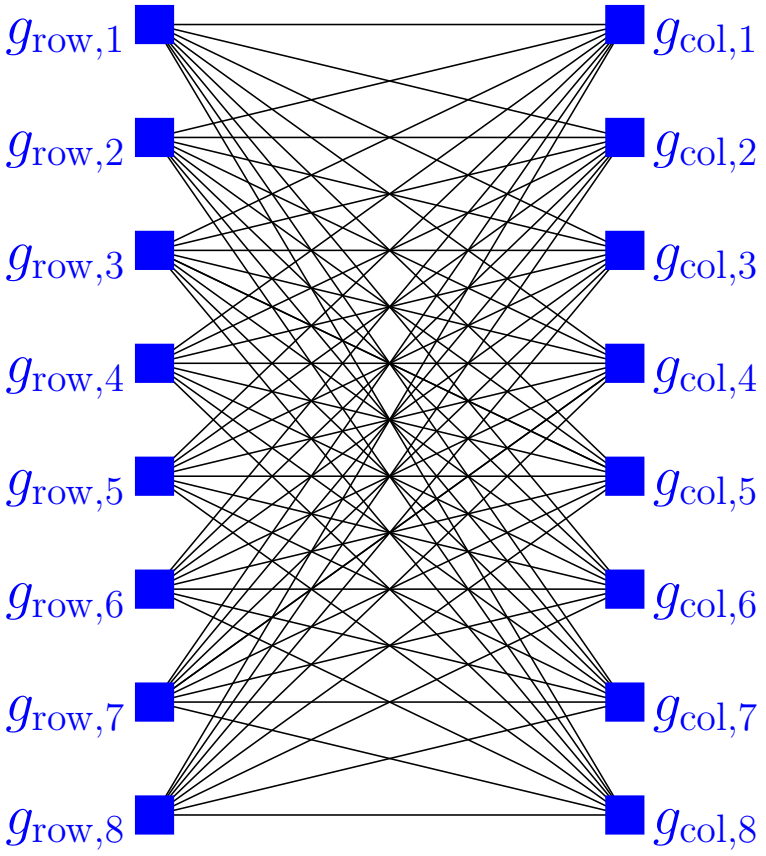


(function nodes are suitably defined based on $\boldsymbol{\theta}$)

(variable nodes have been omitted)

- Many short cycles.
- The vertex degrees are high.

Graphical Model for Permanent



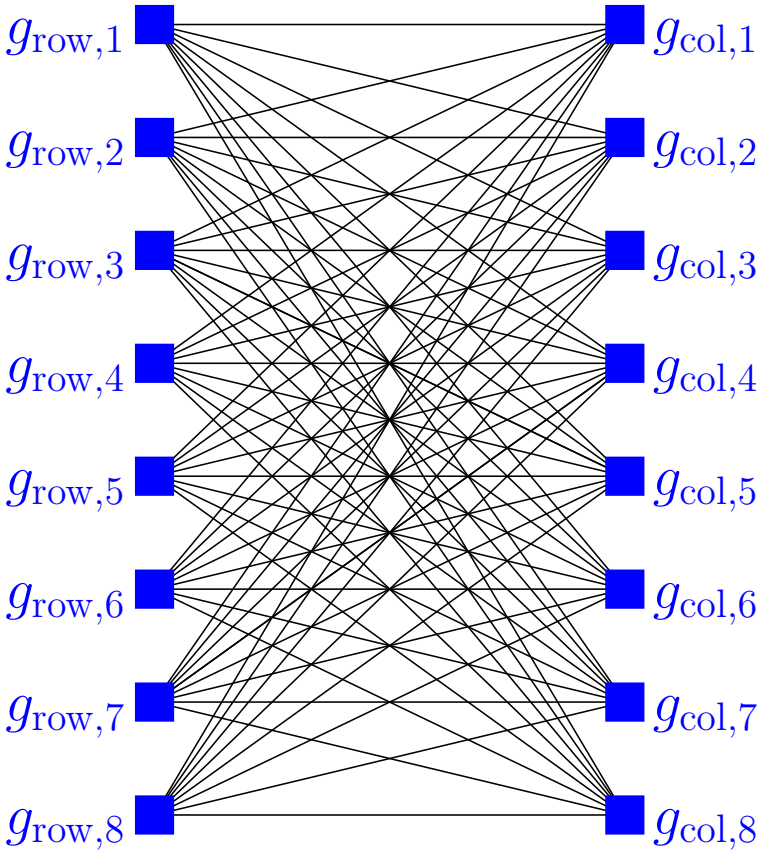
(function nodes are suitably defined based on $\boldsymbol{\theta}$)

(variable nodes have been omitted)

- Many short cycles.
- The vertex degrees are high.

Both facts might suggest that the application of the sum-product algorithm to this factor graph is rather problematic.

Graphical Model for Permanent



(function nodes are suitably defined based on $\boldsymbol{\theta}$)

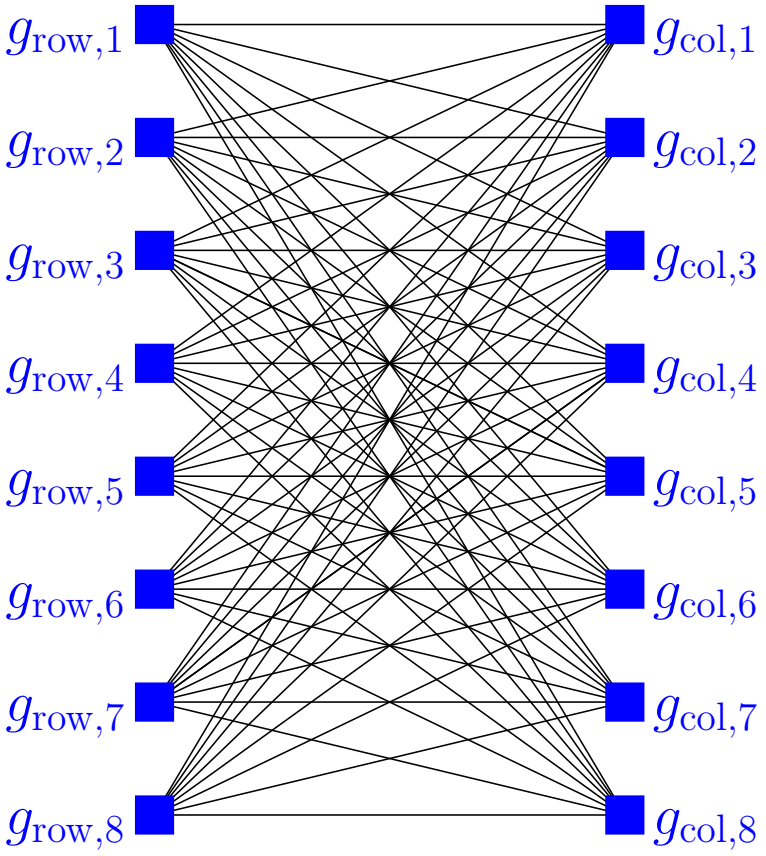
(variable nodes have been omitted)

- Many short cycles.
- The vertex degrees are high.

Both facts might suggest that the application of the sum-product algorithm to this factor graph is rather problematic.

However, luckily this is not the case.

Graphical Model for Permanent



(function nodes are suitably defined based on $\boldsymbol{\theta}$)

(variable nodes have been omitted)

- Many short cycles.
- The vertex degrees are high.

Both facts might suggest that the application of the sum-product algorithm to this factor graph is rather problematic.

However, luckily this is not the case.

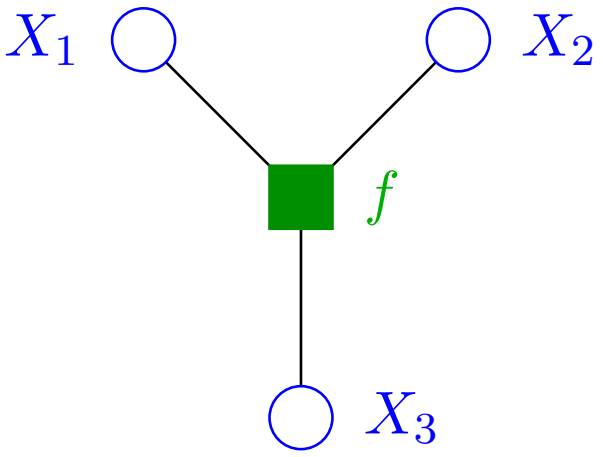
For an SPA suitability assessment, the overall cycle structure and the types of functions nodes are at least as important.

Factor graphs and the sum-product algorithm

Factor Graphs

A factor graph can be used to represent a multivariate function:

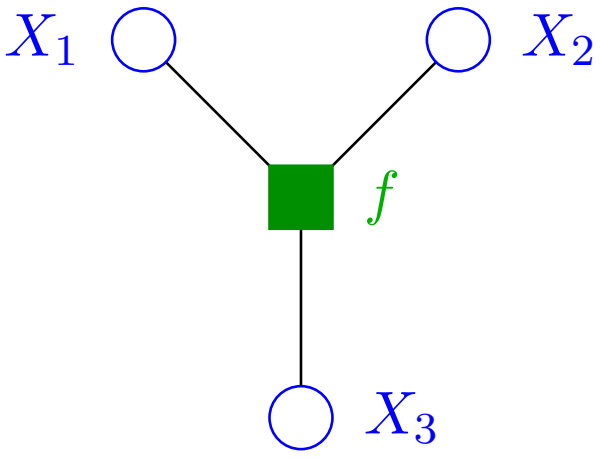
$$f(x_1, x_2, x_3)$$



Factor Graphs

A factor graph can be used to represent a multivariate function:

$$f(x_1, x_2, x_3)$$

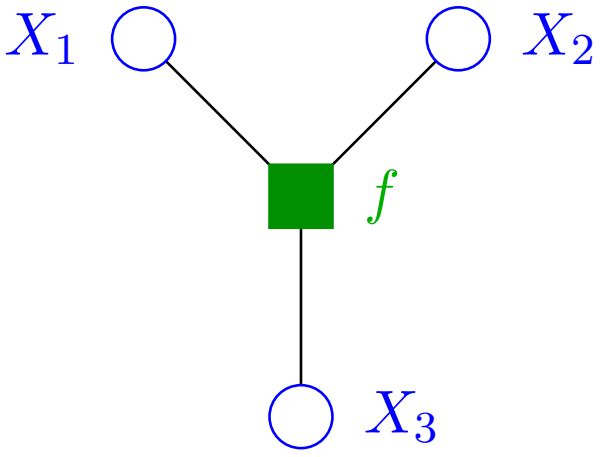


- Variable nodes: for each variable we draw a variable node (empty circles).

Factor Graphs

A factor graph can be used to represent a multivariate function:

$$f(x_1, x_2, x_3)$$

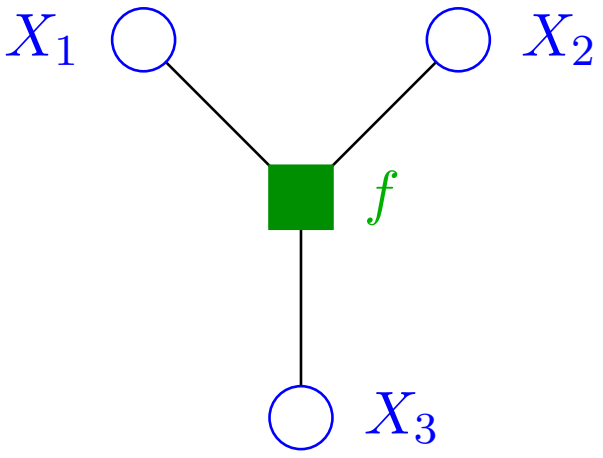


- Variable nodes: for each variable we draw a variable node (empty circles).
- Function nodes: for each function we draw a function node (filled squares).

Factor Graphs

A factor graph can be used to represent a multivariate function:

$$f(x_1, x_2, x_3)$$

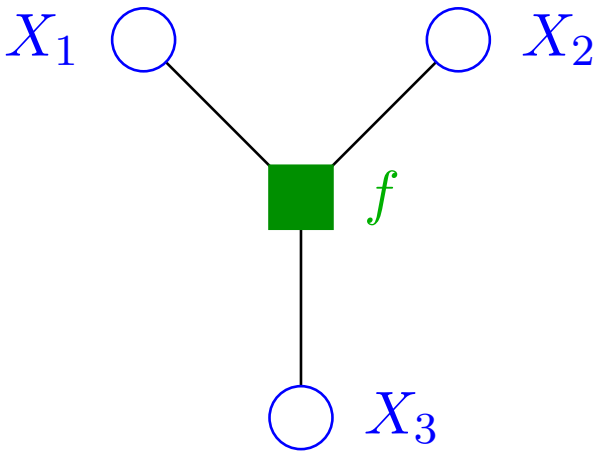


- Variable nodes: for each variable we draw a variable node (empty circles).
- Function nodes: for each function we draw a function node (filled squares).
- Edges: there is an edge between a variable node and a function node if the corresponding variable is an argument of the corresponding function.

Factor Graphs

A factor graph can be used to represent a multivariate function:

$$f(x_1, x_2, x_3)$$



- Variable nodes: for each variable we draw a variable node (empty circles).
- Function nodes: for each function we draw a function node (filled squares).
- Edges: there is an edge between a variable node and a function node if the corresponding variable is an argument of the corresponding function.
- Bipartite graph: the resulting graph is a bipartite graph, i.e. there are only edges between vertices of different types.

Factor Graphs

General references for factor graphs are:

- F. R. Kschischang, B. J. Frey and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” IEEE Trans. Inf. Theory, vol. 47, Feb. 2001.

- H.-A. Loeliger, “An introduction to factor graphs,” IEEE Signal Processing Magazine, Jan. 2004.

Factor Graphs

We assume that we know more about the internal structure of the function $f(x_1, x_2, x_3)$, e.g.

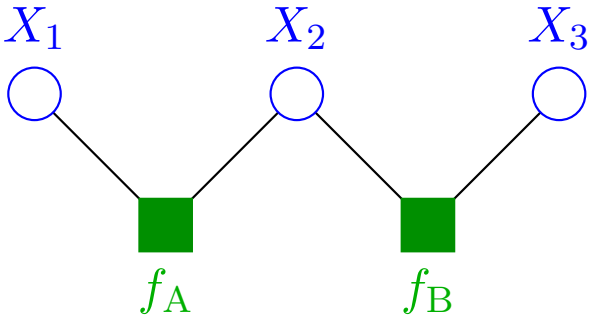
$$f(x_1, x_2, x_3) = f_{\mathrm{A}}(x_1, x_2) \cdot f_{\mathrm{B}}(x_2, x_3).$$

Factor Graphs

We assume that we know more about the internal structure of the function $f(x_1, x_2, x_3)$, e.g.

$$f(x_1, x_2, x_3) = f_{\mathrm{A}}(x_1, x_2) \cdot f_{\mathrm{B}}(x_2, x_3).$$

Then we can take advantage of this fact and the factor graph represents this structure.

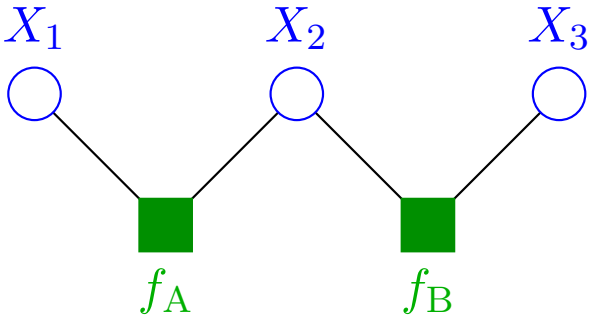


Factor Graphs

We assume that we know more about the internal structure of the function $f(x_1, x_2, x_3)$, e.g.

$$f(x_1, x_2, x_3) = f_{\mathrm{A}}(x_1, x_2) \cdot f_{\mathrm{B}}(x_2, x_3).$$

Then we can take advantage of this fact and the factor graph represents this structure.



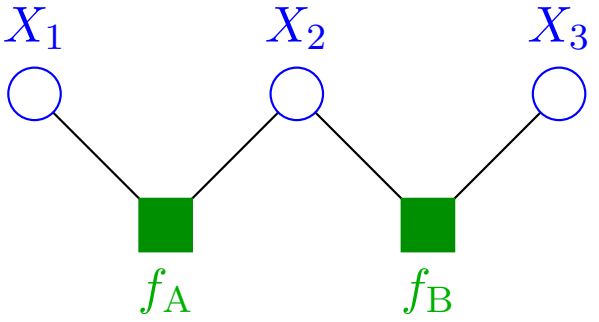
- $f(.,.,.)$ is called the global function.

Factor Graphs

We assume that we know more about the internal structure of the function $f(x_1, x_2, x_3)$, e.g.

$$f(x_1, x_2, x_3) = f_{\mathrm{A}}(x_1, x_2) \cdot f_{\mathrm{B}}(x_2, x_3).$$

Then we can take advantage of this fact and the factor graph represents this structure.



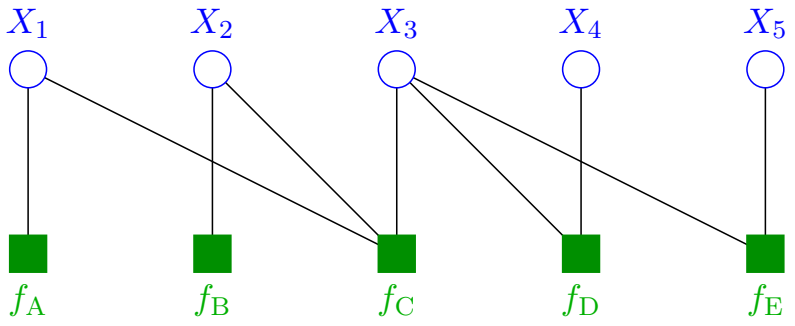
- $f(.,.,.)$ is called the global function.
- $f_{\mathrm{A}}(.,.)$ and $f_{\mathrm{B}}(.,.)$ are called local functions.

Factor Graphs

$$f(x_1, x_2, x_3, x_4, x_5)$$
$$= f_{\mathrm{A}}(x_1) \cdot f_{\mathrm{B}}(x_2) \cdot f_{\mathrm{C}}(x_1, x_2, x_3) \cdot f_{\mathrm{D}}(x_3, x_4) \cdot f_{\mathrm{E}}(x_3, x_5)$$

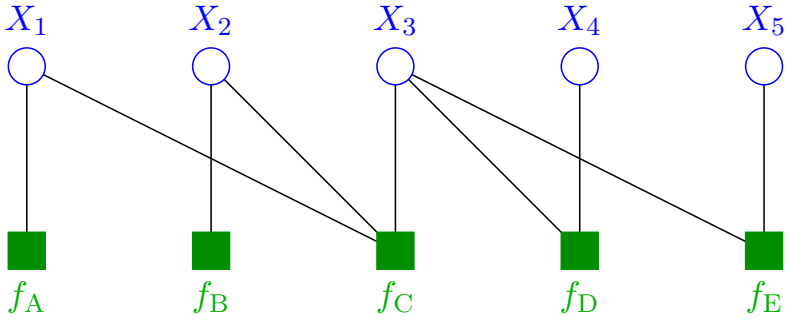
Factor Graphs

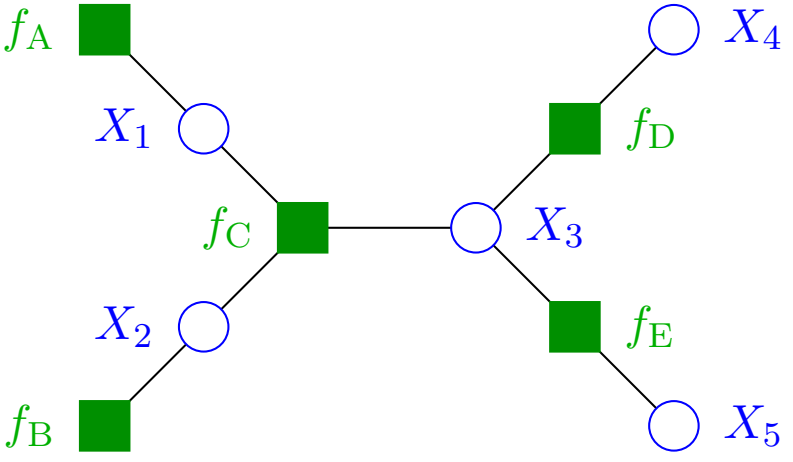
$$f(x_1, x_2, x_3, x_4, x_5)$$
$$= f_{\mathrm{A}}(x_1) \cdot f_{\mathrm{B}}(x_2) \cdot f_{\mathrm{C}}(x_1, x_2, x_3) \cdot f_{\mathrm{D}}(x_3, x_4) \cdot f_{\mathrm{E}}(x_3, x_5)$$



Factor Graphs

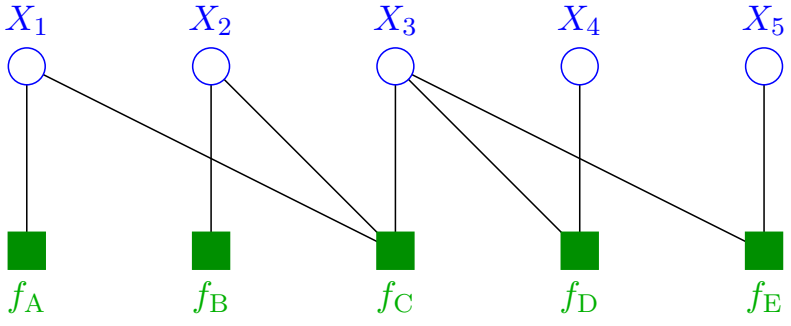
$$f(x_1, x_2, x_3, x_4, x_5)$$
$$= f_{\mathrm{A}}(x_1) \cdot f_{\mathrm{B}}(x_2) \cdot f_{\mathrm{C}}(x_1, x_2, x_3) \cdot f_{\mathrm{D}}(x_3, x_4) \cdot f_{\mathrm{E}}(x_3, x_5)$$

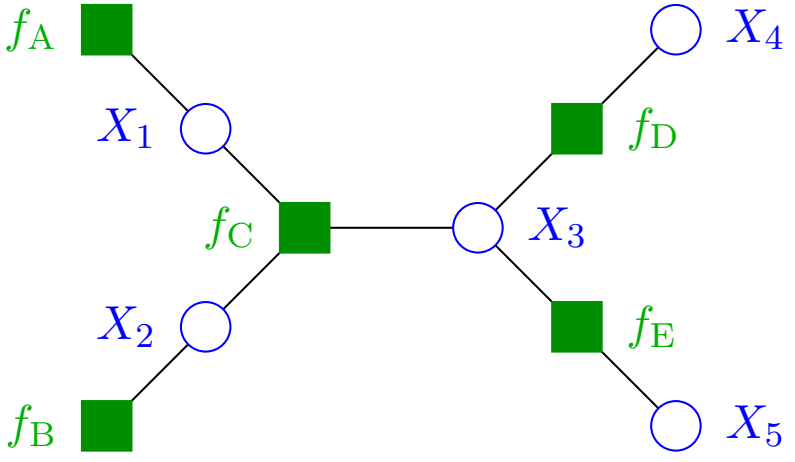




Factor Graphs

$$f(x_1, x_2, x_3, x_4, x_5)$$
$$= f_{\mathrm{A}}(x_1) \cdot f_{\mathrm{B}}(x_2) \cdot f_{\mathrm{C}}(x_1, x_2, x_3) \cdot f_{\mathrm{D}}(x_3, x_4) \cdot f_{\mathrm{E}}(x_3, x_5)$$



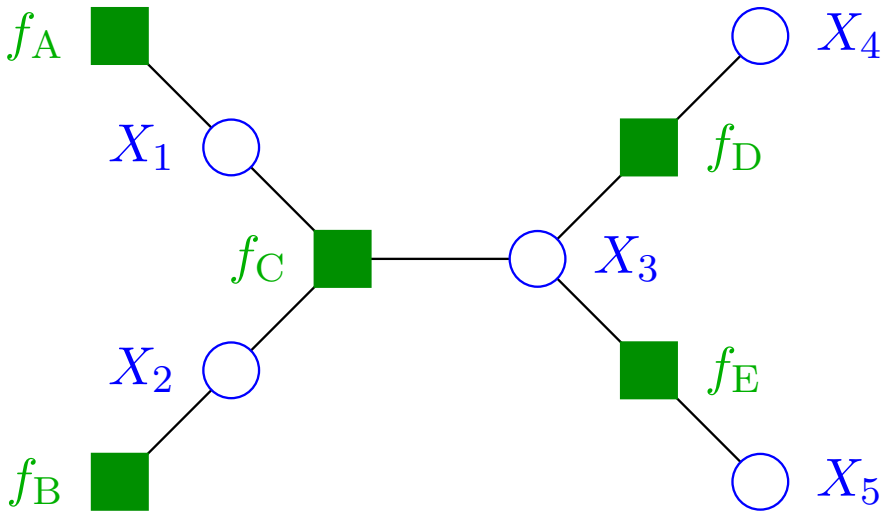


Note: One and the same function can be represented by graphs with different structures: some are more pleasing than others.

The sum-product algorithm

The Sum-Product Algorithm

Let us consider again the following factor graph (which is a tree).



