Adaptive Quantum Error Correction

Kollapudi Jithin   Krunal Vaghela   Yaswanth R Kumar

November 23, 2025

Let's Start with a challenge...

Coin Tossing: An Optimal Strategy Challenge

Defining the Constraint and the Goal

The Setup

We are given three independent coins, C_1, C_2, and C_3, each producing a binary outcome $\{0, 1\}$.

- **Bias:** Each coin C_i has a fixed but **unknown** probability p_i of yielding a 1.

$$P(X = 1 \mid C_i) = p_i$$

- **Budget:** We are limited to a total of $\mathbf{N} = \mathbf{100}$ tosses across all three coins.

The Question

What is the **most optimal strategy** (a sequential rule for choosing the next coin to toss) that maximizes the total expected number of **1s** observed within the 100-toss budget?

Context: The Multi-Armed Bandit (MAB) Problem

The Exploration-Exploitation Trade-off

This scenario is a classic example of a **Stochastic Multi-Armed Bandit (MAB)** problem. The key to the optimal solution lies in balancing two conflicting actions:

Exploration (Learning)

- Action: Toss all coins enough times to estimate their true probabilities ($\hat{p}_i$) accurately.
- Risk: Wasting valuable tosses on a coin that might be sub-optimal.

Exploitation (Earning)

- Action: Focus the majority of the budget on the coin currently appearing to be the best ($\hat{p}_{\max}$).
- Risk: Missing out on a truly optimal coin whose potential was underestimated early on.

What is the Optimal Strategy then?

The Goal of Optimal Strategy: Minimizing Regret

Optimal Strategy

The solution is a sequential strategy (e.g., **UCB** or **Thompson Sampling**) designed to minimize the total expected **Regret** over the 100 tosses.

Definition (Total Expected Regret (R_N))

Regret is the difference between the total reward we **would have** received if we knew the best coin and the total reward received by following our chosen strategy.
Mathematically, for $N = 100$ tosses:

$$R_{100} = \sum_{t=1}^{100} \left[\max_i (p_i) - p_{c(t)} \right]$$

where $p_{c(t)}$ is the true probability of success of the coin chosen at time t.

Goal: A successful MAB strategy seeks to minimize this accumulated loss over time.

UCB Algorithm: Optimistic Exploration

The UCB algorithm is an elegant strategy that prioritizes coins (arms) that have a high estimated reward **or** have been played too few times (high uncertainty). It systematically cycles through exploration and exploitation based on calculated confidence intervals.

The UCB Selection Rule

At each step t, the coin C_i to toss next is the one that maximizes its **UCB value** $A_i(t)$:

$$\text{Choose } C_i = \arg\max_i A_i(t)$$

where $A_i(t)$ is calculated as:

$$A_i(t) = \underbrace{\hat{p}_i}_{\text{Exploitation Term}} + \underbrace{\sqrt{\frac{2\ln t}{n_i(t)}}}_{\text{Exploration Term}}$$

$$A_i(t) = \underbrace{\hat{p}_i}_{\text{Exploitation Term}} + \underbrace{\sqrt{\frac{2\ln t}{n_i(t)}}}_{\text{Exploration Term}}$$

- $\widehat{\mathbf{p}}_{\mathbf{i}}$: The current average reward (estimated probability of getting a 1) for coin C_i. (*The best estimate so far.*)
- $\mathbf{n_i(t)}$: The number of times coin C_i has been tossed up to time t.
- $\ln \mathbf{t}$: The natural logarithm of the total number of tosses made so far.

How it Balances Trade-off

- **Exploitation:** By using $\hat{p}_i$, it favors coins with high success rates.
- **Exploration:** The exploration term decreases as $n_i(t)$ increases. Coins that haven't been played much (low $n_i(t)$) get a large boost, forcing the algorithm to be "optimistic" about their potential and try them out.

Why do all this? Why not just simple toss each coin a few times at the beginning and pick the one with most successes?

