MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Reinforcement-Learning-Guided Multi-Branch Decoding of Quantum LDPC Codes

Nourozi, Vahid; Koike-Akino, Toshiaki; Mitchell, David

TR2026-130 September 17, 2026

Abstract

Belief propagation (BP) is attractive for quantum lowdensity parity-check (QLDPC) codes, yet short cycles, degeneracy, and trapping configurations can cause oscillation, nonconvergence, or convergence to an incorrect logical class. We propose RL-MBOSD, a reinforcement-learning-guided multi-branch decoder for QLDPC codes. For each code matrix, a separately trained 28- parameter linear action-value policy is shared across its variable nodes and ranks sequential message updates using degreenormalized local and global features. Deterministic score perturbations generate B complementary trajectories, each executed for at most T outer sweeps. Bounded component-wise OSD repairs selected stalled branches; the resulting X- and Z-component lists are paired and ranked using the joint negative log-likelihood under the Pauli channel. Syndrome-valid candidates are then grouped by logical equivalence class and selected using an aggregate logicalclass score. On the [[144, 12, 12]] bivariate-bicycle code and the A5 code, the proposed decoder gives lower error-rate point estimates than the re-plotted prior baselines.

IEEE International Conference on Quantum Computing and Engineering (QCE) 2026

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Reinforcement-Learning-Guided Multi-Branch Decoding of Quantum LDPC Codes

Vahid Nourozi[†,*], Toshiaki Koike-Akino[*], and David G. M. Mitchell[†]

[†]Klipsch School of Electrical and Computer Engineering, New Mexico State University
Las Cruces, NM 88003, USA, Email: {nourozi, dgmm}@nmsu.edu.
[*]Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139, USA
Email: {nourozi, koike}@merl.com.

***Abstract*—Belief propagation (BP) is attractive for quantum low-density parity-check (QLDPC) codes, yet short cycles, degeneracy, and trapping configurations can cause oscillation, nonconvergence, or convergence to an incorrect logical class. We propose RL-MB-OSD, a reinforcement-learning-guided multi-branch decoder for QLDPC codes. For each code matrix, a separately trained 28-parameter linear action-value policy is shared across its variable nodes and ranks sequential message updates using degree-normalized local and global features. Deterministic score perturbations generate B complementary trajectories, each executed for at most T outer sweeps. Bounded component-wise OSD repairs selected stalled branches; the resulting X- and Z-component lists are paired and ranked using the joint negative log-likelihood under the Pauli channel. Syndrome-valid candidates are then grouped by logical equivalence class and selected using an aggregate logical-class score. On the $[[144, 12, 12]]$ bivariate-bicycle code and the A5 code, the proposed decoder gives lower error-rate point estimates than the re-plotted prior baselines.**

***Index Terms*—quantum error correction, quantum LDPC codes, reinforcement learning, belief propagation, ordered-statistics decoding, multi-branch decoding.**

I. Introduction

Quantum low-density parity-check (QLDPC) codes combine sparse stabilizer measurements with the ability to encode multiple logical qubits, making them promising for fault-tolerant quantum systems [1], [2]. Their sparse Tanner graphs also invite belief propagation (BP) decoding, but quantum degeneracy and abundant short cycles violate the independence assumptions behind BP, causing oscillating and trapping [3], [4]. Post-processing methods such as BP-OSD [5]–[8] improve convergence through reliability-ordered linear solvers, while guided decimation injects asymmetry by fixing selected variables.

Reinforcement learning-based quaternary serial variable node scheduling (RL-QSVNS) [9], [10] can learn a state-dependent sequential update order for BP. Its results demonstrate that scheduling is a powerful control variable, but one learned trajectory can still be sensitive to an early near-tie between variable-node scores. We introduce an RL-based multi-branch decoder (RL-MB-OSD), which exploits this sensitivity rather than suppressing it. The learned policy defines a preferred trajectory, and a small family of deterministic perturbations explores nearby schedules. The resulting candidates are repaired locally when necessary and judged collectively at the logical-class level.