The global function is

$$f(x_1, x_2, x_3, x_4, x_5)$$
$$= f_A(x_1) \cdot f_B(x_2) \cdot f_C(x_1, x_2, x_3) \cdot f_D(x_3, x_4) \cdot f_E(x_3, x_5).$$

The Sum-Product Algorithm

The global function is

$$f(x_1, x_2, x_3, x_4, x_5) = f_{\mathrm{A}}(x_1) \cdot f_{\mathrm{B}}(x_2) \cdot f_{\mathrm{C}}(x_1, x_2, x_3) \cdot f_{\mathrm{D}}(x_3, x_4) \cdot f_{\mathrm{E}}(x_3, x_5).$$

The Sum-Product Algorithm

The global function is

$$f(x_1, x_2, x_3, x_4, x_5) = f_{\mathrm{A}}(x_1) \cdot f_{\mathrm{B}}(x_2) \cdot f_{\mathrm{C}}(x_1, x_2, x_3) \cdot f_{\mathrm{D}}(x_3, x_4) \cdot f_{\mathrm{E}}(x_3, x_5).$$

Very often one wants to calculate marginal functions. E.g.,

$$\begin{aligned} \eta_{X_1}(x_1) &= \sum_{x_2, x_3, x_4, x_5} f(x_1, x_2, x_3, x_4, x_5) \\ &= \sum_{x_2, x_3, x_4, x_5} f_{\mathrm{A}}(x_1) \cdot f_{\mathrm{B}}(x_2) \cdot f_{\mathrm{C}}(x_1, x_2, x_3) \cdot f_{\mathrm{D}}(x_3, x_4) \cdot f_{\mathrm{E}}(x_3, x_5). \end{aligned}$$

The Sum-Product Algorithm

The global function is

$$f(x_1, x_2, x_3, x_4, x_5) = f_{\mathrm{A}}(x_1) \cdot f_{\mathrm{B}}(x_2) \cdot f_{\mathrm{C}}(x_1, x_2, x_3) \cdot f_{\mathrm{D}}(x_3, x_4) \cdot f_{\mathrm{E}}(x_3, x_5).$$

Very often one wants to calculate marginal functions. E.g.,

$$\begin{aligned}
\eta_{X_1}(x_1) &= \sum_{x_2, x_3, x_4, x_5} f(x_1, x_2, x_3, x_4, x_5) \\
&= \sum_{x_2, x_3, x_4, x_5} f_{\mathrm{A}}(x_1) \cdot f_{\mathrm{B}}(x_2) \cdot f_{\mathrm{C}}(x_1, x_2, x_3) \cdot f_{\mathrm{D}}(x_3, x_4) \cdot f_{\mathrm{E}}(x_3, x_5).
\end{aligned}$$

$$\begin{aligned}
\eta_{X_3}(x_3) &= \sum_{x_1, x_2, x_4, x_5} f(x_1, x_2, x_3, x_4, x_5) \\
&= \sum_{x_1, x_2, x_4, x_5} f_{\mathrm{A}}(x_1) \cdot f_{\mathrm{B}}(x_2) \cdot f_{\mathrm{C}}(x_1, x_2, x_3) \cdot f_{\mathrm{D}}(x_3, x_4) \cdot f_{\mathrm{E}}(x_3, x_5).
\end{aligned}$$

etc.

The Sum-Product Algorithm

$$\eta_{X_1}(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_{\mathrm{A}}(x_1) \cdot f_{\mathrm{B}}(x_2) \cdot f_{\mathrm{C}}(x_1, x_2, x_3) \cdot f_{\mathrm{D}}(x_3, x_4) \cdot f_{\mathrm{E}}(x_3, x_5)$$

The Sum-Product Algorithm

$$\eta_{X_1}(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_{\mathrm{A}}(x_1)\cdot f_{\mathrm{B}}(x_2)\cdot f_{\mathrm{C}}(x_1,x_2,x_3)\cdot f_{\mathrm{D}}(x_3,x_4)\cdot f_{\mathrm{E}}(x_3,x_5)$$

$$= \underbrace{f_{\mathrm{A}}(x_1)}\ \sum_{x_2}\sum_{x_3} f_{\mathrm{C}}(x_1,x_2,x_3)\ \underbrace{f_{\mathrm{B}}(x_2)}\ \sum_{x_4} f_{\mathrm{D}}(x_3,x_4)\cdot\ \underbrace{1}\ \sum_{x_5} f_{\mathrm{E}}(x_3,x_5)\cdot\ \underbrace{1}$$

The Sum-Product Algorithm

$$\eta_{X_1}(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_{\mathrm{A}}(x_1)\cdot f_{\mathrm{B}}(x_2)\cdot f_{\mathrm{C}}(x_1,x_2,x_3)\cdot f_{\mathrm{D}}(x_3,x_4)\cdot f_{\mathrm{E}}(x_3,x_5)$$

$$= \underbrace{f_{\mathrm{A}}(x_1)}\ \sum_{x_2}\sum_{x_3} f_{\mathrm{C}}(x_1,x_2,x_3)\ \underbrace{f_{\mathrm{B}}(x_2)}\ \sum_{x_4} f_{\mathrm{D}}(x_3,x_4)\cdot \underbrace{1}\ \sum_{x_5} f_{\mathrm{E}}(x_3,x_5)\cdot \underbrace{1}_{\mu_{X_5\to f_{\mathrm{E}}}(x_5)}$$

The Sum-Product Algorithm

$$\eta_{X_1}(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_{\mathrm{A}}(x_1)\cdot f_{\mathrm{B}}(x_2)\cdot f_{\mathrm{C}}(x_1,x_2,x_3)\cdot f_{\mathrm{D}}(x_3,x_4)\cdot f_{\mathrm{E}}(x_3,x_5)$$

$$= \underbrace{f_{\mathrm{A}}(x_1)}\ \sum_{x_2}\sum_{x_3} f_{\mathrm{C}}(x_1,x_2,x_3)\ \underbrace{f_{\mathrm{B}}(x_2)}\ \sum_{x_4} f_{\mathrm{D}}(x_3,x_4)\cdot \underbrace{1}\ \sum_{x_5} f_{\mathrm{E}}(x_3,x_5)\cdot \underbrace{1}_{\mu_{X_5\to f_{\mathrm{E}}}(x_5)}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{\mu_{f_{\mathrm{E}}\to X_3}(x_3)}$$

$$\underbrace{\qquad}$$

The Sum-Product Algorithm

$$\eta_{X_1}(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_{\mathrm{A}}(x_1)\cdot f_{\mathrm{B}}(x_2)\cdot f_{\mathrm{C}}(x_1,x_2,x_3)\cdot f_{\mathrm{D}}(x_3,x_4)\cdot f_{\mathrm{E}}(x_3,x_5)$$

$$= \underbrace{f_{\mathrm{A}}(x_1)}\ \underbrace{\sum_{x_2}\sum_{x_3} f_{\mathrm{C}}(x_1,x_2,x_3)\ \underbrace{f_{\mathrm{B}}(x_2)}\ \underbrace{\sum_{x_4} f_{\mathrm{D}}(x_3,x_4)\cdot \underbrace{1}_{\mu_{X_4\to f_{\mathrm{D}}}(x_4)}}\ \underbrace{\sum_{x_5} f_{\mathrm{E}}(x_3,x_5)\cdot \underbrace{1}_{\mu_{X_5\to f_{\mathrm{E}}}(x_5)}}_{\mu_{f_{\mathrm{E}}\to X_3}(x_3)}}_{}}_{}$$

The Sum-Product Algorithm

$$\eta_{X_1}(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_{\mathrm{A}}(x_1)\cdot f_{\mathrm{B}}(x_2)\cdot f_{\mathrm{C}}(x_1,x_2,x_3)\cdot f_{\mathrm{D}}(x_3,x_4)\cdot f_{\mathrm{E}}(x_3,x_5)$$

$$= \underbrace{f_{\mathrm{A}}(x_1)}\ \underbrace{\sum_{x_2}\sum_{x_3} f_{\mathrm{C}}(x_1,x_2,x_3)\ \underbrace{f_{\mathrm{B}}(x_2)}\ \underbrace{\sum_{x_4} f_{\mathrm{D}}(x_3,x_4)\cdot \underbrace{1}_{\mu_{x_4\to f_{\mathrm{D}}}(x_4)}}_{\mu_{f_{\mathrm{D}}\to x_3}(x_3)}\ \underbrace{\sum_{x_5} f_{\mathrm{E}}(x_3,x_5)\cdot \underbrace{1}_{\mu_{x_5\to f_{\mathrm{E}}}(x_5)}}_{\mu_{f_{\mathrm{E}}\to x_3}(x_3)}}_{\mu_{x_3\to f_{\mathrm{C}}}(x_3)}}_{\mu_{f_{\mathrm{C}}\to x_1}(x_1)}$$

The Sum-Product Algorithm

$$\eta_{X_1}(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_{\mathrm{A}}(x_1)\cdot f_{\mathrm{B}}(x_2)\cdot f_{\mathrm{C}}(x_1,x_2,x_3)\cdot f_{\mathrm{D}}(x_3,x_4)\cdot f_{\mathrm{E}}(x_3,x_5)$$

$$= \underbrace{f_{\mathrm{A}}(x_1)}\ \underbrace{\sum_{x_2}\sum_{x_3} f_{\mathrm{C}}(x_1,x_2,x_3)\ \underbrace{f_{\mathrm{B}}(x_2)}\ \underbrace{\sum_{x_4} f_{\mathrm{D}}(x_3,x_4)\cdot \underbrace{1}_{\mu_{x_4\to f_{\mathrm{D}}}(x_4)}}_{\mu_{f_{\mathrm{D}}\to x_3}(x_3)}\ \underbrace{\sum_{x_5} f_{\mathrm{E}}(x_3,x_5)\cdot \underbrace{1}_{\mu_{x_5\to f_{\mathrm{E}}}(x_5)}}_{\mu_{f_{\mathrm{E}}\to x_3}(x_3)}}_{\mu_{f_{X_3}\to f_{\mathrm{C}}}(x_3)}}_{}$$

The Sum-Product Algorithm

$$\eta_{X_1}(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_{\mathrm{A}}(x_1)\cdot f_{\mathrm{B}}(x_2)\cdot f_{\mathrm{C}}(x_1,x_2,x_3)\cdot f_{\mathrm{D}}(x_3,x_4)\cdot f_{\mathrm{E}}(x_3,x_5)$$

$$= \underbrace{f_{\mathrm{A}}(x_1)}\ \underbrace{\sum_{x_2}\sum_{x_3} f_{\mathrm{C}}(x_1,x_2,x_3)\ \underbrace{f_{\mathrm{B}}(x_2)}_{\mu_{f_{\mathrm{B}}\to X_2}(x_2)}\ \underbrace{\sum_{x_4} f_{\mathrm{D}}(x_3,x_4)\cdot \underbrace{1}_{\mu_{X_4\to f_{\mathrm{D}}}(x_4)}}_{\mu_{f_{\mathrm{D}}\to X_3}(x_3)}\ \underbrace{\sum_{x_5} f_{\mathrm{E}}(x_3,x_5)\cdot \underbrace{1}_{\mu_{X_5\to f_{\mathrm{E}}}(x_5)}}_{\mu_{f_{\mathrm{E}}\to X_3}(x_3)}}_{\mu_{f_{X_3}\to f_{\mathrm{C}}}(x_3)}}_{}$$

The Sum-Product Algorithm

$$\eta_{X_1}(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_{\mathrm{A}}(x_1)\cdot f_{\mathrm{B}}(x_2)\cdot f_{\mathrm{C}}(x_1,x_2,x_3)\cdot f_{\mathrm{D}}(x_3,x_4)\cdot f_{\mathrm{E}}(x_3,x_5)$$

$$= \underbrace{f_{\mathrm{A}}(x_1)}\ \underbrace{\sum_{x_2}\sum_{x_3} f_{\mathrm{C}}(x_1,x_2,x_3)\ \underbrace{\underbrace{f_{\mathrm{B}}(x_2)}_{\mu_{f_{\mathrm{B}}\to X_2}(x_2)}}_{\mu_{X_2\to f_{\mathrm{C}}}(x_2)}\ \underbrace{\sum_{x_4} f_{\mathrm{D}}(x_3,x_4)\cdot \underbrace{1}_{\mu_{X_4\to f_{\mathrm{D}}}(x_4)}}_{\mu_{f_{\mathrm{D}}\to X_3}(x_3)}\ \underbrace{\sum_{x_5} f_{\mathrm{E}}(x_3,x_5)\cdot \underbrace{1}_{\mu_{X_5\to f_{\mathrm{E}}}(x_5)}}_{\mu_{f_{\mathrm{E}}\to X_3}(x_3)}}_{\mu_{f_{X_3}\to f_{\mathrm{C}}}(x_3)}}_{}$$

The Sum-Product Algorithm

$$\eta_{X_1}(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_{\mathrm{A}}(x_1)\cdot f_{\mathrm{B}}(x_2)\cdot f_{\mathrm{C}}(x_1,x_2,x_3)\cdot f_{\mathrm{D}}(x_3,x_4)\cdot f_{\mathrm{E}}(x_3,x_5)$$

$$= \underbrace{f_{\mathrm{A}}(x_1)}_{\mu_{f_{\mathrm{C}}\to X_1}(x_1)} \Bigg(\sum_{x_2}\sum_{x_3} f_{\mathrm{C}}(x_1,x_2,x_3) \Bigg(\underbrace{\underbrace{f_{\mathrm{B}}(x_2)}_{\mu_{f_{\mathrm{B}}\to X_2}(x_2)}}_{\mu_{X_2\to f_{\mathrm{C}}}(x_2)} \Bigg(\underbrace{\sum_{x_4} f_{\mathrm{D}}(x_3,x_4)\cdot \underbrace{1}_{\mu_{X_4\to f_{\mathrm{D}}}(x_4)}}_{\mu_{f_{\mathrm{D}}\to X_3}(x_3)} \underbrace{\sum_{x_5} f_{\mathrm{E}}(x_3,x_5)\cdot \underbrace{1}_{\mu_{X_5\to f_{\mathrm{E}}}(x_5)}}_{\mu_{f_{\mathrm{E}}\to X_3}(x_3)} \Bigg)}_{\mu_{f_{X_3}\to f_{\mathrm{C}}}(x_3)} \Bigg)}_{\mu_{f_{\mathrm{C}}\to X_1}(x_1)}$$

The Sum-Product Algorithm

$$\eta_{X_1}(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_{\mathrm{A}}(x_1)\cdot f_{\mathrm{B}}(x_2)\cdot f_{\mathrm{C}}(x_1,x_2,x_3)\cdot f_{\mathrm{D}}(x_3,x_4)\cdot f_{\mathrm{E}}(x_3,x_5)$$

$$= \underbrace{f_{\mathrm{A}}(x_1)}_{\mu_{f_{\mathrm{A}}\to X_1}(x_1)} \underbrace{\sum_{x_2}\sum_{x_3} f_{\mathrm{C}}(x_1,x_2,x_3)\ \underbrace{f_{\mathrm{B}}(x_2)}_{\mu_{f_{\mathrm{B}}\to X_2}(x_2)}}_{\mu_{X_2\to f_{\mathrm{C}}}(x_2)} \underbrace{\sum_{x_4} f_{\mathrm{D}}(x_3,x_4)\cdot \underbrace{1}_{\mu_{X_4\to f_{\mathrm{D}}}(x_4)}}_{\mu_{f_{\mathrm{D}}\to X_3}(x_3)} \underbrace{\sum_{x_5} f_{\mathrm{E}}(x_3,x_5)\cdot \underbrace{1}_{\mu_{X_5\to f_{\mathrm{E}}}(x_5)}}_{\mu_{f_{\mathrm{E}}\to X_3}(x_3)}}_{\mu_{f_{X_3}\to f_{\mathrm{C}}}(x_3)}}_{\mu_{f_{\mathrm{C}}\to X_1}(x_1)}$$

The Sum-Product Algorithm

$$\eta_{X_1}(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_{\mathrm{A}}(x_1)\cdot f_{\mathrm{B}}(x_2)\cdot f_{\mathrm{C}}(x_1,x_2,x_3)\cdot f_{\mathrm{D}}(x_3,x_4)\cdot f_{\mathrm{E}}(x_3,x_5)$$

$$= \underbrace{f_{\mathrm{A}}(x_1)}_{\mu_{f_{\mathrm{A}}\to X_1}(x_1)} \underbrace{\sum_{x_2}\sum_{x_3} f_{\mathrm{C}}(x_1,x_2,x_3)\ \underbrace{f_{\mathrm{B}}(x_2)}_{\mu_{f_{\mathrm{B}}\to X_2}(x_2)}}_{\mu_{X_2\to f_{\mathrm{C}}}(x_2)} \underbrace{\sum_{x_4} f_{\mathrm{D}}(x_3,x_4)\cdot \underbrace{1}_{\mu_{X_4\to f_{\mathrm{D}}}(x_4)}}_{\mu_{f_{\mathrm{D}}\to X_3}(x_3)} \underbrace{\sum_{x_5} f_{\mathrm{E}}(x_3,x_5)\cdot \underbrace{1}_{\mu_{X_5\to f_{\mathrm{E}}}(x_5)}}_{\mu_{f_{\mathrm{E}}\to X_3}(x_3)}}_{\mu_{f_{X_3}\to f_{\mathrm{C}}}(x_3)}}_{\mu_{f_{\mathrm{C}}\to X_1}(x_1)}$$

The objects $\mu_{X_i\to f_j}(x_i)$ and $\mu_{f_j\to X_i}(x_i)$:

The Sum-Product Algorithm

$$\eta_{X_1}(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_{\mathrm{A}}(x_1)\cdot f_{\mathrm{B}}(x_2)\cdot f_{\mathrm{C}}(x_1,x_2,x_3)\cdot f_{\mathrm{D}}(x_3,x_4)\cdot f_{\mathrm{E}}(x_3,x_5)$$

$$= \underbrace{f_{\mathrm{A}}(x_1)}_{\mu_{f_{\mathrm{A}}\to X_1}(x_1)} \underbrace{\sum_{x_2}\sum_{x_3} f_{\mathrm{C}}(x_1,x_2,x_3)\ \underbrace{f_{\mathrm{B}}(x_2)}_{\mu_{f_{\mathrm{B}}\to X_2}(x_2)}}_{\mu_{X_2\to f_{\mathrm{C}}}(x_2)} \underbrace{\sum_{x_4} f_{\mathrm{D}}(x_3,x_4)\cdot \underbrace{1}_{\mu_{X_4\to f_{\mathrm{D}}}(x_4)}}_{\mu_{f_{\mathrm{D}}\to X_3}(x_3)} \underbrace{\sum_{x_5} f_{\mathrm{E}}(x_3,x_5)\cdot \underbrace{1}_{\mu_{X_5\to f_{\mathrm{E}}}(x_5)}}_{\mu_{f_{\mathrm{E}}\to X_3}(x_3)}}_{\mu_{f_{X_3}\to f_{\mathrm{C}}}(x_3)}}_{\mu_{f_{\mathrm{C}}\to X_1}(x_1)}$$

The objects $\mu_{X_i\to f_j}(x_i)$ and $\mu_{f_j\to X_i}(x_i)$:

- They are called messages.

The Sum-Product Algorithm

$$\eta_{X_1}(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_{\mathrm{A}}(x_1)\cdot f_{\mathrm{B}}(x_2)\cdot f_{\mathrm{C}}(x_1,x_2,x_3)\cdot f_{\mathrm{D}}(x_3,x_4)\cdot f_{\mathrm{E}}(x_3,x_5)$$

$$= \underbrace{f_{\mathrm{A}}(x_1)}_{\mu_{f_{\mathrm{A}}\to X_1}(x_1)} \underbrace{\sum_{x_2}\sum_{x_3} f_{\mathrm{C}}(x_1,x_2,x_3)\ \underbrace{f_{\mathrm{B}}(x_2)}_{\mu_{f_{\mathrm{B}}\to X_2}(x_2)}}_{\mu_{X_2\to f_{\mathrm{C}}}(x_2)} \underbrace{\sum_{x_4} f_{\mathrm{D}}(x_3,x_4)\cdot \underbrace{1}_{\mu_{X_4\to f_{\mathrm{D}}}(x_4)}}_{\mu_{f_{\mathrm{D}}\to X_3}(x_3)} \underbrace{\sum_{x_5} f_{\mathrm{E}}(x_3,x_5)\cdot \underbrace{1}_{\mu_{X_5\to f_{\mathrm{E}}}(x_5)}}_{\mu_{f_{\mathrm{E}}\to X_3}(x_3)}}_{\mu_{f_{X_3}\to f_{\mathrm{C}}}(x_3)}}_{\mu_{f_{\mathrm{C}}\to X_1}(x_1)}$$

The objects $\mu_{X_i\to f_j}(x_i)$ and $\mu_{f_j\to X_i}(x_i)$:

- They are called messages.
- They can be associated with the edge between the vertex X_i and the vertex f_j.

The Sum-Product Algorithm

$$\eta_{X_1}(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_{\mathrm{A}}(x_1)\cdot f_{\mathrm{B}}(x_2)\cdot f_{\mathrm{C}}(x_1,x_2,x_3)\cdot f_{\mathrm{D}}(x_3,x_4)\cdot f_{\mathrm{E}}(x_3,x_5)$$

$$= \underbrace{f_{\mathrm{A}}(x_1)}_{\mu_{f_{\mathrm{A}}\to X_1}(x_1)} \sum_{x_2}\sum_{x_3} f_{\mathrm{C}}(x_1,x_2,x_3) \underbrace{f_{\mathrm{B}}(x_2)}_{\mu_{f_{\mathrm{B}}\to X_2}(x_2)} \underbrace{\sum_{x_4} f_{\mathrm{D}}(x_3,x_4)\cdot \underbrace{1}_{\mu_{X_4\to f_{\mathrm{D}}}(x_4)}}_{\mu_{f_{\mathrm{D}}\to X_3}(x_3)} \underbrace{\sum_{x_5} f_{\mathrm{E}}(x_3,x_5)\cdot \underbrace{1}_{\mu_{X_5\to f_{\mathrm{E}}}(x_5)}}_{\mu_{f_{\mathrm{E}}\to X_3}(x_3)}}_{\mu_{X_2\to f_{\mathrm{C}}}(x_2)} \Bigg}_{\mu_{f_{X_3}\to f_{\mathrm{C}}}(x_3)} \Bigg\}_{\mu_{f_{\mathrm{C}}\to X_1}(x_1)}$$

The objects $\mu_{X_i\to f_j}(x_i)$ and $\mu_{f_j\to X_i}(x_i)$:

- They are called messages.
- They can be associated with the edge between the vertex X_i and the vertex f_j.
- They are functions of x_i, i.e., their domain is the alphabet of X_i.

The Sum-Product Algorithm

$$\eta_{X_1}(x_1) = \sum_{x_2}\sum_{x_3}\sum_{x_4}\sum_{x_5} f_{\mathrm{A}}(x_1)\cdot f_{\mathrm{B}}(x_2)\cdot f_{\mathrm{C}}(x_1,x_2,x_3)\cdot f_{\mathrm{D}}(x_3,x_4)\cdot f_{\mathrm{E}}(x_3,x_5)$$

$$= \underbrace{f_{\mathrm{A}}(x_1)}_{\mu_{f_{\mathrm{A}}\to X_1}(x_1)} \sum_{x_2}\sum_{x_3} f_{\mathrm{C}}(x_1,x_2,x_3) \underbrace{f_{\mathrm{B}}(x_2)}_{\mu_{f_{\mathrm{B}}\to X_2}(x_2)} \underbrace{\sum_{x_4} f_{\mathrm{D}}(x_3,x_4)\cdot \underbrace{1}_{\mu_{X_4\to f_{\mathrm{D}}}(x_4)}}_{\mu_{f_{\mathrm{D}}\to X_3}(x_3)} \underbrace{\sum_{x_5} f_{\mathrm{E}}(x_3,x_5)\cdot \underbrace{1}_{\mu_{X_5\to f_{\mathrm{E}}}(x_5)}}_{\mu_{f_{\mathrm{E}}\to X_3}(x_3)}}_{\mu_{X_2\to f_{\mathrm{C}}}(x_2)} \Bigg}_{\mu_{f_{X_3}\to f_{\mathrm{C}}}(x_3)} \Bigg\}_{\mu_{f_{\mathrm{C}}\to X_1}(x_1)}$$

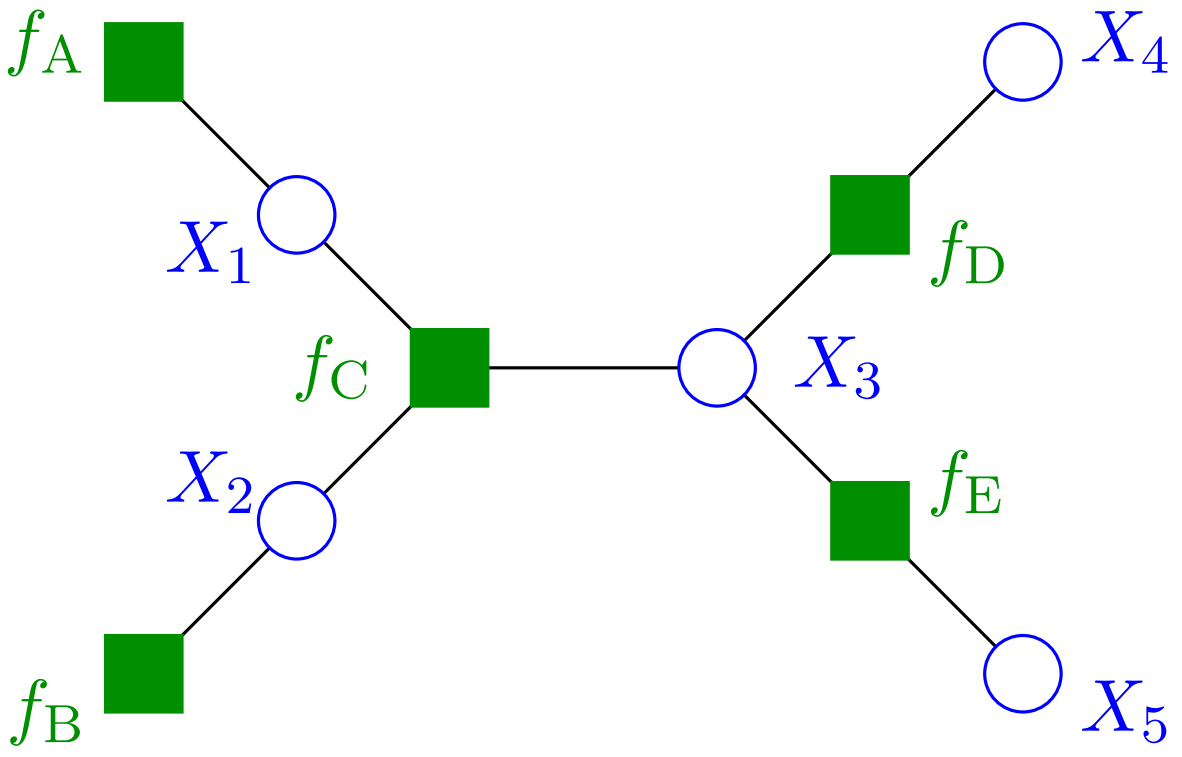
The objects $\mu_{X_i\to f_j}(x_i)$ and $\mu_{f_j\to X_i}(x_i)$:

- They are called messages.
- They can be associated with the edge between the vertex X_i and the vertex f_j.
- They are functions of x_i, i.e., their domain is the alphabet of X_i.