Let's see a few numbers

Table: Total Cumulative Regret for MAB Strategies

Testcase	ϵ-Greedy Regret	UCB Regret	KL-UCB Regret	Thompson Regret
1	55.10	24.73	7.32	4.78
2	410.85	263.19	75.25	51.59
3	950.40	571.33	134.06	74.61

Averaged results from simulations for 30 coins for an horizon of 10000-250000 tosses

Back to QEC...

Motivation

Classical wireless standards (Wi-Fi/5G) use **Adaptive Modulation and Coding (AMC)** to handle varying noise.

- *Low Noise:* Use high-rate schemes like **64-QAM**.
- *High Noise:* Switch to robust schemes like **QPSK** or **BPSK**.

We aim to apply this same adaptive principle to Quantum Error Correction.

Can we do this in Quantum Networks?

Motivation

Problem: Transmitting quantum information over noisy channels

- Depolarizing channel with unknown noise rate p
- Multiple QEC codes available with different costs
- **Goal:** Maximize fidelity while minimizing physical qubit overhead

Key Challenge: How do we adaptively choose the right quantum code in real-time?

Depolarizing Channel Model

The depolarizing channel applies noise to transmitted qubits:

$$\mathcal{E}_p(\rho) = (1 - p)\rho + \frac{p}{3}(X\rho X + Y\rho Y + Z\rho Z) \tag{1}$$

- p is the **physical error rate** (unknown to us)
- With probability $(1 - p)$: no error
- With probability $p/3$ each: bit-flip (X), phase-flip (Z), or both (Y)

Challenge: We must learn p online while transmitting qubits

Choosing the Quantum Error-Correcting Codes

Say we have three quantum error-correcting codes to choose from:

Code Name	**Qubits C_k**	**Trade-off**
5-qubit perfect	5	Lowest cost, less robust
7-qubit Steane	7	Medium cost/protection
9-qubit Shor	9	Highest cost, most robust

These codes were selected because they all share a crucial property, a minimum distance of $d = 3$. This distance guarantees that each code can correct **at least one arbitrary single-qubit Pauli error** (X, Y, or Z).

Key Insight: The "best" code depends on the unknown noise level p

- Low noise $\rightarrow$ cheaper codes suffice
- High noise $\rightarrow$ need expensive protection

Multi-Armed Bandit Formulation

Bandit Framework:

- **Arms:** $k \in \{5, 7, 9\}$ (the three QEC codes)
- **Time horizon:** T logical qubits to transmit
- **Action at time t:** Choose code k_t
- **Observation:** Logical error indicator $\ell_t \in \{0, 1\}$
- **Fidelity:** $f_t = 1 - \ell_t$ (success $= 1$, failure $= 0$)

Expected fidelity for code k under channel p:

$$F_k(p) = 1 - L_k(p) \tag{2}$$

where $L_k(p) = \Pr(\text{logical error} \mid \text{code } k,\ \text{noise } p)$

What is a reward in this setting?

What is a reward in this setting?

Less number of qubits?

What is a reward in this setting?

Less number of qubits?
High Fidelity?

Reward Definition: Fidelity vs Cost

We define a scalar reward trading off fidelity against qubit cost:

$$\boxed{r_t = f_t - \lambda C_{k_t}} \tag{3}$$

where:

- f_t: observed fidelity (0 or 1)
- C_{k_t}: physical qubits used
- $\lambda > 0$: cost weight (how many fidelity points = one qubit)

Expected reward for code k under noise p:

$$\mu_k(p) = \mathbb{E}[r_t \mid k_t = k] = F_k(p) - \lambda C_k \tag{4}$$

Higher $\lambda \rightarrow$ prefer cheaper codes | Lower $\lambda \rightarrow$ prefer robust codes

Oracle and Regret Definition

Best fixed code in hindsight (knows true p):

$$k^*(p) = \arg \max_{k \in \{5,7,9\}} \left(F_k(p) - \lambda C_k \right) \tag{5}$$

Regret: Cumulative loss compared to oracle

$$\boxed{R_T = \sum_{t=1}^{T} \mu_{k^*(p)}(p) - \sum_{t=1}^{T} r_t} \tag{6}$$

Equivalently:

$$R_T = \sum_{t=1}^{T} \left[\mu_{k^*(p)}(p) - \mu_{k_t}(p) \right] \tag{7}$$

Goal: Minimize R_T (learn the best code quickly)

Regret Decomposition

We can break regret into two interpretable components:

Fidelity Regret:

$$R_T^{(F)} = \sum_{t=1}^{T} F_{k^*(p)}(p) - \sum_{t=1}^{T} f_t \tag{8}$$

Cost (Qubit) Regret:

$$R_T^{(Q)} = \sum_{t=1}^{T} C_{k_t} - \sum_{t=1}^{T} C_{k^*(p)} \tag{9}$$

Relationship:

$$R_T = R_T^{(F)} - \lambda \cdot R_T^{(Q)} \tag{10}$$

Estimating the Channel: $\hat{p}_t$

Model-Based Approach: Estimate p from observed logical errors

Given data up to time t: $\mathcal{D}_t = \{(k_\tau, \ell_\tau)\}_{\tau=1}^{t}$

Maximum Likelihood Estimation:

$$\hat{p}_t = \arg \max_{p \in [0, p_{\max}]} \prod_{\tau=1}^{t} L_{k_\tau}(p)^{\ell_\tau} \cdot (1 - L_{k_\tau}(p))^{1-\ell_\tau} \tag{11}$$

Or equivalently, maximize log-likelihood:

$$\hat{p}_t = \arg\max_{p} \sum_{\tau=1}^{t} \left[\ell_\tau \log L_{k_\tau}(p) + (1 - \ell_\tau) \log(1 - L_{k_\tau}(p)) \right] \tag{12}$$

Alternative: Least-squares fit to observed fidelities

$$\hat{p}_t = \arg\min_{p} \sum_{\tau=1}^{t} \left(f_\tau - F_{k_\tau}(p) \right)^2 \tag{13}$$

Strategy 1: Pure Exploitation (Greedy)

Given current estimate $\hat{p}_t$, compute estimated rewards:

$$\hat{\mu}_{k,t} = F_k(\hat{p}_t) - \lambda C_k \tag{14}$$

Greedy policy: Choose the code with maximum estimated reward

$$\boxed{k_{t+1} = \arg \max_{k \in \{5,7,9\}} \hat{\mu}_{k,t}} \tag{15}$$

Pros:

- Simple to implement
- Exploits current knowledge optimally

Cons:

- No explicit exploration
- Can get stuck on suboptimal code if early estimates mislead

Strategy 2: Model-Free UCB1

Direct bandit learning without estimating p:

Track for each arm k:

- Number of pulls: $N_{k,t} = \sum_{\tau=1}^{t} \mathbb{I}\{k_\tau = k\}$
- Empirical mean reward: $\hat{\mu}_{k,t} = \frac{1}{N_{k,t}} \sum_{\tau:k_\tau=k} r_\tau$

UCB1 policy:

$$\boxed{k_{t+1} = \arg\max_k \left(\hat{\mu}_{k,t} + \sqrt{\frac{2\log t}{N_{k,t}}} \right)} \tag{16}$$

- Exploration bonus $\propto 1/\sqrt{N_{k,t}}$
- Proven $O(\log T)$ regret bound
- Doesn't require knowing $F_k(p)$ functional form

Strategies : Comparison

- **Greedy (Exploitation):** This strategy is Model-Based (uses $\hat{p}_t$) but includes zero explicit exploration. It risks long-term losses by relying entirely on the accuracy of the current best guess of the noise, potentially getting stuck on a suboptimal code.
- **Model-Free UCB1 (Guaranteed Exploration):** This strategy is Model-Free (ignores $\hat{p}_t$), learning only from observed rewards. It uses an explicit exploration bonus based on pull count, offering superior robustness and providing provable $\mathbf{O}(\log \mathbf{T})$ regret bounds.

Connection to Our Heuristic

Current approach: Window-based syndrome counting

1. Count syndrome-triggered logical errors in a recent window.
2. Form an empirical noise estimate $\tilde{p}_t$.
3. Apply fixed decision thresholds:
   - $\tilde{p}_t$ small $\Rightarrow$ use 5-qubit code,
   - $\tilde{p}_t$ moderate $\Rightarrow$ use 7-qubit code,
   - $\tilde{p}_t$ large $\Rightarrow$ use 9-qubit code.

Connection to Our Heuristic Contd.

Mathematical interpretation:
This rule is equivalent to a **greedy parametric bandit policy** with a piecewise threshold structure:

$$k_{t+1} = \begin{cases} 5, & \tilde{p}_t < p_{(5,7)}, \\ 7, & p_{(5,7)} \leq \tilde{p}_t < p_{(7,9)}, \\ 9, & \tilde{p}_t \geq p_{(7,9)}, \end{cases}$$

where $p_{(5,7)}$ and $p_{(7,9)}$ are the noise values at which the expected rewards satisfy

$$\mu_5(p_{(5,7)}) = \mu_7(p_{(5,7)}), \qquad \mu_7(p_{(7,9)}) = \mu_9(p_{(7,9)}),$$

with $\mu_k(p) = F_k(p) - \lambda C_k$.

Interpreting the Window Size W

- A window size $W = 4$ **does not** mean sending classical feedback every 4 *physical* qubits. **It means feedback is generated once every 4 logical qubits**.
- The choice of W reflects an exploration–exploitation tradeoff: smaller W gives faster adaptation and better early estimates of p, but increases classical communication overhead.
- If total communication time is part of the objective, we can refine the regret metric by incorporating both:

$$t_Q \propto C_k \quad \text{(quantum transmission time)},$$

$$t_C \propto \frac{1}{W} \quad \text{(classical feedback rate)}.$$

- To reduce overhead as the estimate of p stabilizes, we may let the window size grow with time, e.g.

$$W_t = W_0 + \alpha \ln(t+1),$$

providing rapid early exploration and slower, less costly feedback at later times.

Interpreting the Window Size W (Contd.)

Consider a time instant t where we transmit a logical state $\psi_L^{(t)}$ encoded using the code $C_k^{(t)}$. A natural question is:

Does the window size W need to be the same for all codes?

- Higher–redundancy codes (e.g., the 9-qubit Shor code) produce fewer independent syndrome events per logical transmission. Hence they provide **less information per logical qubit** about the underlying noise rate p.
- Lower–redundancy codes (e.g., the 5-qubit perfect code) accumulate statistically useful syndrome data more rapidly.
- It is therefore reasonable to use a **code-dependent window size**

$$W_k \quad \text{with} \quad W_9 < W_7 < W_5,$$

  where more redundant codes use smaller windows to compensate for their lower information rate.
- A practical choice is: $W_9 = 1,\ W_7 = 2,\ W_5 = 3$, ensuring that each code contributes approximately comparable statistical confidence in its noise estimate.

Non-Stationary Channels: Sliding Windows

Real-world complication: Physical error rate p may drift over time

- Temperature fluctuations
- Hardware degradation
- External interference

Solution: Use **sliding window** estimation

- Only use recent W logical transmissions to estimate $\hat{p}_t$
- Discards old data that may reflect outdated channel conditions
- Trade-off: smaller W tracks changes faster but has higher variance

Modified MLE:

$$\hat{p}_t = \arg\max_p \sum_{\tau=\max(1,t-W+1)}^{t} \left[\ell_\tau \log L_{k_\tau}(p) + (1-\ell_\tau)\log(1-L_{k_\tau}(p))\right] \tag{17}$$

Algorithm Summary

Algorithm 1 Adaptive QEC Code Selection

Input: Codes $\{5, 7, 9\}$, cost weight λ, horizon T, window W
Initialize: $\hat{p}_0 = 0.5$, $\mathcal{D} = \emptyset$
for $t = 1$ to T **do**
  Estimate channel: $\hat{p}_t = \text{MLE}(\mathcal{D}_{\text{recent}})$
  Compute rewards: $\hat{\mu}_k = F_k(\hat{p}_t) - \lambda C_k$ for each k
  Choose code: $k_t = \arg\max_k \hat{\mu}_k$ (or UCB variant)
  Encode logical qubit using code k_t
  Transmit through depolarizing channel
  Decode and measure: observe $\ell_t \in \{0, 1\}$
  Compute fidelity: $f_t = 1 - \ell_t$
  Compute reward: $r_t = f_t - \lambda C_{k_t}$
  Update dataset: $\mathcal{D} \leftarrow \mathcal{D} \cup \{(k_t, \ell_t)\}$
  Keep only recent window: $\mathcal{D}_{\text{recent}} \leftarrow$ last W samples
end for

Performance Metrics

What can we measure:

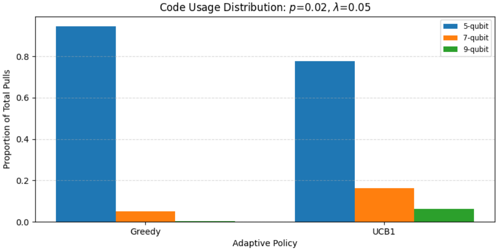
1. **Cumulative Regret:** $R_T = \sum_{t=1}^{T}(\mu_{k^*} - r_t)$
   - Should grow sublinearly: $R_T = O(\log T)$ for good policies
2. **Regret Decomposition:** $(R_T^{(Q)}, R_T^{(F)})$
3. **Convergence Rate:** Time to identify $k^*(p)$
   - How quickly does $\hat{p}_t \to p$?

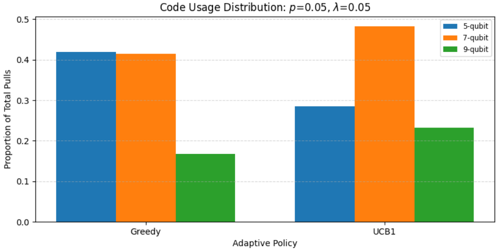
Expected Results

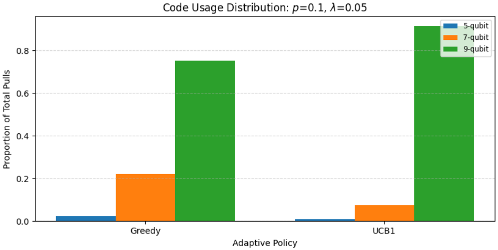
Hypothesis: Our bandit-based approach will:

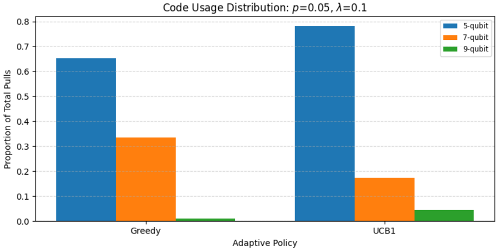
- **Outperform fixed policies** that always use one code
  - Lower regret across different noise regimes
- **Approach oracle performance** as $T \to \infty$
  - $R_T/T \to 0$ (per-step regret vanishes)
- **Adapt to changing conditions** with sliding windows
  - Track non-stationary $p(t)$ effectively
- **Balance exploration vs exploitation**
  - UCB policies explore efficiently early
  - Exploit optimal code once identified

Simulation results : Code frequencies for different channel conditions, λ







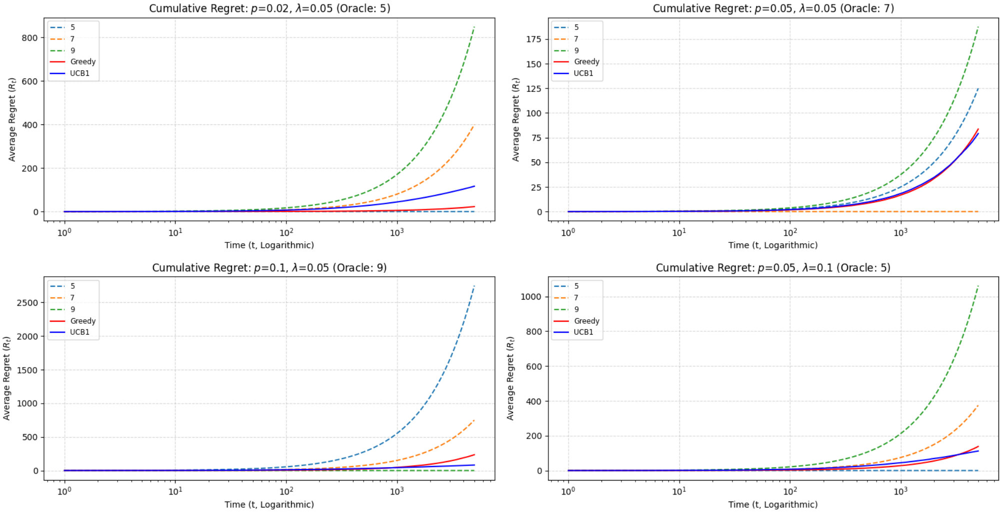


Simulation results

Table: Cumulative Regret Comparison of Fixed vs. Adaptive QEC Policies ($\mathbf{T = 5000}$ Logical Qubits)

Channel		**Optimal**	**Oracle**	**Fixed Policy Regret**			**Adaptive Policy Regret**	
P	λ	**Code**	**Reward**	**5-qubit**	**7-qubit**	**9-qubit**	**Greedy**	**UCB1**
0.02	0.05	5-qubit	0.7100	0.00	400.00	850.00	22.85	116.53
0.05	0.05	7-qubit	0.5250	125.00	0.00	187.50	83.67	79.16
0.10	0.05	9-qubit	0.3000	2750.00	750.00	0.00	234.93	81.66
0.05	0.10	5-qubit	0.2500	0.00	375.00	1062.50	138.33	112.22

Simulation results : Cumulative Regrets



Simulation results : Total Physical Qubits Transmitted

Table: Total Physical Qubits Transmitted by Fixed and Adaptive Policies (**T** = **5000** Logical Qubits)

Channel		**Optimal Code**	**Fixed Policy Qubits**			**Adaptive Policy Qubits**	
P	λ		**5-qubit**	**7-qubit**	**9-qubit**	**Greedy**	**UCB1**
0.02	0.05	5-qubit	25000	35000	45000	25568	27837
0.05	0.05	7-qubit	25000	35000	45000	32475	34481
0.10	0.05	9-qubit	25000	35000	45000	42281	44065
0.05	0.10	5-qubit	25000	35000	45000	28589	27624

Discussion: Why This Matters

Broader Impact:

- **Resource efficiency:** Use minimum qubits for desired fidelity
  - Critical for NISQ-era devices with limited qubits
- **Robustness:** Automatically adapt to hardware variations
  - No manual recalibration needed
- **Scalability:** Framework extends to more codes
  - Can incorporate surface codes, color codes, etc.
- **Theoretical guarantees:** Provable regret bounds
  - Not just heuristics—mathematically principled

Future work: Contextual bandits (incorporate syndrome patterns), multi-objective optimization

Limitations and Extensions

Current limitations:

- Assumes i.i.d. noise within windows
- Binary fidelity model (success/failure)
- Fixed code set (3 options)
- Doesn't leverage syndrome structure directly

Possible extensions:

- **Contextual bandits:** Use syndrome patterns as context
- **Continuous fidelity:** