Modified Belief-Propagation Decoder Exploiting the Degeneracy of Quantum Surface Codes

(an extended work of DOI:10.1109/JSAIT.2020.3011758)

Kao-Yueh Kuo and Ching-Yi Lai

Institute of Communications Engineering, National Chiao Tung University,
Hsinchu 30010, Taiwan

August, 2020

Outline

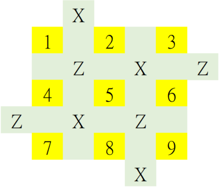
Sparse Quantum Codes

- Quantum states are very sensitive.
- Techniques similar to classical error-correction can be used for protection
- **Stabilizer codes** is a major class of quantum error-correcting codes
  - A stabilizer code is the fixed subspace of a set of Pauli operators.
  - An operator is called **sparse** if it has a low weight
    - ★ e.g., $I \otimes I \otimes I \otimes X \otimes I \otimes Z$ has a weight 2.
- Stabilizer codes defined by sparse Pauli operators have good performance and low encoding/decoding complexity.[1]
- **Topological codes** (due to Kitaev): toric codes, surface codes, etc.
  - suitable for superconducting implementation
  - based on local measurements
  - every operator's weight ≤ 4 (independent of the code length)

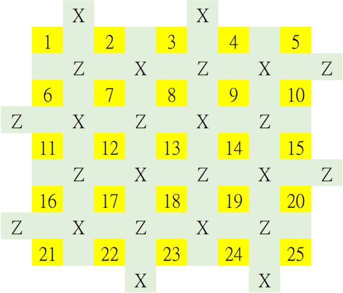
[1] There are other sparse quantum codes, as discussed in the JSAIT paper. Here we focus on surface codes.

Surface Codes

- A surface code encodes 1 information qubit by $N = L^2$ qubits.
- There are $M = N - 1$ (projective) measurement operators.
- The most small surface code has 9 qubits (LHS figure):
  - A yellow box is a qubit.
  - The label X between qubits 1, 2 represents
    $X_1X_2 \quad = \quad X \otimes X \otimes I \otimes I \otimes I \otimes I \otimes I \otimes I \otimes I$
  - The label Z between qubits 1, 2, 4, 5 represents
    $Z_1Z_2Z_4Z_5 = Z \otimes Z \otimes I \otimes Z \otimes Z \otimes I \otimes I \otimes I \otimes I$
  - and so on



$L = 3$ (up) and $L = 5$ (right).



Some Decoding Strategies

N: the code length. (Given the measurement results:)

- **Minimum-weight matching** (MWM) finds a valid error with *minimum-weight*,[2] for decoding topological codes
  - with a complexity $O(N^2)$ after simplification
- **Renormalization group** (RG) algorithm decodes topological codes by treating them as *concatenated codes* (divide and conquer)
  - with a complexity $\propto N \log(\sqrt{N})$
- **Belief Propagation** (BP) is an efficient & powerful algorithm used in coding and AI communities (due to Gallager and Pearl)
  - can be used to decode any sparse codes
  - with a complexity $O(Nj\tau)$, which could be nearly linear in N, so we are interested in this algorithm
    - ★ j is the mean column-weight of the check matrix: due to sparsity, $j \ll N$ or even fixed, e.g. $j \leq 4$ for toric/surface codes
    - ★ τ is the average number of iterations, with $\tau = O(\log \log N)$.[3]

[2] "*minimum-weight*" is optimal in the classical sense before considering the quantum **degeneracy**.

[3] This happens if BP converges well for most errors. In the JSAIT paper, we refined BP to well converge for decoding bicycle codes and hypergraph-product codes. Here we extend the result for decoding topological codes.

BP Decoding of Quantum Codes

Some BP issues for decoding quantum codes:

- Performance: a stabilizer check matrix has many **short cycles**, which affect the decoding convergence & degrade the decoding performance.
- Complexity: handling I, X, Y, Z needs a **quaternary BP** (BP_4).
  - It is 16 times complex than the classical **binary BP** (BP_2).