Note: similar manipulations can be performed for calculating $\eta_{X_2}(x_2)$, $\eta_{X_3}(x_3)$, $\eta_{X_4}(x_4)$, $\eta_{X_5}(x_5)$.

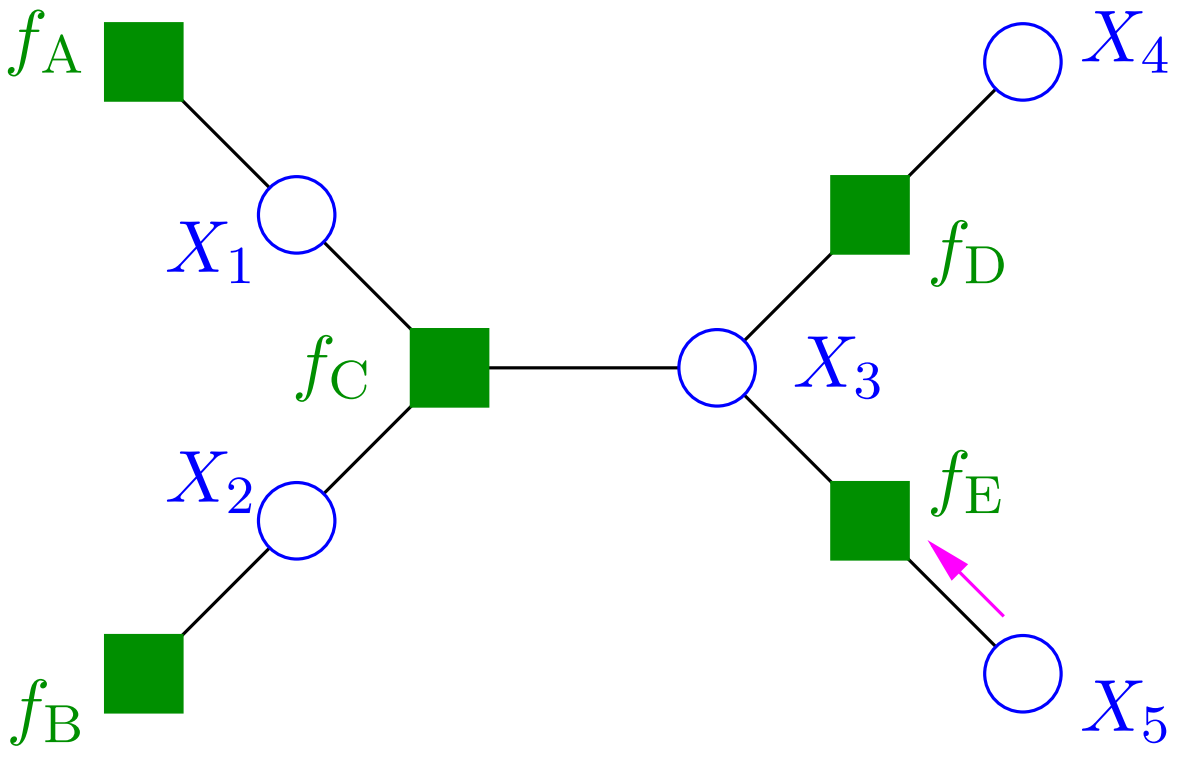
The Sum-Product Algorithm

Messages necessary for calculating $\eta_{X_1}(x_1)$.



The Sum-Product Algorithm

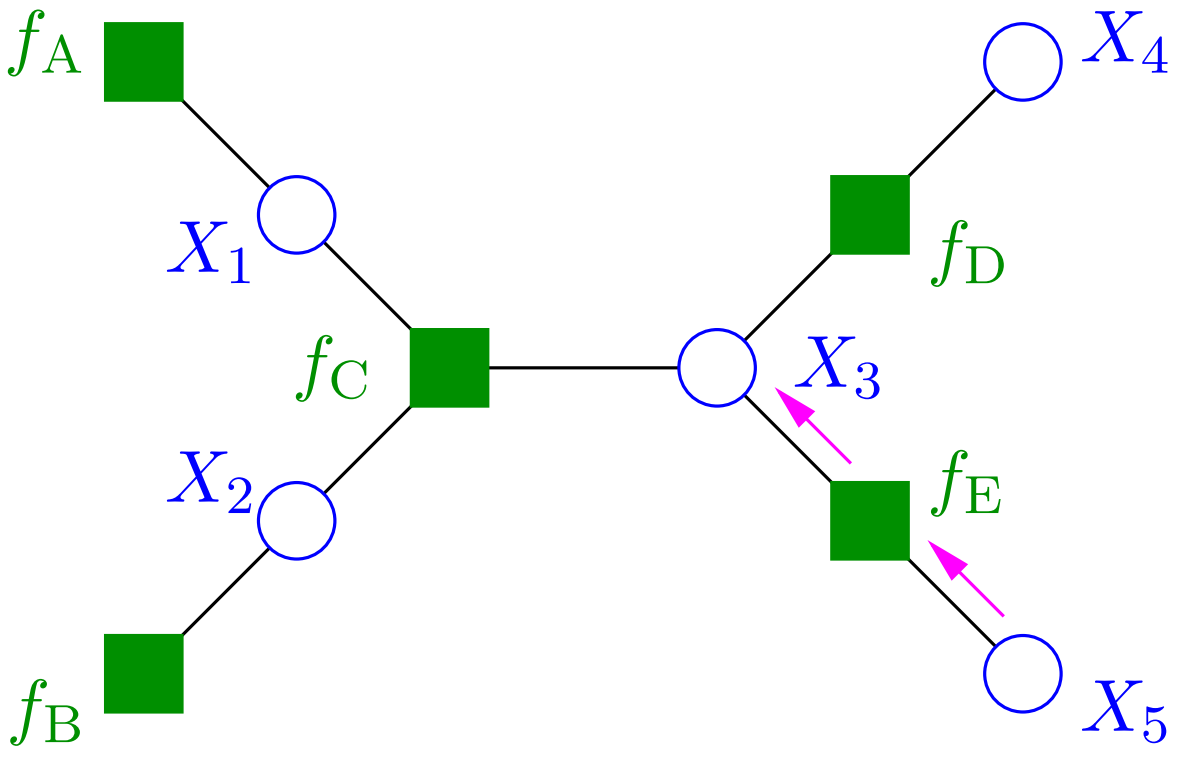
Messages necessary for calculating $\eta_{X_1}(x_1)$.



$$\mu_{X_5 \to f_{\mathrm{E}}}(x_5) \triangleq 1$$

The Sum-Product Algorithm

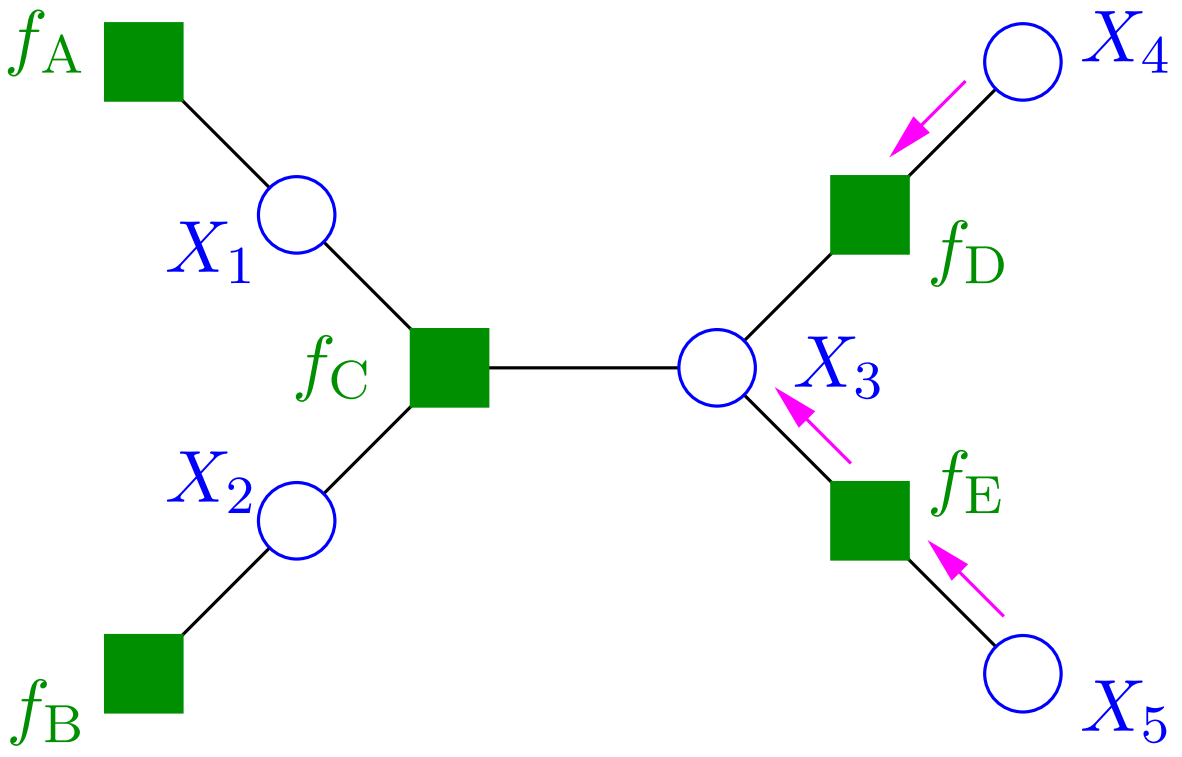
Messages necessary for calculating $\eta_{X_1}(x_1)$.



$$\mu_{f_{\mathrm{E}} \to X_3}(x_3) \triangleq \sum_{x_5} f_{\mathrm{E}}(x_3, x_5) \cdot \mu_{X_5 \to f_{\mathrm{E}}}(x_5)$$

The Sum-Product Algorithm

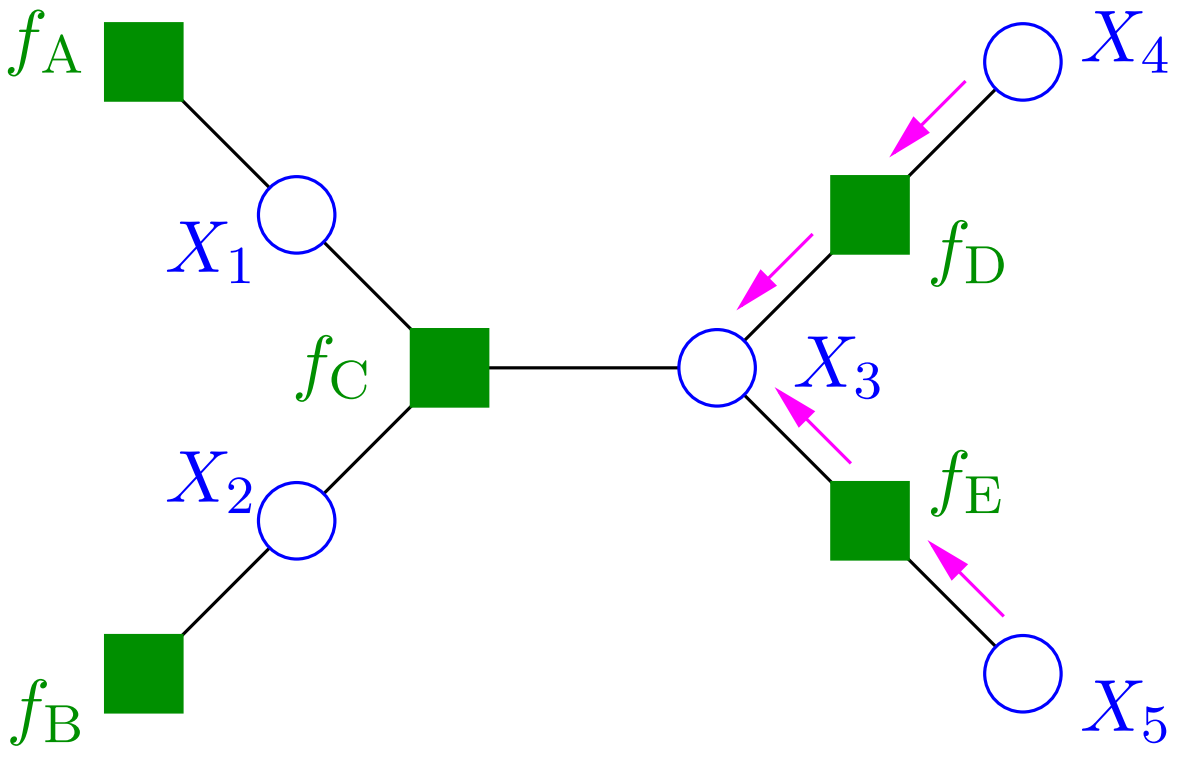
Messages necessary for calculating $\eta_{X_1}(x_1)$.



$$\mu_{X_4 \to f_{\mathrm{D}}}(x_4) \triangleq 1$$

The Sum-Product Algorithm

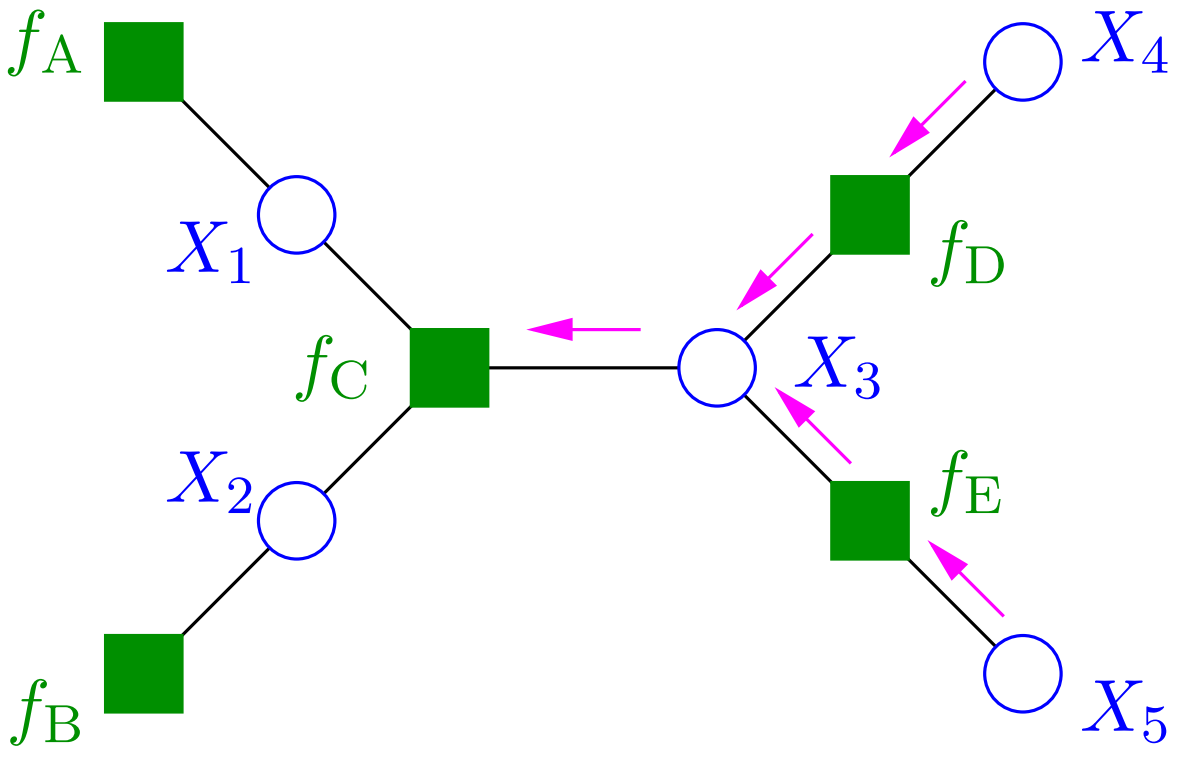
Messages necessary for calculating $\eta_{X_1}(x_1)$.



$$\mu_{f_{\mathrm{D}} \to X_3}(x_3) \triangleq \sum_{x_4} f_{\mathrm{D}}(x_3, x_4) \cdot \mu_{X_4 \to f_{\mathrm{D}}}(x_4)$$

The Sum-Product Algorithm

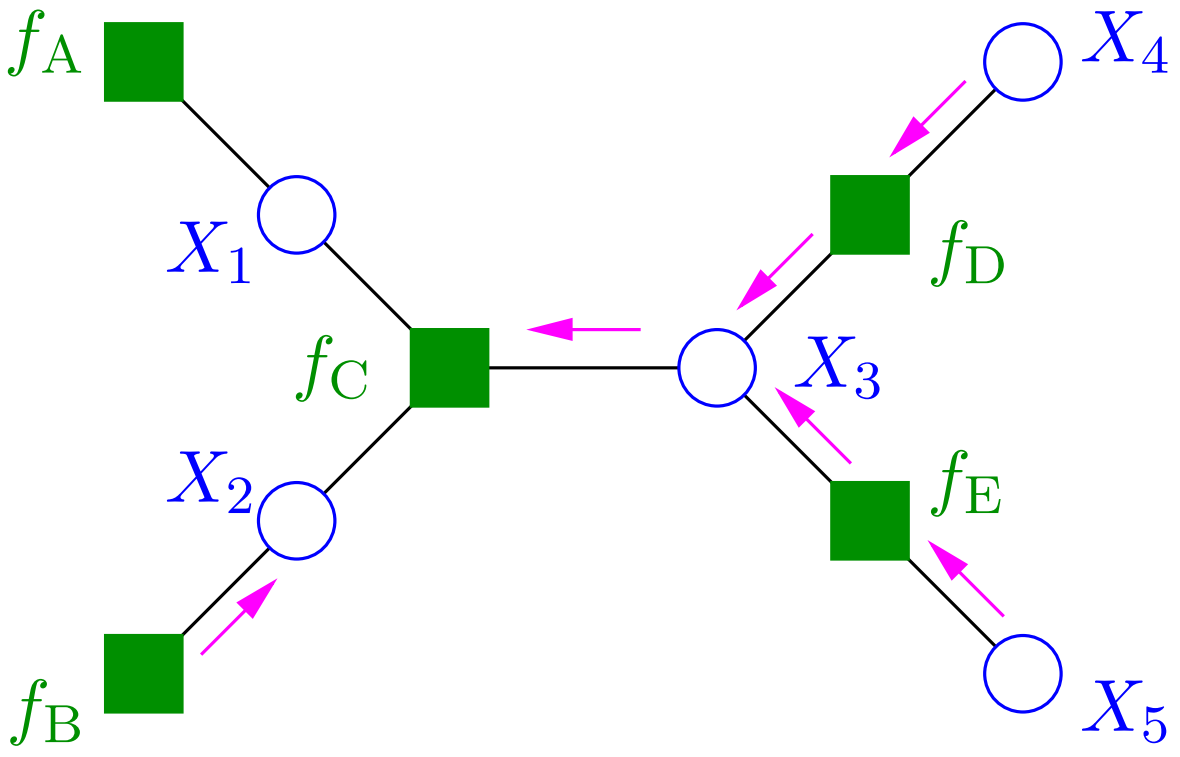
Messages necessary for calculating $\eta_{X_1}(x_1)$.



$$\mu_{X_3 \to f_{\mathrm{C}}}(x_3) \triangleq \mu_{f_{\mathrm{D}} \to X_3}(x_3) \cdot \mu_{f_{\mathrm{E}} \to X_3}(x_3)$$

The Sum-Product Algorithm

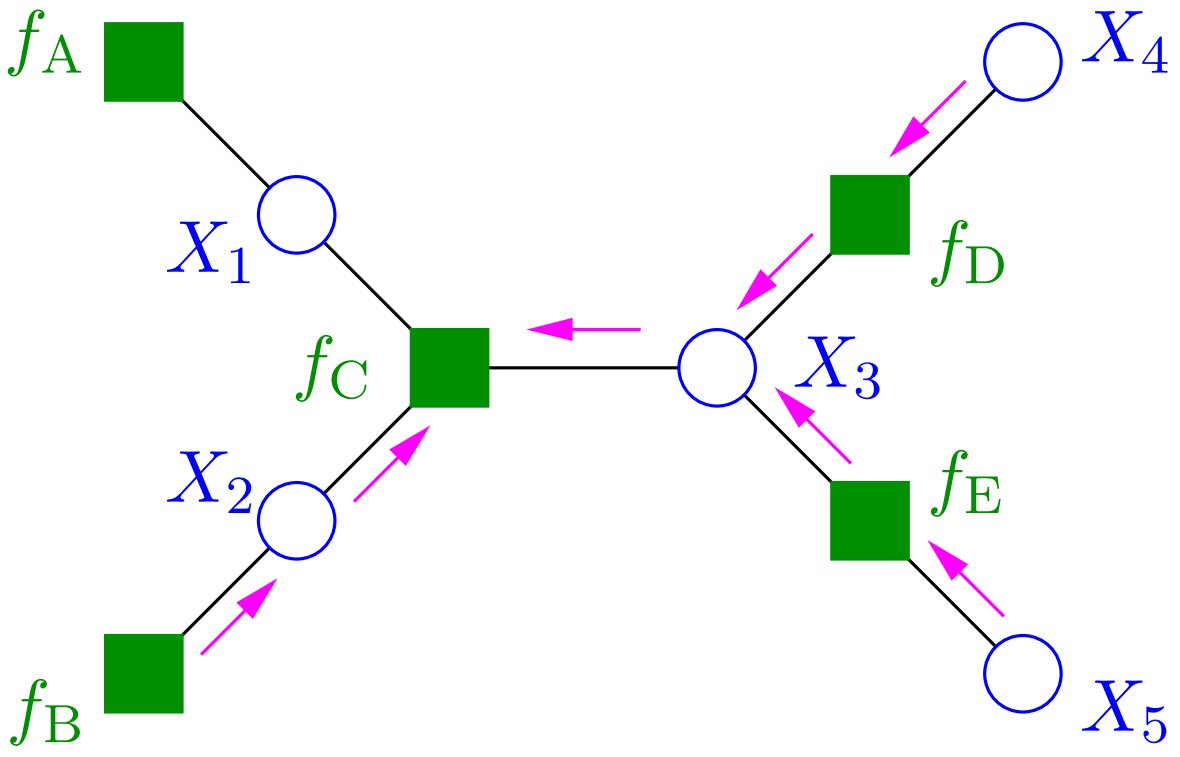
Messages necessary for calculating $\eta_{X_1}(x_1)$.



