SUDOKU Codes, a class of non-linear iteratively decodable codes

Jossy Sayir

UNIVERSITY OF CAMBRIDGE
Department of Engineering

Chinese University of Hong Kong, 21 October 2014

Outline

- SUDOKU as channel codes
- Efficient decoding / encoding of SUDOKU
- Density Evolution
- Rate of SUDOKU codes

SUDOKU Puzzles

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

SUDOKU Puzzles

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

SUDOKU Puzzles

3		4	
	2		
			1

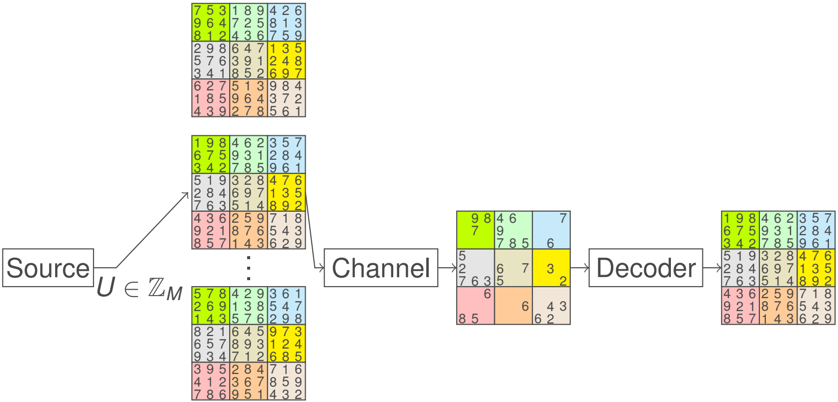
SUDOKU Puzzles

3	1	4	2
4	2	1	3
2	4	3	1
1	3	2	4

Coding by SUDOKU

Wikipedia "The Mathematics of SUDOKU"

There are $M = 6{,}670{,}903{,}752{,}021{,}072{,}936{,}960$ valid SUDOKU grids.



Background

Literature

- P. Farrell, “Sudoku Codes: a Tutorial” (Ambleside 2009) - looked at distance properties of SUDOKU codes
- T. Moon & al., BP and Sinkhorn for SUDOKU solving (2006, 2009) - algorithms to solve SUDOKU puzzles (not SUDOKU as codes)

- use in lectures since 2006 to illustrate BP decoding
- invaluable didactic tool to illustrate the use of factor graphs, trellis decoding, arithmetic decoding and other techniques
- 2 student projects in 2013/14 and strong student interest
- Proxy for the study of non-linear codes with local constraints

Non-linear Codes

Theory

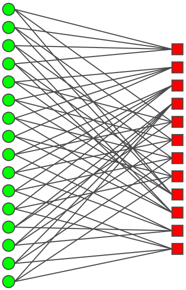
- Used in achievability proofs (fixed-composition codes, typical sequence arguments, etc.)
- No practical encoders, decoders, etc.

Applications

- Simple constrained sequences (e.g. for magnetic recording)
- Other contraints, e.g., low Peak-to-Average Power Ratio (PAPR), can translate into non-linear constraints

