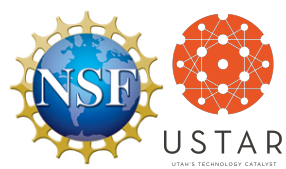
Recent Advances on Stochastic and Noise Enhanced Methods

in Error Correction Decoders

Chris Winstead, Tasnuva Tithi (Utah State University)
Emmanuel Boutillon (Lab-STICC, Université de Bretagne Sud)
Fakhreddine Ghaffari (ETIS Lab, Université Cergy-Pontoise)



Outline

ML decoding

The ML decoding can be expressed as

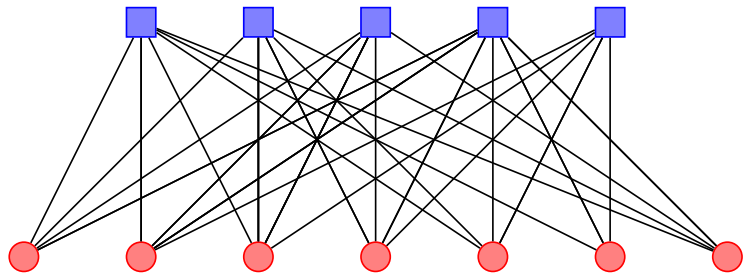
$$\hat{x} = \arg\max_{x \in \mathcal{C}} \sum_{k=0}^{N-1} x_k y_k$$

This is a classical research optimization problem. It may be solved efficiently by heuristics using noise (simulated annealing for example).

How noise can be used to improve the decoding performance of an iterative decoder (partial survey)

Introduction: Non-deterministic Decoding Algorithms

Message Passing (MP) algorithms are sub-optimal on loopy code graphs.



Example: With LDPC codes, absorbing sets act as local minima in MP algorithms. Non-determinism assists in escaping local minima.

Non-determinism dramatically improves bit-flipping (BF) algorithms.

- Enables low-cost, low-energy, high-performance decoders
- Can exploit native non-determinism in nano-scale devices

Outline

Outline

Message Passing Decoders helped by noise

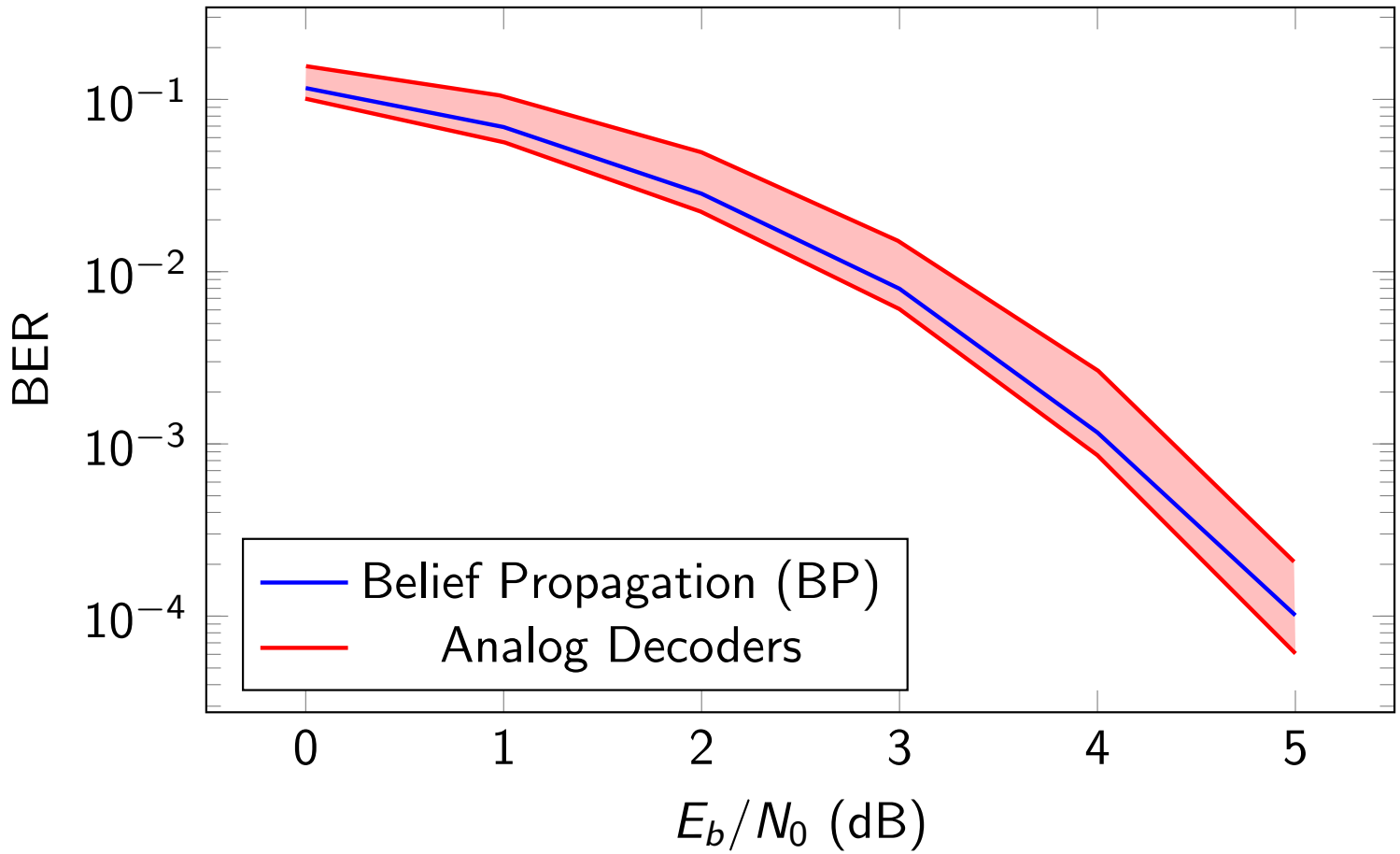
Several non-deterministic BP, MS, and other decoding algorithms have been studied:

- Analog Decoders
- Dithered BP[1]
- Noisy Min-Sum[2]
- Stochastic algorithm[3]

[1] Leduc-Primeau et al. 2012.
[2] Ngassa, Savin, and Declercq 2014; Cochachin et al. 2017.
[3] Gaudet and Rapley 2003.

Structural Non-Determinism

Benefits from random mismatch were noticed in analog decoders in 2001[4]. These effects were mysterious, and could not be exploited in design.



[4] Lustenberger and Loeliger 2001.

Dithered BP

Let $\vec{y}$ be the noisy received codeword.

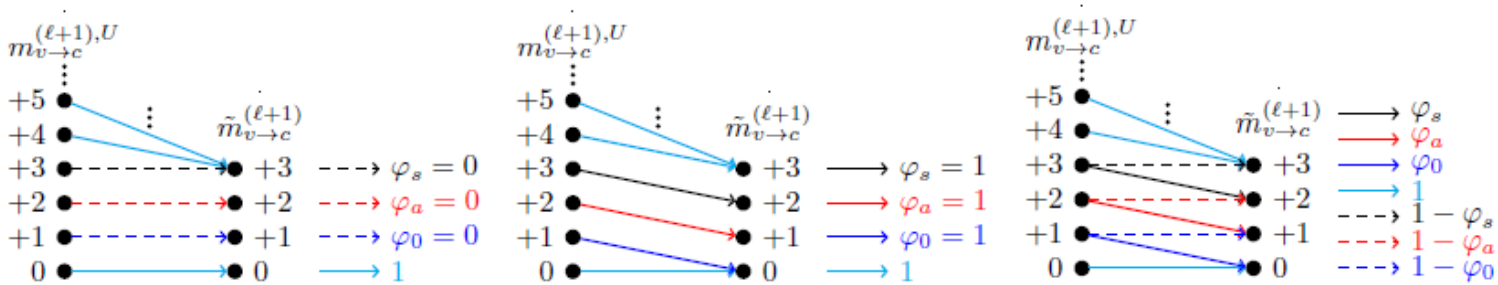
Dithered decoding is described as:

decode($\vec{y}$)
while **failed**:
    decode($\vec{y} + \vec{w}$)
    where $\vec{w}$ is a vector of random perturbations.

Very simple technique, but effective.[5]

[5] Leduc-Primeau et al. 2012.

Noise Aided-Min Sum decoders



(a) MS decoder. (b) OMS decoder. (c) NAN-MS decoder.

(d_v, d_c)-regular LDPC code, BI-AWGN channel						
		MS Decoder	OMS decoders	NAN-MS decoders		
(d_v, d_c)	q	δ_{db}	δ_{db}	$\tilde{\delta}_{db}$	DE gain	SNR gain
$(3, 6)$	3	**1.7888**	2.2039	**1.5711**	**0.2177**	**0.2**
	4	1.6437	**1.3481**	**1.2877**	**0.0604**	**0.05**
$(4, 8)$	3	2.7360	**2.3219**	**2.1056**	**0.2163**	**0.2**
	4	2.5389	**1.7509**	**1.7411**	**0.0098**	**0.000**

Noise injection significantly helps low precision decoder in the waterfall region.[6]

[6] Ngassa, Savin, and Declercq 2014; Cochachin et al. 2017.

Stochastic Decoding

Algorithm concept

Based on stochastic computing for probability computations[7]:

- Input: stationary random IID binary sequence. E.g.:

$$P(A = 0.2) \Rightarrow A = \ldots 000100100000010 \ldots$$

- Outputs: converge in mean to arithmetic results

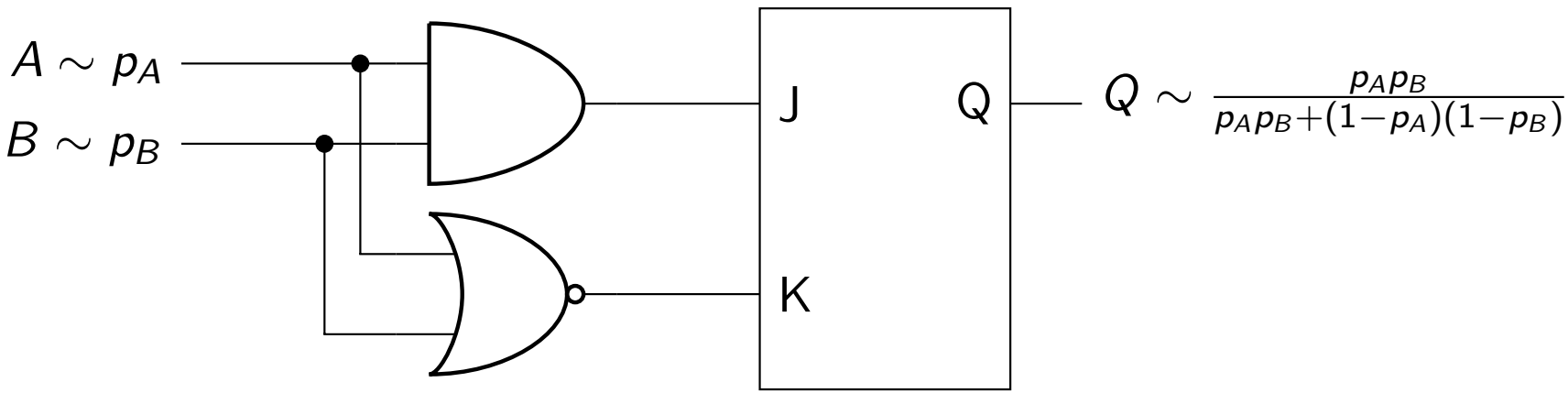
Stochastic decoders approximate belief propagation.

Can benefit from non-determinism, e.g. repeated decoding yields different results on the same frame.