$$\mu_{f_{\mathrm{B}} \to X_2}(x_2) \triangleq f_{\mathrm{B}}(x_2)$$

The Sum-Product Algorithm

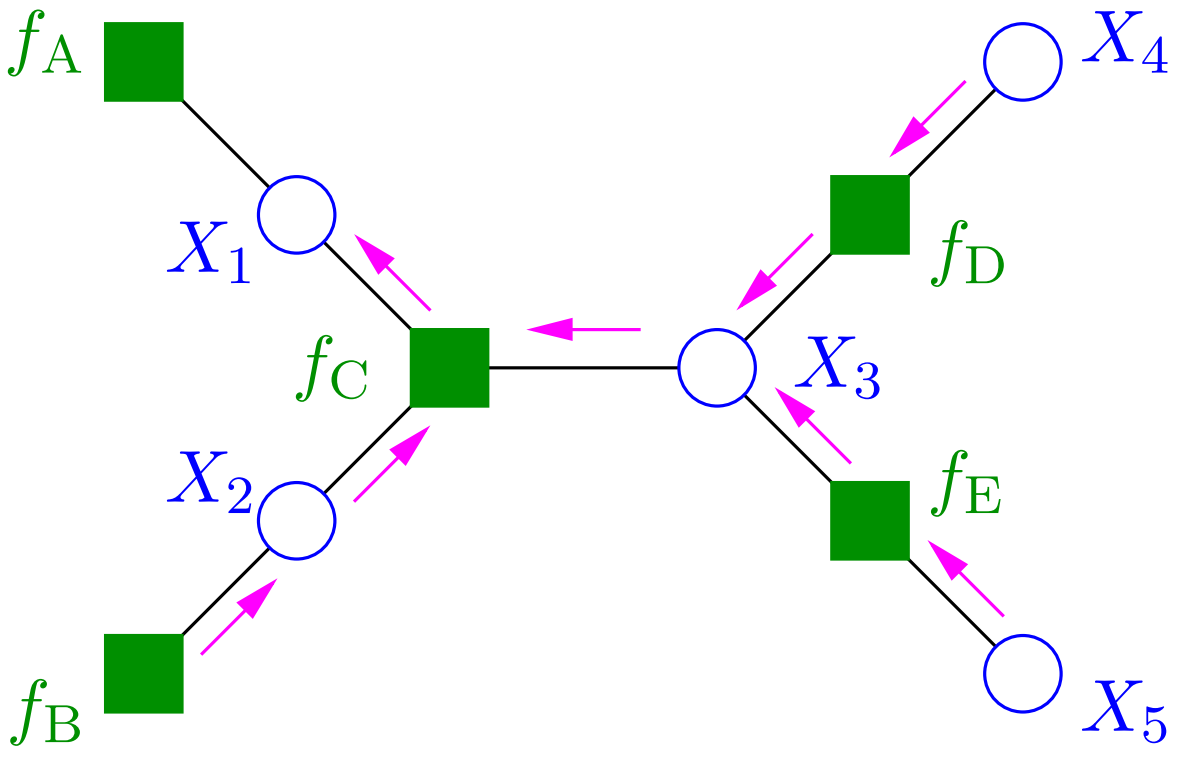
Messages necessary for calculating $\eta_{X_1}(x_1)$.



$$\mu_{X_2 \to f_{\mathrm{C}}}(x_2) \triangleq \mu_{f_{\mathrm{B}} \to X_2}(x_2)$$

The Sum-Product Algorithm

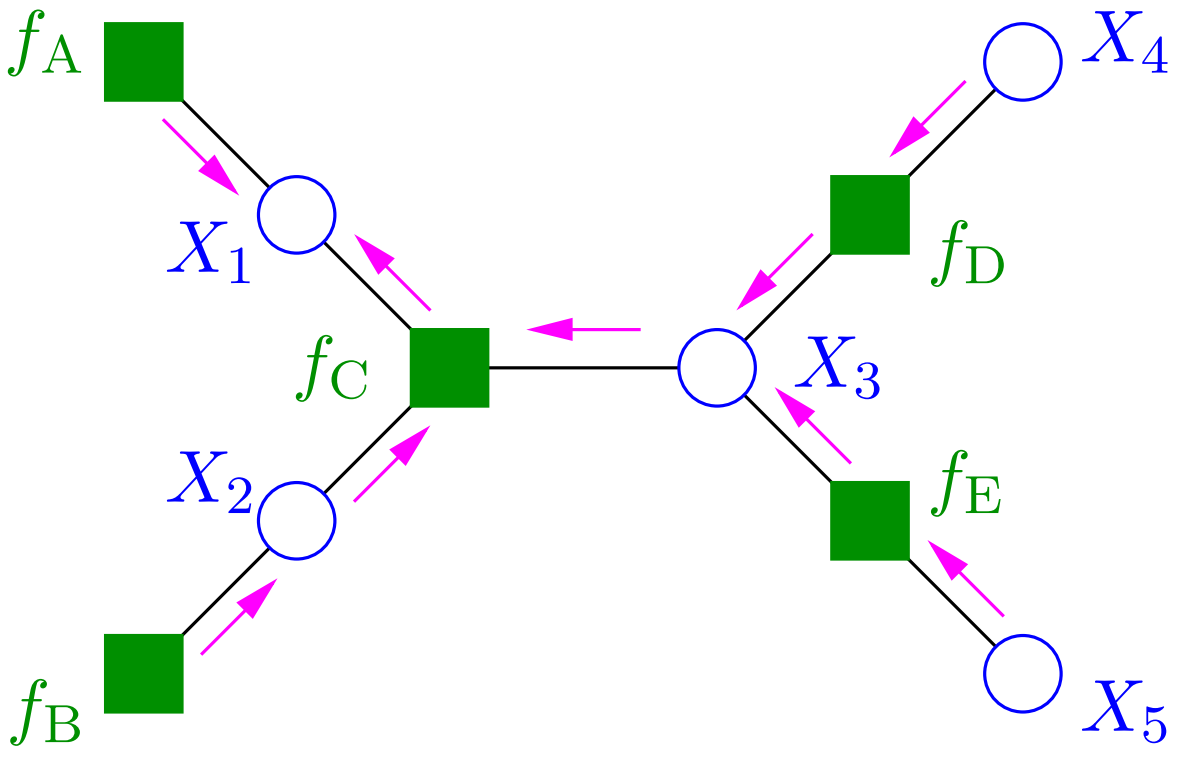
Messages necessary for calculating $\eta_{X_1}(x_1)$.



$$\mu_{f_{\mathrm{C}} \to X_1}(x_1) \triangleq \sum_{x_2} \sum_{x_3} f_{\mathrm{C}}(x_1, x_2, x_3) \cdot \mu_{X_2 \to f_{\mathrm{C}}}(x_2) \cdot \mu_{X_3 \to f_{\mathrm{C}}}(x_3)$$

The Sum-Product Algorithm

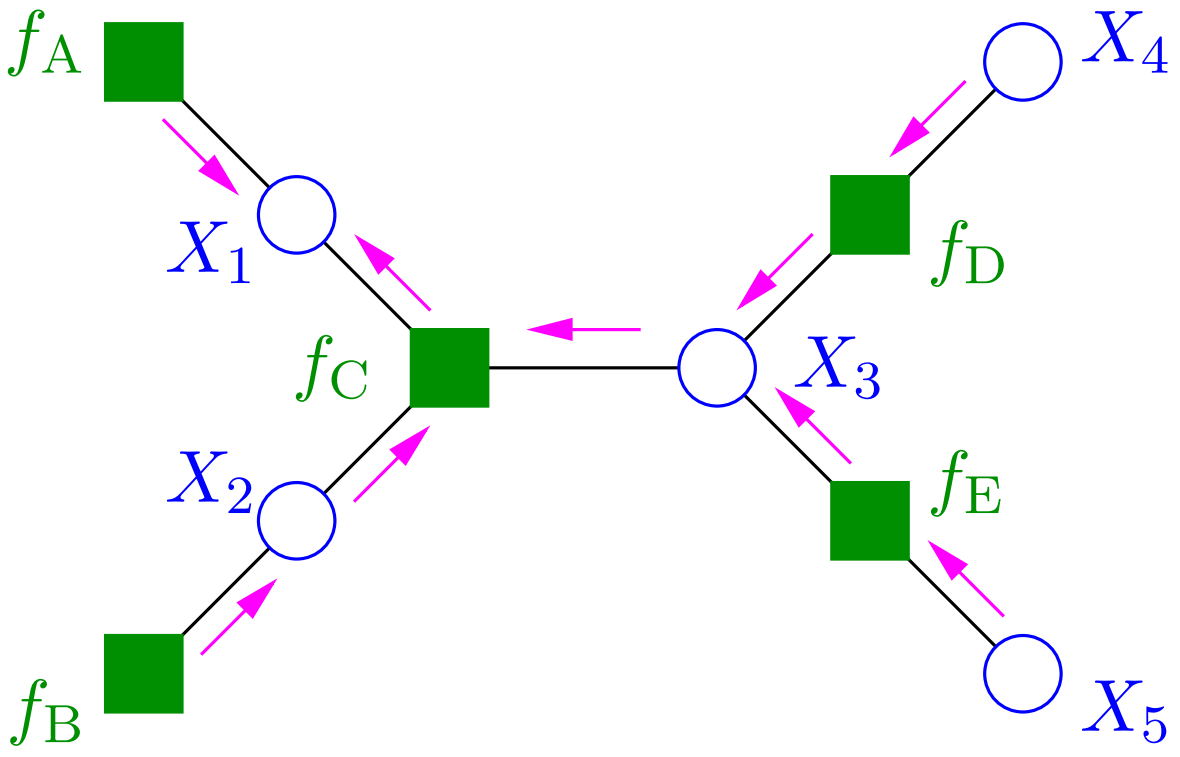
Messages necessary for calculating $\eta_{X_1}(x_1)$.



$$\mu_{f_{\mathrm{A}} \to X_1}(x_1) \triangleq f_{\mathrm{A}}(x_1)$$

The Sum-Product Algorithm

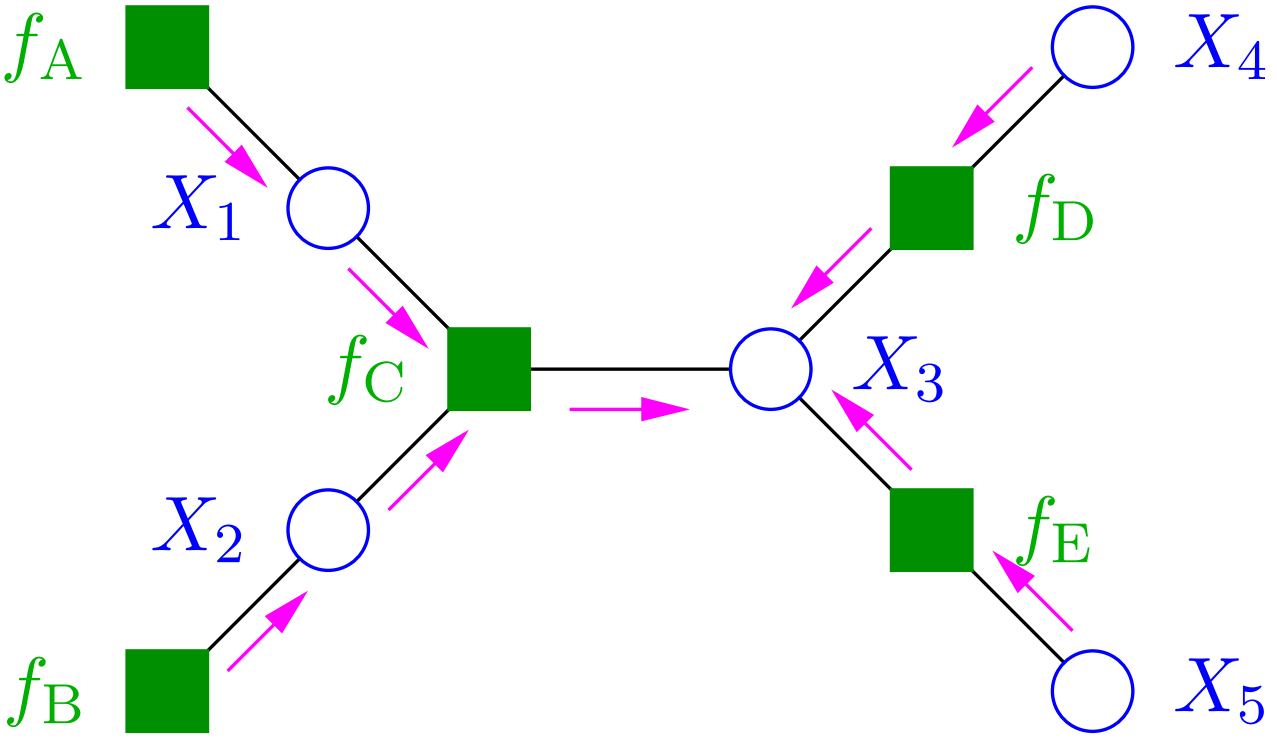
Messages necessary for calculating $\eta_{X_1}(x_1)$.



$$\eta_{X_1}(x_1) = \mu_{f_A \to X_1}(x_1) \cdot \mu_{f_C \to X_1}(x_1)$$

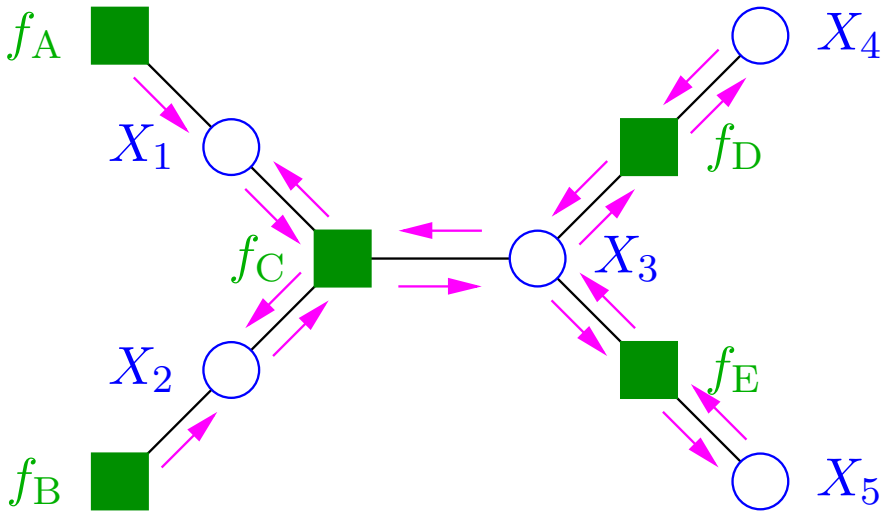
The Sum-Product Algorithm

Messages necessary for calculating $\eta_{X_3}(x_3)$.



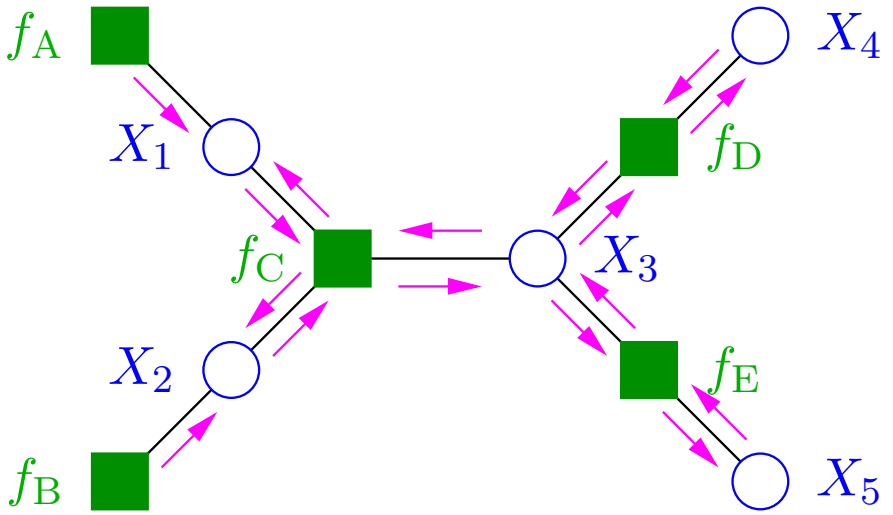
The Sum-Product Algorithm

The figure shows the messages that are necessary for calculating $\eta_{X_1}(x_1)$, $\eta_{X_2}(x_2)$, $\eta_{X_3}(x_3)$, $\eta_{X_4}(x_4)$, and $\eta_{X_5}(x_5)$.



The Sum-Product Algorithm

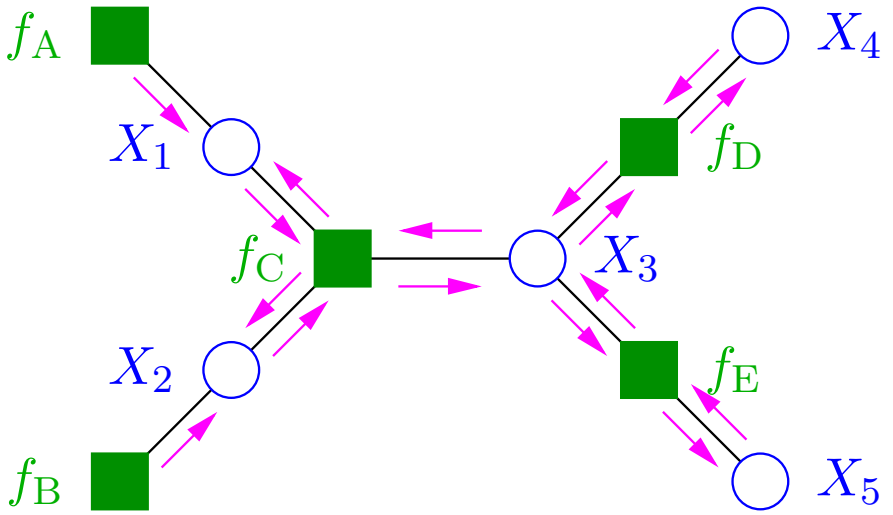
The figure shows the messages that are necessary for calculating $\eta_{X_1}(x_1)$, $\eta_{X_2}(x_2)$, $\eta_{X_3}(x_3)$, $\eta_{X_4}(x_4)$, and $\eta_{X_5}(x_5)$.



- Edges: Messages are sent along edges.

The Sum-Product Algorithm

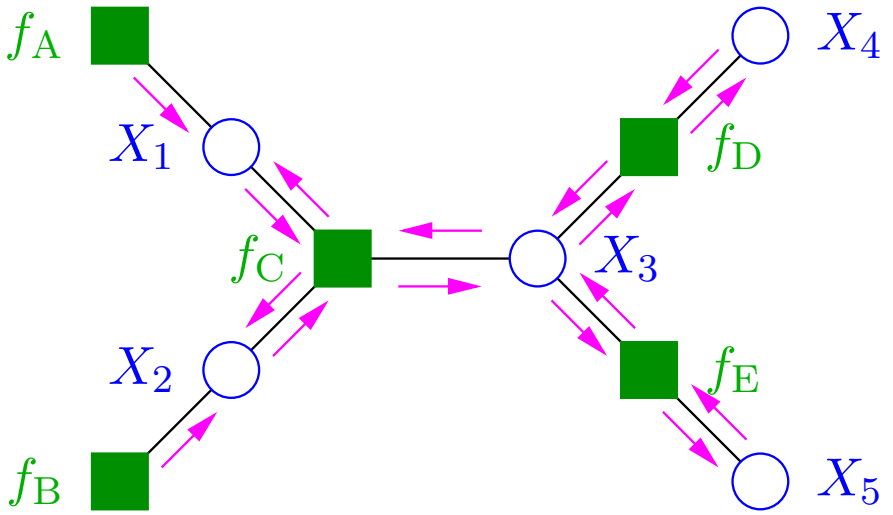
The figure shows the messages that are necessary for calculating $\eta_{X_1}(x_1)$, $\eta_{X_2}(x_2)$, $\eta_{X_3}(x_3)$, $\eta_{X_4}(x_4)$, and $\eta_{X_5}(x_5)$.



- Edges: Messages are sent along edges.
- Processing: Taking products and doing summations is done in the vertices.

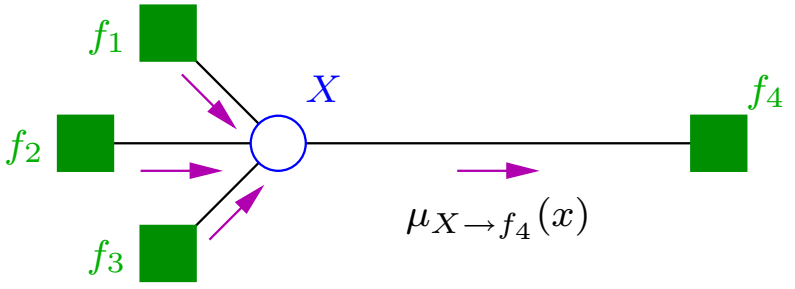
The Sum-Product Algorithm

The figure shows the messages that are necessary for calculating $\eta_{X_1}(x_1)$, $\eta_{X_2}(x_2)$, $\eta_{X_3}(x_3)$, $\eta_{X_4}(x_4)$, and $\eta_{X_5}(x_5)$.



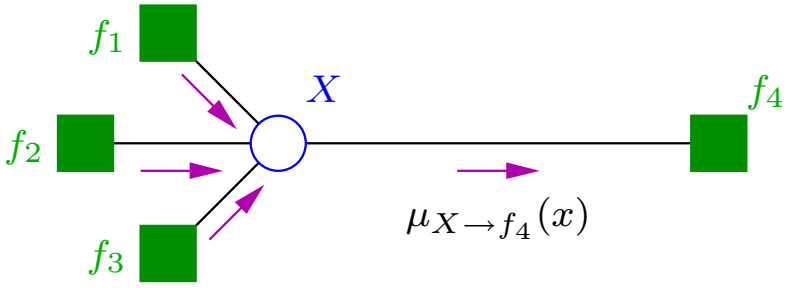
- Edges: Messages are sent along edges.
- Processing: Taking products and doing summations is done in the vertices.
- Reuse of messages: We see that messages can be “reused” in the sense that many partial calculations are the same; so it suffices to perform them only once.

The Sum-Product Algorithm

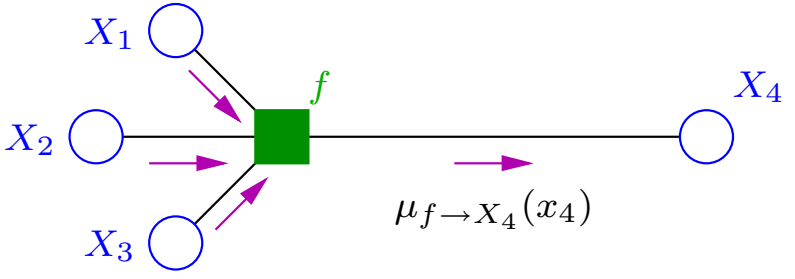


$$\mu_{X \to f_4}(x) = \mu_{f_1 \to X}(x) \cdot \mu_{f_2 \to X}(x) \cdot \mu_{f_3 \to X}(x)$$

The Sum-Product Algorithm

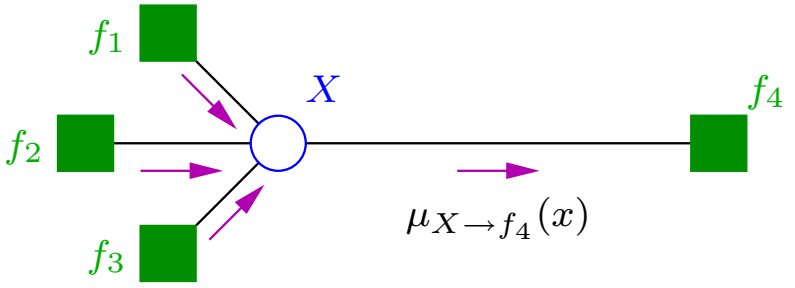


$$\mu_{X \to f_4}(x) = \mu_{f_1 \to X}(x) \cdot \mu_{f_2 \to X}(x) \cdot \mu_{f_3 \to X}(x)$$

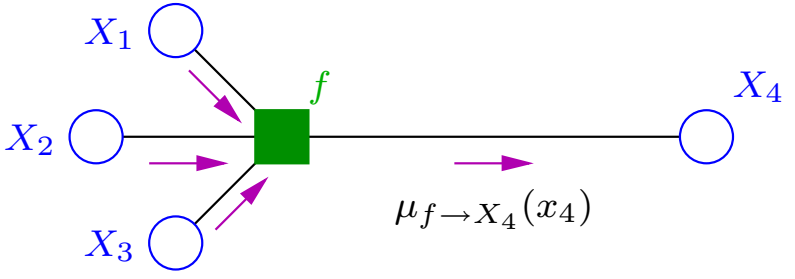


$$\mu_{f \to X_4}(x_4) = \sum_{x_1} \sum_{x_2} \sum_{x_3} f(x_1, x_2, x_3, x_4) \cdot \mu_{X_1 \to f}(x_1) \cdot \mu_{X_2 \to f}(x_2) \cdot \mu_{X_3 \to f}(x_3)$$

The Sum-Product Algorithm



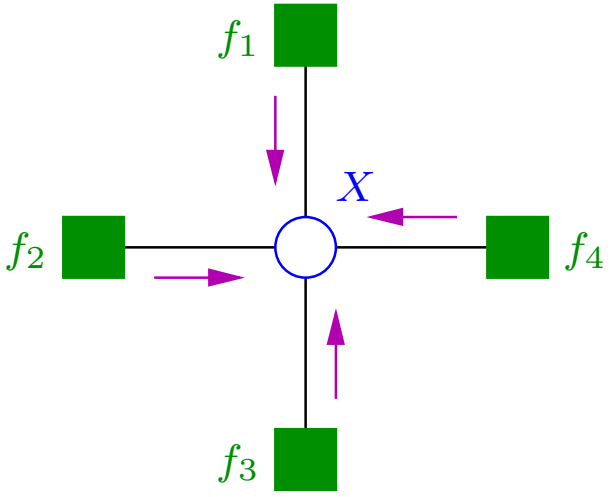
$$\mu_{X \to f_4}(x) = \mu_{f_1 \to X}(x) \cdot \mu_{f_2 \to X}(x) \cdot \mu_{f_3 \to X}(x)$$



$$\mu_{f \to X_4}(x_4) = \sum_{x_1} \sum_{x_2} \sum_{x_3} f(x_1, x_2, x_3, x_4) \cdot \mu_{X_1 \to f}(x_1) \cdot \mu_{X_2 \to f}(x_2) \cdot \mu_{X_3 \to f}(x_3)$$

Depending on the setup, non-negative scaling factors can be included in both update rules.