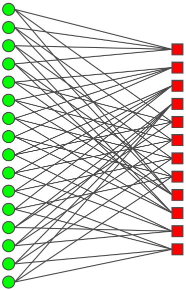
My work on SUDOKU codes

- Efficient decoding
- Efficient encoding
- Density Evolution
- Rate of the code



My work on SUDOKU codes

- Efficient decoding
- Efficient encoding
- Density Evolution
- Rate of the code



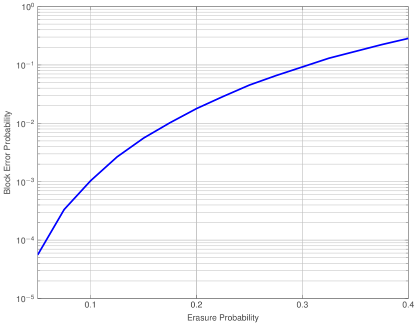
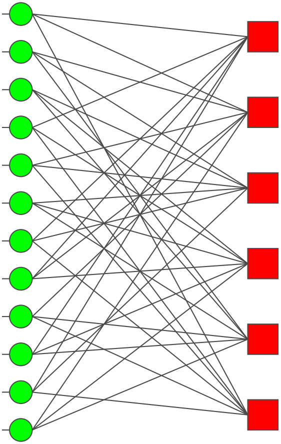
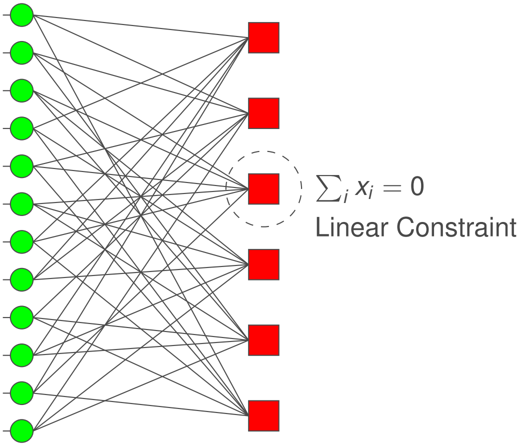


Figure : Classic 9x9 SUDOKU simulated performance averaged over 49 codewords

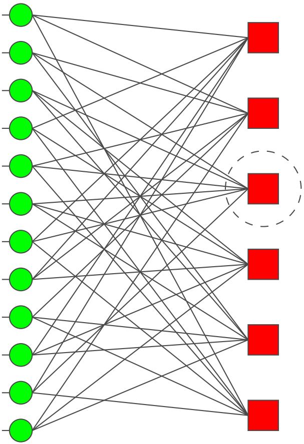
Local constraints: factor graphs



Local constraints: factor graphs



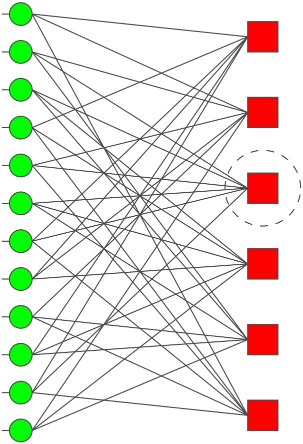
Local constraints: factor graphs



Non-linear constraint

e.g., $x_1 \neq x_2 \neq \ldots x_d$

Local constraints: factor graphs



Non-linear constraint

e.g., $x_1 \neq x_2 \neq \dots x_d$

q: alphabet size

d: node degree

if $q = d$, $\{x_1, \dots, x_q\} \in \mathcal{S}_q$

SUDOKU (permutation) constraint

Belief propagation for permutation constraints

- Messages are q-ary probability mass functions
- Variable nodes: product of incoming probabilities
- Constraint nodes:

$$P(X_i = k | m_{v\sim i\to c}) = \sum_{i' \neq i} \prod_{k' \neq k} P(X_{i'} = k' | m_{v_{i'}\to c})$$

- Let **P** be the matrix of incoming messages to a constraint node, i.e., p_{ik} is the probability that the variable corresponding to the i-th message takes on value k

Constraint Node Computation: Permanents


Augustin-Louis Cauchy

- Permanent of a matrix,

$$\text{per}\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = a(ei+hf)+b(di+gf)+c(dh+ge),$$

same as a determinant except all "+"

- For a constraint node,

$$m_{c\to vi} = \frac{1}{\text{per}\,\mathbf{P}}[\text{per}(\mathbf{P}_{\sim i1}), \text{per}(\mathbf{P}_{\sim i2}), \ldots, \text{per}(\mathbf{P}_{\sim iq})],$$

where $\mathbf{P}_{\sim ij}$ denotes the matrix $\mathbf{P}$ with its i-th row and j-th column removed