[7] Gaudet and Rapley 2003.

Stochastic Decoding

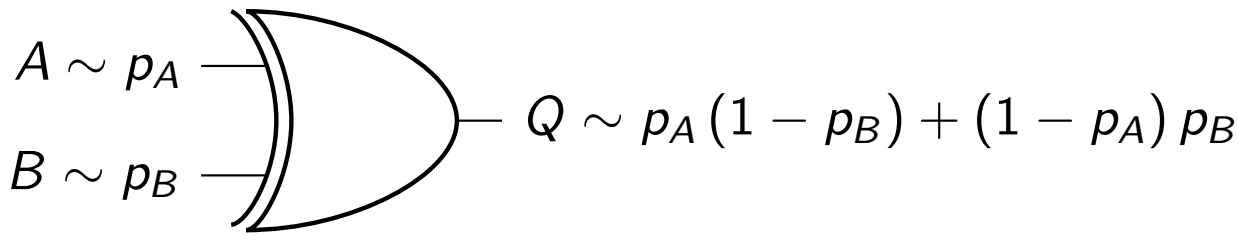
Example: Bayes' Rule



- Implement symbol node computations for Belief Propagation (BP)
- Output samples at Q are not independent due to flip-flop memory.

Stochastic Decoding

Example: Parity Check



- Implements check-node computations for BP.

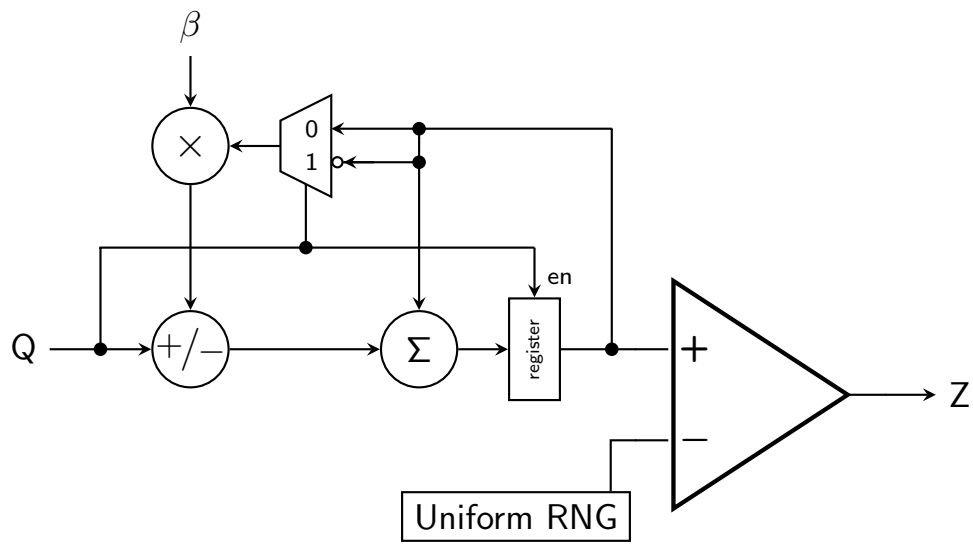
Stochastic Decoding

Edge memories

Because of flip-flop memory, stochastic decoders lock into fixed states[8].

The solution is to apply re-randomization on every edge[9].

Solved with edge memories or tracking forecast memories (TFMs)[10], which add significant complexity:



[8] Winstead et al. 2005.

[9] Tehrani, Gross, and Mannor 2006.

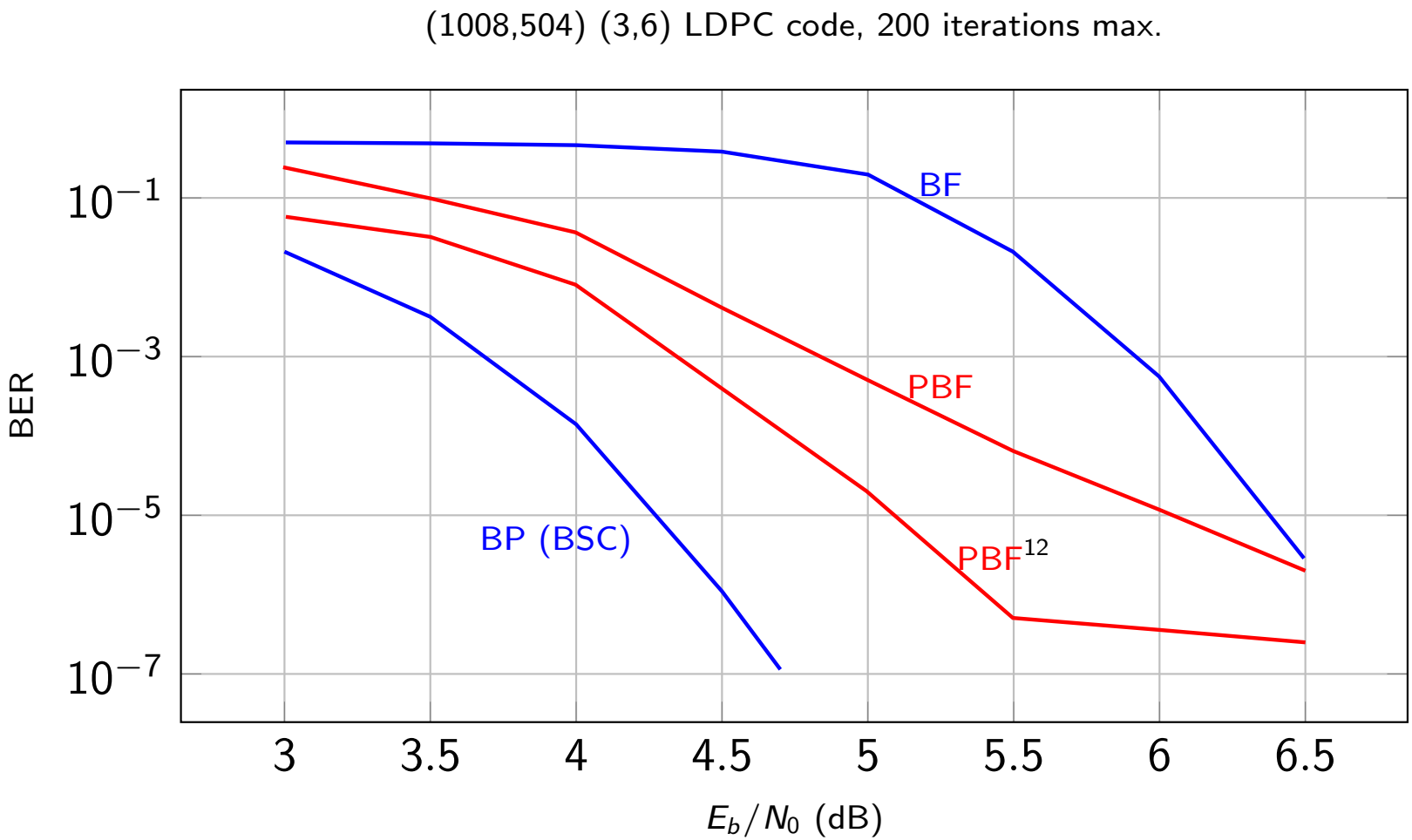
[10] Tehrani et al. 2010.

Outline

Probabilistic Bit Flipping

Performance enhancement from stochastic behavior

The Probabilistic Bit Flipping (PBF) algorithm was introduced in 2005[11], where "Bit Flipping" refers to the Gallager A decoding algorithm.



[11] Miladinovic and Fossorier 2005.

[12] 51 parallel decoders with re-initialization heuristic.

Probabilistic Bit Flipping

Algorithm Concept

For symbol node k at iteration ℓ:

$$\begin{aligned} F_{k,\mathrm{BF}}(\ell) &= \text{Flip indicator in the normal BF algorithm} \\ r_k(\ell) &= \text{Random bit with probability } p_{\mathrm{flip}} \end{aligned}$$

For PBF, the flip indicator is revised:

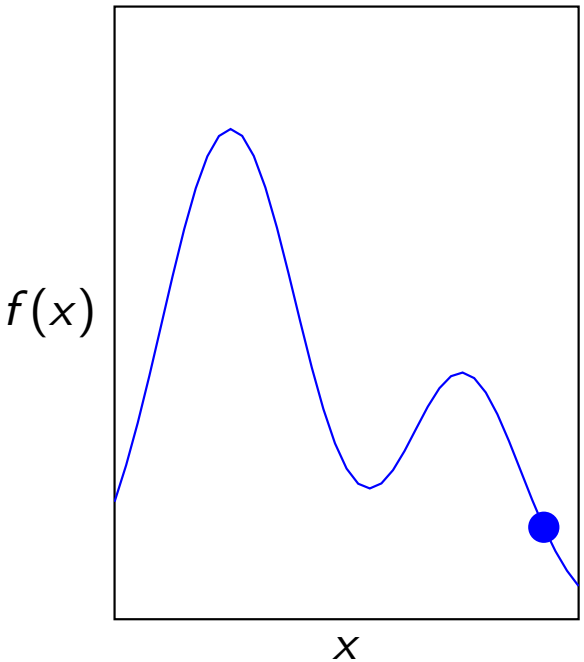
$$F_{k,\mathrm{PBF}}(\ell) = r_k(\ell) F_{k,\mathrm{BF}}(\ell)$$

I.e. flip only with probability p_{flip}.

Outline

Gradient Descent (or Gradient Ascent)

BF is related to Gradient Descent Optimization[13].



Received samples y give initial guess x.

Decoding optimizes a global reliability metric, called the objective function:

$$f(x, y) = \sum_{i=1}^{n} x_i y_i + \sum_{j=1}^{m} s_j$$

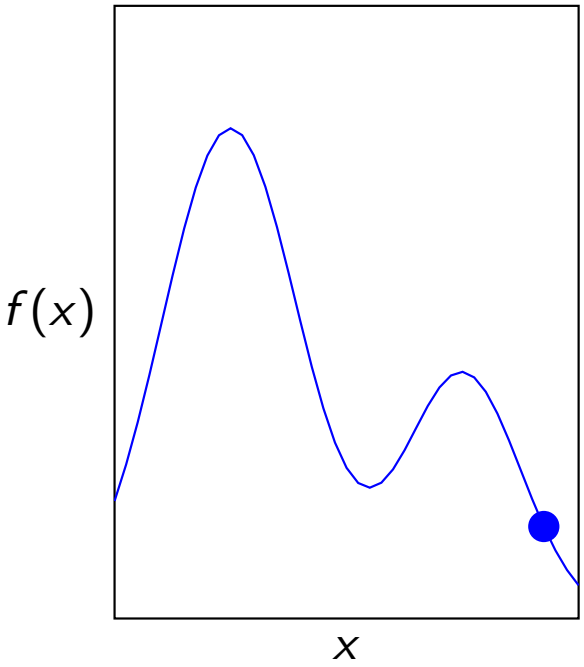
The first part, $\sum_{i=1}^{n} x_i y_i$, represents the Maximum Likelihood problem.

The second part, $\sum_{j=1}^{m} s_j$, is the sum over all bipolar parity checks. If the sequence is valid, then all parity checks equal $+1$.

[13] Wadayama et al. 2010.

Gradient Descent (or Gradient Ascent)

BF is related to Gradient Descent Optimization[13].