Our approach:

- Refine and Modify

$$\text{(Original) BP}_4 \xrightarrow{\text{refine}} \text{Refined BP}_4 \xrightarrow{\text{modify}} \text{Refined BP}_4\ (\alpha_i)$$

(equivalent to BP_4 but is a BP_2-like algorithm)
lower complexity

(introduce a parameter α_i to adjust the algorithm)
improved performance

Outline

Stabilizer Code and Check Matrix

- An $[[N, K]]$ stabilizer code is a 2^K-dim. subspace in $\mathbb{C}^{2^N}$ that is the common $(+1)$-eigenspace of a stabilizer group $\mathcal{S}$, where: $\mathcal{S}$ is a commutative subgroup of the N-fold Pauli group $\{\pm 1, \pm i\} \times \{I, X, Y, Z\}^{\otimes N}$ such that $\mathcal{S}$ has $N - K$ independent generators and $-I^{\otimes N} \notin \mathcal{S}$.
- We can construct an $M \times N$ check matrix by $M \geq N - K$ operators. ($M = N - K$ if we only choose independent generators.)
- For example:
  if $\mathcal{S}$ is generated by $X \otimes Y \otimes I$ and $Z \otimes Z \otimes Y$,

  then $S = \begin{bmatrix} X & Y & I \\ Z & Z & Y \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} & S_{13} \\ S_{21} & S_{22} & S_{23} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}$.

  - $\mathcal{S} = \{III, XYI, ZZY, YXY\}$ is the (productive) rowspace of S.

The Encoding/Decoding Scenario

- There is an Enc_Proc associated with S, and if we assume a noisy Pauli channel:[4]

$$K \text{ qubits} \xrightarrow{\text{Enc\_Proc}} \rho\ (N \text{ qubits}) \xrightarrow{\textbf{Pauli ch.}} \mathcal{E}(\rho)$$

  subject to an unknown error $E = E_1 E_2 \ldots E_N \in \{I, X, Y, Z\}^N$.

- Measurement (by S) and Decoding:

$$\mathcal{E}(\rho) \xrightarrow{\text{meas. by } S} z\ (M \text{ syndrome bits}) \xrightarrow{\textbf{decoder}} \hat{E} \quad \text{(to apply to } \mathcal{E}(\rho))$$

  (post-measurement state is still $\mathcal{E}(\rho)$.)

  such that $\hat{E} \in E\mathcal{S}$ with a probability as higher as possible.

- Unlike classical error-correction that needs $\hat{E} = E$, here it needs to maximize the probability $\hat{E} \in E\mathcal{S}$ due to **degeneracy**.[5]

---

[4] This is sufficient according to the error discretization theorem.

[5]
- Maximizing $P(\hat{E} = E)$ was shown to be NP-hard.
- Maximizing $P(\hat{E} \in E\mathcal{S})$ was also shown to be NP-hard, or even harder, #P-hard.

The Binary Syndrome

- The syndrome $z = (z_1, z_2, \ldots, z_M) \in \{0,1\}^M$ is a **binary vector** such that the m-th syndrome bit satisfies

$$z_m = \sum_{n=1}^{N} \langle E_n, S_{mn} \rangle \mod 2$$

which indicates whether the unknown E and the m-th stabilizer S_m commute ($z_m = 0$) or anticommute ($z_m = 1$).

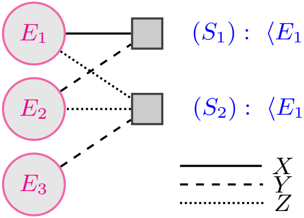
- Since E has $4^N = 2^{2N}$ possibilities, this is a bit like solving an Ising Model with $2N$ variables and M constraints.

Table: Commutation Relations of Pauli Operators (0: commute, 1: anticommute)

$\langle E_n, F_n \rangle$	$F_n = I$	$F_n = X$	$F_n = Y$	$F_n = Z$
$E_n = I$	0	0	0	0
$E_n = X$	0	0	1	1
$E_n = Y$	0	1	0	1
$E_n = Z$	0	1	1	0

BP: Message Passing on Tanner Graph

- BP decoding is an iterative **message-passing** algorithm run on a bipartite graph (called **Tanner graph**) defined by S.
- For example, $S = \begin{bmatrix} X & Y & I \\ Z & Z & Y \end{bmatrix}$ has a Tanner graph:



$(S_1): \ \langle E_1, X\rangle + \langle E_2, Y\rangle = z_1$

$(S_2): \ \langle E_1, Z\rangle + \langle E_2, Z\rangle + \langle E_3, Y\rangle = z_2$

- BP_4 starts with an initial belief (for the error in each qubit)

$$\boldsymbol{p}_n = (p_n^I, p_n^X, p_n^Y, p_n^Z) \quad \left(\text{e.g.} = (1-\epsilon, \frac{\epsilon}{3}, \frac{\epsilon}{3}, \frac{\epsilon}{3}) \text{ for a depolarizing ch.}\right) \qquad (1)$$

(given S and z) to compute an updated belief

$$\boldsymbol{q}_n = (q_n^I, q_n^X, q_n^Y, q_n^Z) \qquad (2)$$

and infer $\hat{E}_n = \arg\max_{W \in \{I,X,Y,Z\}} q_n^W$.