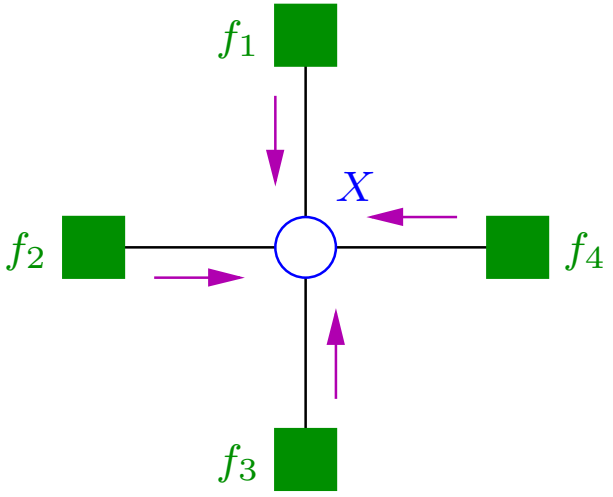
The Sum-Product Algorithm



Computation of marginal at variable node:

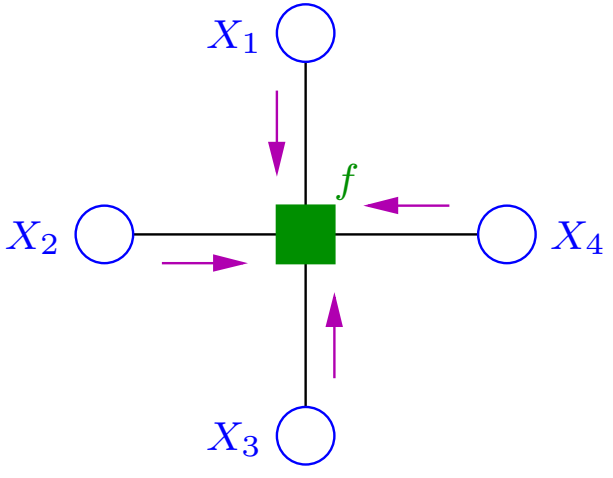
$$\eta_X(x) = \mu_{f_1 \to X}(x) \cdot \mu_{f_2 \to X}(x)$$
$$\cdot\ \mu_{f_3 \to X}(x) \cdot \mu_{f_4 \to X}(x)$$

The Sum-Product Algorithm



Computation of marginal at variable node:

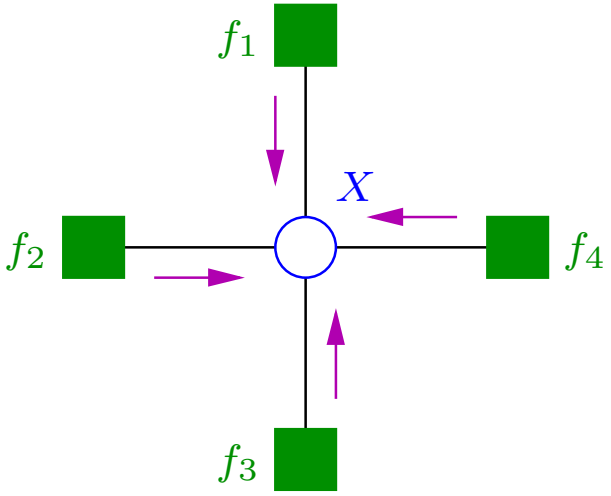
$$\eta_X(x) = \mu_{f_1 \to X}(x) \cdot \mu_{f_2 \to X}(x) \cdot \mu_{f_3 \to X}(x) \cdot \mu_{f_4 \to X}(x)$$



Computation of marginal at function node:

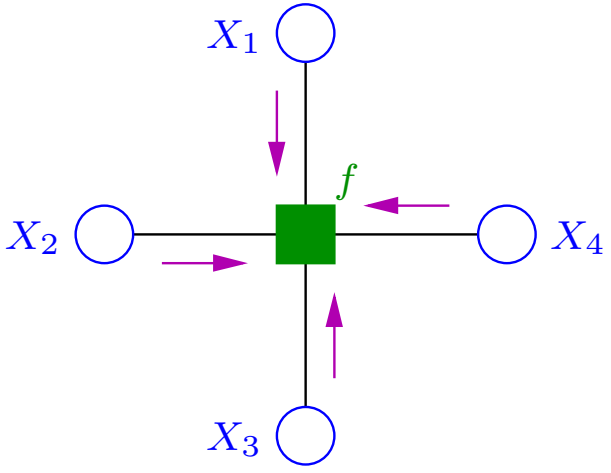
$$\eta_f(x_1, x_2, x_3, x_4) = f(x_1, x_2, x_3, x_4) \cdot \mu_{X_1 \to f}(x_1) \cdot \mu_{X_2 \to f}(x_2) \cdot \mu_{X_3 \to f}(x_3) \cdot \mu_{X_4 \to f}(x_4)$$

The Sum-Product Algorithm



Computation of marginal at variable node:

$$\eta_X(x) = \mu_{f_1 \to X}(x) \cdot \mu_{f_2 \to X}(x) \cdot \mu_{f_3 \to X}(x) \cdot \mu_{f_4 \to X}(x)$$



Computation of marginal at function node:

$$\eta_f(x_1, x_2, x_3, x_4) = f(x_1, x_2, x_3, x_4) \cdot \mu_{X_1 \to f}(x_1) \cdot \mu_{X_2 \to f}(x_2) \cdot \mu_{X_3 \to f}(x_3) \cdot \mu_{X_4 \to f}(x_4)$$

Depending on the setup, non-negative scaling factors can be included in both marginal rules.

The Sum-Product Algorithm

- Factor graph without cycles: in this case it is obvious what messages have to be calculated when.

$\Rightarrow$ “Mode of operation 1”

The Sum-Product Algorithm

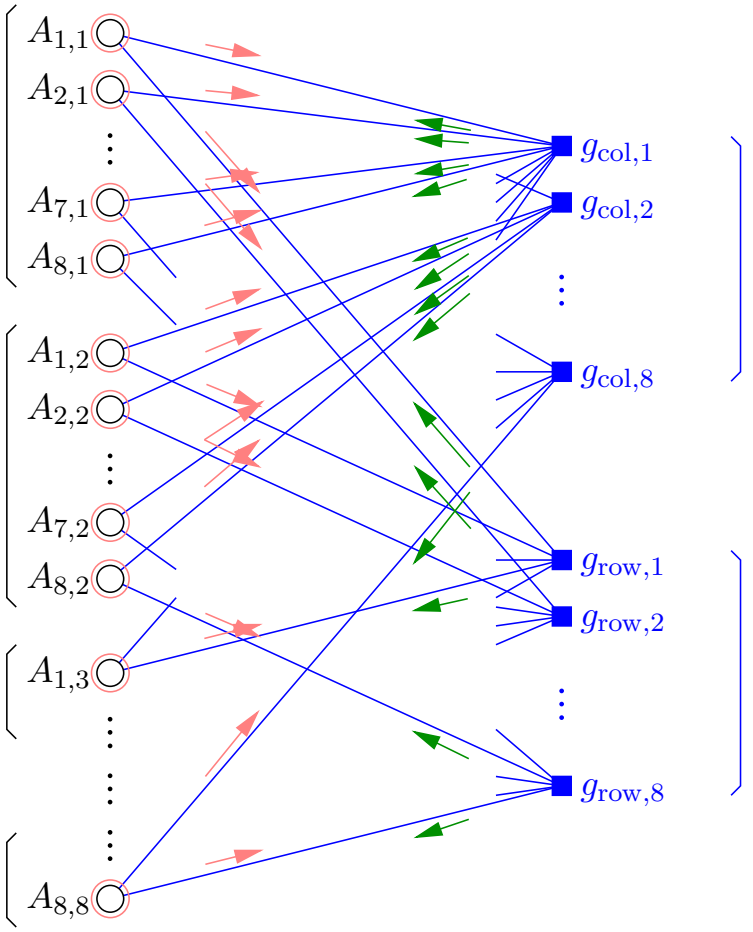
- Factor graph without cycles: in this case it is obvious what messages have to be calculated when.

  $\Rightarrow$ "Mode of operation 1"

- Factor graph with cycles: one has to decide what update schedule to take.

  $\Rightarrow$ "Mode of operation 2"

Comments on the Sum-Product Algorithm



Global function:

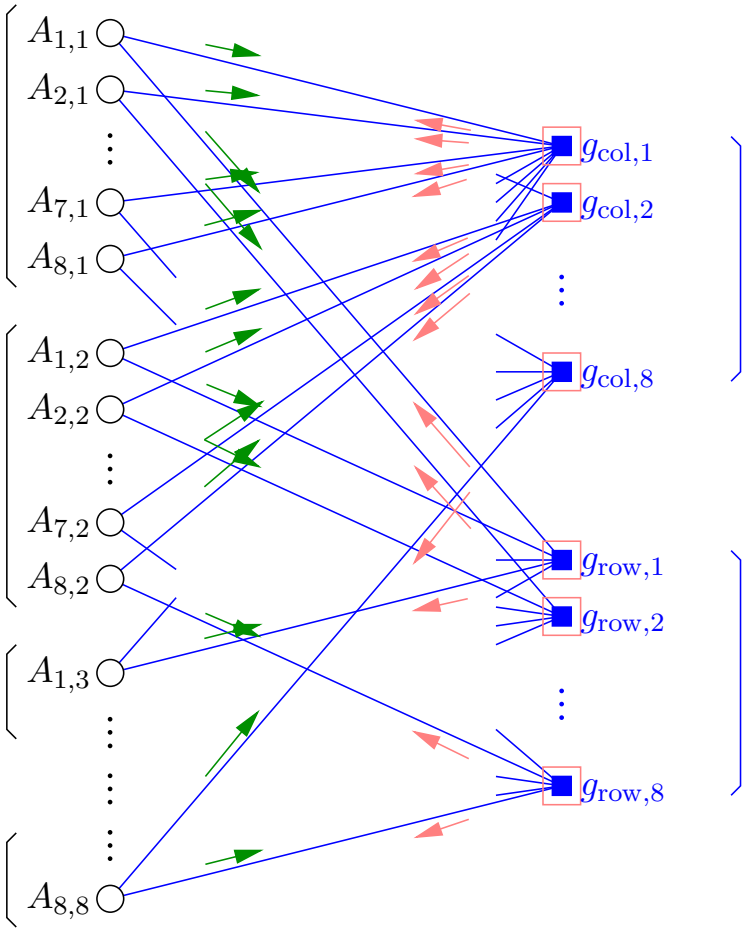
$$g(a_{1,1}, \ldots, a_{8,8}) = \prod_j g_{\text{col},j}(a_{1,j}, \ldots, a_{8,j}) \times \prod_i g_{\text{row},i}(a_{i,1}, \ldots, a_{i,8})$$

Total sum (partition function):

$$Z = \sum_{a_{1,1},\ldots,a_{8,8}} g(a_{1,1}, \ldots, a_{8,8})$$

Use of loopy belief propagation for approximating Z?

Comments on the Sum-Product Algorithm



Global function:

$$g(a_{1,1}, \ldots, a_{8,8}) = \prod_j g_{\text{col},j}(a_{1,j}, \ldots, a_{8,j}) \times \prod_i g_{\text{row},i}(a_{i,1}, \ldots, a_{i,8})$$

Total sum (partition function):

$$Z = \sum_{a_{1,1},\ldots,a_{8,8}} g(a_{1,1}, \ldots, a_{8,8})$$

Use of loopy belief propagation for approximating Z?

Comments on the Sum-Product Algorithm

- If the factor graph has no cycles, then it is obvious what messages have to be calculated when.
- If the factor graphs has cycles, then one has to decide what update schedule to take.
- Depending on the underlying semi-ring, one gets different versions of the sum-product algorithm.
  - For $\langle \mathbb{R}, +, \cdot \rangle$ one gets the sum-product algorithm. (This is the case discussed above.)
  - For $\langle \mathbb{R}^+, \max, \cdot \rangle$ one gets the max-product algorithm.
  - For $\langle \mathbb{R}, \min, + \rangle$ one gets the min-sum algorithm.
  - etc.

Partition function (total sum)

Partition Function

$$Z = \sum_{x_1, x_2, x_3, x_4, x_5} f(x_1, x_2, x_3, x_4, x_5)$$

Partition Function

$$Z = \sum_{x_1, x_2, x_3, x_4, x_5} f(x_1, x_2, x_3, x_4, x_5)$$

Recall:

$$\eta_{X_1}(x_1) = \sum_{x_2, x_3, x_4, x_5} f(x_1, x_2, x_3, x_4, x_5)$$

$$\eta_{X_2}(x_2) = \sum_{x_1, x_3, x_4, x_5} f(x_1, x_2, x_3, x_4, x_5)$$

$$\vdots \qquad\qquad \vdots$$

Partition Function

$$Z = \sum_{x_1,x_2,x_3,x_4,x_5} f(x_1, x_2, x_3, x_4, x_5)$$

Recall:

$$\eta_{X_1}(x_1) = \sum_{x_2,x_3,x_4,x_5} f(x_1, x_2, x_3, x_4, x_5)$$

$$\eta_{X_2}(x_2) = \sum_{x_1,x_3,x_4,x_5} f(x_1, x_2, x_3, x_4, x_5)$$

$$\vdots \qquad \vdots$$

Define:

$$Z_{X_1} = \sum_{x_1} \eta_{X_1}(x_1)$$

$$Z_{X_2} = \sum_{x_2} \eta_{X_2}(x_2)$$

$$\vdots \qquad \vdots$$

Partition Function

$$Z = \sum_{x_1,x_2,x_3,x_4,x_5} f(x_1, x_2, x_3, x_4, x_5)$$

Recall:

$$\eta_{X_1}(x_1) = \sum_{x_2,x_3,x_4,x_5} f(x_1, x_2, x_3, x_4, x_5)$$

$$\eta_{X_2}(x_2) = \sum_{x_1,x_3,x_4,x_5} f(x_1, x_2, x_3, x_4, x_5)$$

$$\vdots \qquad \vdots$$

Define:

$$Z_{X_1} = \sum_{x_1} \eta_{X_1}(x_1) = Z$$

$$Z_{X_2} = \sum_{x_2} \eta_{X_2}(x_2) = Z$$

$$\vdots \qquad \vdots$$

Partition Function

$$Z = \sum_{x_1, x_2, x_3, x_4, x_5} f(x_1, x_2, x_3, x_4, x_5)$$

Partition Function

$$Z = \sum_{x_1, x_2, x_3, x_4, x_5} f(x_1, x_2, x_3, x_4, x_5)$$

Recall:

$$\begin{aligned} &\vdots && \vdots \\ \eta_{f_C}(x_1, x_2, x_3) &= \sum_{x_4, x_5} f(x_1, x_2, x_3, x_4, x_5) \\ &\vdots && \vdots \end{aligned}$$

Partition Function

$$Z = \sum_{x_1, x_2, x_3, x_4, x_5} f(x_1, x_2, x_3, x_4, x_5)$$

Recall:

$$\vdots \qquad \vdots$$

$$\eta_{f_C}(x_1, x_2, x_3) = \sum_{x_4, x_5} f(x_1, x_2, x_3, x_4, x_5)$$

$$\vdots \qquad \vdots$$

Define:

$$\vdots \qquad \vdots$$

$$Z_{f_C} = \sum_{x_1, x_2, x_3} \eta_{f_C}(x_1, x_2, x_3)$$

$$\vdots \qquad \vdots$$

Partition Function

$$Z = \sum_{x_1, x_2, x_3, x_4, x_5} f(x_1, x_2, x_3, x_4, x_5)$$

Recall:

$$\vdots \qquad \vdots$$

$$\eta_{f_C}(x_1, x_2, x_3) = \sum_{x_4, x_5} f(x_1, x_2, x_3, x_4, x_5)$$

$$\vdots \qquad \vdots$$

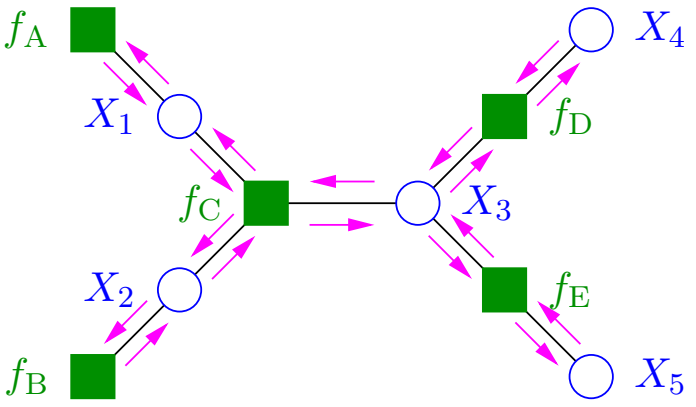
Define:

$$\vdots \qquad \vdots$$

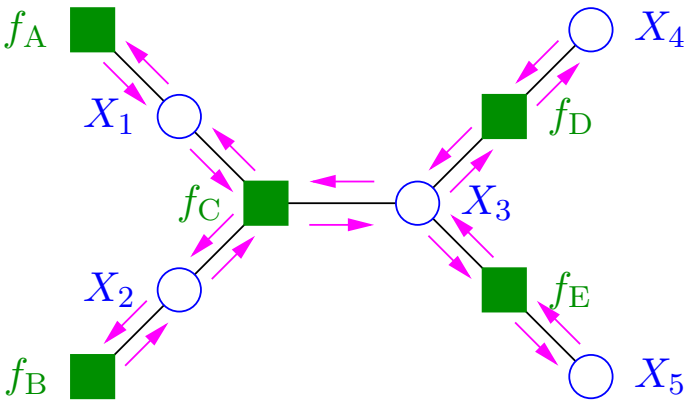
$$Z_{f_C} = \sum_{x_1, x_2, x_3} \eta_{f_C}(x_1, x_2, x_3) = Z$$

$$\vdots \qquad \vdots$$

Partition Sum

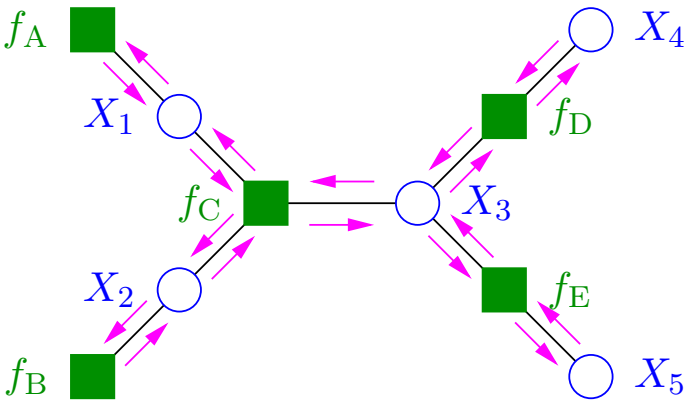


Partition Sum



$$Z = Z_{X_1} = Z_{X_2} = Z_{X_3} = Z_{X_4} = Z_{X_5} = Z_{f_A} = Z_{f_B} = Z_{f_C} = Z_{f_D} = Z_{f_E}$$

Partition Sum



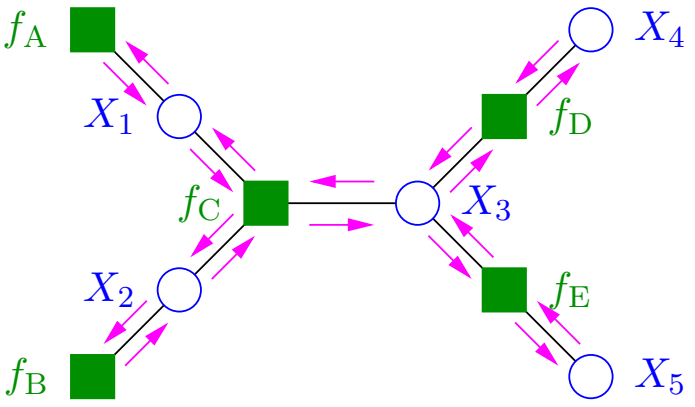
$$Z = Z_{X_1} = Z_{X_2} = Z_{X_3} = Z_{X_4} = Z_{X_5} = Z_{f_A} = Z_{f_B} = Z_{f_C} = Z_{f_D} = Z_{f_E}$$

Claim:

$$Z = \frac{Z_{f_A} \cdot Z_{f_B} \cdot Z_{f_C} \cdot Z_{f_D} \cdot Z_{f_E} \cdot Z_{X_1} \cdot Z_{X_2} \cdot Z_{X_3} \cdot Z_{X_4} \cdot Z_{X_5}}{Z_{X_1}^2 \cdot Z_{X_2}^2 \cdot Z_{X_3}^3 \cdot Z_{X_4}^1 \cdot Z_{X_5}^1}$$

(Note: exponents in denominator equal variable node degrees.)

Partition Sum



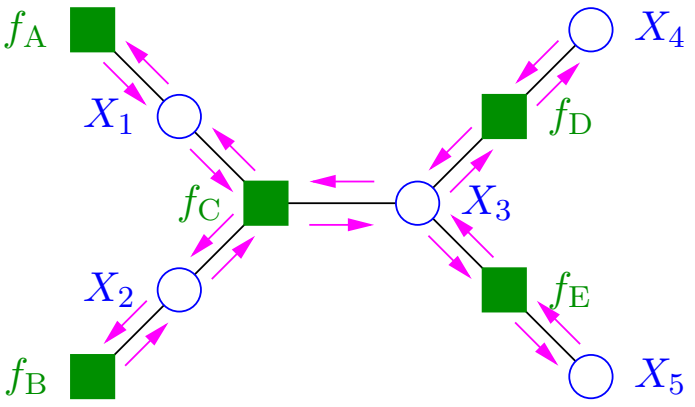
$$Z = Z_{X_1} = Z_{X_2} = Z_{X_3} = Z_{X_4} = Z_{X_5} = Z_{f_A} = Z_{f_B} = Z_{f_C} = Z_{f_D} = Z_{f_E}$$

Claim:

$$Z = \frac{Z^{\#\text{vertices}}}{Z^{\#\text{edges}}} = \frac{Z_{f_A} \cdot Z_{f_B} \cdot Z_{f_C} \cdot Z_{f_D} \cdot Z_{f_E} \cdot Z_{X_1} \cdot Z_{X_2} \cdot Z_{X_3} \cdot Z_{X_4} \cdot Z_{X_5}}{Z_{X_1}^2 \cdot Z_{X_2}^2 \cdot Z_{X_3}^3 \cdot Z_{X_4}^1 \cdot Z_{X_5}^1}$$

(Note: exponents in denominator equal variable node degrees.)

Partition Sum



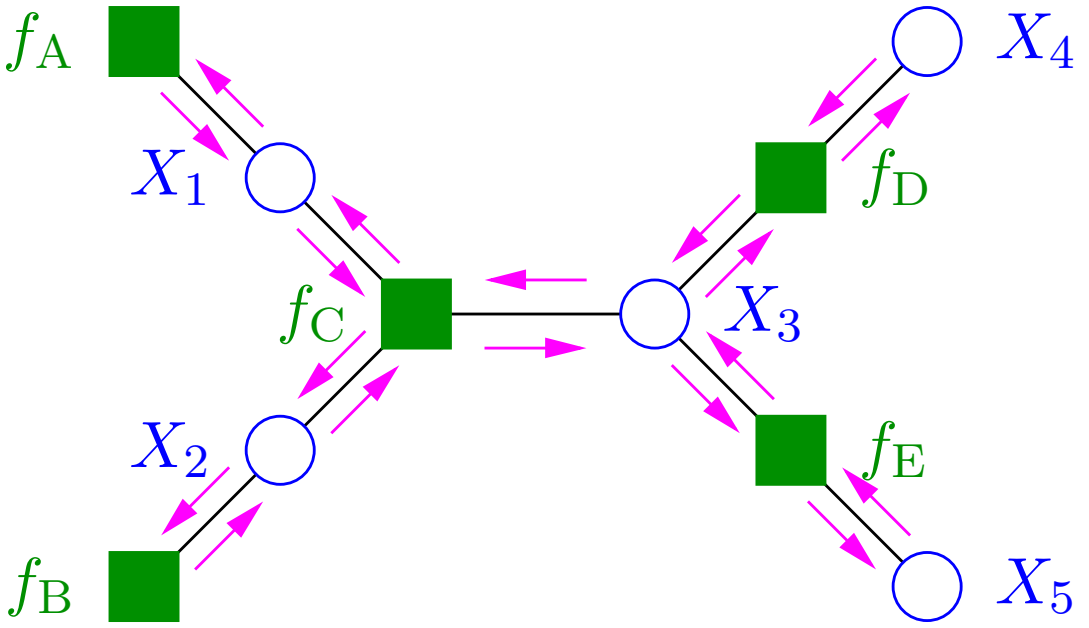
$$Z = Z_{X_1} = Z_{X_2} = Z_{X_3} = Z_{X_4} = Z_{X_5} = Z_{f_A} = Z_{f_B} = Z_{f_C} = Z_{f_D} = Z_{f_E}$$

Claim:

$$Z = \frac{Z^{\#\text{vertices}}}{Z^{\#\text{edges}}} = \frac{Z_{f_A} \cdot Z_{f_B} \cdot Z_{f_C} \cdot Z_{f_D} \cdot Z_{f_E} \cdot Z_{X_1} \cdot Z_{X_2} \cdot Z_{X_3} \cdot Z_{X_4} \cdot Z_{X_5}}{Z_{X_1}^2 \cdot Z_{X_2}^2 \cdot Z_{X_3}^3 \cdot Z_{X_4}^1 \cdot Z_{X_5}^1}$$

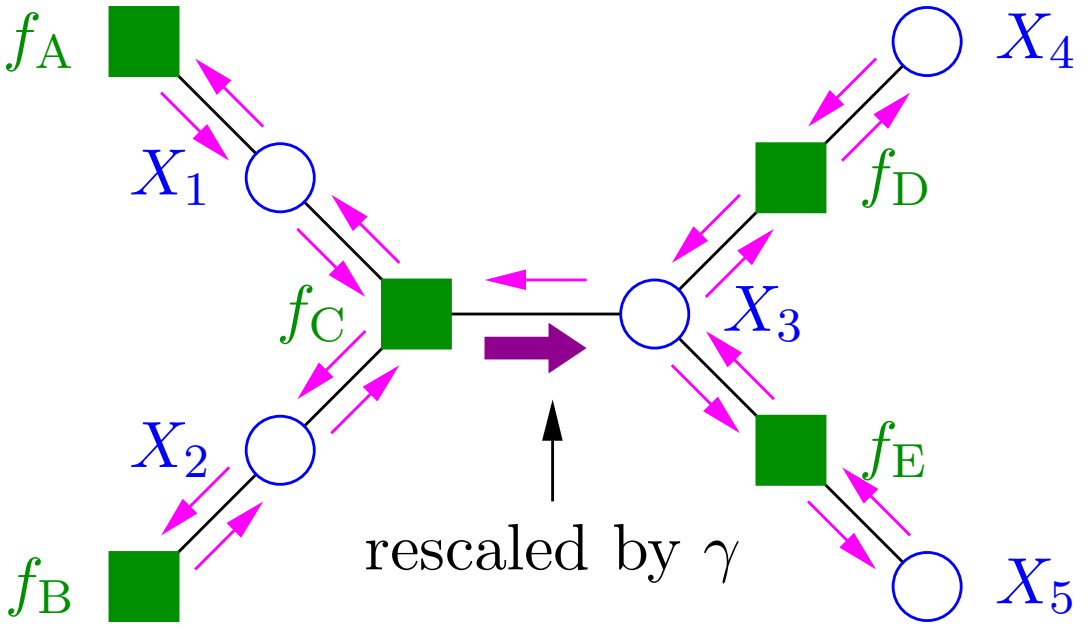
(Here we used the fact that for a graph with one component and no cycles it holds that $\#\text{vertices} = \#\text{edges} + 1$.)