Complexity of the constraint node operation

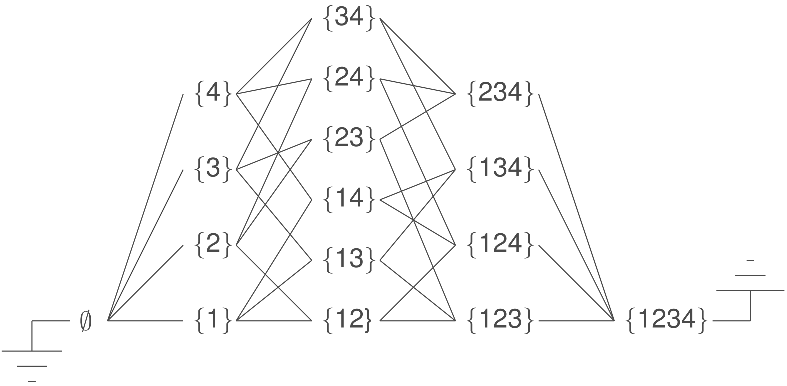
- Each constraint node at each iteration requires the computation of a permanent
- Brute force computation: sum of $q!$ products of q factors, i.e.,

Alphabet Size q	Multiplications $(q-1) \times q!$	Additions $q! - 1$
4	72	23
9	2'903'040	362'879
16	3.14×10^{14}	2.09×10^{13}

- *From Wikipedia:* The permanent is more difficult to compute than the determinant. Gaussian elimination cannot be used to compute the permanent. Computing the permanent of a 0-1 matrix (matrix whose entries are 0 or 1) is $\sharp P$-complete. $FP = \sharp P$ is stronger than $P = NP$. When the entries of A are nonnegative, however, the permanent can be computed approximately in probabilistic polynomial time, up to an error of εM, where M is the value of the permanent and $\varepsilon > 0$ is arbitrary.[1]

[1] Jerrum, M.; Sinclair, A.; Vigoda, E. (2004), "A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries", Journal of the ACM

Trellis-based permanent computation

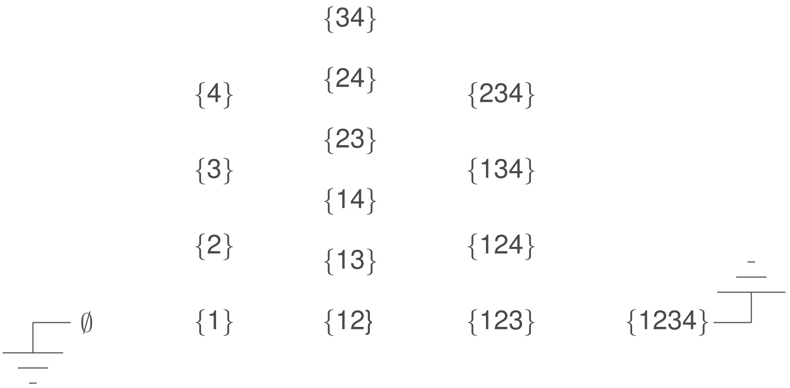


- Forward multiply and add yields the permanent
- Full BCJR yields the subpermanents we need
- thanks Gottfried Lechner