Adjust the guess along the objective function gradient:

$$\Delta x_i \propto x_i \frac{df}{dx_i} = x_i \left(y_i + \sum_{j \in M(i)} \prod_{k \in N(j) \setminus i} x_j \right)$$

Result: symbol-wise "energy function":

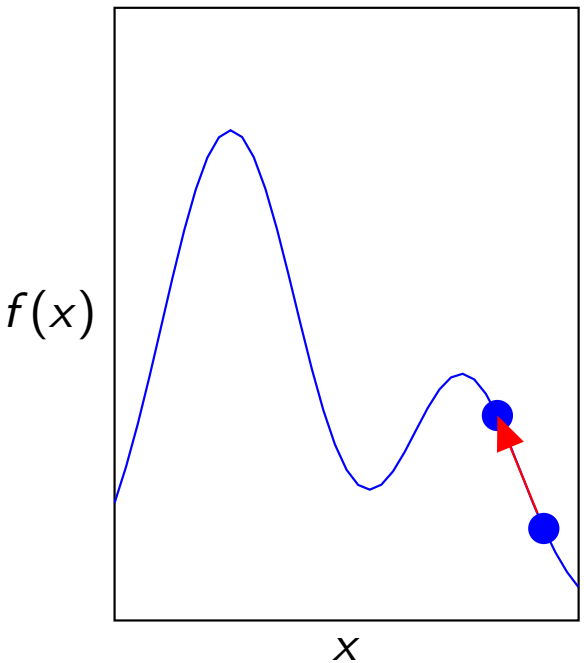
$$E_i = x_i y_i + \sum_{j \in M(i)} s_j$$

Flip bit(s) w/ most negative E_i to maximize f ().

[13] Wadayama et al. 2010

Gradient Descent (or Gradient Ascent)

BF is related to Gradient Descent Optimization[13].

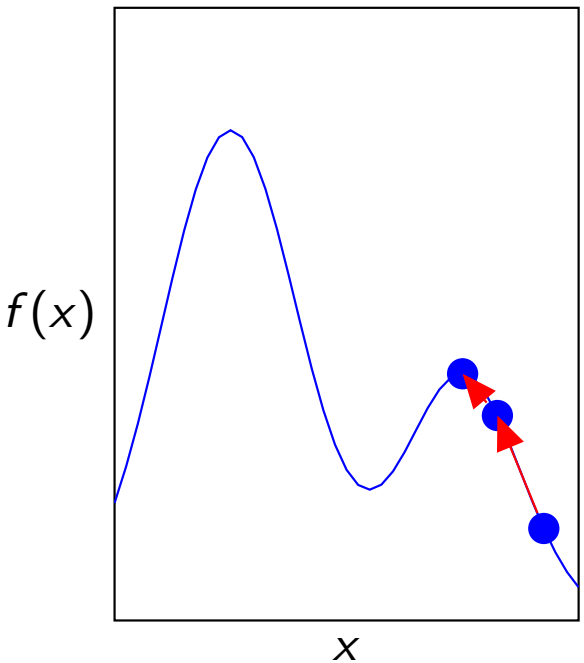


BF incrementally increases the objective function, following the positive slope.

[13] Wadayama et al. 2010.

Gradient Descent (or Gradient Ascent)

BF is related to Gradient Descent Optimization[13].



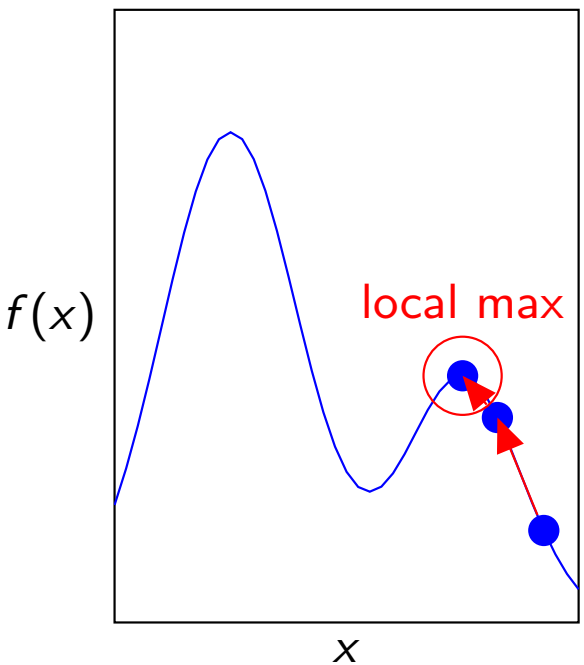
BF incrementally increases the objective function, following the positive slope.

This procedure tends to get stuck at local maxima.

[13] Wadayama et al. 2010.

Gradient Descent (or Gradient Ascent)

BF is related to Gradient Descent Optimization[13].



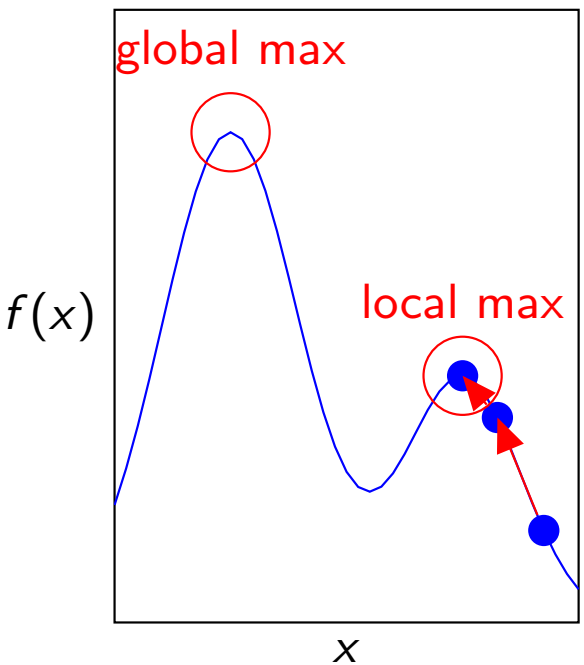
BF incrementally increases the objective function, following the positive slope.

This procedure tends to get stuck at local maxima.

[13] Wadayama et al. 2010.

Gradient Descent (or Gradient Ascent)

BF is related to Gradient Descent Optimization[13].



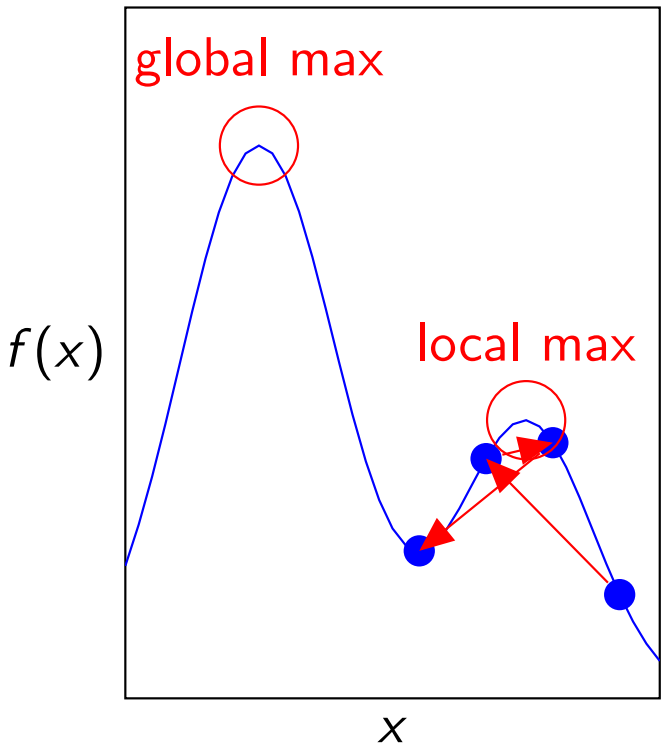
BF incrementally increases the objective function, following the positive slope.

This procedure tends to get stuck at local maxima.

[13] Wadayama et al. 2010.

Noisy Gradient Descent (or Ascent)

Noisy Gradient Descent can escape from local maxima[14].

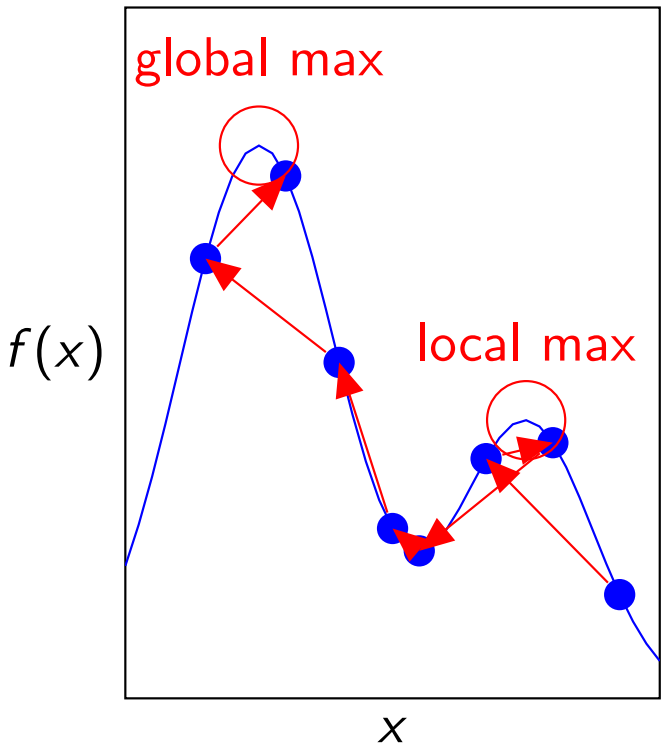


The guess x gets a random perturbation at each step.

[14] Sundararajan, Winstead, and Boutillon 2014.

Noisy Gradient Descent (or Ascent)

Noisy Gradient Descent can escape from local maxima[14].



The guess x gets a random perturbation at each step.

The algorithm can randomly escape the local maximum, and is more likely to reach the global maximum.

[14] Sundararajan, Winstead, and Boutillon 2014.

Noisy Bit-Flipping: BSC

PGDBF algorithm

Probabilistic Gradient Descent Bit Flipping (PGDBF)[15]:

$$\begin{aligned} E_k &= x_k y_k + \sum_{j \in \mathcal{N}_k} s_j \\ \text{flip } x_k \text{ if } E_k &< \theta \text{ with probability } p_{\text{flip}} \end{aligned}$$