Original BP$_4$: (every message is a vector)

- To complete the 1st iteration:
  variable node n passes to check node m the message
  $$\boldsymbol{q}_{n\to m} = (q^I_{mn}, q^X_{mn}, q^Y_{mn}, q^Z_{mn}) = \boldsymbol{p}_n,$$
  and check node m passes to variable node n the message
  $$\boldsymbol{r}_{m\to n} = (r^I_{mn}, r^X_{mn}, r^Y_{mn}, p^Z_{mn}), \text{ with}$$
  $$r^W_{mn} = \sum_{\substack{E|_{\mathcal{N}(m)}:\ E_n=W,\\ \langle E|_{\mathcal{N}(m)}, S_m|_{\mathcal{N}(m)}\rangle = z_m}} \left(\prod_{n'\in\mathcal{N}(m)\setminus n} q^{E_{n'}}_{mn'} \right)$$
  for $W \in \{I, X, Y, Z\}$, where $\mathcal{N}(m) = \{n \mid S_{mn} \neq I\}$.
- For any next iteration, $\boldsymbol{q}_{n\to m} = (q^I_{mn}, q^X_{mn}, q^Y_{mn}, q^Z_{mn})$ with
  $$q^W_{mn} \propto p^W_n \prod_{m'\in\mathcal{M}(n)\setminus m} r^W_{m'n}$$
  where $\mathcal{M}(n) = \{m \mid S_{mn} \neq I\}$.
- To infer $\hat{E}_n$ is by $\boldsymbol{q}_n = (q^I_n, q^X_n, q^Y_n, q^Z_n)$ with $q^W_n = p^W_n \prod_{m\in\mathcal{M}(n)} r^W_{mn}$.

Refine and Modify the Algorithm

To refine: (to have a lower complexity)

- An observation: $\langle E_1, S_{m1}\rangle = z_m + \sum_{n=2}^{N} \langle E_2, S_{mn}\rangle \mod 2$
  - In other words, the message from a neighboring check will tell us more likely whether the error E_1 commutes or anti-commutes with S_{m1}
- We derived a refined algorithm by, e.g., if $S_{mn} = X$, then passing $d_{n\to m} = (q_{mn}^{I} + q_{mn}^{X}) - (q_{mn}^{Y} + q_{mn}^{Z})$ is sufficient for computation.
- Every message becomes a scalar, and the check-node efficiency is 16 times improved.

To modify: (to have improved performance)

- A stabilizer check matrix has many **short cycles**, which cause the *wrong belief worsely propagated* in the decoding network.
- We modify the algorithm by introducing a parameter α_i.
  - to suppress the wrong belief,
  - to create inhibition between nodes.[6]

[6] BP is like a recurrent neural network (RNN)—inhibition between nodes enhances the perception capability, as found for a Hopfield net (network with symmetric connections).

Refined BP_4, with Modification by α_i

- **The most high-complexity step** is refined.
- **The computation in (3) and (4)** is modified for performance improvement.

(The nonlinear function can be efficiently implemented by Schraudolph's approximation)

Algorithm 1 : Quaternary BP (BP_4) with message normalization and inhibition between nodes controlled by α_i.

Input: $S \in \{I, X, Y, Z\}^{M \times N}$, $\{\boldsymbol{p}_n = (p_n^I, p_n^X, p_n^Y, p_n^Z)\}_{n=1}^N$, target $z \in \{0,1\}^M$, and a real parameter α_i.

Initialization. For $n = 1, 2, \dots, N$ and $m \in \mathcal{M}(n)$, let

$$d_{n\to m} = q_{n\to m}^{(0)} - q_{n\to m}^{(1)},$$

where $q_{n\to m}^{(0)} = p_n^I + p_n^{S_{mn}}$ and $q_{n\to m}^{(1)} = 1 - q_{n\to m}^{(0)}$.

Horizontal Step. For $m = 1, 2, \dots, M$ and $n \in \mathcal{N}(m)$, compute

$$\delta_{m\to n} = (-1)^{z_m} \prod_{n' \in \mathcal{N}(m)\setminus n} d_{n'\to m},$$

Vertical Step. For $n = 1, 2, \dots, N$ and $m \in \mathcal{M}(n)$, do:

- Compute

$$r_{m\to n}^{(0)} = \left(\tfrac{1+\delta_{m\to n}}{2}\right)^{1/\alpha_i}, \quad r_{m\to n}^{(1)} = \left(\tfrac{1-\delta_{m\to n}}{2}\right)^{1/\alpha_i}, \quad (3)$$

$$q_{n\to m}^I = p_n^I \prod_{m' \in \mathcal{M}(n)\setminus m} r_{m'\to n}^{(0)},$$

$$q_{n\to m}^W = p_n^W \prod_{m' \in \mathcal{M}(n)\setminus m} r_{m'\to n}^{\langle W, S_{m'n}\rangle}, \text{ for } W \in \{X, Y, Z\}.$$

- Let

$$q_{n\to m}^{(0)} = a_{mn}\left(q_{n\to m}^I + q_{n\to m}^{S_{mn}}\right) / \left(\tfrac{1+\delta_{m\to n}}{2}\right)^{1-1/\alpha_i},$$
$$q_{n\to m}^{(1)} = a_{mn}\left(\textstyle\sum_{W'} q_{n\to m}^{W'}\right) / \left(\tfrac{1-\delta_{m\to n}}{2}\right)^{1-1/\alpha_i}, \quad (4)$$

where $W' \in \{X, Y, Z\} \setminus S_{mn}$ and a_{mn} is a chosen scalar such that $q_{n\to m}^{(0)} + q_{n\to m}^{(1)} = 1$.

- Update: $d_{n\to m} = q_{n\to m}^{(0)} - q_{n\to m}^{(1)}$.

Hard Decision. For $n = 1, 2, \dots, N$, compute

$$q_n^I = p_n^I \textstyle\prod_{m \in \mathcal{M}(n)} r_{mn}^{(0)}$$
$$q_n^W = p_n^W \textstyle\prod_{m \in \mathcal{M}(n)} r_{mn}^{\langle W, S_{mn}\rangle}, \text{ for } W \in \{X, Y, Z\}.$$

Let $\hat{E}_n = \arg\max_{W \in \{I,X,Y,Z\}} q_n^W$.

- Let $\hat{E} = \hat{E}_1 \hat{E}_2 \cdots \hat{E}_N$.
  - If $\langle \hat{E}, S_m \rangle = z_m$ for $m = 1, 2, \dots, M$, halt and return "SUCCESS";
  - otherwise, if a maximum number of iterations is reached, halt and return "FAIL";
  - otherwise, repeat from the horizontal step.

The Parameter α_i

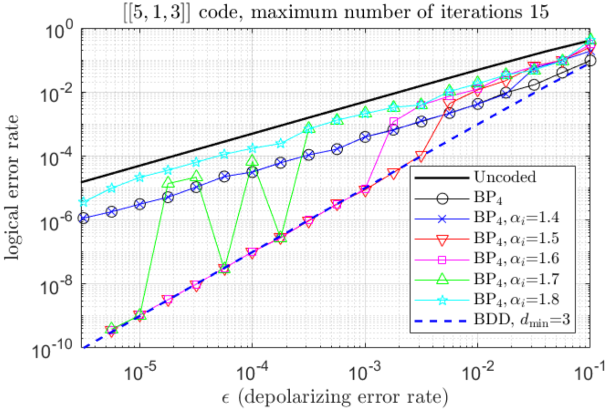
Two features:

- Against short cycles: original BP will decouple the $n \to m$ message and the $m \to n$ message **that are passed on the same edge**.
  - This is suitable for the case of less short cycles; here we need a different strategy.
  - Introducing α_i (especially in (4)) breaks the decoupling rule and creates strong memory effect at check-node side (fed back from variable-node side).
- Flexibility (assume $\alpha_i > 0$):
  - a larger $\alpha_i > 1$ corresponds to a careful (smaller-step) search with stronger memory at check-node side,
  - a smaller $\alpha_i < 1$ corresponds to an aggregate (larger-step) search with weaker memory at check-node side.

Outline

The famous five-qubit code, $[[N = 5,\, K = 1,\, d_{\min} = 3]]$

- This code has $S = \begin{bmatrix} X & Z & Z & X & I \\ I & X & Z & Z & X \\ X & I & X & Z & Z \\ Z & X & I & X & Z \end{bmatrix}$, and can correct any weight-one errors.
  - BP_4 without α_i (or say $\alpha_i = 1$) cannot correct the error $IIIYI$.
  - BP_4 with $\alpha_i \approx 1.5$ successfully corrects any weight-one errors.
- The logical error rate (the lower the better) at different ϵ:

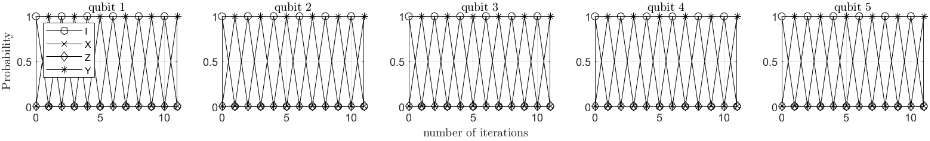


How variable nodes converge (for decoding $IIIYI$ at $\epsilon = 0.003$)

$\boldsymbol{q}_n = (q_n^I, q_n^X, q_n^Y, q_n^Z)$ is the local observation of variable node n.

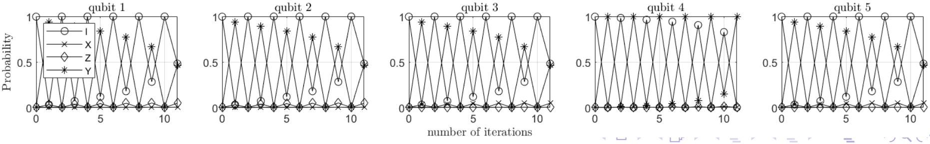
We can normalize it as a distribution and plot it (for each qubit when the iteration increases):

- Without α_i, the trajectory of each $\boldsymbol{q}_n$, $n = 1, 2, 3, 4, 5$, keeps oscillating:



- With $\alpha_i = 1.5$, it slowly **suppress the wrong belief** to converge correctly:

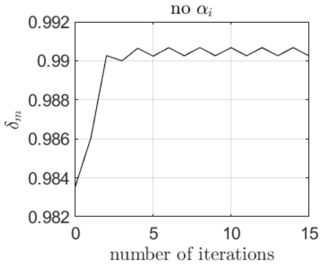
I I I Y I



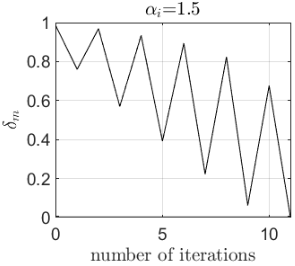
How check nodes converge (for decoding $IIIYI$ at $\epsilon = 0.003$)

We can define an observation δ_m for each check node m:

- By the previous $\boldsymbol{q}_n$, let $\hat{q}_{nm}^{(0)} = q_n^I + q_n^{S_{mn}}$ and $\hat{q}_{nm}^{(1)} = 1 - \hat{q}_{nm}^{(0)}$.
- Define $\delta_m \triangleq \prod_{n \in \mathcal{N}(m)} (\hat{q}_{nm}^{(0)} - \hat{q}_{nm}^{(1)})$ and plot its trajectory:
  ($IIIYI$ causes all $z_m = 1$ and the same trajectory δ_m $\forall$ m with a target $\delta_m < 0$)



(a) without α_i (note: the *swing is very tiny*).



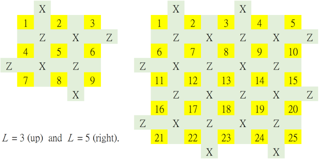
(b) with $\alpha_i = 1.5$.

Recall that α_i **introduces memory-effect**:

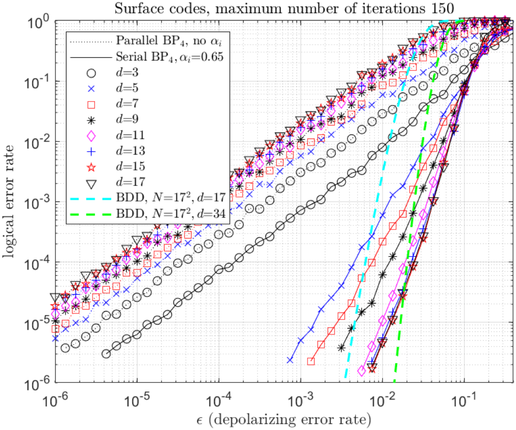
- This is like a simplified **long short-term memory** (LSTM) method,
- or can be understood as: BP is able to utilize the **momentum**.

Surface Codes, $[[N = L^2, K = 1, d = L]]$

- The code structure has strong symmetry: using a serial update schedule helps (known from Hopfield nets).
- The decoder output is usually trapped near origin: we need an $\alpha_i < 1$.
- We plot the decoding logical error rate (the lower the better):



L = 3 (up) and L = 5 (right).



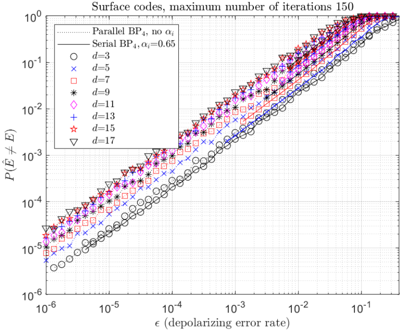
The improvement is from exploiting the degeneracy

- Write the logical error rate as:

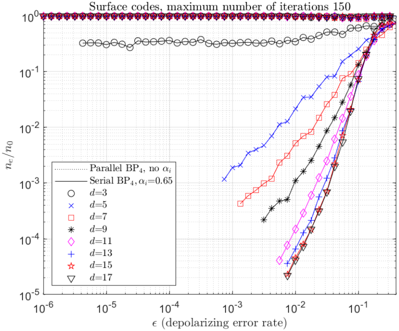
$$P(\hat{E} \notin E\mathcal{S}) = P(\hat{E} \notin E\mathcal{S}, \hat{E} \neq E)$$
$$= P(\hat{E} \neq E) \times P(\hat{E} \notin E\mathcal{S} \mid \hat{E} \neq E) = \frac{n_0}{n} \times \frac{n_e}{n_0}.$$

- We plot $\frac{n_0}{n}$ and $\frac{n_e}{n_0}$ (both the lower the better, and the lower $\frac{n_e}{n_0}$ means the more the decoder exploits the degeneracy):

Two schemes have similar $P(\hat{E} \neq E) = \frac{n_0}{n}$.

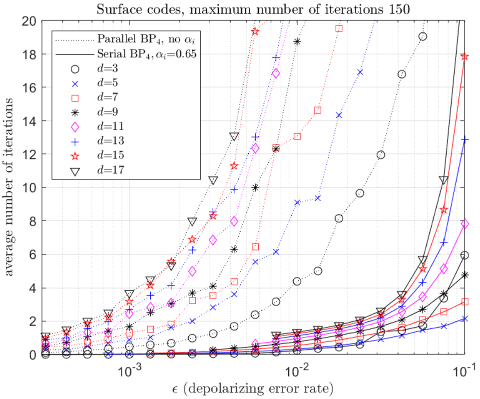


The proposed scheme has a much lower $\frac{n_e}{n_0}$.



The convergence behavior

- How well the algorithm converges can be evaluated by the **average number of iterations** (the lower the better (smaller complexity)):



- The improvement is achieved by better algorithm convergence, rather than paying complexity at doing more iterations.

Outline

Conclusion and Future Work

Conclusion:

- We refine the BP_4 decoding of quantum codes to have a lower complexity.
- We modify the refined BP_4 to have adjustable step-size and inhibition strength, controlled by one parameter α_i.
- We simulate the decoding of the $[[5, 1, 3]]$ code and surface codes.
- The proposed BP scheme exploits the degeneracy and significantly improves the performance.
- The improvement comes from better algorithm convergence, resulting in very low average numbers of iterations.

Future work:

- To further improve for the $d \geq 15$ surface codes.

Main References

R. G. Gallager, *Low-Density Parity-Check Codes*, Cambridge, MA: MIT Press, 1963.

D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, pp. 399–431, 1999.

R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. 27, pp. 533–547, 1981.

J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, pp. 498–519, 2001.

A. Y. Kitaev, "Fault-tolerant quantum computation by anyons," *Ann. Phys.*, vol. 303, pp. 2–30, 2003.

D. Poulin and Y. Chung, "On the iterative decoding of sparse quantum codes," *Quantum Inf. Comput.*, vol. 8, pp. 987–1000, 2008.

K.-Y. Kuo and C.-Y. Lai, "Refined belief propagation decoding of sparse-graph quantum codes," accepted by *IEEE J. Sel. Areas Inf. Theory*, 2020. (DOI:10.1109/JSAIT.2020.3011758)

J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci.*, vol. 81, pp. 3088–3092, 1984.

S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1139–1147.

N. N. Schraudolph, "A fast, compact approximation of the exponential function," *Neural Comput.*, vol. 11, no. 4, pp. 853–862, 1999.