The contributions are as follows. First, for each code matrix, one separately trained multi-parameter linear policy is shared across all variable nodes through degree-normalized local, global, reliability, graph, and search-context features. Here, *shared* means shared among the variable nodes of the same code; the same parameter vector is not assumed to transfer between different code matrices. Second, deterministic perturbations of the learned priority scores generate B reproducible sequential BP trajectories, each executed for at most T outer sweeps. Third, selected stalled trajectories are repaired by bounded component-wise OSD with explicit component-list and Pauli-pair limits. Fourth, syndrome-valid candidates are grouped by logical equivalence class and selected using an aggregate logical-class score that combines the negative log-likelihood and candidate multiplicity. Results are reported for the $[[144, 12, 12]]$ bivariate-bicycle (BB) code and the $[[180, 10, 15 \leq d \leq 18]]$ A5 code. We show the significant benefit of RL-MB-OSD over QBP, QBPGD, and RL-QSVNS decoding.

II. Decoder Formulation

A. CSS model and objective

Let $H_X \in \mathbb{F}_2^{m_X \times n}$ and $H_Z \in \mathbb{F}_2^{m_Z \times n}$ satisfy $H_X H_Z^{\mathsf{T}} = 0$. A Pauli error is represented by $e = (e_X, e_Z) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$, with $Y = (1, 1)$, and produces

$$s_X = H_Z e_X^{\mathsf{T}}, \qquad s_Z = H_X e_Z^{\mathsf{T}}. \tag{1}$$

For the depolarizing channel, $P(I) = 1 - p$ and $P(X) = P(Y) = P(Z) = p/3$. For a Pauli correction c, define its negative log-likelihood

$$C_p(c) = -\sum_{v=1}^{n} \log P(E_v = c_v). \tag{2}$$

A decoder must return an $\hat{e}$ consistent with (1). In simulation, a frame is successful only if the residual $e+\hat{e}$ also has zero logical error; both nonconvergence and a nontrivial logical residual contribute to frame error rates (FER).

B. Shared action-value policy

At sweep t of branch b, selecting an eligible variable node v is the scheduling action. The corresponding decoder state is represented by $\phi(v, t, b) \in \mathbb{R}^{28}$. The vector $\theta \in \mathbb{R}^{28}$ contains the learned feature weights and is shared by all variable nodes

TABLE I
EXACT DEFINITION OF THE SHARED-POLICY FEATURES.

Index	Feature entries
1	Constant bias 1.
2–5	$u_{H_X}(v)$, $u_{H_Z}(v)$, $\bar{u}_{H_X}(v)$, and $\bar{u}_{H_Z}(v)$.
6–8	$\frac{1}{2}(u_{H_X}+u_{H_Z})$, $u_{H_X}u_{H_Z}$, and $\left[\frac{1}{2}(u_{H_X}+u_{H_Z})\right]^2$.
9–12	$\|\delta_Z\|_1/m_X$, $\|\delta_X\|_1/m_Z$, their mean, and their absolute difference.
13–15	Sweep fraction $(t-1)/T$, channel feature $\min(1, p/p_{\mathrm{ref}})$, where $p_{\mathrm{ref}} = 0.15$, and quantum rate k/n.
16–18	Current hard components $\hat{e}_{X,v}$, $\hat{e}_{Z,v}$, and $\hat{e}_{X,v}\hat{e}_{Z,v}$.
19–21	$1-\min(1, \lvert\ell_Z(v)\rvert/L_{\max})$, $1-\min(1, \lvert\ell_X(v)\rvert/L_{\max})$, and their product.
22–23	Indicator that the previous visit changed either hard component, and $\min(1, N_{\mathrm{flip}}(v)/4)$.
24–25	$\min\{1, \log(1+d_{X/Z}(v))/\log(1+d_{\mathrm{ref}})\}$, where $d_{\mathrm{ref}} = 32$
26–27	Mean neighboring check weight divided by the maximum check weight in the corresponding component graph.
28	Branch level $b/(B-1)$ for $B > 1$, and zero for $B = 1$.

of one code. Their inner product gives the scalar scheduling priority

$$Q_\theta(v,t,b) = \theta^{\mathsf{T}}\phi(v,t,b). \tag{3}$$

The highest-scoring unvisited variable is updated next. The BB and A5 codes use separately trained vectors θ, and no feature contains an absolute variable-node index. Define the local mismatch fractions

$$u_{H_X}(v) = \frac{1}{|N_{H_X}(v)|}\sum_{a\in N_{H_X}(v)}\delta_Z(a), \tag{4}$$

$$u_{H_Z}(v) = \frac{1}{|N_{H_Z}(v)|}\sum_{a\in N_{H_Z}(v)}\delta_X(a), \tag{5}$$

with zero used when the corresponding neighborhood is empty. Their inverse-check-degree-weighted counterparts are denoted by $\bar{u}_{H_X}(v)$ and $\bar{u}_{H_Z}(v)$. The exact 28-dimensional feature vector is summarized in Table I. All bounded features are clipped to $[0,1]$.