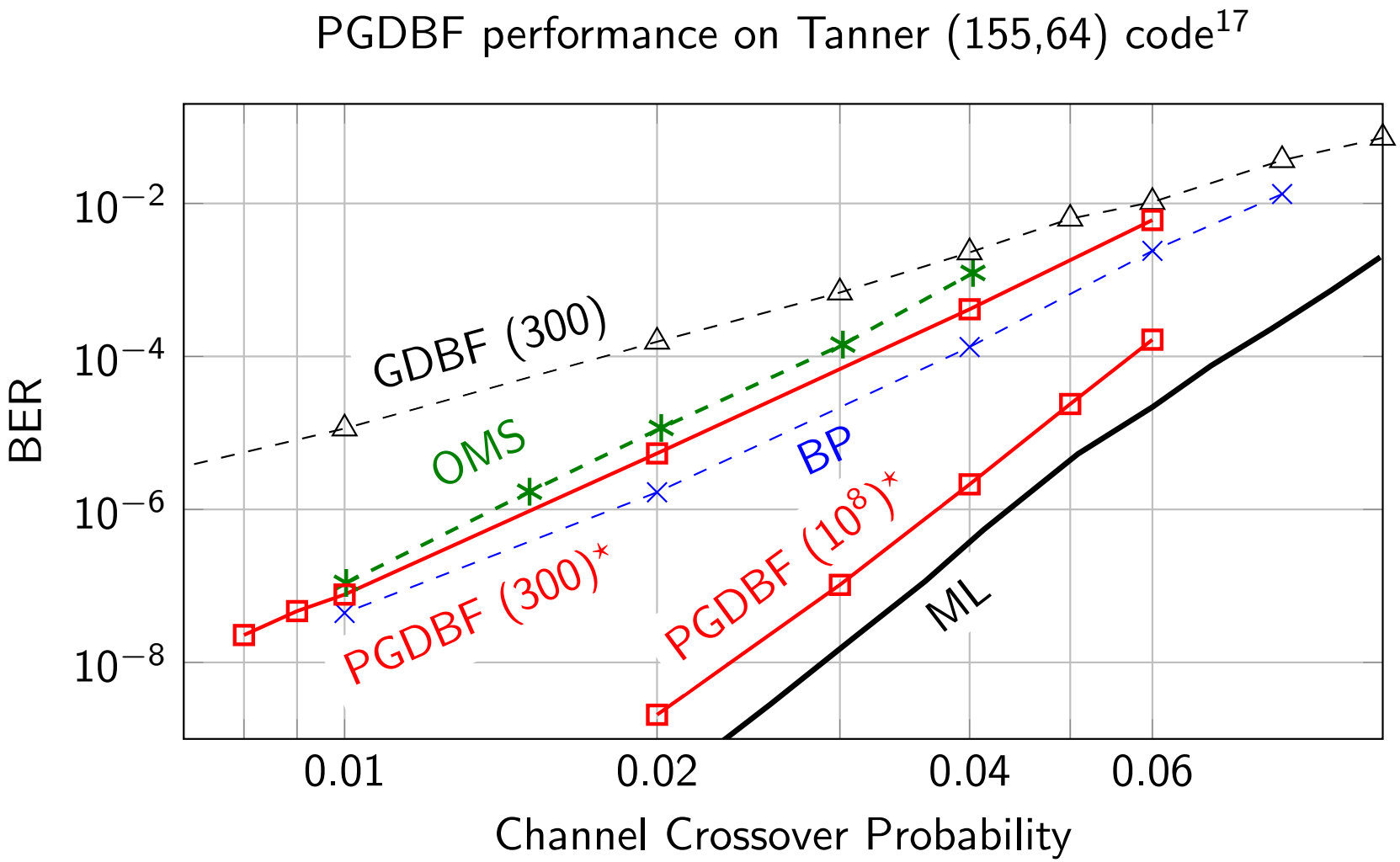
where θ is a flipping threshold[16].
The concept is closely related to the PBF algorithm.

[15] Al Rasheed, Ivanis, and Vasic 2014.

[16] Several algorithms are distinguished by their method of choosing θ

Noisy Bit-Flipping: BSC

PGDBF performance



⋆ data is shown for the DDS-PGDBF variant

(.) parenthesis indicate maximum iterations

[17] Declercq et al. 2016.

Noisy Bit-Flipping: AWGN

NGDBF algorithm

Noisy Gradient Descent Bit Flipping (NGDBF)[18]:

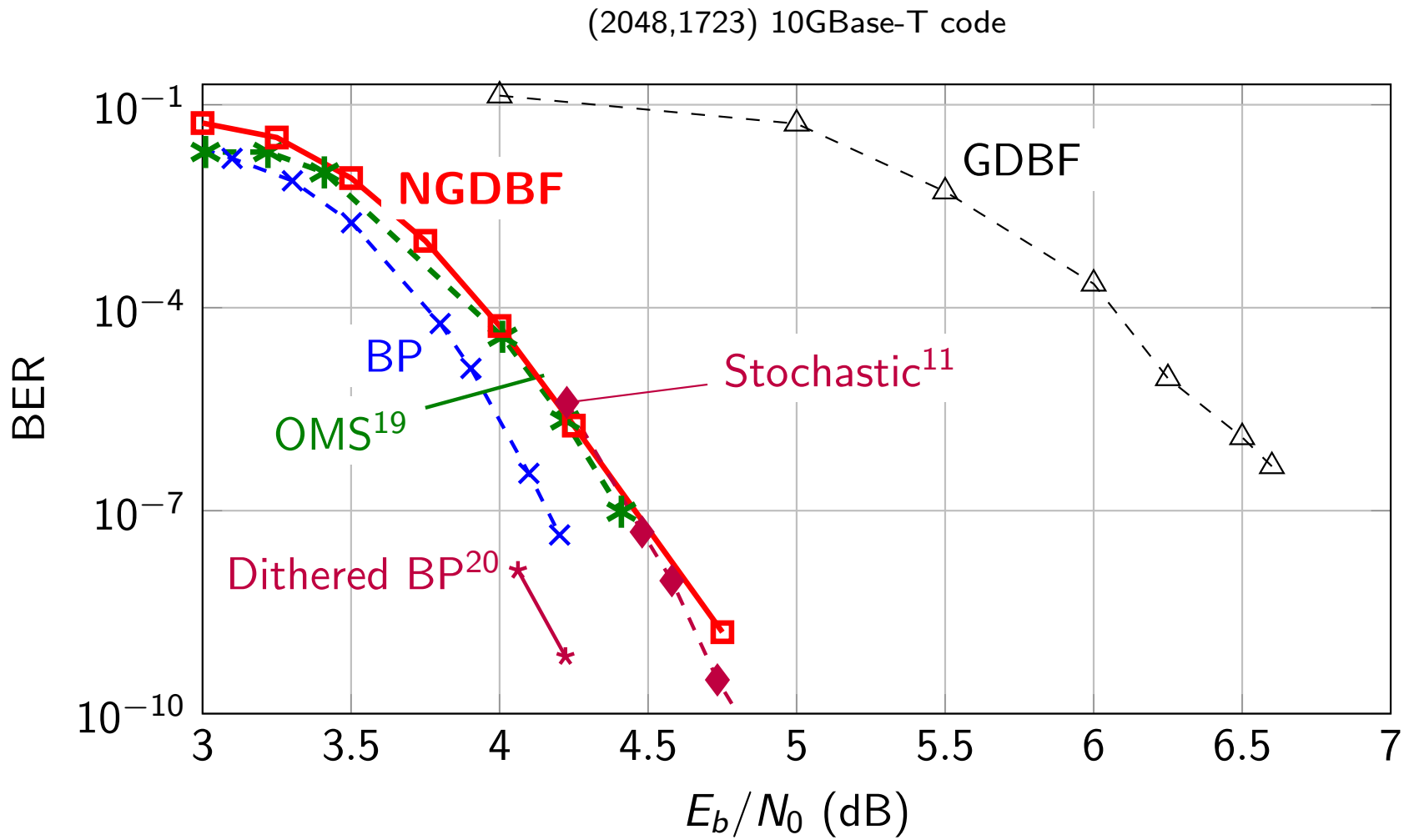
$$\begin{aligned} E_k &= x_k y_k + \sum_{j \in \mathcal{N}_k} s_j + q_k \\ \text{flip } x_k \text{ if } E_k &< \theta \end{aligned}$$

$$\text{where } q_k = \text{noise perturbation } \in \mathbb{R}$$

[18] Sundararajan, Winstead, and Boutillon 2014.

Noisy Bit-Flipping: AWGN

NGDBF performance



[19] Zhang et al. 2009.

[20] Leduc-Primeau et al. 2012.

Outline

Noisy Decoding: Absorbing Sets

Why is noise beneficial?

One explanation: as seen with PBF, correct states may be unreachable from some initial conditions.

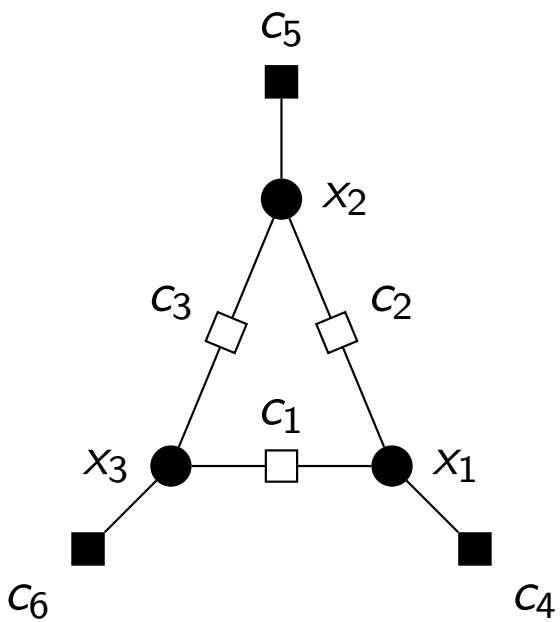
Non-determinism makes all states reachable.

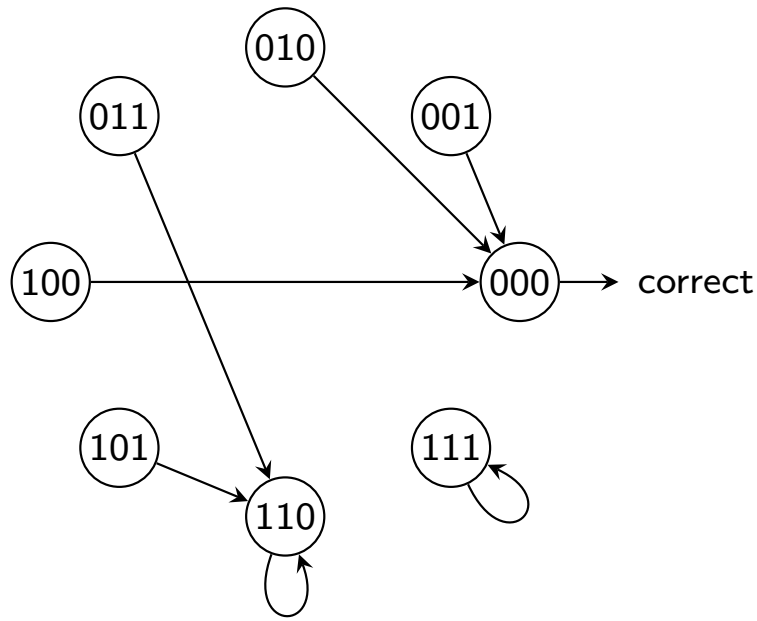
Noisy Decoding: Absorbing Sets

Example: (3,3) absorbing set

Consider a (3,3) subgraph with received channel samples $(-\epsilon, -\epsilon, +\epsilon)$, where $0 < \epsilon < 1$. The flipping threshold is $\theta = -\epsilon - 1$.