University of Cambridge
Department of Engineering

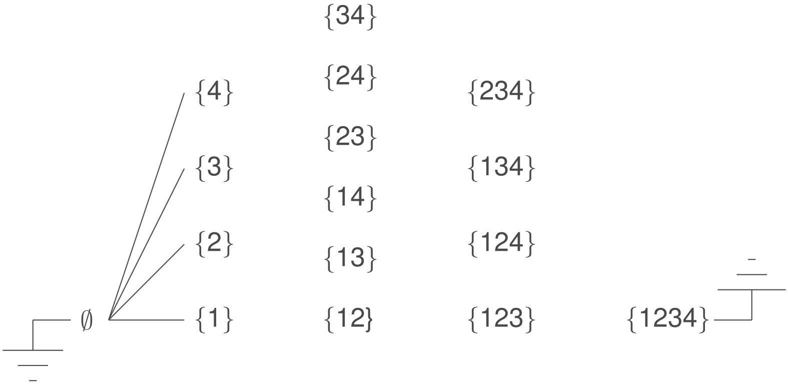
Trellis-based erasure decoding

$$\mathbf{T}_{\text{in}} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$



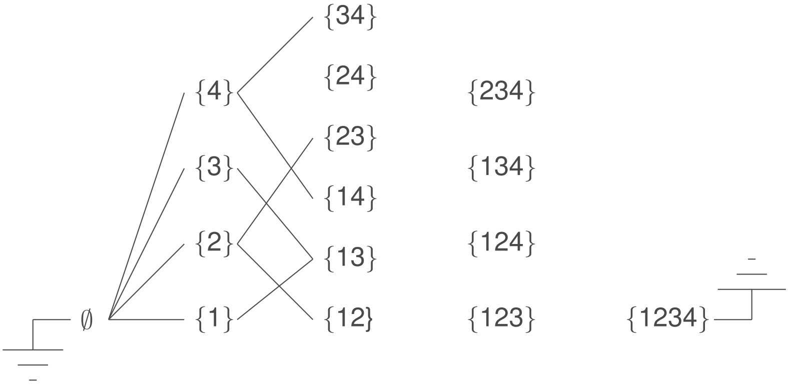
Trellis-based erasure decoding

$$\mathbf{T}_{\text{in}} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$



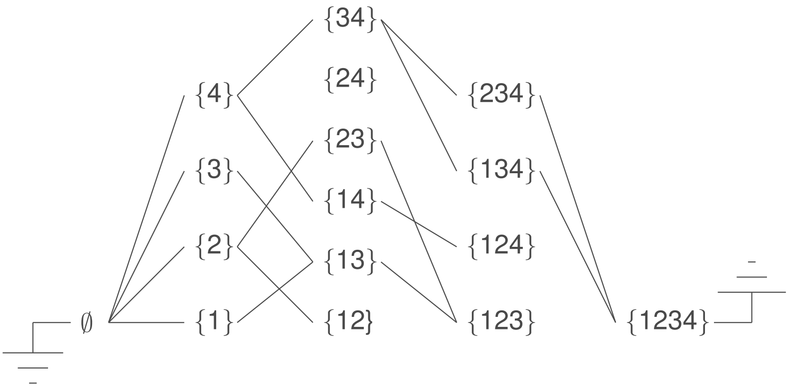
Trellis-based erasure decoding

$$\mathbf{T}_{\text{in}} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$



Trellis-based erasure decoding

$$\mathbf{T}_{\text{in}} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$



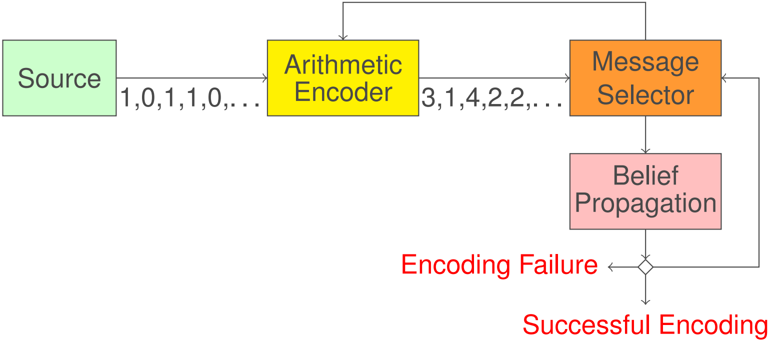
Trellis-based erasure decoding

$$\mathbf{T}_{\text{in}} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Trellis-based erasure decoding

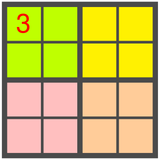
$$\mathbf{T}_{\text{in}} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \qquad \mathbf{T}_{\text{out}} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Universal Encoder for Codes with a Factor Graph Description