Partition Sum



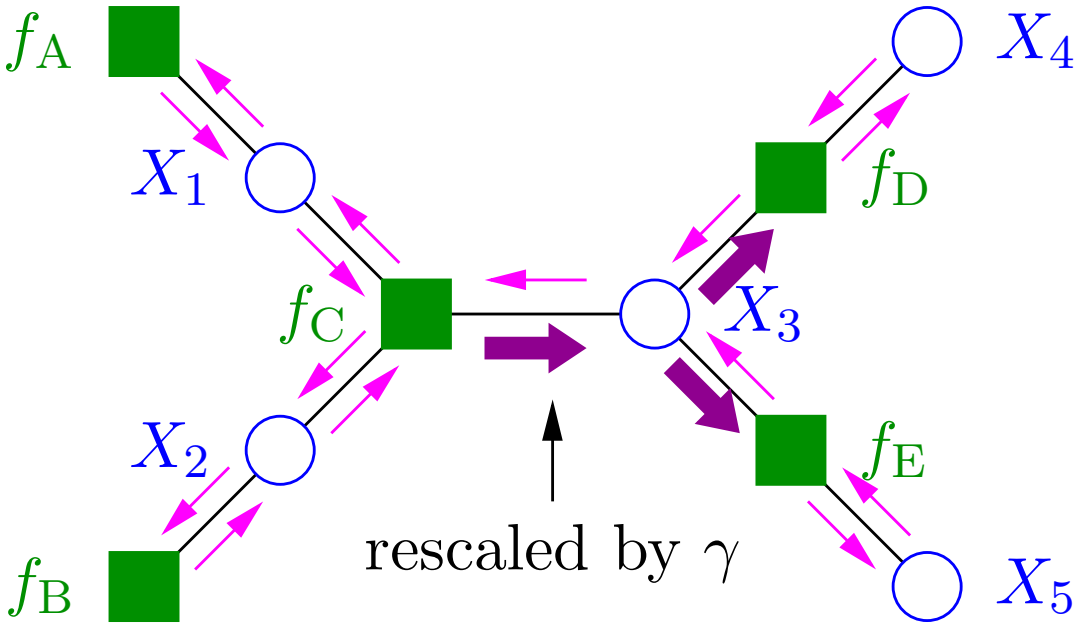
$$Z = \frac{Z_{f_A} \cdot Z_{f_B} \cdot Z_{f_C} \cdot Z_{f_D} \cdot Z_{f_E} \cdot Z_{X_1} \cdot Z_{X_2} \cdot Z_{X_3} \cdot Z_{X_4} \cdot Z_{X_5}}{Z_{X_1}^2 \cdot Z_{X_2}^2 \cdot Z_{X_3}^3 \cdot Z_{X_4}^1 \cdot Z_{X_5}^1}$$

Partition Sum



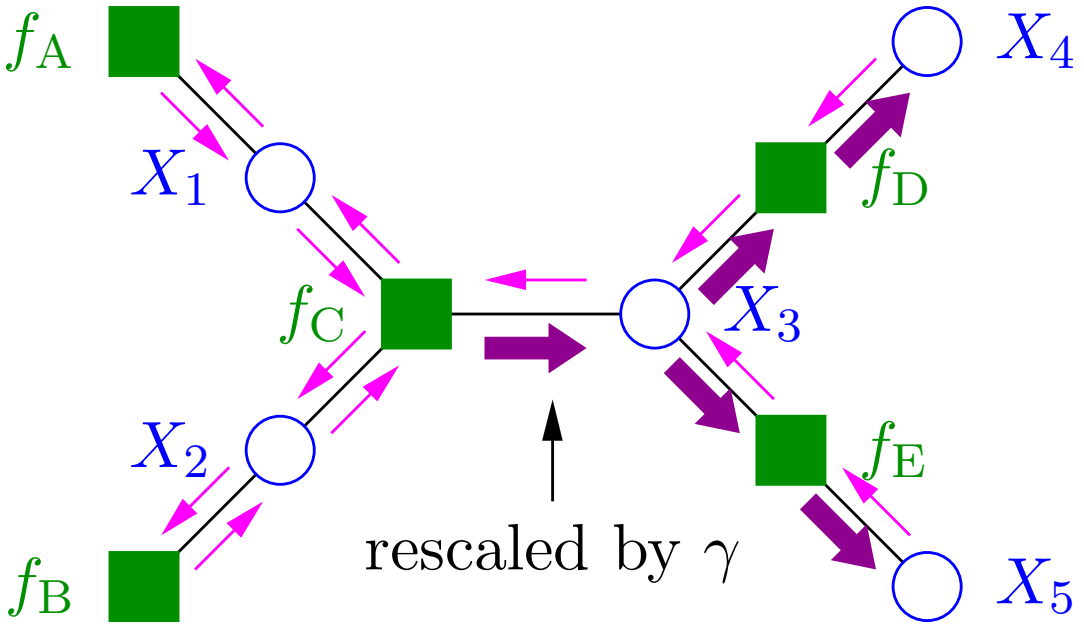
$$Z = \frac{Z_{f_A} \cdot Z_{f_B} \cdot Z_{f_C} \cdot Z_{f_D} \cdot Z_{f_E} \cdot Z_{X_1} \cdot Z_{X_2} \cdot Z_{X_3} \cdot Z_{X_4} \cdot Z_{X_5}}{Z_{X_1}^2 \cdot Z_{X_2}^2 \cdot Z_{X_3}^3 \cdot Z_{X_4}^1 \cdot Z_{X_5}^1}$$

Partition Sum



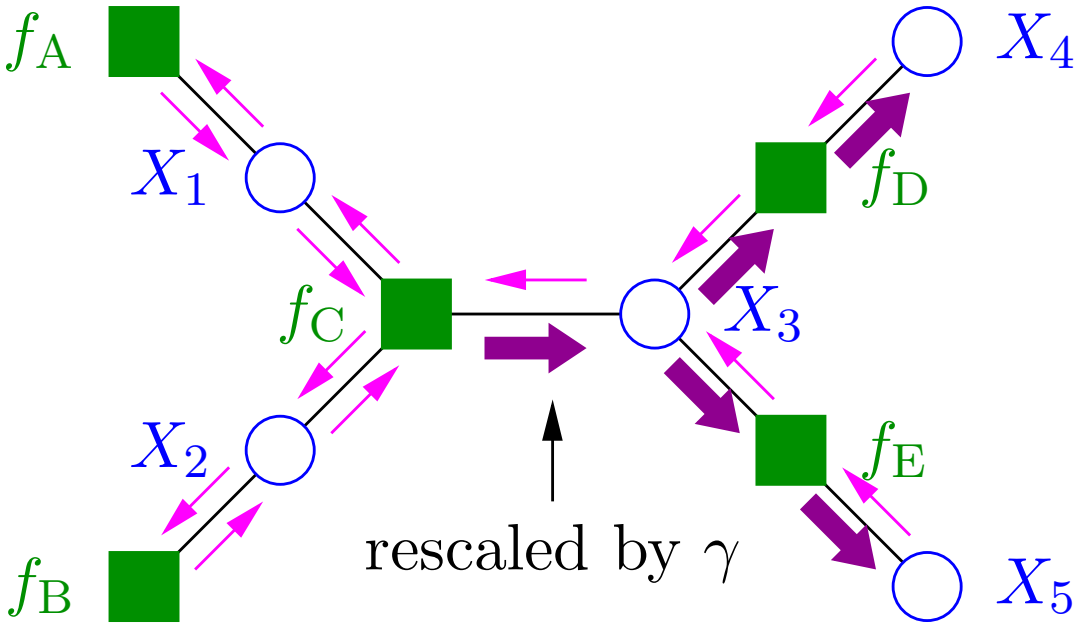
$$Z = \frac{Z_{f_A} \cdot Z_{f_B} \cdot Z_{f_C} \cdot Z_{f_D} \cdot Z_{f_E} \cdot Z_{X_1} \cdot Z_{X_2} \cdot Z_{X_3} \cdot Z_{X_4} \cdot Z_{X_5}}{Z_{X_1}^2 \cdot Z_{X_2}^2 \cdot Z_{X_3}^3 \cdot Z_{X_4}^1 \cdot Z_{X_5}^1}$$

Partition Sum



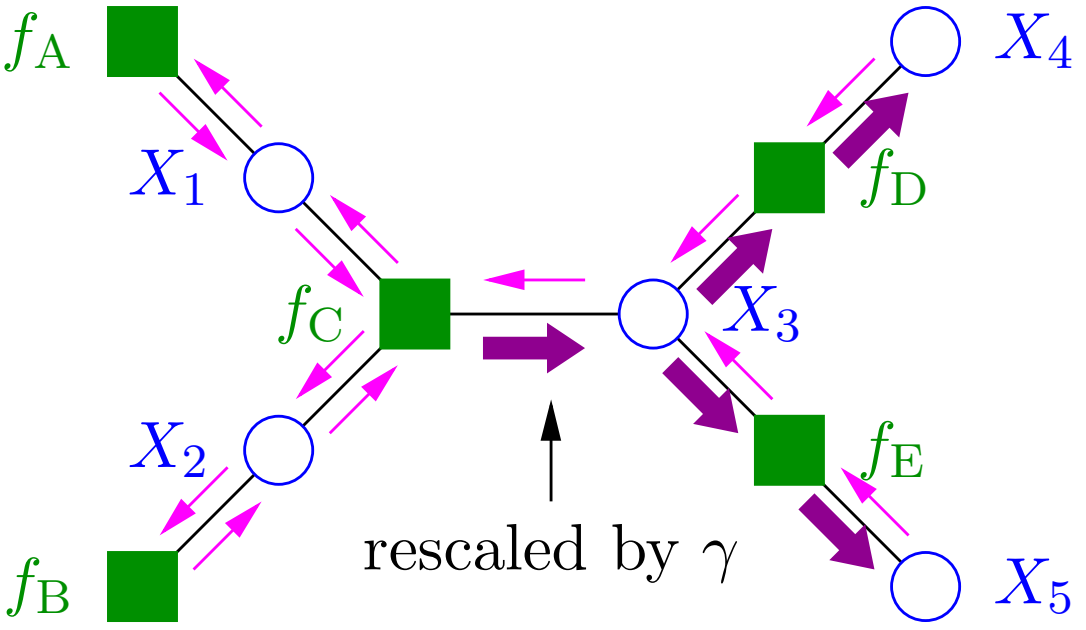
$$Z = \frac{Z_{f_A} \cdot Z_{f_B} \cdot Z_{f_C} \cdot Z_{f_D} \cdot Z_{f_E} \cdot Z_{X_1} \cdot Z_{X_2} \cdot Z_{X_3} \cdot Z_{X_4} \cdot Z_{X_5}}{Z_{X_1}^2 \cdot Z_{X_2}^2 \cdot Z_{X_3}^3 \cdot Z_{X_4}^1 \cdot Z_{X_5}^1}$$

Partition Sum



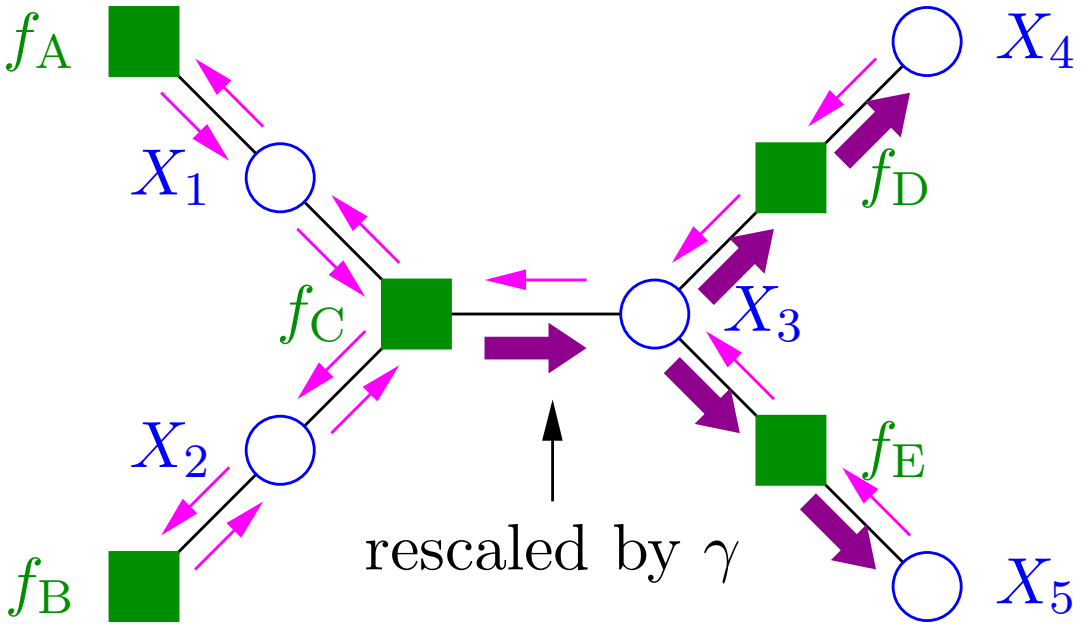
$$Z = \frac{Z_{f_A} \cdot Z_{f_B} \cdot Z_{f_C} \cdot \boxed{\hat{Z}_{f_D}} \cdot \boxed{\hat{Z}_{f_E}} \cdot Z_{X_1} \cdot Z_{X_2} \cdot \boxed{\hat{Z}_{X_3}} \cdot \boxed{\hat{Z}_{X_4}} \cdot \boxed{\hat{Z}_{X_5}}}{Z_{X_1}^2 \cdot Z_{X_2}^2 \cdot \boxed{\hat{Z}_{X_3}^3} \cdot \boxed{\hat{Z}_{X_4}^1} \cdot \boxed{\hat{Z}_{X_5}^1}}$$

Partition Sum



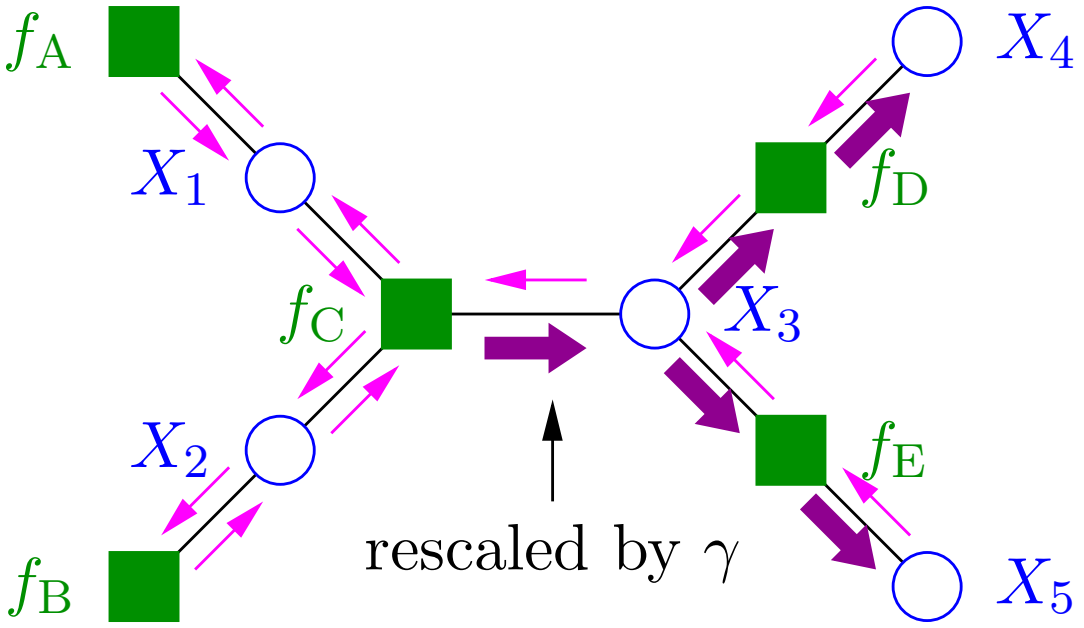
$$Z = \frac{Z_{f_A} \cdot Z_{f_B} \cdot Z_{f_C} \cdot \boxed{\gamma Z_{f_D}} \cdot \boxed{\gamma Z_{f_E}} \cdot Z_{X_1} \cdot Z_{X_2} \cdot \boxed{\gamma Z_{X_3}} \cdot \boxed{\gamma Z_{X_4}} \cdot \boxed{\gamma Z_{X_5}}}{Z_{X_1}^2 \cdot Z_{X_2}^2 \cdot \boxed{\gamma^3 Z_{X_3}^3} \cdot \boxed{\gamma^1 Z_{X_4}^1} \cdot \boxed{\gamma^1 Z_{X_5}^1}}$$

Partition Sum



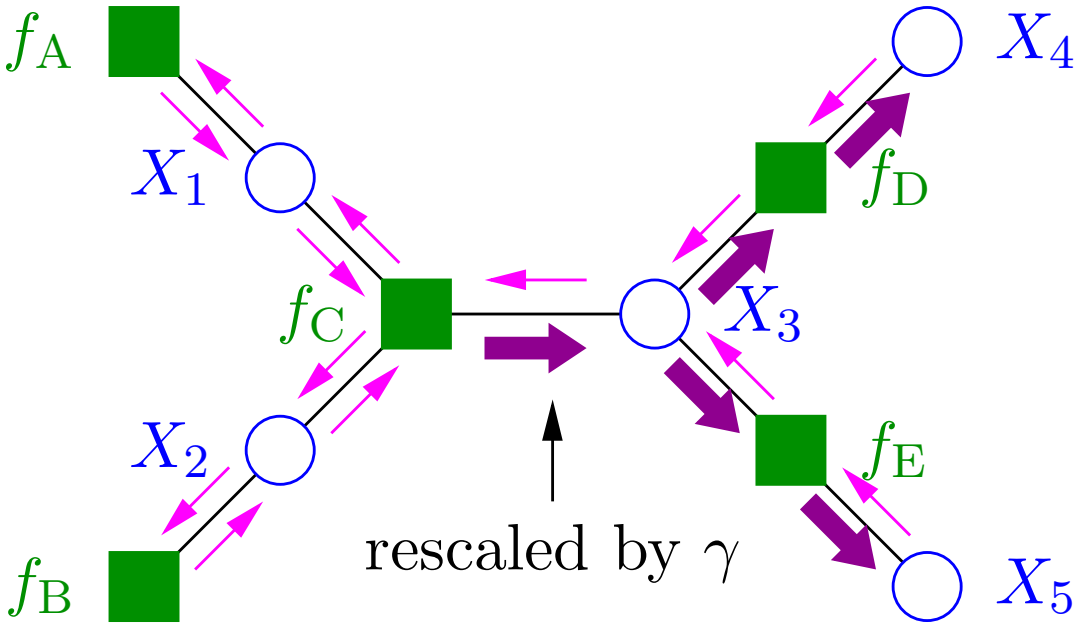
$$Z = \frac{\gamma^5}{\gamma^5} \cdot \frac{Z_{f_A} \cdot Z_{f_B} \cdot Z_{f_C} \cdot Z_{f_D} \cdot Z_{f_E} \cdot Z_{X_1} \cdot Z_{X_2} \cdot Z_{X_3} \cdot Z_{X_4} \cdot Z_{X_5}}{Z_{X_1}^2 \cdot Z_{X_2}^2 \cdot Z_{X_3}^3 \cdot Z_{X_4}^1 \cdot Z_{X_5}^1}$$

Partition Sum



$$Z = \frac{Z_{f_A} \cdot Z_{f_B} \cdot Z_{f_C} \cdot Z_{f_D} \cdot Z_{f_E} \cdot Z_{X_1} \cdot Z_{X_2} \cdot Z_{X_3} \cdot Z_{X_4} \cdot Z_{X_5}}{Z_{X_1}^2 \cdot Z_{X_2}^2 \cdot Z_{X_3}^3 \cdot Z_{X_4}^1 \cdot Z_{X_5}^1}$$

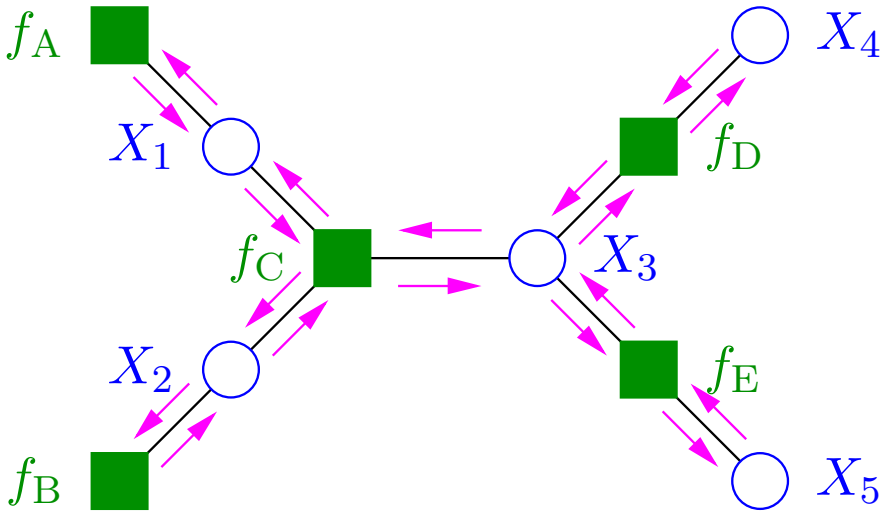
Partition Sum



$$Z = \frac{Z_{f_A} \cdot Z_{f_B} \cdot Z_{f_C} \cdot Z_{f_D} \cdot Z_{f_E} \cdot Z_{X_1} \cdot Z_{X_2} \cdot Z_{X_3} \cdot Z_{X_4} \cdot Z_{X_5}}{Z_{X_1}^2 \cdot Z_{X_2}^2 \cdot Z_{X_3}^3 \cdot Z_{X_4}^1 \cdot Z_{X_5}^1}$$

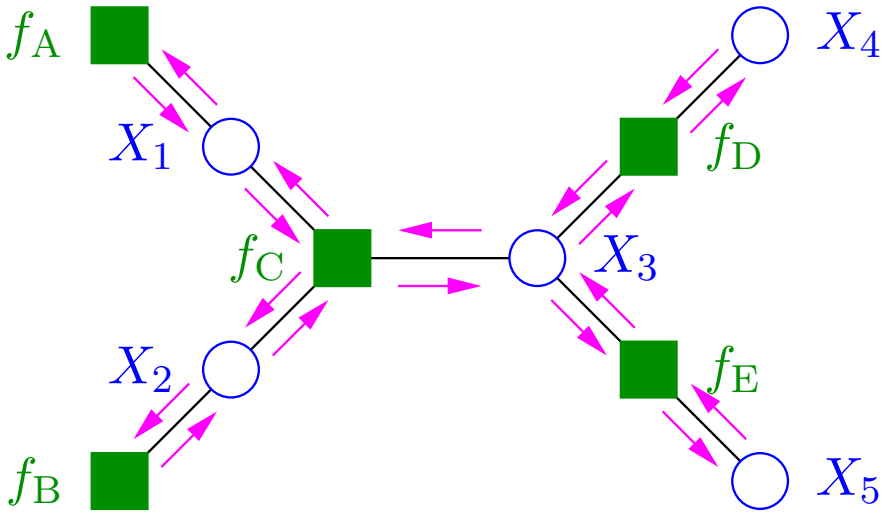
Remarkable: this expression is invariant to rescaling of *function-node-to-variable-node* messages!