For deterministic GDBF:



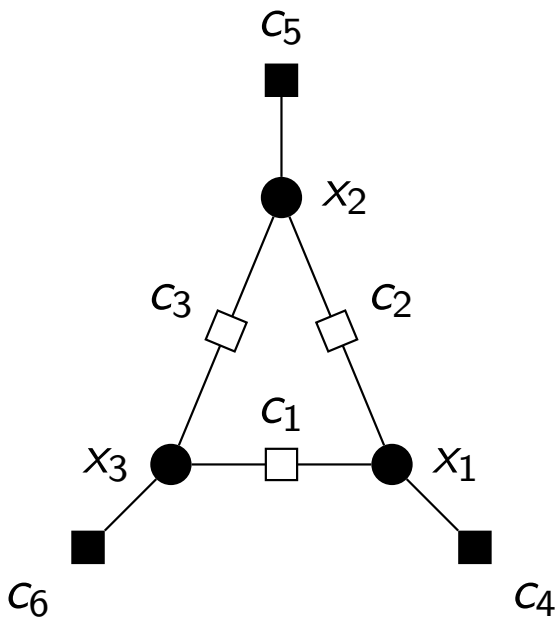


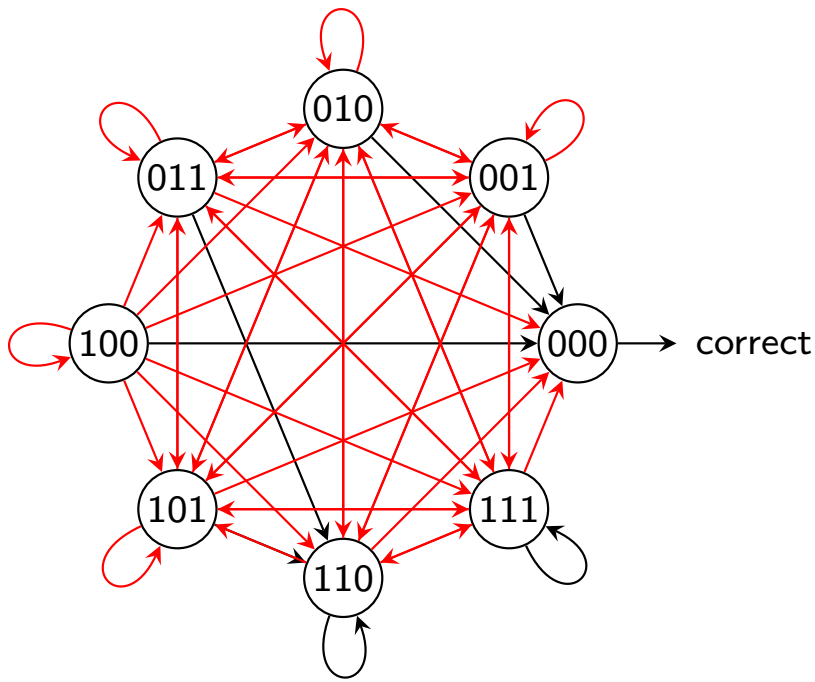
Noisy Decoding: Absorbing Sets

Example: (3,3) absorbing set

Consider a (3,3) subgraph with received channel samples $(-\epsilon, -\epsilon, +\epsilon)$, where $0 < \epsilon < 1$. The flipping threshold is $\theta = -\epsilon - 1$.

For Noisy GDBF





Noisy Decoding: Absorbing Sets

Markov Analysis

The noisy decoder is a homogeneous Markov process.

For given channel samples $\vec{y}$ we can compute:

- $P_c\left(\vec{y};\ell\right)$ – prob. of reaching the correct state in ℓ iterations.
- $P_e\left(\vec{y};\ell\right)$ – prob. of leaving the correct state within ℓ iterations.

Using importance sampling to marginalize $\vec{y}$, we obtain an estimate for the error floor associated with the absorbing set:

$$FER \approx N_{as}\left(1 - P_c\left(\ell\right)\right),$$

where N_{as} is the multiplicity of the absorbing set.

Noisy Decoding: Absorbing Sets

Error Overbound

Even if the correct state is reached, it is possible that noisy decoding can create spontaneous errors elsewhere in the frame.

From $P_e(\ell)$ we estimate the probability of failure due to spontaneous errors:

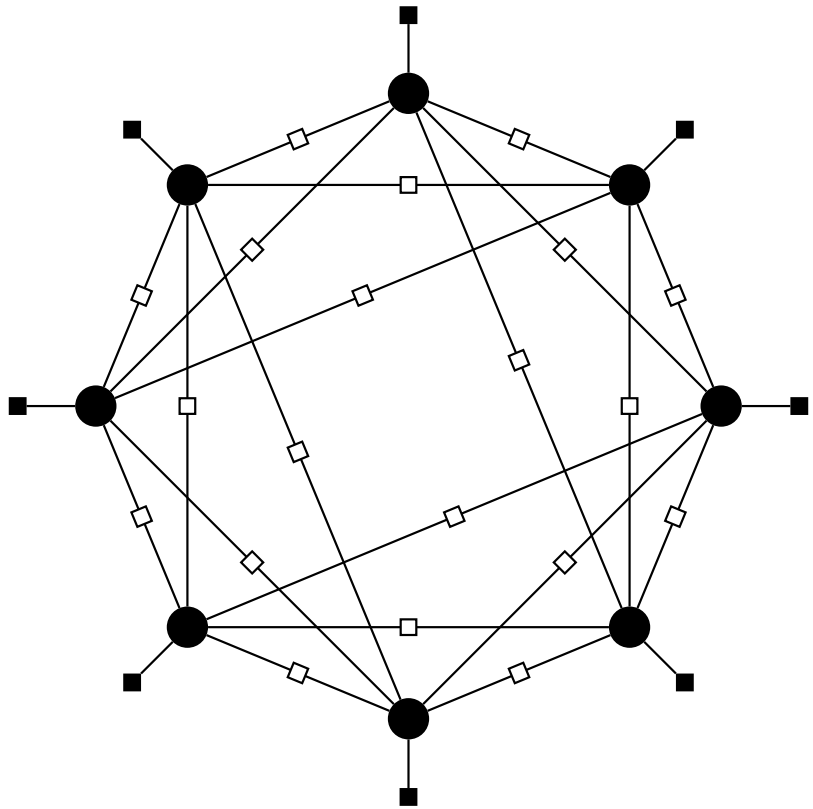
$$P_{se}(\ell) < N_{as} P_e(\ell).$$

This is an upper bound since spontaneous errors can also be corrected.

Noisy Decoding: Absorbing Sets

Bigger example: (8,8) set

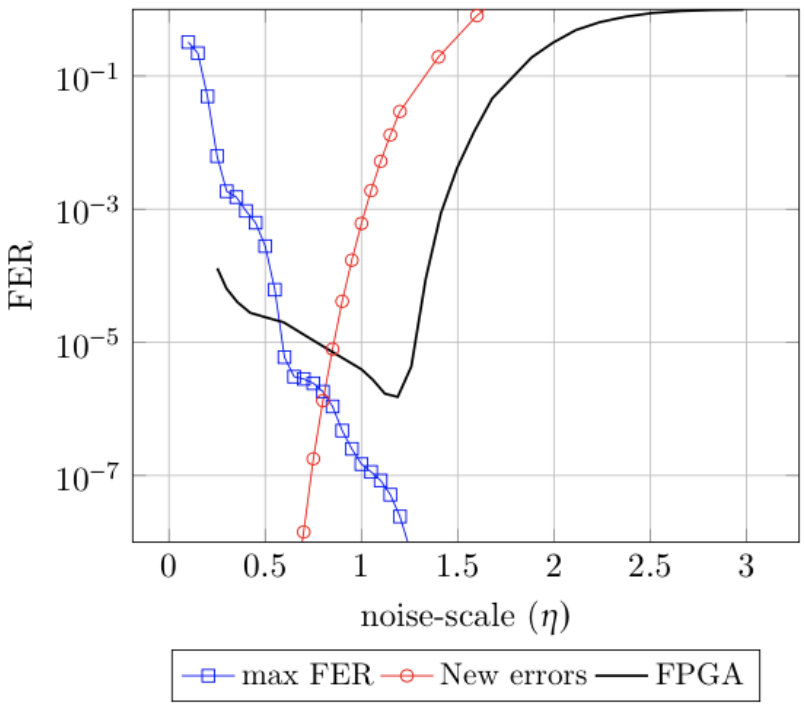
This (8,8) absorbing set is a dominant contributor to error floors in the 10GBase-T LDPC code.



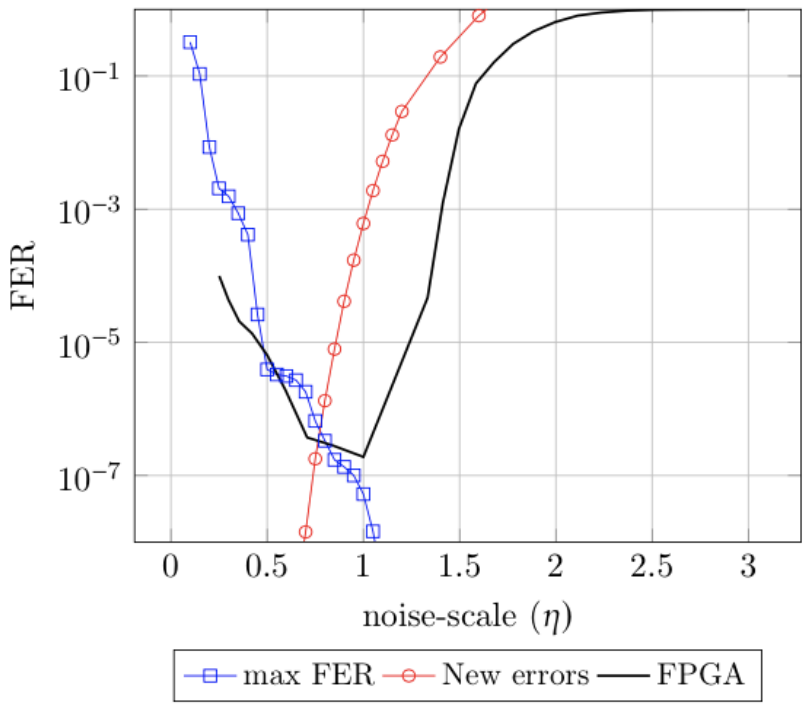
Absorbing Sets

Theoretical and Measured floor results: NGDBF for 10GBase-T LDPC code

FPGA decoder (black), Markov analysis (blue), and overbound (red)[21]:



(a) 600 iteration



(b) 5000 iterations

[21] Tithi 2019.

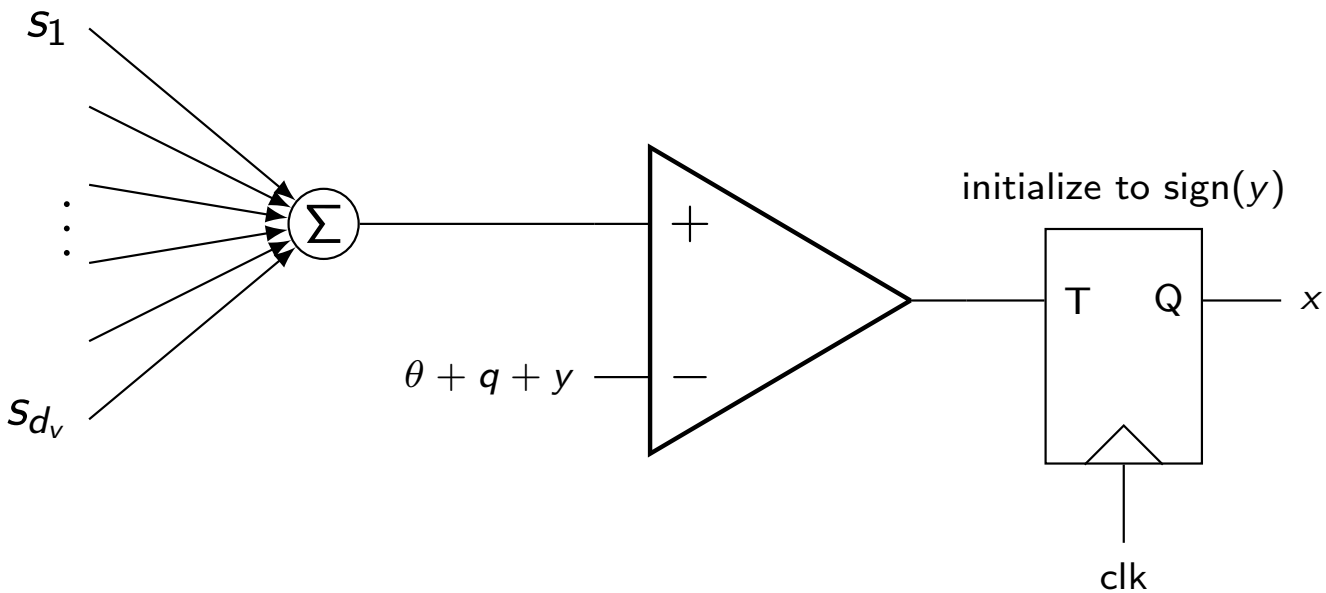
Outline

Architecture Comparisons

Symbol node processing

Stochastic and (N/P)GDBF have similar symbol node complexity.