The variable-degree and channel features use the normalized forms

$$\phi_d(v) = \min\left\{1, \frac{\log(1+d(v))}{\log(1+d_{\mathrm{ref}})}\right\}, \quad \phi_p = \min\left\{1, \frac{p}{p_{\mathrm{ref}}}\right\},$$

here, $d_{\mathrm{ref}} = 32$ and $p_{\mathrm{ref}} = 0.15$ are fixed reference values used to normalize the variable-node degree and physical error rate, respectively.

a) Offline policy training: The policy is learned by semi-gradient Q-learning [11]. With $W_t = \|\delta_X^{(t)}\|_1 + \|\delta_Z^{(t)}\|_1$, the reward favors $W_t - W_{t+1}$, penalizes unnecessary or oscillatory updates, and assigns terminal values for logical success, wrong-logical-class convergence, and nonconvergence. For the selected variable v_t,

$$\theta \leftarrow \theta + \alpha\big[r_t + \gamma\max_u Q_\theta(u, t+1, b) - Q_\theta(v_t,t,b)\big]\phi(v_t,t,b), \tag{6}$$

with zero bootstrap value at termination. The BB and A5 policies are trained separately using the same feature definition; θ is fixed during inference.

C. Multi-branch sequential search

Branch zero follows the learned priority scores. Branch b uses the perturbed score

$$\widetilde{Q}_\theta(v,t,b) = Q_\theta(v,t,b) + \eta\frac{b}{B-1}\xi(s,b,t,v), \qquad B > 1, \tag{7}$$

Algorithm 1 One RL-guided sequential branch

Input: Syndrome s, channel p, policy θ, branch b, branch count B, sweep budget T
Output: Hard correction, final beliefs, convergence flag

```
Initialize coupled messages, hard Pauli decisions, mismatch vectors, and flip counters
for t = 1, ..., T do
    Compute φ(v, t, b) for all variables
    Build a max-heap using Q̃_θ(v, t, b)
    while the heap is nonempty do
        if both residual mismatch vectors are zero then
            break
        end if
        Remove the highest-scoring unvisited variable v
        Update all coupled messages incident to v
        Update the hard Pauli decision at v
        if the hard Pauli decision changed then
            Toggle the adjacent mismatch bits
            Refresh the keys of affected unvisited variables
        end if
    end while
    if both CSS syndrome equations are satisfied then
        break
    end if
end for
Refresh the final beliefs
return branch correction, beliefs, and convergence flag
```

where $\xi(s,b,t,v) \in [-1,1]$ is a deterministic hash of the syndrome, branch, sweep, and variable. For $B = 1$, the perturbation is zero. Thus, branch zero follows the learned order exactly, while later branches progressively reorder near-tied actions without introducing run-to-run randomness.

D. Sweep budget, bounded OSD repair, and class decision

Each branch is initialized once and executed for at most T outer sequential sweeps, subject to early syndrome stopping. Increasing T extends the maximum trajectory length. It does not rerun a ladder of smaller sweep budgets and does not automatically retain candidates from independent smaller-T runs.

A stalled branch supplies reliabilities to a low-order OSD step [5], [6], [12]. For a stalled branch with hard decision $\hat{e} = (\hat{e}_X, \hat{e}_Z)$, the X-component repair vector d_X satisfies

$$H_Z d_X^T = s_X + H_Z\hat{e}_X^T, \tag{8}$$

and the repaired component is $c_X = \hat{e}_X + d_X$. Similarly,

$$H_X d_Z^T = s_Z + H_X\hat{e}_Z^T, \tag{9}$$

and $c_Z = \hat{e}_Z + d_Z$.

For each component, the columns are reliability ordered and Gaussian elimination identifies pivot and nonpivot columns. If q_{np} is the number of nonpivot columns, the decoder sets

$$o' = \min(o, q_{\mathrm{np}}) \tag{10}$$

and enumerates all $2^{o'}$ assignments on the first o' ordered nonpivot positions. All remaining nonpivot repair variables are set to zero, and the pivot variables are obtained by back substitution. At most K_c candidates are retained for each component.

Algorithm 2 RL-MB candidate generation and selection

Input: $H_X, H_Z, s, p, \theta, T, B, o, K_c, K_p, \tau, \beta$
Output: Correction $\hat{e}$ or declared nonconvergence

Initialize the raw candidate multiset $\mathcal{C} \leftarrow \varnothing$
for $b = 0, \ldots, B-1$ **do**
  Run Algorithm 1 with sweep budget T
  if the branch correction satisfies both syndrome equations **then**
    Add the correction to $\mathcal{C}$
  else if branch b is OSD-eligible **then**
    Generate at most K_c X-component candidates
    Generate at most K_c Z-component candidates
    Form Cartesian Pauli pairs and score them by C_p
    Add at most K_p syndrome-valid pairs to $\mathcal{C}$
  end if
end for
if the auxiliary BP-derived candidate is enabled **then**
  Run BP for at most T_{BP} iterations
  if its correction satisfies both syndrome equations **then**
    Add it to $\mathcal{C}$
  end if
end if
Remove syndrome-invalid candidates
Deduplicate exact corrections and retain multiplicities
if $\mathcal{C} = \varnothing$ **then**
  return declared nonconvergence
end if
Group candidates by logical label
return the minimum-cost representative of the highest-scoring logical class

The Cartesian X/Z pairs are then scored by the joint negative log-likelihood C_p, and at most K_p syndrome-valid pairs are retained.

Thus, the algebraic solves are componentwise, while the dependence among X, Z, and Y is retained through the coupled quaternary BP beliefs and the joint negative log-likelihood assigned to each paired candidate. A Y correction is represented by the component pair $(1,1)$. With $o = 10$, the implementation examines at most 2^{10} masks per component before list truncation; it does not enumerate the full classical order-10 OSD list.

After syndrome validation, identical corrections are merged and their multiplicities retained. Choose $c_{\mathrm{ref}} \in \mathcal{C}$. The logical label of $c_j \in \mathcal{C}$ is

$$\lambda_j = \left(L_Z(c_{X,j} + c_{X,\mathrm{ref}})^{\mathsf{T}}, L_X(c_{Z,j} + c_{Z,\mathrm{ref}})^{\mathsf{T}}\right). \tag{11}$$

Candidates with the same label belong to the same logical equivalence class. The aggregate class score is

$$S(\lambda) = \mathrm{LSE}_{j:\lambda_j=\lambda}[-C_p(c_j)/\tau + \beta \log m_j], \tag{12}$$

where m_j is multiplicity. The decoder returns the minimum-cost representative of $\arg\max_\lambda S(\lambda)$. Setting $\beta = 0$ removes multiplicity weighting, whereas minimum-cost selection omits logical-class aggregation.

Reproducibility settings. The reported policies were trained separately for each code matrix using the same 28-feature definition.

TABLE II
AVAILABLE TRAINING, INFERENCE, AND EVALUATION SETTINGS FOR THE REPORTED RL-MB-OSD RUNS.

Setting	Value	Setting	Value
Training episodes	50,000	T_{tr}	100
Training p grid	.03–.07	Training seed	12
Channel	Depolarizing	Normalized min-sum factor	0.625
LLR clipping magnitude	25	Branches	$B = 4$
Branch perturbation	$\eta = 0.35$	Sweep budget	As stated
Early syndrome stopping	Enabled	OSD search width	$o = 10$
OSD-eligible branches	1	OSD trigger	Stalled branch only
Component/pair limits	$K_c = 12, K_p = 64$	Class-score parameters	$\tau = 1, \beta = .35$
Auxiliary BP candidate	Enabled, $T_{\mathrm{BP}} = 32$	RL-S candidate/table	Not used
Target errors/max frames	$100/10^9$	Evaluation seed	12
Threads/batch size	$100/64$	Checkpoint period	10,000 frames

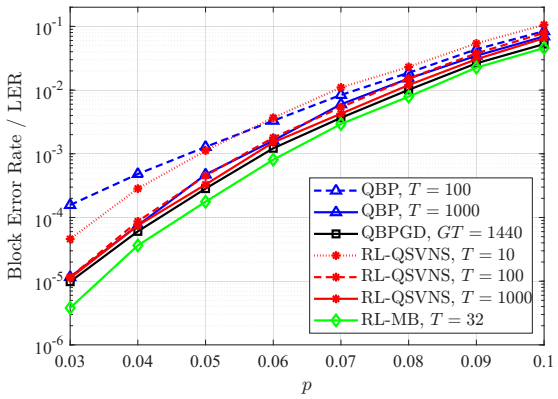

Fig. 1. FER for the $[[144, 12, 12]]$ BB code. RL-MB-OSD, $T = 32$, is from this work. All other curves are re-plotted from Fig. 4 of Moradi *et al.* [9].

The normalized min-sum factor scales the check-to-variable messages, and all LLR magnitudes are clipped at 25.

III. RESULTS

We use code-capacity depolarizing noise and two CSS codes: the $[[144, 12, 12]]$ BB code and the $[[180, 10, 15 \le d \le 18]]$ A5 code. The policies are trained independently for the two code matrices. Each RL-MB-OSD point is obtained by direct Monte Carlo simulation with a target of 100 frame errors. The reported RL-MB-OSD configuration uses $B = 4$, bounded OSD search width $o = 10$, component and Pauli-pair limits $K_c = 12$ and $K_p = 64$, and one auxiliary syndrome-validated BP-derived candidate with $T_{\mathrm{BP}} = 32$.

The QBP, QBPGD, and RL-QSVNS curves are external comparison curves from [9]. They were not evaluated on the same physical-error frames as the new RL-MB-OSD curves. Consequently, the comparisons below refer to plotted point estimates and do not constitute paired statistical tests or comparisons at equal computational cost.

Figure 1 compares independently generated FER point estimates on the $[[144, 12, 12]]$ BB code. The RL-MB-OSD point estimates with sweep budget $T = 32$ lie below the re-plotted baseline estimates over the displayed channel range. At $p = 0.03$, the RL-MB-OSD and lowest re-plotted baseline estimates are approximately 3.8×10^{-6} and 9.8×10^{-6}, respectively. At $p = 0.05$, the corresponding estimates are approximately 1.8×10^{-4} and 2.8×10^{-4}.

For A5 at $p = 0.04$, the supplied records give $F = 100$ errors in $N = 34{,}266{,}727$ frames for $T = 100$ and $N = 80{,}238{,}590$ frames for $T = 1000$. The FER estimates are 2.92×10^{-6} and

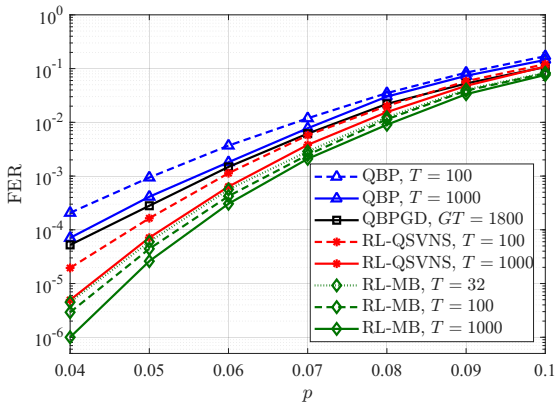

Fig. 2. FER for the $[[180, 10, 15 \leq d \leq 18]]$ A5 code. The RL-MB-OSD curves are from this work; all other curves are re-plotted from Fig. 3 of Moradi *et al.* [9].

1.25×10^{-6}, with Wilson 95% intervals $[2.40, 3.55] \times 10^{-6}$ and $[1.02, 1.52] \times 10^{-6}$. No nonconvergences or invalid converged syndromes occurred. These values show the within-decoder sweep-budget effect, not an equal-latency comparison with the external decoders.

A. Monte Carlo protocol

The decoder receives only the syndrome and channel parameter; the sampled physical error is used afterward only to test the logical residual. Each run stops after 100 frame errors or the maximum frame count. The evaluator reports F, N, $\widehat{\mathrm{FER}} = F/N$, and a Wilson 95% interval. Equivalent machine-readable records should accompany every plotted point.

IV. Complexity and Discussion

Let $F = 28$ be the policy dimension and let

$$|E| = \mathrm{nnz}(H_X) + \mathrm{nnz}(H_Z)$$

be the total number of Tanner-graph edges. At the beginning of one sweep, evaluating all variable-node feature scores costs $O(Fn)$, and building the max-heap costs $O(n)$. At most n variables are removed from the heap, giving $O(n \log n)$ priority-queue work. Under bounded variable degree and check weight, incident message updates and affected score refreshes contribute $O(|E|)$. The worst-case sequential cost of B branches with sweep budget T is therefore

$$O(BT\,[|E| + Fn + n \log n])\,. \tag{13}$$

Early syndrome stopping can substantially reduce the realized cost. The 28 learned coefficients require constant policy storage, but this does not make the complete inference cost constant because graph messages, feature evaluation, and priority-queue operations remain code dependent.

For a triggered componentwise OSD call, reliability ordering costs $O(n \log n)$, followed by Gaussian elimination and at most

$$N_{\mathrm{TEP}} = 2^{\min(o, q_{\mathrm{np}})} \tag{14}$$

test-error patterns before the component-list and Pauli-pair limits are applied. Here q_{np} is the number of nonpivot columns. This replaces the full-order expression $\sum_{i=0}^{o} \binom{q}{i}$, which is not the search implemented in the reported decoder.

The aggregate FER curves do not isolate the contribution of the learned order, branch perturbation, OSD, or logical-class aggregation. Matched-frame ablations should compare random, fixed, and residual-based schedules; $B = 1$, $\eta = 0$, OSD disabled, $\beta = 0$, and minimum-cost selection. They should also report variable-node and edge updates, tested OSD patterns, and mean/p95 latency. The available logs contain OSD invocation counts and aggregate runtime, but not the number of tested patterns or tail latency.

All reported curves use training seed 12, so policy-training variation is not measured. The BB and A5 policies are trained separately; the fixed 28-dimensional representation removes code-indexed policy storage but does not establish direct cross-code reuse. The external baselines and new curves were not generated on common error frames, so the plotted ordering is a point-estimate comparison rather than a paired test.

V. Conclusion

RL-MB-OSD combines a code-specific, variable-node-shared linear scheduling policy with deterministic branches, bounded componentwise OSD, and logical-class aggregation. On the BB and A5 codes, its FER point estimates are lower than the independently re-plotted baselines under the stated settings, but the comparison is not equal-cost or paired. Matched-compute ablations, multiple training seeds, and direct transfer tests remain important next steps.

Acknowledgement

D.G.M. Mitchell is partly supported by the National Science Foundation under Grant No. CCF-2145917.

References

[1] N. P. Breuckmann and J. N. Eberhardt, "Quantum low-density parity-check codes," *PRX Quantum*, vol. 2, p. 040101, 2021.
[2] S. Bravyi, A. W. Cross, J. M. Gambetta, D. Maslov, P. Rall, and T. J. Yoder, "High-threshold and low-overhead fault-tolerant quantum memory," *Nature*, vol. 627, pp. 778–782, 2024.
[3] D. Poulin and Y. Chung, "On the iterative decoding of sparse quantum codes," *Quantum Inf. Comput.*, vol. 8, pp. 987–1000, 2008.
[4] N. Raveendran and B. Vasić, "Trapping sets of quantum LDPC codes," *Quantum*, vol. 5, p. 562, 2021.
[5] J. Roffe, D. R. White, S. Burton, and E. Campbell, "Decoding across the quantum low-density parity-check code landscape," *Phys. Rev. Research*, vol. 2, p. 043423, 2020.
[6] T. Hillmann, L. Berent, A. O. Quintavalle, J. Eisert, R. Wille, and J. Roffe, "Localized statistics decoding for quantum low-density parity-check codes," *Nature Communications*, vol. 16, p. 8214, 2025.
[7] H. Yao, W. A. Laban, C. Hager, A. Graell i Amat, and H. D. Pfister, "Belief propagation decoding of quantum LDPC codes with guided decimation," in *Proc. IEEE ISIT*, 2024, pp. 2478–2483.
[8] M. Ye, D. Wecker, and N. Delfosse, "Beam search decoder for quantum LDPC codes," arXiv:2512.07057, 2025.
[9] M. Moradi, V. Nourozi, S. Habib, and D. G. M. Mitchell, "Learning to decode quantum LDPC codes via belief propagation," arXiv:2603.10192, 2026.
[10] M. Moradi, S. Habib, V. Nourozi, and D. G. M. Mitchell, "Sequential BP-based decoding of QLDPC codes," arXiv:2602.13420, 2026.
[11] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.
[12] M. P. C. Fossorier and S. Lin, "Soft-decision decoding of linear block codes based on ordered statistics," *IEEE Trans. Inf. Theory*, vol. 41, pp. 1379–1396, 1995.
[13] P. Panteleev and G. Kalachev, "Degenerate quantum LDPC codes with good finite length performance," *Quantum*, vol. 5, p. 585, 2021.