Partition Sum



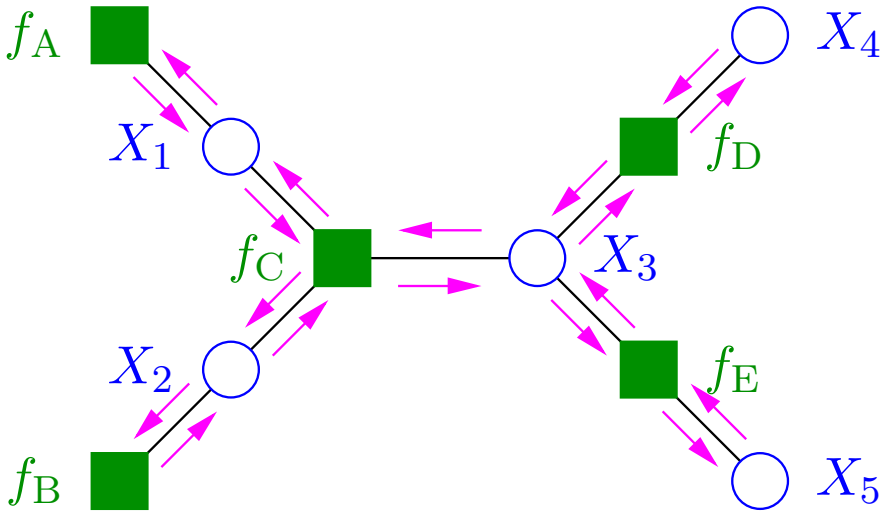
$$Z = \frac{Z_{f_A} \cdot Z_{f_B} \cdot Z_{f_C} \cdot Z_{f_D} \cdot Z_{f_E} \cdot Z_{X_1} \cdot Z_{X_2} \cdot Z_{X_3} \cdot Z_{X_4} \cdot Z_{X_5}}{Z_{X_1}^2 \cdot Z_{X_2}^2 \cdot Z_{X_3}^3 \cdot Z_{X_4}^1 \cdot Z_{X_5}^1}$$

Partition Sum



$$Z = \frac{\prod_f Z_f \cdot \prod_X Z_X}{\prod_X Z_X^{\deg(X)}}$$

Partition Sum

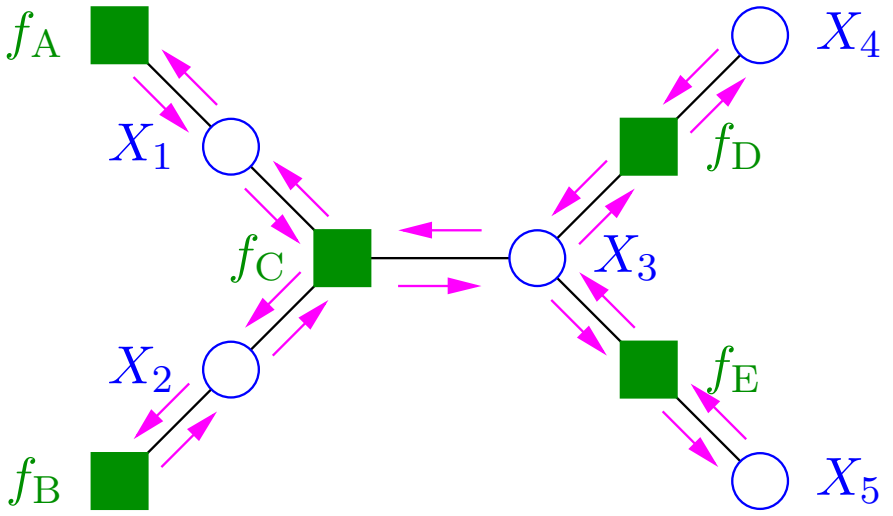


$$Z = \frac{\prod_f Z_f \cdot \prod_X Z_X}{\prod_X Z_X^{\deg(X)}}$$

Bethe approximation:

Use the above type of expression also when factor graph has cycles.

Partition Sum



$$Z = \frac{\prod_f Z_f \cdot \prod_X Z_X}{\prod_X Z_X^{\deg(X)}}$$

Bethe approximation:

Use the above type of expression also when factor graph has cycles.

$$\rightarrow Z'_{\text{Bethe}}$$

Bethe Partition Sum

Bethe Partition Sum

- Basically, we can evaluate the expresion for Z'_{Bethe} at any iteration of the SPA.

Bethe Partition Sum

- Basically, we can evaluate the expresion for Z'_{Bethe} at any iteration of the SPA.
- Factor graph without cycles:

  We have $Z'_{\mathrm{Bethe}} = Z$ only at a fixed point of the SPA.

Bethe Partition Sum

- Basically, we can evaluate the expresion for Z'_{Bethe} at any iteration of the SPA.
- Factor graph without cycles:

  We have $Z'_{\mathrm{Bethe}} = Z$ only at a fixed point of the SPA.

- Factor graph with cycles:

  Therefore, we call Z'_{Bethe} a (local) Bethe partition function only if we are at a fixed point of the SPA.

Bethe Partition Sum

- Basically, we can evaluate the expresion for Z'_{Bethe} at any iteration of the SPA.
- Factor graph without cycles:

  We have $Z'_{\text{Bethe}} = Z$ only at a fixed point of the SPA.

- Factor graph with cycles:

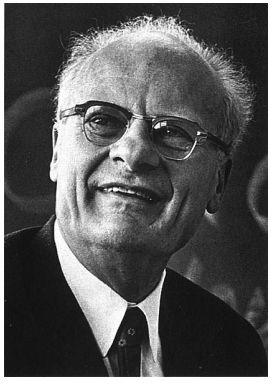
  Therefore, we call Z'_{Bethe} a (local) Bethe partition function only if we are at a fixed point of the SPA.

- Factor graph with cycles: the SPA can have multiple fixed points. We define the Bethe partition sum to be

$$Z_{\text{Bethe}} \triangleq \max_{\text{fixed points of SPA}} Z'_{\text{Bethe}}.$$

Bethe and Peierls

The Bethe approximation is sometimes also called Bethe–Peierls approximation.



Hans A. Bethe (1906–2005)

H. A. Bethe, "Statistical theory of superlattices," Proc. Royal Society London A, vol. 150, no. 871, pp. 552–575, Jul. 1935.

Nobel Prize in Physics 1967 for his work on the theory of stellar nucleosynthesis.

Rudolf Peierls (1907–1995)

R. Peierls, "On Ising's model of ferromagnetism," Math. Proc. Cambr. Phil. Soc., vol. 32, no. 3, pp. 477-481, Oct. 1936.

Bethe and Peierls

The Bethe approximation is sometimes also called Bethe–Peierls approximation.

Hans A. Bethe (1906–2005)

H. A. Bethe, "Statistical theory of superlattices," Proc. Royal Society London A, vol. 150, no. 871, pp. 552–575, Jul. 1935.

Nobel Prize in Physics 1967 for his work on the theory of stellar nucleosynthesis.

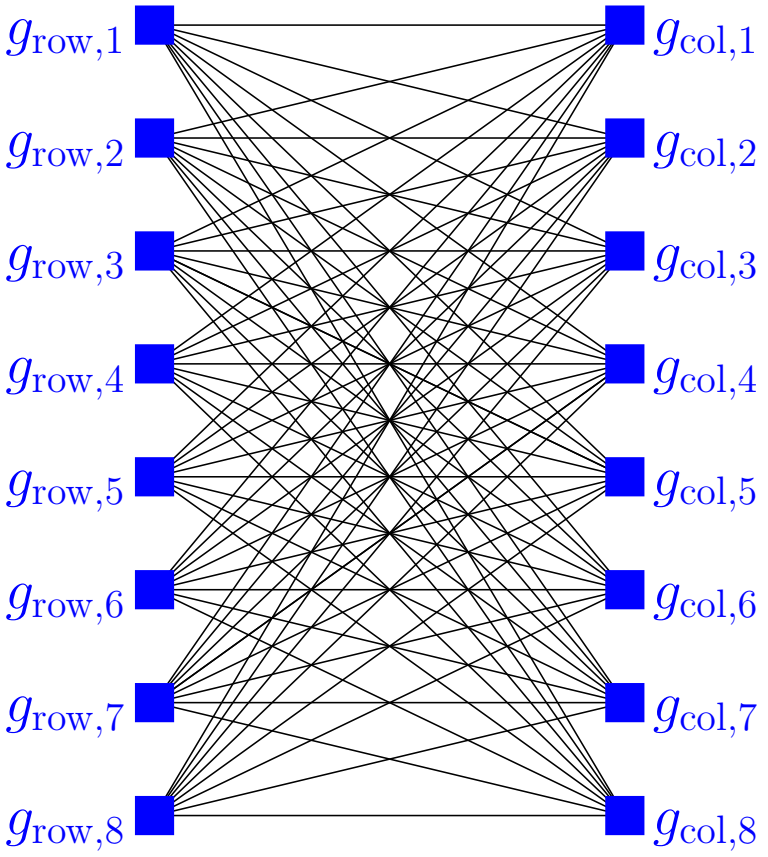
Rudolf Peierls (1907–1995)

R. Peierls, "On Ising's model of ferromagnetism," Math. Proc. Cambr. Phil. Soc., vol. 32, no. 3, pp. 477-481, Oct. 1936.

Curious fact: the abbreviation of Bethe–Peierls is BP.

(Recall that Belief Propagation, another name for the SPA, is also abbreviated BP.)

Graphical Model for Permanent



(function nodes are suitably defined based on $\boldsymbol{\theta}$)

(variable nodes have been omitted)

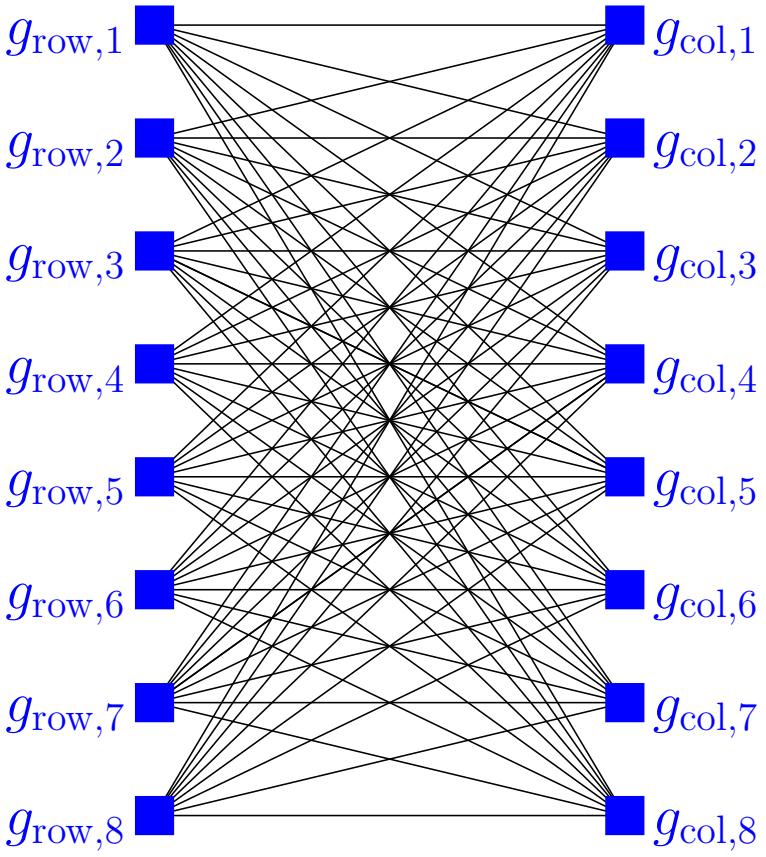
Global function:

$$g(a_{1,1}, \ldots, a_{8,8}) = \prod_j g_{\text{col},j}(a_{1,j}, \ldots, a_{8,j}) \times \prod_i g_{\text{row},i}(a_{i,1}, \ldots, x_{i,8})$$

Permanent:

$$\text{perm}(\boldsymbol{\theta}) = Z = \sum_{a_{1,1},\ldots,a_{8,8}} g(a_{1,1}, \ldots, a_{8,8})$$

Graphical Model for Permanent



(function nodes are suitably defined based on $\boldsymbol{\theta}$)

(variable nodes have been omitted)

Global function:

$$g(a_{1,1}, \ldots, a_{8,8})$$
$$= \prod_j g_{\text{col},j}(a_{1,j}, \ldots, a_{8,j}) \times$$
$$\prod_i g_{\text{row},i}(a_{i,1}, \ldots, x_{i,8})$$

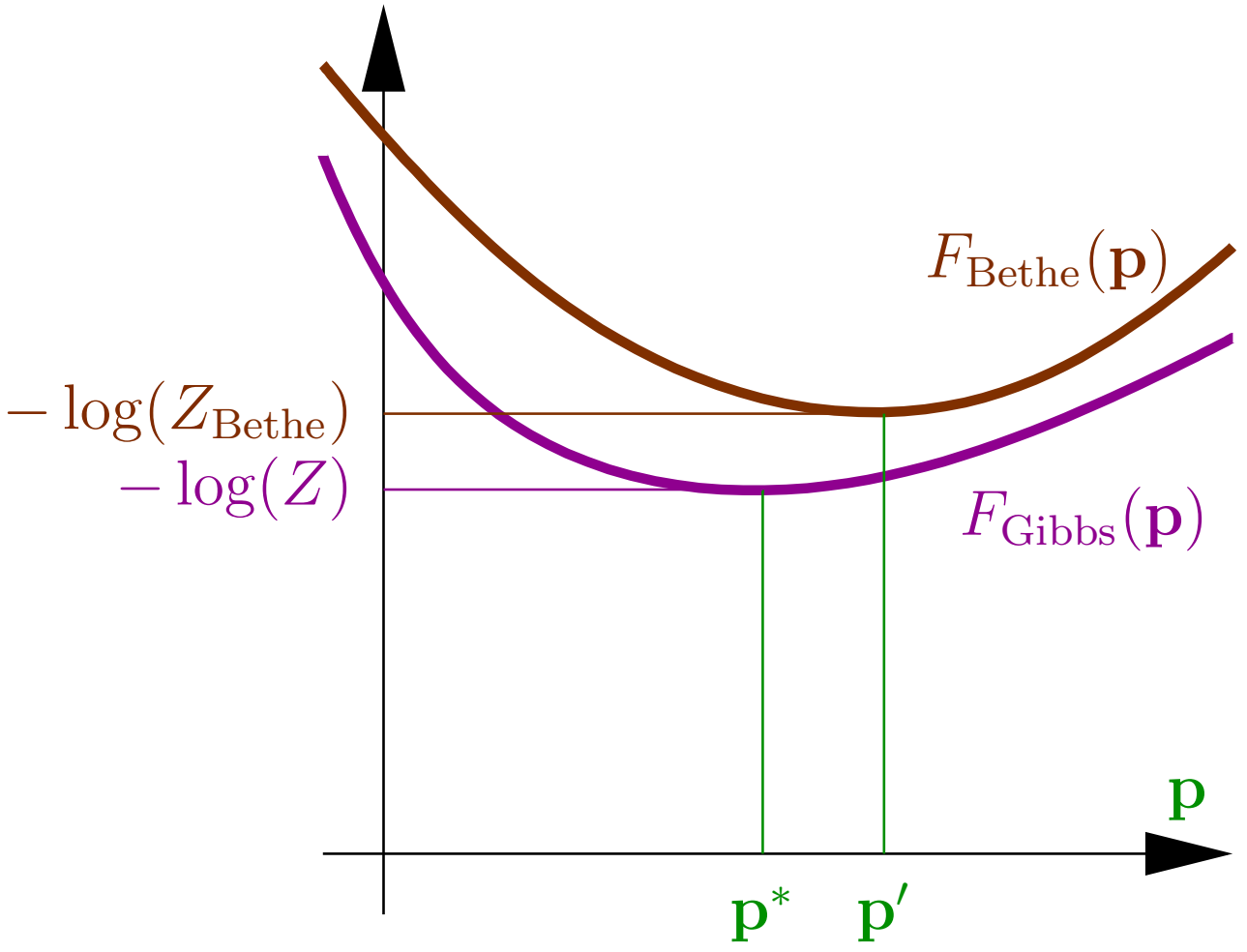
Bethe Permanent:

$$\text{perm}_{\text{B}}(\boldsymbol{\theta}) \triangleq Z_{\text{Bethe}}$$

Comments on the

Bethe approximation of the partition sum

Gibbs vs. Bethe Free Energy Function



Note: this picture is not quite correct as, strictly speaking, the optimization of F_{Gibbs} is over a different domain than F_{Bethe}.

Bethe Approximation

In general, the approach of replacing the Gibbs free energy by the Bethe free energy comes with very few guarantees:

Bethe Approximation

In general, the approach of replacing the Gibbs free energy by the Bethe free energy comes with very few guarantees:

- The Bethe free energy function might have multiple local minima.

Bethe Approximation

In general, the approach of replacing the Gibbs free energy by the Bethe free energy comes with very few guarantees:

- The Bethe free energy function might have multiple local minima.
- It is unclear how close the (global) minimum of the Bethe free energy is to the minimum of the Gibbs free energy.

Bethe Approximation

In general, the approach of replacing the Gibbs free energy by the Bethe free energy comes with very few guarantees:

- The Bethe free energy function might have multiple local minima.
- It is unclear how close the (global) minimum of the Bethe free energy is to the minimum of the Gibbs free energy.
- It is unclear if the sum-product algorithm converges (even to a local minimum of the Bethe free energy).

Bethe Approximation

Luckily, in the case of the permanent approximation problem, the above-mentioned factor graph $\mathrm{N}(\theta)$ is such that the Bethe free energy function is very well behaved. In particular, one can show that:

Bethe Approximation

Luckily, in the case of the permanent approximation problem, the above-mentioned factor graph $\mathrm{N}(\theta)$ is such that the Bethe free energy function is very well behaved. In particular, one can show that:

- The Bethe free energy function (for a suitable parametrization) is convex and therefore has no local minima [V., 2010, 2013].

Bethe Approximation

Luckily, in the case of the permanent approximation problem, the above-mentioned factor graph $\mathrm{N}(\theta)$ is such that the Bethe free energy function is very well behaved. In particular, one can show that:

- The Bethe free energy function (for a suitable parametrization) is convex and therefore has no local minima [V., 2010, 2013].
- The minimum of the Bethe free energy is quite close to the minimum of the Gibbs free energy. *(More details later.)*

Bethe Approximation

Luckily, in the case of the permanent approximation problem, the above-mentioned factor graph $\mathrm{N}(\theta)$ is such that the Bethe free energy function is very well behaved. In particular, one can show that:

- The Bethe free energy function (for a suitable parametrization) is convex and therefore has no local minima [V., 2010, 2013].
- The minimum of the Bethe free energy is quite close to the minimum of the Gibbs free energy. *(More details later.)*
- The sum-product algorithm converges to the minimum of the Bethe free energy. *(More details later.)*

Relationship between Permanent and Bethe Permanent

Theorem (Gurvits, 2011) **Theorem (Anari and Rezaei, 2019)**

$$\begin{array}{ccccc} & \downarrow & & \downarrow & \\ \mathrm{perm}_{\mathrm{B}}(\boldsymbol{\theta}) & \leq & \mathrm{perm}(\boldsymbol{\theta}) & \leq & \sqrt{2}^{n} \cdot \mathrm{perm}_{\mathrm{B}}(\boldsymbol{\theta}) \end{array}$$

Relationship between Permanent and Bethe Permanent

Theorem (Gurvits, 2011) **Theorem (Anari and Rezaei, 2019)**

$$\begin{array}{ccccc} & \downarrow & & \downarrow & \\ \mathrm{perm}_{\mathrm{B}}(\boldsymbol{\theta}) & \leq & \mathrm{perm}(\boldsymbol{\theta}) & \leq & \sqrt{2}^{\,n} \cdot \mathrm{perm}_{\mathrm{B}}(\boldsymbol{\theta}) \end{array}$$

This can be rewritten as follows:

$$\frac{1}{n}\log \mathrm{perm}_{\mathrm{B}}(\boldsymbol{\theta}) \;\leq\; \frac{1}{n}\log \mathrm{perm}(\boldsymbol{\theta}) \;\leq\; \frac{1}{n}\log \mathrm{perm}_{\mathrm{B}}(\boldsymbol{\theta}) + \log(\sqrt{2})$$

Sum-Product Algorithm Convergence

Theorem: Modulo some minor technical conditions on the initial messages, the sum-product algorithm converges to the (global) minimum of the Bethe free energy function [V., 2010, 2013].

Sum-Product Algorithm Convergence

Theorem: Modulo some minor technical conditions on the initial messages, the sum-product algorithm converges to the (global) minimum of the Bethe free energy function [V., 2010, 2013].

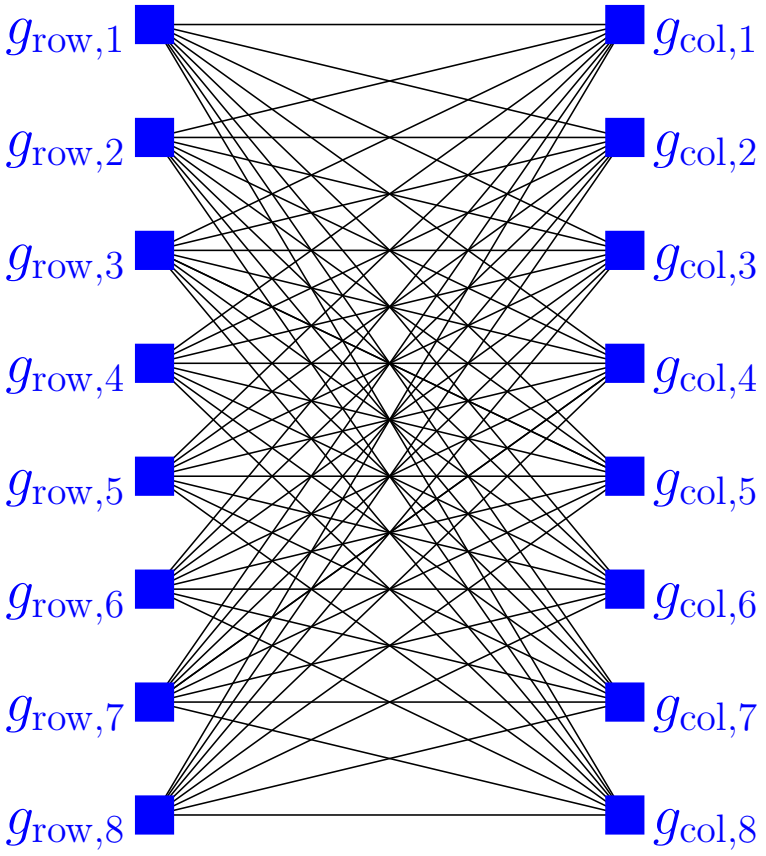
Comment: the first part of the proof of the above theorem is very similar to the SPA convergence proof in

Bayati and Nair, "*A rigorous proof of the cavity method for counting matchings*," Allerton 2006.

Note that they consider matchings, not perfect matchings. (Although the perfect matching case can be seen as a limiting case of the matching setup, the convergence proof of the SPA is incomplete for that case.)

Beyond the permanent setup

Beyond the permanent setup



(variable nodes have been omitted)

Global function:

$$g(a_{1,1}, \ldots, a_{n,n}) = \prod_j g_{\text{col},j}(a_{1,j}, \ldots, a_{n,j}) \times \prod_i g_{\text{row},i}(a_{i,1}, \ldots, a_{i,n})$$

Partition sum:

$$Z = \sum_{a_{1,1},\ldots,a_{n,n}} g(a_{1,1}, \ldots, a_{n,n})$$

Beyond the permanent setup

For $j \in \{1, \ldots, n\}$, define the polynomial

$$G_{\mathrm{col},j}(\mathsf{X}_{1,j}, \ldots, \mathsf{X}_{n,j}) \triangleq \sum_{a_{1,j},\ldots,a_{n,j}} g_{\mathrm{col},j}(a_{1,j}, \ldots, a_{n,j}) \cdot \mathsf{X}_{1,j}^{a_{1,j}} \cdots \mathsf{X}_{n,j}^{a_{n,j}} .$$

For $i \in \{1, \ldots, n\}$, define the polynomial

$$G_{\mathrm{row},i}(\mathsf{X}_{i,1}, \ldots, \mathsf{X}_{i,n}) \triangleq \sum_{a_{i,1},\ldots,a_{i,n}} g_{\mathrm{row},i}(a_{i,1}, \ldots, a_{i,n}) \cdot \mathsf{X}_{i,1}^{a_{i,1}} \cdots \mathsf{X}_{i,n}^{a_{i,n}} .$$

Beyond the permanent setup

For $j \in \{1, \ldots, n\}$, define the polynomial

$$G_{\mathrm{col},j}(\mathsf{X}_{1,j}, \ldots, \mathsf{X}_{n,j}) \triangleq \sum_{a_{1,j},\ldots,a_{n,j}} g_{\mathrm{col},j}(a_{1,j}, \ldots, a_{n,j}) \cdot \mathsf{X}_{1,j}^{a_{1,j}} \cdots \mathsf{X}_{n,j}^{a_{n,j}} .$$

For $i \in \{1, \ldots, n\}$, define the polynomial

$$G_{\mathrm{row},i}(\mathsf{X}_{i,1}, \ldots, \mathsf{X}_{i,n}) \triangleq \sum_{a_{i,1},\ldots,a_{i,n}} g_{\mathrm{row},i}(a_{i,1}, \ldots, a_{i,n}) \cdot \mathsf{X}_{i,1}^{a_{i,1}} \cdots \mathsf{X}_{i,n}^{a_{i,n}} .$$

For the permanent setup, we obtain the following polynomials.

For $j \in \{1, \ldots, n\}$, we get

$$G_{\mathrm{col},j}(\mathsf{X}_{1,j}, \ldots, \mathsf{X}_{n,j}) = \sqrt{\theta_{1,j}} \cdot \mathsf{X}_{1,j} + \cdots + \sqrt{\theta_{n,j}} \cdot \mathsf{X}_{n,j} .$$

For $i \in \{1, \ldots, n\}$, we get

$$G_{\mathrm{row},i}(\mathsf{X}_{i,1}, \ldots, \mathsf{X}_{i,n}) = \sqrt{\theta_{i,1}} \cdot \mathsf{X}_{i,1} + \cdots + \sqrt{\theta_{i,n}} \cdot \mathsf{X}_{i,n} .$$

Beyond the permanent setup

For $j \in \{1, \ldots, n\}$, define the polynomial

$$G_{\mathrm{col},j}(\mathsf{X}_{1,j}, \ldots, \mathsf{X}_{n,j}) \triangleq \sum_{a_{1,j},\ldots,a_{n,j}} g_{\mathrm{col},j}(a_{1,j}, \ldots, a_{n,j}) \cdot \mathsf{X}_{1,j}^{a_{1,j}} \cdots \mathsf{X}_{n,j}^{a_{n,j}} .$$

For $i \in \{1, \ldots, n\}$, define the polynomial

$$G_{\mathrm{row},i}(\mathsf{X}_{i,1}, \ldots, \mathsf{X}_{i,n}) \triangleq \sum_{a_{i,1},\ldots,a_{i,n}} g_{\mathrm{row},i}(a_{i,1}, \ldots, a_{i,n}) \cdot \mathsf{X}_{i,1}^{a_{i,1}} \cdots \mathsf{X}_{i,n}^{a_{i,n}} .$$

Many of the above results about Z_{Bethe} for the permanent setup can be generalized when the

local functions $g_{\mathrm{col},j}(a_{1,j}, \ldots, a_{n,j})$ and $g_{\mathrm{row},i}(a_{i,1}, \ldots, a_{i,n})$

are such that

$G_{\mathrm{col},j}(\mathsf{X}_{1,j}, \ldots, \mathsf{X}_{n,j})$ and $G_{\mathrm{row},i}(\mathsf{X}_{i,1}, \ldots, \mathsf{X}_{i,n})$

are multi-affine (homogeneous) real stable polynomials.

Real stable polynomials

Real stable polynomials

Definition: Consider a polynomial $h(\mathsf{Z}) = h(\mathsf{Z}_1, \ldots, \mathsf{Z}_n) \in \mathbb{C}\big[\mathsf{Z}_1, \ldots, \mathsf{Z}_n\big]$.

- We say that $h(\mathsf{Z})$ is $\mathcal{H}$-stable if

$$h(z_1, \ldots, z_n) \neq 0 \quad \text{for all} \quad (z_1, \ldots z_n) \in \mathcal{H}^n \ ,$$

where $\mathcal{H}$ is the (open) upper half plane of the complex plane, i.e.,

$$\mathcal{H} \ \triangleq \ \big\{\zeta \in \mathbb{C} \ \big| \ \mathrm{Im}(\zeta) > 0\big\} \ .$$

Real stable polynomials

Definition: Consider a polynomial $h(\mathsf{Z}) = h(\mathsf{Z}_1, \ldots, \mathsf{Z}_n) \in \mathbb{C}\big[\mathsf{Z}_1, \ldots, \mathsf{Z}_n\big]$.

- We say that $h(\mathsf{Z})$ is $\mathcal{H}$-stable if

$$h(z_1, \ldots, z_n) \neq 0 \quad \text{for all} \quad (z_1, \ldots z_n) \in \mathcal{H}^n \,,$$

where $\mathcal{H}$ is the (open) upper half plane of the complex plane, i.e.,

$$\mathcal{H} \;\triangleq\; \big\{\zeta \in \mathbb{C} \;\big|\; \mathrm{Im}(\zeta) > 0\big\} \,.$$

Example: Consider the polynomial $h(\mathsf{Z}) = h(\mathsf{Z}_1, \mathsf{Z}_2) = 1 - \mathsf{Z}_1\mathsf{Z}_2$.

We claim that this polynomial is $\mathcal{H}$-stable. Indeed, the expression

$$1 - z_1 z_2 \;=\; 0$$

can be rewritten as

$$z_1 \;=\; z_2^{-1} \,.$$

However, this means that if $z_2 \in \mathcal{H}$, then $z_1 \notin \mathcal{H}$.

Real stable polynomials

Motivation for stable polynomials:

- Consider the ordinary differential equation

$$a_n \cdot \left(\frac{\mathrm{d}}{\mathrm{d}t}\right)^n y(t) \;+\; \cdots \;+\; a_1 \cdot \frac{\mathrm{d}}{\mathrm{d}t} y(t) \;+\; a_0 \cdot y(t) \;=\; 0\,, \quad t \geq 0\,,$$

  with suitable initial conditions at $t = 0$.

Real stable polynomials

Motivation for stable polynomials:

- Consider the ordinary differential equation

$$a_n \cdot \left(\frac{\mathrm{d}}{\mathrm{d}t}\right)^n y(t) + \cdots + a_1 \cdot \frac{\mathrm{d}}{\mathrm{d}t} y(t) + a_0 \cdot y(t) = 0\,, \quad t \geq 0\,,$$

  with suitable initial conditions at $t = 0$.

  Will $|y(t)|$ go to infinity, stay bounded, or go to zero as $t \to \infty$?

Real stable polynomials

Motivation for stable polynomials:

- Consider the ordinary differential equation

$$a_n \cdot \left(\frac{\mathrm{d}}{\mathrm{d}t}\right)^n y(t) \; + \; \cdots \; + \; a_1 \cdot \frac{\mathrm{d}}{\mathrm{d}t} y(t) \; + \; a_0 \cdot y(t) \;\; = \;\; 0 \, , \quad t \geq 0 \, ,$$

  with suitable initial conditions at $t = 0$.

  Will $|y(t)|$ go to infinity, stay bounded, or go to zero as $t \to \infty$?

- Introduce the characteristic polynomial

$$a(\mathsf{Z}) \;\; \triangleq \;\; a_n \mathsf{Z}^n \; + \; \cdots \; + \; a_1 \mathsf{Z} \; + \; a_0 \, .$$

  If all roots of $a(\mathsf{Z})$ have a negative real part, then

$$|y(t)| \to 0 \;\; \text{as} \;\; t \to \infty \, ,$$

  independent of the initial conditions.

Real stable polynomials

Definition: Consider a polynomial $h(\mathsf{Z}) = h(\mathsf{Z}_1, \ldots, \mathsf{Z}_n) \in \mathbb{C}\big[\mathsf{Z}_1, \ldots, \mathsf{Z}_n\big]$.

- We say that $h(\mathsf{Z})$ is real stable if

  $h(\mathsf{Z}) \in \mathbb{R}\big[\mathsf{Z}_1, \ldots, \mathsf{Z}_n\big]$ and $h(\mathsf{Z})$ is $\mathcal{H}$-stable.

- We say that $h(\mathsf{Z})$ is multi-affine if

  the exponenets in a term are all either equal to zero or one.

- We say that $h(\mathsf{Z})$ is homogeneous if

  the total degree of every term is the same.

Real stable polynomials

Comments w.r.t. real stable polynomials:

- Real stable polynomials have various interesting properties.

  In applications, one typically uses these properties, and does not directly work with the condition $h(z) \neq 0$ for all $z \in \mathcal{H}^n$ in the definition.

- It is not always straightforward to directly prove that a polynomial is real stable. However, there are various rules that allow one to

  transform a real stable polynomial into another real stable polynomial.

  So, in order to prove that a polynomial is real stable, one can try to show that it is the result of a transformation of a polynomial that is known to be real stable.

Real stable polynomials

Properties of real stable polynomials:

- Let $h(\mathrm{Z})$ be a multi-affine homogeneous real stable (MAHRS) polynomial.

  Then the

  set of exponent vectors of the nonzero terms of the polynomial

  equals

  the set of the indicator vectors of the bases of a matroid.

Real stable polynomials

Properties of real stable polynomials:

- Let $h(\mathrm{Z})$ be a multi-affine homogeneous real stable (MAHRS) polynomial.

  Then the

  set of exponent vectors of the nonzero terms of the polynomial

  equals

  the set of the indicator vectors of the bases of a matroid.

Example: One can show that the following polynomial is a MAHRS polynomial:

$$h(\mathrm{Z}) = \sum_{\substack{i,j=1 \\ i<j}}^{5} \mathrm{Z}_i \mathrm{Z}_j \ .$$

Exponent vectors:

$$(1,1,0,0,0),\ (1,0,1,0,0),\ (1,0,0,1,0),\ (1,0,0,0,1),\ (0,1,1,0,0),\ (0,1,0,1,0),\ \ldots,(0,0,0,1,1)\,.$$

With this, the following are the bases of a matroid:

$$\{1,2\},\ \{1,3\},\ \{1,4\},\ \{1,5\},\ \{2,3\},\ \{2,4\},\ \ldots,(4,5)\,.$$

Matroids

Matroids

Definition of Matroid in Terms of Independent Sets:

In terms of independence, a finite matroid $\mathcal{M}$ is a pair $(\mathcal{E}, \mathcal{I})$, where

- $\mathcal{E}$ is a finite set (called the ground set) and
- $\mathcal{I}$ is a family of subsets of $\mathcal{E}$ (called the independent sets)

with the following properties:

(I1) The empty set is independent, i.e., $\emptyset \in \mathcal{I}$.

(I2) Every subset of an independent set is independent, i.e., for each $A' \subseteq A \subseteq \mathcal{E}$,

$$\text{if } A \in \mathcal{I} \text{ then } A' \in \mathcal{I}.$$

This is called the hereditary property, or the downward-closed property.

(I3) If A and B are two independent sets (i.e., each set is independent) and A has more elements than B, then

$$\text{there exists } x \in A \setminus B \text{ such that } B \cup \{x\} \text{ is in } \mathcal{I}.$$

This is called the augmentation propery or the independent set exchange property.

Matroids

Example 1 (representable or linear matroid):

Consider the following matrix over $\mathbb{F}_2$:

$$\mathbf{H} \triangleq \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix} .$$

- Let $\mathcal{E} \triangleq \{1, 2, 3, 4, 5\}$.
- Let $\mathcal{I}$ consist of all sets $A \subseteq \mathcal{E}$ such that

  $\mathbf{H}_A$ consists of linearly independent columns.

  Here, $\mathbf{H}_A$ is the submatrix of $\mathbf{H}$ consisting of the columns of $\mathbf{H}$ with index in A.

Clearly,

$$\mathcal{I} = \big\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{1,2\}, \{1,3\}, \{1,4\}, \{1,5\}, \{2,3\}, \{3,4\}, \{3,5\}\big\} .$$

Matroids

Example 2 (representable or linear matroid):

Let q be a prime power with $q \geq 5$, and let α be a primitive element of $\mathbb{F}_q$.

Consider the following matrix over $\mathbb{F}_q$:

$$\mathbf{H} \triangleq \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & \alpha & \alpha^2 & \alpha^3 \end{pmatrix}.$$

(Note that this is the parity-check matrix of an MDS code.)

- Let $\mathcal{E} \triangleq \{1, 2, 3, 4, 5\}$.
- Let $\mathcal{I}$ consist of all sets $A \subseteq \mathcal{E}$ such that $\mathbf{H}_A$ consists of linearly independent columns.

Clearly,

$$\mathcal{I} = \bigcup_{i \in \mathcal{E}} \{i\} \cup \bigcup_{\substack{i,j \in \mathcal{E} \\ i<j}} \{i, j\}.$$

Matroids

Definition:

- A subset of the ground set $\mathcal{E}$ that is not independent is called dependent.
- A maximal independent set — that is, an independent set that becomes dependent upon adding any element of $\mathcal{E}$ — is called a basis for the matroid.
- A circuit in a matroid $\mathcal{M}$ is defined to be a minimal dependent subset of $\mathcal{E}$ — that is, a dependent set whose proper subsets are all independent. The term arises because the circuits of graphic matroids are cycles in the corresponding graphs.

Note: Let $\mathcal{C}$ be a code over $\mathbb{F}_q$ described by the parity-check matrix $\mathbf{H}$. Then

the support set of a minimal codeword of $\mathcal{C}$

is

a circuit of the matroid described by the matrix $\mathbf{H}$.

Matroids

The dependent sets, the bases, or the circuits of a matroid characterize the matroid completely:

a set is independent

if and only if

it is not dependent,

if and only if

it is a subset of a basis,

if andonly if

it does not contain a circuit.

The collections of dependent sets, of bases, and of circuits each have simple properties that may be taken as axioms for a matroid.

Matroids

Definition of Matroid in Terms of Bases:

A matroid $\mathcal{M}$ is a pair $(\mathcal{E}, \mathcal{B})$, where $\mathcal{E}$ is a finite set as before and $\mathcal{B}$ is a collection of subsets of $\mathcal{E}$, called bases, with the following properties:

(B1) B is nonempty.

(B2) If A and B are distinct members of $\mathcal{B}$ and $a \in A \setminus B$, then there exists an element $b \in B \setminus A$ such that $(A \setminus \{a\}) \cup \{b\} \in \mathcal{B}$.

This is called the basis exchange property. It follows from this property that no member of $\mathcal{B}$ can be a proper subset of any other.

Note: one can show that all $B \in \mathcal{B}$ have the same size.

Matroids

Example 1 revisited (representable or linear matroid):

Consider the following matrix over $\mathbb{F}_2$:

$$\mathbf{H} \triangleq \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix} .$$

- Let $\mathcal{E} \triangleq \{1, 2, 3, 4, 5\}$.
- Let $\mathcal{I}$ consist of all sets $A \subseteq \mathcal{E}$ such that

  $\mathbf{H}_A$ consists of linearly independent columns.

  Here, $\mathbf{H}_A$ is the submatrix of $\mathbf{H}$ consisting of the columns of $\mathbf{H}$ with index in A.

Clearly,

$$\begin{aligned} \mathcal{I} &= \big\{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \\ &\qquad \{1,2\}, \{1,3\}, \{1,4\}, \{1,5\}, \{2,3\}, \{3,4\}, \{3,5\}\big\} , \\ \mathcal{B} &= \big\{\{1,2\}, \{1,3\}, \{1,4\}, \{1,5\}, \{2,3\}, \{3,4\}, \{3,5\}\big\} . \end{aligned}$$

Matroids

Example 2 revisited (representable or linear matroid):

Let q be a prime power with $q \geq 5$, and let α be a primitive element of $\mathbb{F}_q$.

Consider the following matrix over $\mathbb{F}_q$:

$$\mathbf{H} \triangleq \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & \alpha & \alpha^2 & \alpha^3 \end{pmatrix} .$$

(Note that this is the parity-check matrix of an MDS code.)

- Let $\mathcal{E} \triangleq \{1, 2, 3, 4, 5\}$.
- Let $\mathcal{I}$ consist of all sets $A \subseteq \mathcal{E}$ such that $\mathbf{H}_A$ consists of linearly independent columns.

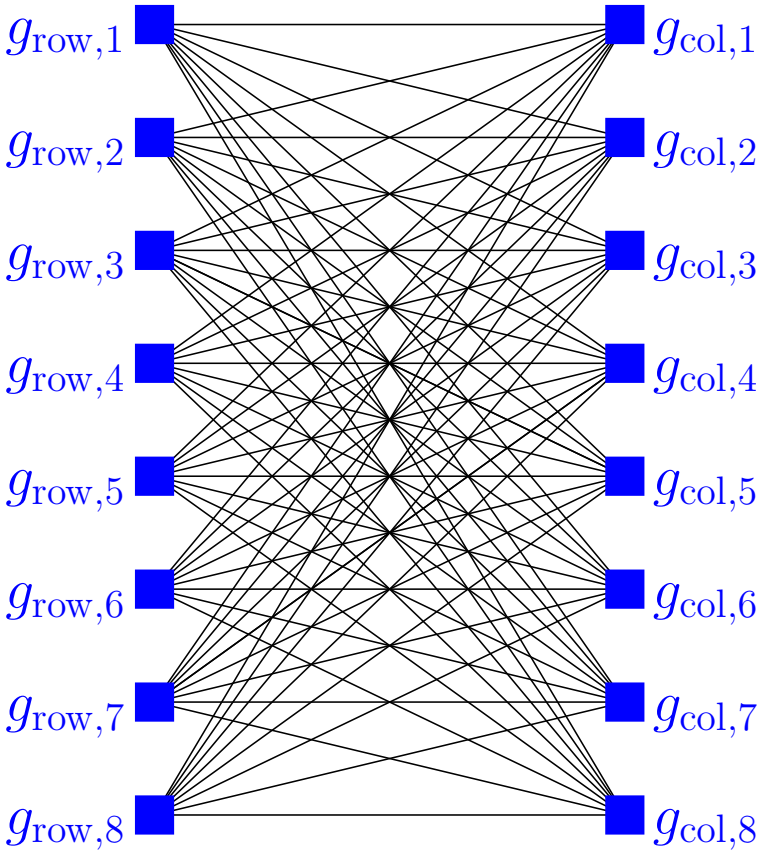
Clearly,

$$\mathcal{I} = \bigcup_{i \in \mathcal{E}} \{i\} \cup \bigcup_{\substack{i,j \in \mathcal{E} \\ i<j}} \{i, j\} ,$$

$$\mathcal{B} = \bigcup_{i,j \in \mathcal{E}:\, i<j} \{i, j\} .$$

A graphical model based on MAHRS polynomials

A Graphical Model Based on MAHRS Polynomials



(variable nodes have been omitted)

Global function:

$$g(a_{1,1}, \ldots, a_{n,n}) = \prod_j g_{\text{col},j}(a_{1,j}, \ldots, a_{n,j}) \times \prod_i g_{\text{row},i}(a_{i,1}, \ldots, a_{i,n})$$

Partition sum:

$$Z = \sum_{a_{1,1},\ldots,a_{n,n}} g(a_{1,1}, \ldots, a_{n,n})$$

A Graphical Model Based on MAHRS Polynomials

For $j \in \{1, \ldots, n\}$, define the polynomial

$$G_{\mathrm{col},j}(\mathsf{X}_{1,j}, \ldots, \mathsf{X}_{n,j}) \triangleq \sum_{a_{1,j},\ldots,a_{n,j}} g_{\mathrm{col},j}(a_{1,j}, \ldots, a_{n,j}) \cdot \mathsf{X}_{1,j}^{a_{1,j}} \cdots \mathsf{X}_{n,j}^{a_{n,j}} .$$

For $i \in \{1, \ldots, n\}$, define the polynomial

$$G_{\mathrm{row},i}(\mathsf{X}_{i,1}, \ldots, \mathsf{X}_{i,n}) \triangleq \sum_{a_{i,1},\ldots,a_{i,n}} g_{\mathrm{row},i}(a_{i,1}, \ldots, a_{i,n}) \cdot \mathsf{X}_{i,1}^{a_{i,1}} \cdots \mathsf{X}_{i,n}^{a_{i,n}} .$$

Assume that the

local functions $g_{\mathrm{col},j}(a_{1,j}, \ldots, a_{n,j})$ and $g_{\mathrm{row},i}(a_{i,1}, \ldots, a_{i,n})$

are such that

the polynomials $G_{\mathrm{col},j}(\mathsf{X}_{1,j}, \ldots, \mathsf{X}_{n,j})$ and $G_{\mathrm{row},i}(\mathsf{X}_{i,1}, \ldots, \mathsf{X}_{i,n})$

are multi-affine (homogeneous) real stable polynomials.

A Graphical Model Based on MAHRS Polynomials

Example 1 (exactly one rook per row and per column):

Choose the local functions $g_{\mathrm{col},j}(a_{1,j},\ldots,a_{n,j})$ and $g_{\mathrm{row},i}(a_{i,1},\ldots,a_{i,n})$ such that the polynomials $G_{\mathrm{col},j}(\mathsf{X}_{1,j},\ldots,\mathsf{X}_{n,j})$ and $G_{\mathrm{row},i}(\mathsf{X}_{i,1},\ldots,\mathsf{X}_{i,n})$ are as follows.

For $j \in \{1,\ldots,n\}$, we have

$$G_{\mathrm{col},j}(\mathsf{X}_{1,j},\ldots,\mathsf{X}_{n,j}) \;=\; \mathsf{X}_{1,j}+\cdots+\mathsf{X}_{n,j}\,.$$

For $i \in \{1,\ldots,n\}$, we have

$$G_{\mathrm{row},i}(\mathsf{X}_{i,1},\ldots,\mathsf{X}_{i,n}) \;=\; \mathsf{X}_{i,1}+\cdots+\mathsf{X}_{i,n}\,.$$

A Graphical Model Based on MAHRS Polynomials

Example 2 (exactly two rooks per row and per column):

Choose the local functions $g_{\mathrm{col},j}(a_{1,j},\ldots,a_{n,j})$ and $g_{\mathrm{row},i}(a_{i,1},\ldots,a_{i,n})$ such that the polynomials $G_{\mathrm{col},j}(\mathsf{X}_{1,j},\ldots,\mathsf{X}_{n,j})$ and $G_{\mathrm{row},i}(\mathsf{X}_{i,1},\ldots,\mathsf{X}_{i,n})$ are as follows.

For $j \in \{1,\ldots,n\}$, we get

$$G_{\mathrm{col},j}(\mathsf{X}_{1,j},\ldots,\mathsf{X}_{n,j}) \;=\; \sum_{\substack{i_1,i_2=1\\ i_1<i_2}}^{n} \mathsf{X}_{i_1,j}\mathsf{X}_{i_2,j} \;.$$

For $i \in \{1,\ldots,n\}$, we get

$$G_{\mathrm{row},i}(\mathsf{X}_{i,1},\ldots,\mathsf{X}_{i,n}) \;=\; \sum_{\substack{j_1,j_2=1\\ j_1<j_2}}^{n} \mathsf{X}_{i,j_1}\mathsf{X}_{i,j_2} \;.$$

A Graphical Model Based on MAHRS Polynomials

Properties:

- [Straszak and Vishnoi, 2019] It holds that

$$Z_{\mathrm{Bethe}} \leq Z \,.$$

(Actually, they do not require the polynomials to be homogeneous.)

A Graphical Model Based on MAHRS Polynomials

Properties:

- [Huang and V., 2024] The local marginal polytope (the domain of the Bethe free energy function) is a tight relaxation of the convex hull of the polytope describing all valid configurations of the graphical model.

  (The proof is based on using results about the intersection of two matroids.

  While the intersection of two matroids is in general not a matroid, it still has nice properties.)

A Graphical Model Based on MAHRS Polynomials

Properties:

- [Huang and V., 2024] The local marginal polytope (the domain of the Bethe free energy function) is a tight relaxation of the convex hull of the polytope describing all valid configurations of the graphical model.

  (The proof is based on using results about the intersection of two matroids.
  While the intersection of two matroids is in general not a matroid, it still has nice properties.)

- [Huang and V., 2024] The sum-product algorithm converges to the minimum of the Bethe free energy function.

  (There are some technical conditions, but we expect the general statement to hold.)

  (The proof is based, in part, on showing that the SPA message update rules can be rewritten in terms of evaluations of the polynomials $G_{\mathrm{col},j}(\mathsf{X}_{1,j}, \ldots, \mathsf{X}_{n,j})$ and $G_{\mathrm{row},i}(\mathsf{X}_{i,1}, \ldots, \mathsf{X}_{i,n})$.)

Conclusions

Conclusions

Conclusions

- Loopy belief propagagion is no silver bullet.

Conclusions

- Loopy belief propagagion is no silver bullet.
- However, there are interesting setups where it works very well.

Conclusions

- Loopy belief propagagion is no silver bullet.
- However, there are interesting setups where it works very well.
- Complexity of the permanent estimation based on the SPA is remarkably low. (Hard to be beaten by any standard convex optimization algorithm that minimizes the Bethe free energy.)

Conclusions

- Loopy belief propagagion is no silver bullet.
- However, there are interesting setups where it works very well.
- Complexity of the permanent estimation based on the SPA is remarkably low. (Hard to be beaten by any standard convex optimization algorithm that minimizes the Bethe free energy.)
- If the Bethe approximation does not work well, one can try better approximations, e.g., the Kikuchi approximation.

References

- P. O. Vontobel, "The Bethe permanent of a non-negative matrix," IEEE Trans. Inf. Theory, vol. 59, no. 3, pp. 1866–1901, Mar. 2013.

- D. Straszak and N. K. Vishnoi, "Belief propagation, Bethe approximation, and polynomials," IEEE Trans. Inf. Theory, vol. 65, no. 7, pp. 4353–4363, Jul. 2019.

- Y. Huang and P. O. Vontobel, "The Bethe partition function and the SPA for factor graphs based on homogeneous real stable polynomials, Proc. ISIT 2024. (Link to long version in that paper.)

pascal.vontobel@ieee.org