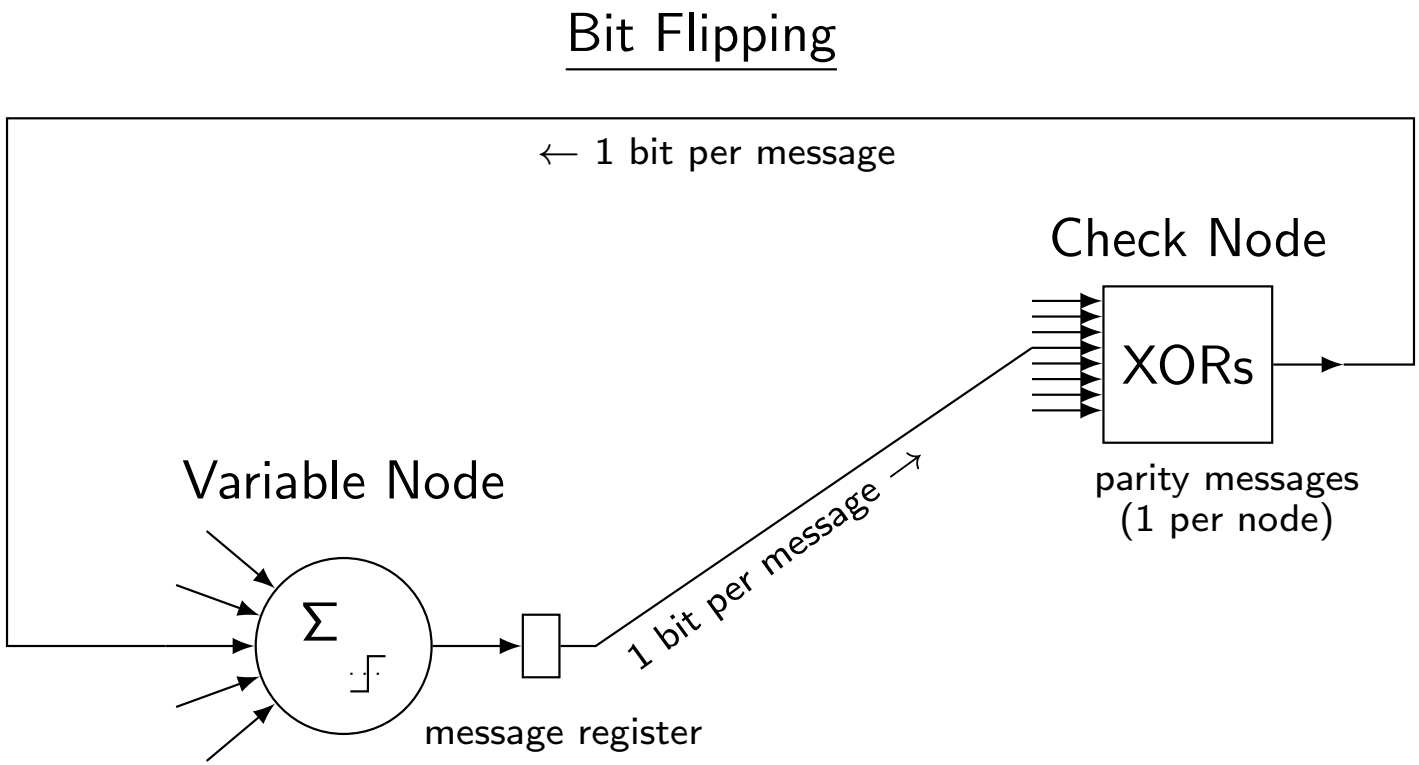
NGDBF symbol node:



Architecture Comparisons

BF Architecture

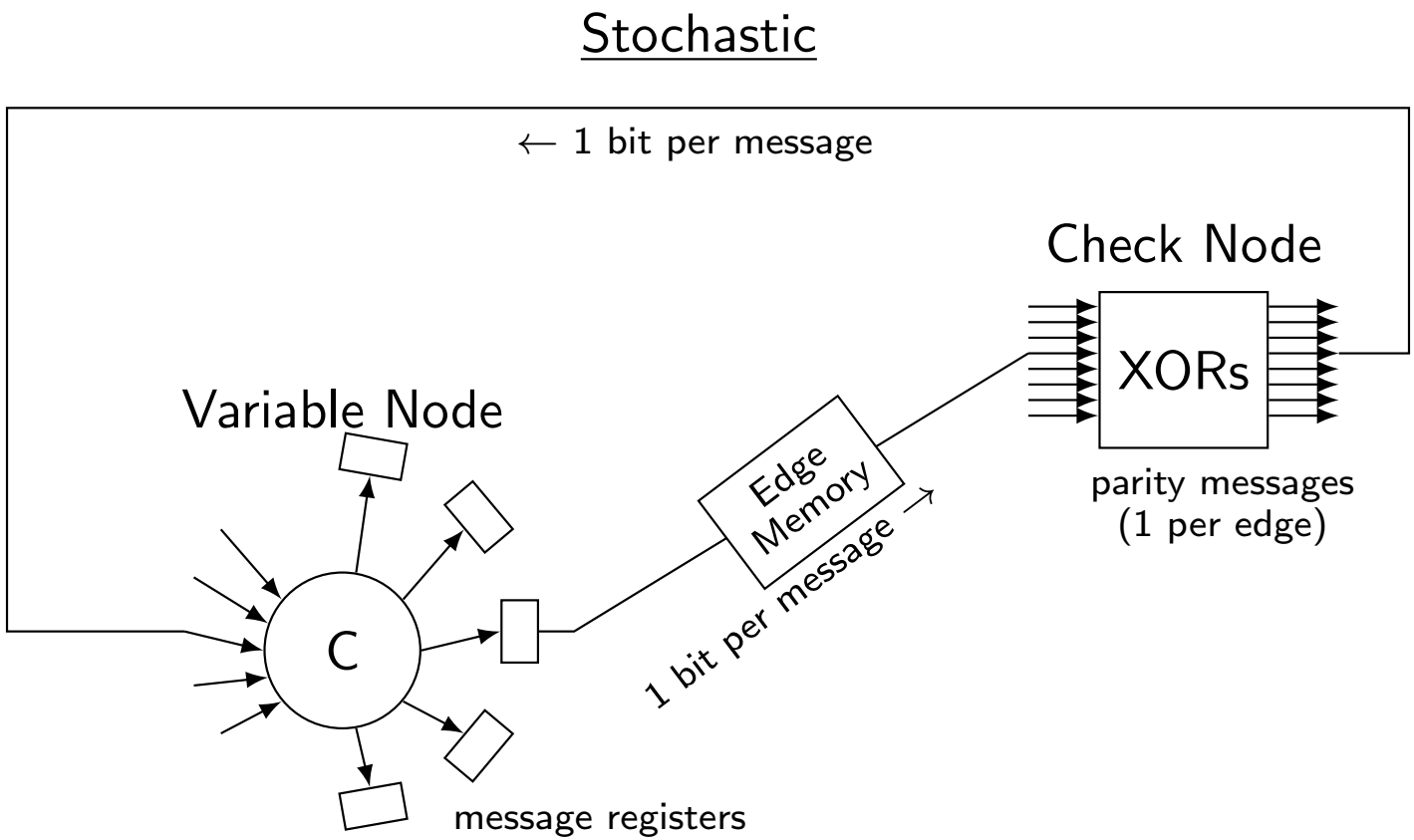
BF decoders compute one bit per node per iteration.



Architecture Comparisons

Stochastic Architecture

Stochastic decoders compute one bit per edge per iteration, and require edge memories.



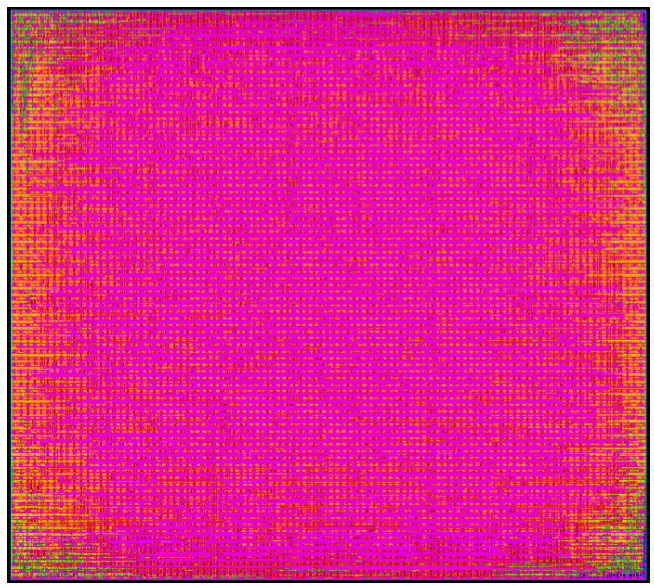
Application to the 802.3an 10GBaseT standard

Rate 0.8143 $(2048, 1723)$ with regular $(6, 32)$ degree distribution.

We implemented an NGDBF decoder in a 65 nm technology[22].

Close comparisons are available in the literature for 65 nm decoders, which include

- Offset Min-Sum (OMS)[23]
- OMS Split-Row architecture[24]
- Stochastic[25]
- Improved Differential Binary (IDB) (a deterministic BF algorithm)



Chip layout from Encounter

[22] Sundararajan and Winstead 2016.
[23] Zhang et al. 2010.
[24] Mohsenin 2010.
[25] Cushon et al. 2014.

ASIC Comparison for the 802.3 code

Direct comparisons (65nm):

Design:	NGDBF	IDB[26]	Split-Row MS[27]	OMS[28]
Quantization (bits)	7	6	5	4
Area (mm^2)	0.81	1.44	4.84	5.35
Clock (MHz)	188.67	520	195	700
E_b/N_0 at $\mathrm{BER} = 10^{-7}$	4.45	4.5	4.55 4.25	
At $\mathrm{SNR} = 4.55\,\mathrm{dB}$:				
Power (mW)	61.6	462	1359	-
Throughput (Gbps)	14.6	126.3	92.8	-
EpB (pJ/bit)	4.21	3.65	14.6	-
At $\mathrm{SNR} = 5.5\,\mathrm{dB}$:				
Power (mW)	63	478	-	2800
Throughput (Gbps)	36.4	171.8	-	47.7
EpB (pJ/bit)	1.73	2.78	-	58.7
Bit-flipping disadvantage:				
Worst-case Throughput (Gbps)	0.645	3.38	36.3	14.9

[26] Cushon et al. 2014.

[27] Mohsenin 2010.

[28] Zhang et al. 2010.

ASIC Comparison for the 802.3 code

Scaled comparisons (from 90nm):

Design:	NGDBF	Stochastic[26]	Layered OMS[27]
Quantization (bits)	7	6	4
Area (mm^2)	0.81	3.33	2.79
Clock (MHz)	188.67	500	85
E_b/N_0 at $\mathrm{BER} = 10^{-7}$	4.45	4.45	4.4
At $\mathrm{SNR} = 4.55\,\mathrm{dB}$:			
Power (mW)	61.6	-	-
Throughput (Gbps)	14.6	-	11.7
EpB (pJ/bit)	4.21	-	-
At $\mathrm{SNR} = 5.5\,\mathrm{dB}$:			
Power (mW)	63	-	-
Throughput (Gbps)	36.4	61.3	11.7
EpB (pJ/bit)	1.73	-	-
Bit-flipping disadvantage:			
Worst-case Throughput (Gbps)	0.645	2.56	11.7

[26] Cushon et al. 2014.

[27] Cevrero 2010.

PGDBF Implementations

Several PGDBF-based decoders have been implemented for the BSC.

These results are for 90nm CMOS, (1296,648) LDPC code:

Algorithm:	PPBF[28]	NS-PPBF	PGDBF[29]	GDBF
Clock Freq (MHz)	476	576	357	385
Area (mm^2)	0.367	0.349	0.367	0.360
At $\mathrm{FER} = 10^{-5}$:				
Throughput (Gbps)	80	95.1	84.4	222.8
$Gbps/mm^2$	218	272.4	230	618.9

PPBF: "Probabilistic Parallel Bit Flipping" (a PGDBF variant)

NS-PPBF: "No-Syndrome PPBF", improved critical path

[28] Le et al. 2018a.

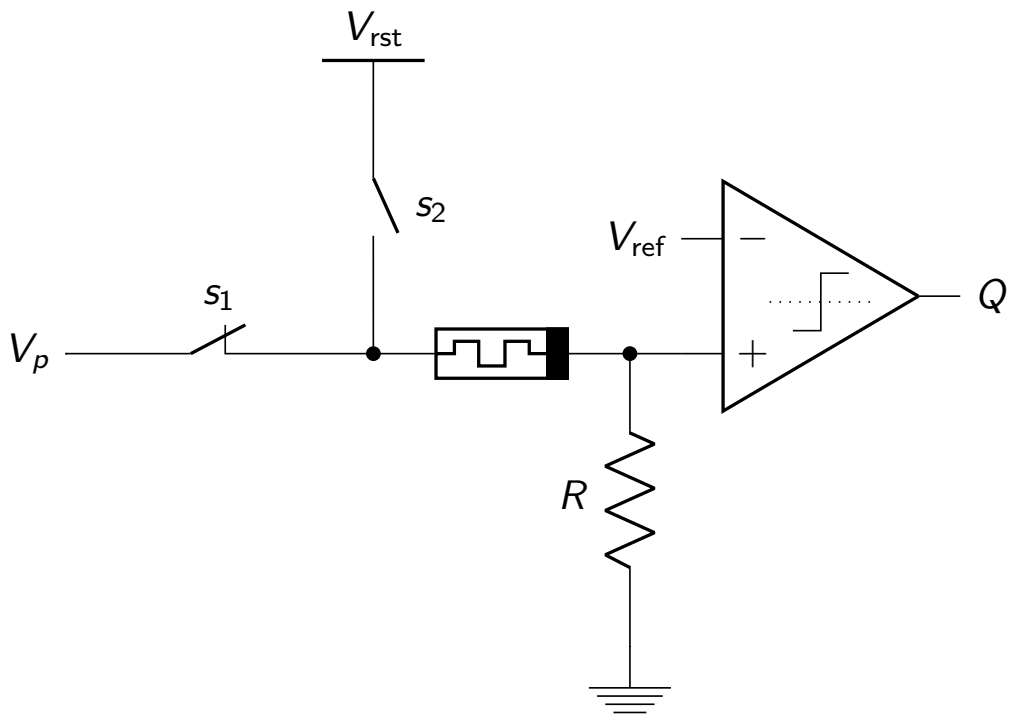
[29] Le et al. 2018b.

Outline

Non-Deterministic Electronics

Stochastic memristors

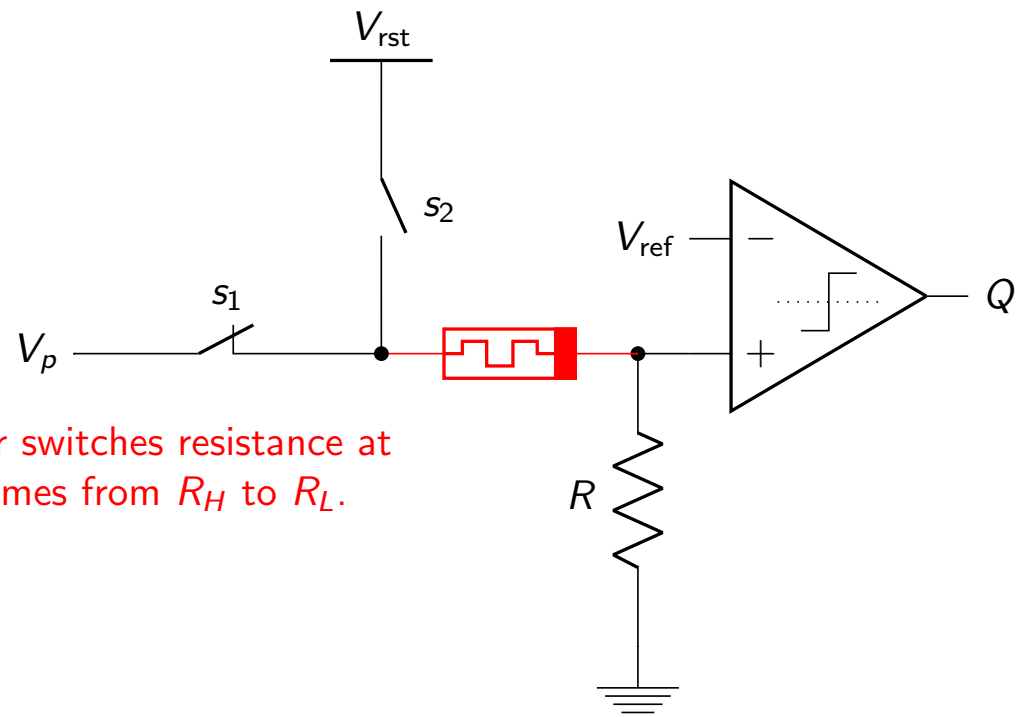
Memristors can be used to generate a true random Poisson variable.



Non-Deterministic Electronics

Stochastic memristors

Memristors can be used to generate a true random Poisson variable.

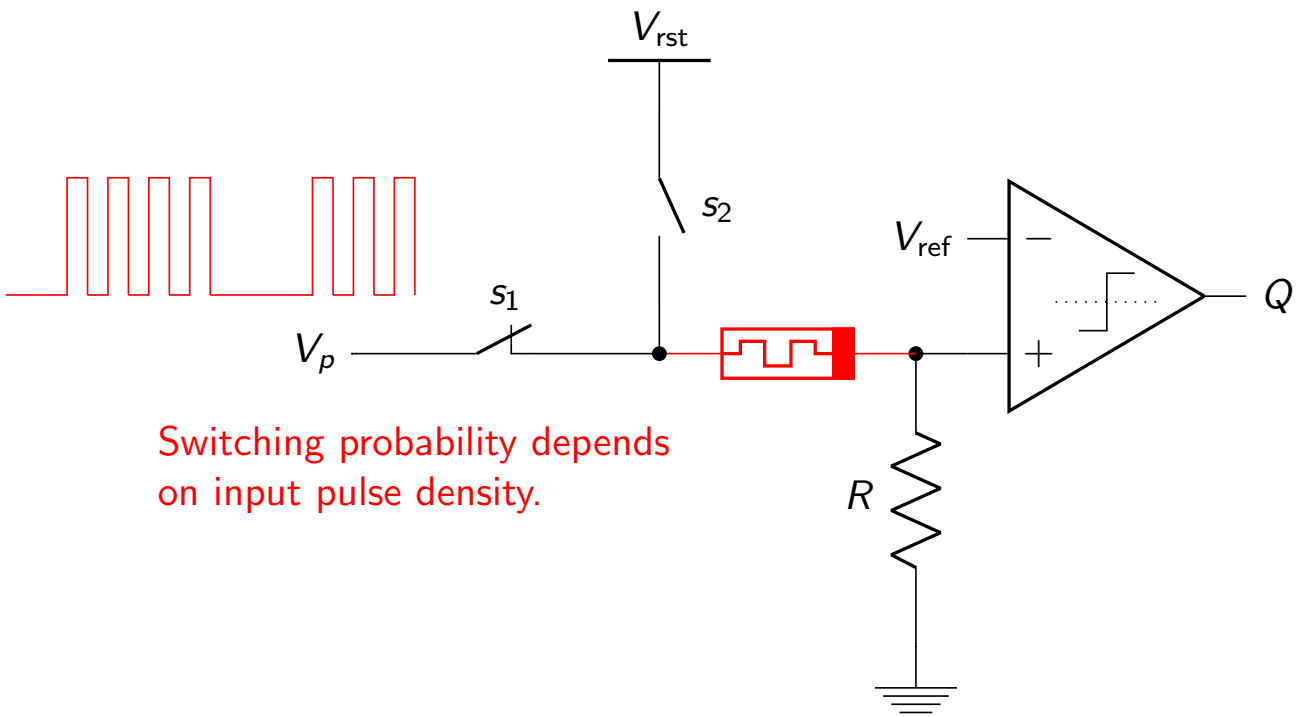


Memristor switches resistance at random times from R_H to R_L.

Non-Deterministic Electronics

Stochastic memristors

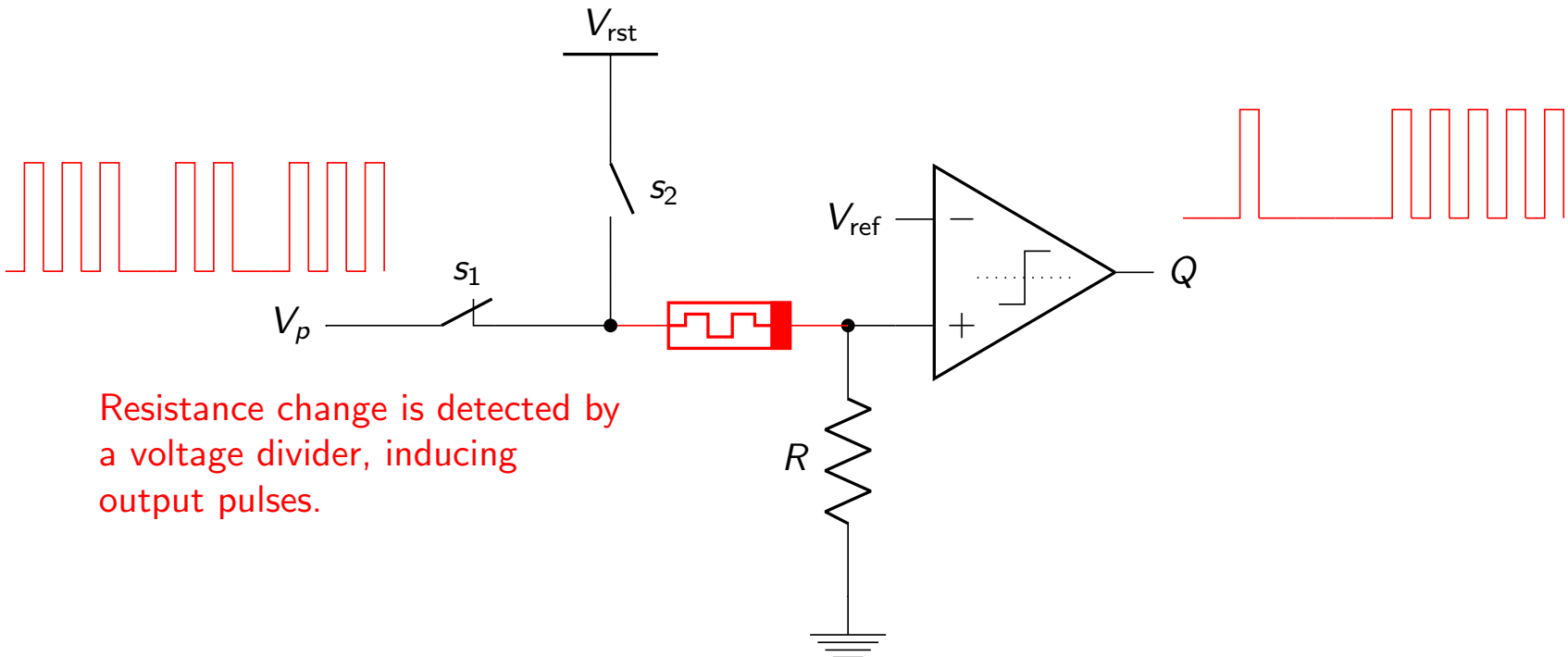
Memristors can be used to generate a true random Poisson variable.



Non-Deterministic Electronics

Stochastic memristors

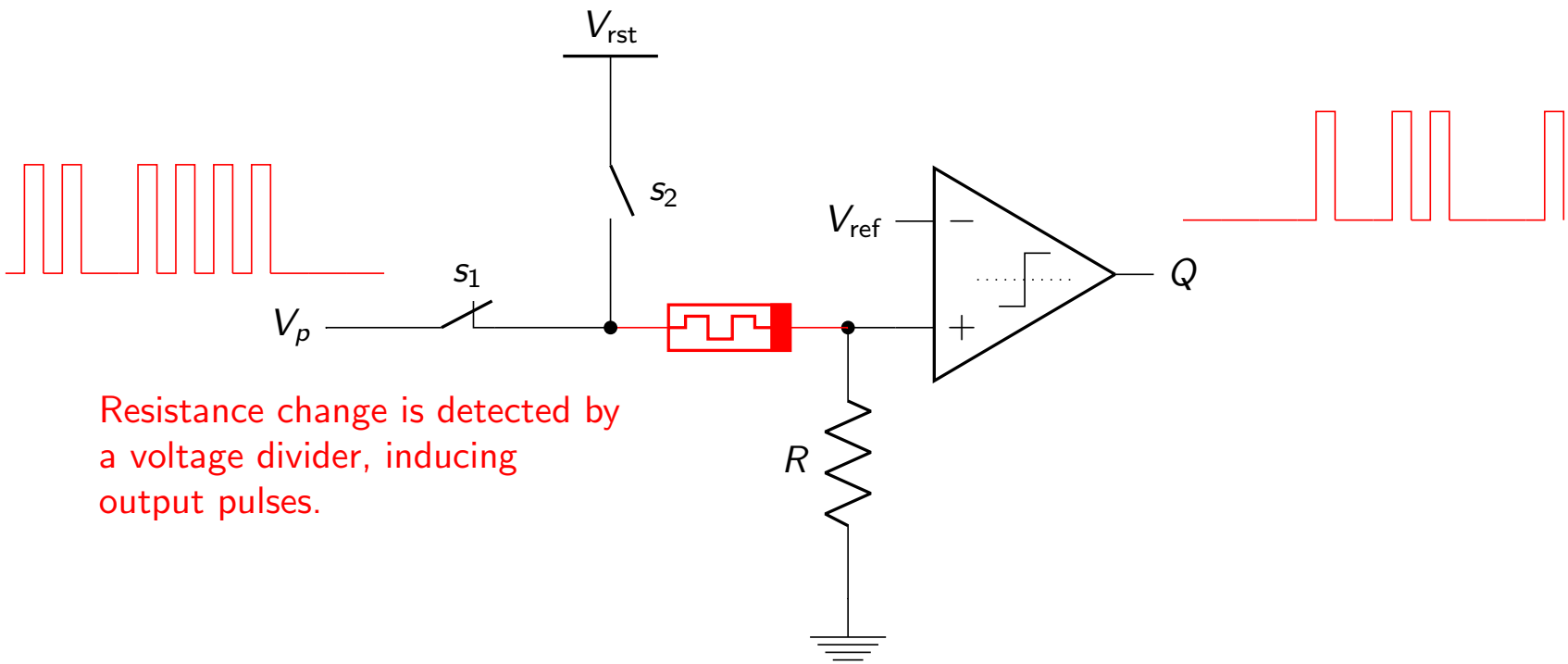
Memristors can be used to generate a true random Poisson variable.



Non-Deterministic Electronics

Stochastic memristors

Memristors can be used to generate a true random Poisson variable.



A source of cheap randomness for noisy decoders, and can (theoretically) replace edge memories in stochastic decoders.

Non-Deterministic Electronics

Other stochastic nano-devices

Other nano-scale device types have stochastic behavior:

- Phase-change devices[30]
- Magnetic tunnel junctions[31]
- Spin Orbit Torque devices[32]

There are now many articles applying stochastic nano-devices to Bayesian and neural networks.

[30] Piccinini, Brunetti, and Rudan 2017.
[31] Sengupta et al. 2016.
[32] Shim et al. 2017.

Outline

Conclusions

"Think Noisy"

Open problem: how Noisy Bit Flipping behaves in the waterfall region?

Outline

References I

- Leduc-Primeau, François et al. (2012). "Dithered belief propagation decoding". In: *IEEE Transactions on Communications* 60.8, pp. 2042–2047.
- Ngassa, C. L. K., V. Savin, and D. Declercq (2014). "Unconventional behavior of the noisy min-sum decoder over the binary symmetric channel". In: *Information Theory and Applications Workshop (ITA)*. DOI: 10.1109/ITA.2014.6804283.
- Cochachin, F. et al. (2017). "Density evolution thresholds for noise-against-noise min-sum decoders". In: *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–7. DOI: 10.1109/PIMRC.2017.8292326.
- Gaudet, Vincent C and Anthony C Rapley (2003). "Iterative decoding using stochastic computation". In: *Electronics Letters* 39.3, pp. 299–301.
- Lustenberger, Felix and H-A Loeliger (2001). "On mismatch errors in analog-VLSI error correcting decoders". In: *IEEE International Symposium on Circuits and Systems (ISCAS)*. Vol. 4, pp. 198–201.

References II


Winstead, Chris et al. (2005). "Stochastic iterative decoders". In: *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*. IEEE, pp. 1116–1120.


Tehrani, S. Sharifi, W. J. Gross, and S. Mannor (2006). "Stochastic decoding of LDPC codes". In: *IEEE Communications Letters* 10.10, pp. 716–718. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2006.060570.


Tehrani, S. Sharifi et al. (2010). "Majority-Based Tracking Forecast Memories for Stochastic LDPC Decoding". In: *IEEE Transactions on Signal Processing* 58.9, pp. 4883–4896. ISSN: 1053-587X. DOI: 10.1109/TSP.2010.2051434.


Miladinovic, Nenad and Marc PC Fossorier (2005). "Improved bit-flipping decoding of low-density parity-check codes". In: *IEEE Transactions on Information Theory* 51.4, pp. 1594–1606.


Wadayama, T. et al. (2010). "Gradient descent bit flipping algorithms for decoding LDPC codes". In: *IEEE Transactions on Communications* 58.6, pp. 1610–1614. ISSN: 0090-6778. DOI: 10.1109/TCOMM.2010.06.090046.

References III

- Sundararajan, G., C. Winstead, and E. Boutillon (2014). "Noisy Gradient Descent Bit-Flip Decoding for LDPC Codes". In: *IEEE Transactions on Communications* 62.10, pp. 3385–3400. ISSN: 0090-6778. DOI: 10.1109/TCOMM.2014.2356458.
- Al Rasheed, O., P. Ivanis, and B. Vasic (2014). "Fault-Tolerant Probabilistic Gradient-Descent Bit Flipping Decoder". In: *Communications Letters, IEEE* 18.9, pp. 1487–1490. ISSN: 1089-7798. DOI: 10.1109/LCOMM.2014.2344031.
- Declercq, David et al. (2016). "Noise-aided gradient descent bit-flipping decoders approaching maximum likelihood decoding". In: *International Symp. on Turbo Codes and Iterative Information Processing (ISTC)*. IEEE, pp. 300–304.
- Zhang, Z. et al. (2009). "Design of LDPC decoders for improved low error rate performance: quantization and algorithm choices". In: *IEEE Transactions on Communications* 57.11, pp. 3258–3268. ISSN: 0090-6778. DOI: 10.1109/TCOMM.2009.11.080105.
- Tithi, Tasnuva (2019). "Error floors of the 802.3 LDPC code for noise assisted decoding". PhD thesis. Utah State University.

References IV

- Sundararajan, Gopalakrishnan and Chris Winstead (2016). *ASIC Design of a Noisy Gradient Descent Bit Flip Decoder for 10GBASE-T Ethernet Standard*. arXiv:1608.06272.
- Zhang, Z. et al. (2010). "An Efficient 10GBASE-T Ethernet LDPC Decoder Design With Low Error Floors". In: *IEEE Journal of Solid-State Circuits* 45.4, pp. 843–855. ISSN: 0018-9200. DOI: 10.1109/JSSC.2010.2042255.
- Mohsenin, T. et al. (2010). "A Low-Complexity Message-Passing Algorithm for Reduced Routing Congestion in LDPC Decoders". In: *IEEE Trans. Circ. Syst. I, Reg. Papers* 57, pp. 1048–1061. ISSN: 1549-8328. DOI: 10.1109/TCSI.2010.2046957.
- Cushon, K. et al. (2014). "High-Throughput Energy-Efficient LDPC Decoders Using Differential Binary Message Passing". In: *IEEE Transactions on Signal Processing* 62.3, pp. 619–631. ISSN: 1053-587X. DOI: 10.1109/TSP.2013.2293116.
- Cevrero, A. et al. (2010). "A 5.35 mm2 10GBASE-T Ethernet LDPC decoder chip in 90 nm CMOS". In: *IEEE Solid State Circ. Conf.* DOI: 10.1109/ASSCC.2010.5716619.

References V

- Le, K. et al. (2018a). "A Probabilistic Parallel Bit-Flipping Decoder for Low-Density Parity-Check Codes". In: *IEEE Transactions on Circuits and Systems I: Regular Papers*, pp. 1–14. ISSN: 1549-8328. DOI: 10.1109/TCSI.2018.2849679.
- Le, K. et al. (2018b). "Variable-Node-Shift Based Architecture for Probabilistic Gradient Descent Bit Flipping on QC-LDPC Codes". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 65.7, pp. 2183–2195. ISSN: 1549-8328. DOI: 10.1109/TCSI.2017.2777802.
- Piccinini, E., R. Brunetti, and M. Rudan (2017). "Self-Heating Phase-Change Memory-Array Demonstrator for True Random Number Generation". In: *IEEE Transactions on Electron Devices* 64.5, pp. 2185–2192. ISSN: 0018-9383. DOI: 10.1109/TED.2017.2673867.
- Sengupta, Abhronil et al. (2016). "Magnetic Tunnel Junction Mimics Stochastic Cortical Spiking Neurons". In: *Scientific Reports* 6, 30039 EP –. URL: https://doi.org/10.1038/srep30039.

References VI

Shim, Yong et al. (2017). "Stochastic Spin-Orbit Torque Devices as Elements for Bayesian Inference". In: *Scientific Reports* 7.1, p. 14101. DOI: 10.1038/s41598-017-14240-z. URL: https://doi.org/10.1038/s41598-017-14240-z.