Encoding Examples

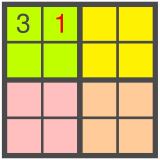
Encoding Examples



$\{1, 2, 3, 4\}$

$\log 4$

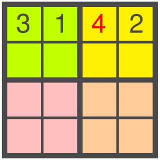
Encoding Examples



$\{1, 2, 4\}$

$\log 4 + \log 3$

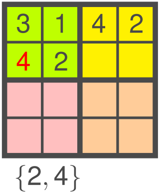
Encoding Examples



$\{2, 4\}$

$$\log 4 + \log 3 + \log 2$$

Encoding Examples



$$\log 4 + \log 3 + \log 2 + \log 2$$

Encoding Examples

$\{1, 3\}$

$$\log 4 + \log 3 + \log 2 + \log 2 + \log 2$$

Encoding Examples

3	1	4	2
4	2	1	3
2			
1			

$\{1, 2\}$

$\log 4 + \log 3 + \log 2 + \log 2 + \log 2 + \log 2$

UNIVERSITY OF CAMBRIDGE
Department of Engineering

Encoding Examples

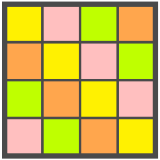
3	1	4	2
4	2	1	3
2	4	3	1
1	3	2	4

$\{3, 4\}$

$$\log 4 + \log 3 + \log 2 + \log 2 + \log 2 + \log 2 + \log 2 = 8.59 \text{ bits}$$

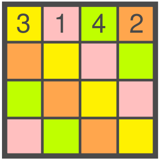
$$R = \frac{\log_4(4 \cdot 3 \cdot 2^6)}{16} = 0.30$$

Encoding Examples



$$R = \frac{\log_4(4 \cdot 3 \cdot 2^6)}{16} = 0.30$$

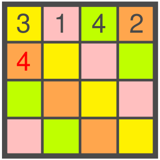
Encoding Examples



$$\log(4 \cdot 3 \cdot 2)$$

$$R = \frac{\log_4(4 \cdot 3 \cdot 2^6)}{16} = 0.30$$

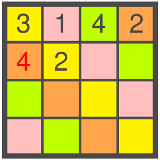
Encoding Examples



$$\log(4 \cdot 3 \cdot 2^2)$$

$$R = \frac{\log_4(4 \cdot 3 \cdot 2^6)}{16} = 0.30$$

Encoding Examples



$$\log(4 \cdot 3 \cdot 2^2)$$

$$R = \frac{\log_4(4 \cdot 3 \cdot 2^6)}{16} = 0.30$$

Encoding Examples

3	1	4	2
4	2	1	3
2	4	3	1
1	3	2	4

3	1	4	2
4	2	3	

$$\log(4 \cdot 3 \cdot 2^2)$$

$$R = \frac{\log_4(4 \cdot 3 \cdot 2^6)}{16} = 0.30$$

Encoding Examples

$\log(4 \cdot 3 \cdot 2^2)$

$$R = \frac{\log_4(4 \cdot 3 \cdot 2^6)}{16} = 0.30$$

Encoding Examples

$$\log(4 \cdot 3 \cdot 2^2)$$

$$R = \frac{\log_4(4 \cdot 3 \cdot 2^6)}{16} = 0.30$$

Encoding Examples

$$\log(4 \cdot 3 \cdot 2^2)$$

$$R = \frac{\log_4(4 \cdot 3 \cdot 2^6)}{16} = 0.30$$

Encoding Examples

$$\log(4 \cdot 3 \cdot 2^2)$$

$$R = \frac{\log_4(4 \cdot 3 \cdot 2^6)}{16} = 0.30$$

Encoding Examples

3	1	4	2
4	2	1	3
2	4	3	1
1	3	2	4

3	1	4	2
4	2	3	1
2	3	1	4

$$\log(4 \cdot 3 \cdot 2^2)$$

$$R = \frac{\log_4(4 \cdot 3 \cdot 2^6)}{16} = 0.30$$

Encoding Examples

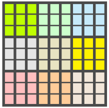
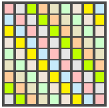
3	1	4	2
4	2	1	3
2	4	3	1
1	3	2	4

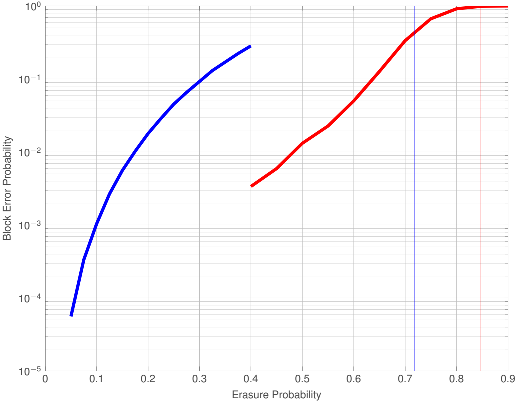
$$R = \frac{\log_4(4 \cdot 3 \cdot 2^6)}{16} = 0.30$$

3	1	4	2
4	2	3	1
2	3	1	4
!!!			

$$R = 0$$

Simulation Measurements

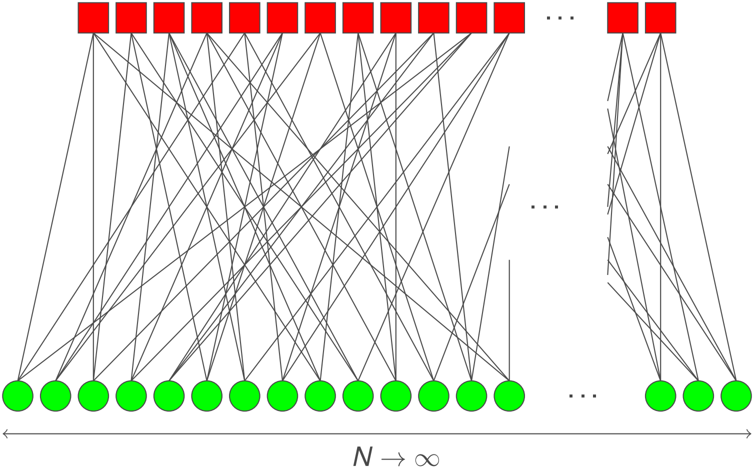
Factor Graph		
True Rate	$R = 0.2824$	$R = 0.1527$
Probability of Decoding Failure	0.016	0.9995



Diagonal SUDOKUs

Alphabet size q	Number M of valid grids	Rate $R = \log M/q^2$
3	6	0.1812
4	0	0
5	360	0.1463
6	0	0
7	3,200,400	0.1571

Asymptotic Analysis



Density Evolution

Non-linear codes with local constraints vs. linear (LDPC) codes

- Concentration of the error performance
- Convergence to a cycle-free case
- Simplification by restriction to the all-one (all zero) codeword

SUDOKU constraints for the q-ary Erasure channel

- Messages = subsets of $\{1, \ldots, q\}$
- Some interesting symmetries

Symmetries of the SUDOKU decoder

Proposition

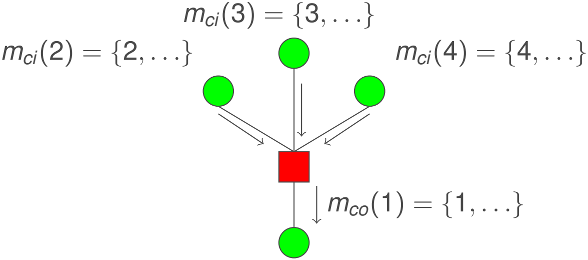
All operations symmetric under alphabet and edge permutations

Proposition

The probability distribution of the cardinalities of messages $\#m$ at iteration k is a sufficient statistic for the probability distribution of the actual messages

- $P_k(\#m)$ is a sufficient statistic for $P_{k+1}(\#m)$
- $P_k(\#m)$ is a sufficient statistic for the block error probability at iteration k

Density Evolution: the calculation



Density evolution: the calculation

input #	multipl.	output # 1	2	3	4
$(\mathbf{1},\mathbf{1},\mathbf{1})$	1	1	0	0	0
$(\mathbf{1},\mathbf{1},\mathbf{2})$	3	2/3	1/3	0	0
$(\mathbf{1},\mathbf{1},\mathbf{3})$	3	1/3	2/3	0	0
$(\mathbf{1},\mathbf{1},\mathbf{4})$	3	0	1	0	0
$(\mathbf{1},\mathbf{2},\mathbf{2})$	3	4/9	2/9	1/3	0
$(\mathbf{1},\mathbf{2},\mathbf{3})$	6	2/9	2/9	5/9	0
$(\mathbf{1},\mathbf{2},\mathbf{4})$	6	0	1/3	2/3	0
$(\mathbf{1},\mathbf{3},\mathbf{3})$	3	1/9	0	8/9	0
$(\mathbf{1},\mathbf{3},\mathbf{4})$	6	0	0	1	0
$(\mathbf{1},\mathbf{4},\mathbf{4})$	3	0	0	1	0
$(\mathbf{2},\mathbf{2},\mathbf{2})$	1	8/27	1/9	0	16/27
$(\mathbf{2},\mathbf{2},\mathbf{3})$	3	4/27	2/27	0	21/27
$(\mathbf{2},\mathbf{2},\mathbf{4})$	3	0	1/9	0	8/9
$(\mathbf{2},\mathbf{3},\mathbf{3})$	3	2/27	0	0	25/27
⋮ ⋮ ⋮	⋮	⋮	⋮	⋮	⋮

Density evolution: the calculation

$$\begin{aligned}
P_{co}(\mathbf{1}) =& (P_{ci}(\mathbf{1}))^3 + 2P_{ci}(\mathbf{1})^2 P_{ci}(\mathbf{2}) + P_{ci}(\mathbf{1})^2 P_{ci}(\mathbf{3}) + \frac{4}{3} P_{ci}(\mathbf{1}) P_{ci}(\mathbf{2})^2 \\
&+ \frac{4}{3} P_{ci}(\mathbf{1}) P_{ci}(\mathbf{2}) P_{ci}(\mathbf{3}) + \frac{1}{3} P_{ci}(\mathbf{1}) P_{ci}(\mathbf{3})^2 + \frac{8}{27} P_{ci}(\mathbf{2})^3 \\
&+ \frac{4}{9} P_{ci}(\mathbf{2})^2 P_{ci}(\mathbf{3}) + \frac{2}{9} P_{ci}(\mathbf{2}) P_{ci}(\mathbf{3})^2 + \frac{1}{27} P_{ci}(\mathbf{3})^3 \\
P_{co}(\mathbf{2}) =& P_{ci}(\mathbf{1})^2 P_{ci}(\mathbf{2}) + 2P_{ci}(\mathbf{1})^2 P_{ci}(\mathbf{3}) + 3P_{ci}(\mathbf{1})^2 P_{ci}(\mathbf{4}) + \frac{2}{3} P_{ci}(\mathbf{1}) P_{ci}(\mathbf{2})^2 \\
&+ \frac{4}{3} P_{ci}(\mathbf{1}) P_{ci}(\mathbf{2}) P_{ci}(\mathbf{3}) + 2P_{ci}(\mathbf{1}) P_{ci}(\mathbf{2}) P_{ci}(\mathbf{4}) + \frac{1}{9} P_{ci}(\mathbf{2})^3 \\
&+ \frac{2}{9} P_{ci}(\mathbf{2})^2 P_{ci}(\mathbf{3}) + \frac{1}{3} P_{ci}(\mathbf{2})^2 P_{ci}(\mathbf{4}) \\
P_{co}(\mathbf{3}) =& P_{ci}(\mathbf{1}) P_{ci}(\mathbf{2})^2 + \frac{10}{3} P_{ci}(\mathbf{1}) P_{ci}(\mathbf{2}) P_{ci}(\mathbf{3}) + 4P_{ci}(\mathbf{1}) P_{ci}(\mathbf{2}) P_{ci}(\mathbf{4}) \\
&+ \frac{8}{3} P_{ci}(\mathbf{1}) P_{ci}(\mathbf{3})^2 + 6P_{ci}(\mathbf{1}) P_{ci}(\mathbf{3}) P_{ci}(\mathbf{4}) + \ldots
\end{aligned}$$

Density evolution: results for regular $d_v = 3$ graphs

Alphabet q	Threshold	Run time
3	0.8836	<1s
4	0.7251	<1s
5	0.6209	<1s
6	0.5492	<10s
7	0.4965	<1min
8	0.4559	3 weeks
9	?	10^8 years

Rate as blocklength $N \to \infty$

The rate of a SUDOKU-type code for $N \to \infty$ is currently unknown. The quantity defined below may give an indication of what the true rate might be.

Definition

For a constraint-regular factor graph with constraint degree d_c equal to the alphabet size q, and variable degree distribution $\lambda(x)$, the "cycle-free rate" of a code with SUDOKU type constaints is

$$R_{cf} = \frac{\log_q((q-1)!)}{q-1}$$

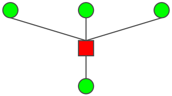
Rate as blocklength $N \to \infty$

The rate of a SUDOKU-type code for $N \to \infty$ is currently unknown. The quantity defined below may give an indication of what the true rate might be.

Definition

For a constraint-regular factor graph with constraint degree d_c equal to the alphabet size q, and variable degree distribution $\lambda(x)$, the "cycle-free rate" of a code with SUDOKU type constaints is

$$R_{cf} = \frac{\log_q((q-1)!)}{q-1}$$



$\frac{\log(q!)}{q}$

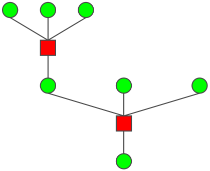
Rate as blocklength $N \to \infty$

The rate of a SUDOKU-type code for $N \to \infty$ is currently unknown. The quantity defined below may give an indication of what the true rate might be.

Definition

For a constraint-regular factor graph with constraint degree d_c equal to the alphabet size q, and variable degree distribution $\lambda(x)$, the "cycle-free rate" of a code with SUDOKU type constaints is

$$R_{cf} = \frac{\log_q ((q-1)!)}{q-1}$$



$$\frac{\log(q!)+\log((q-1)!)}{q+(q-1)}$$

Density evolution: results for regular $d_v = 3$ graphs

Alphabet q	Threshold	$1 - R_{cf}$
3	0.8836	0.6845
4	0.7251	0.5692
5	0.6209	0.5063
6	0.5492	0.4656
7	0.4965	0.4365
8	0.4559	0.4143
9	?	0.3967

Pascal Vontobel's Bethe approximation of the partition function of the factor graph

Rate

$$R = \max\left\{0, \frac{d_v}{q}\log_2(q!) - (d_v - 1)\log_2 q\right\}$$
$$\approx \max\left\{0, \log_2\left(\frac{q(2\pi q)^{d_v/(2q)}}{e^{d_v}}\right)\right\}$$

$R = 0$ for $d_v = 3$ and $q < 12$

Conclusion

- I calculated using density evolution an erasure threshold for $d_v = 3$ and $q = 3, \ldots, 8$, but Pascal proved that there are in fact no codewords for these dimensions (or, as he put it more precisely, sub-exponentially many codewords)
- Asymptotic analysis seems stuck between a combinatorial explosion and the requirement to go to higher alphabets
- Study specific structures like the diagonal SUDOKU, devise encoding methods and analyse performance
- Non-linear codes with local constraints are fun: they test the limit of our abilities, pose interesting problems, and the brand name "SUDOKU" seems to attract good students
- Current student project: linear codes with added non-linear constraints for joint synchronisation and coding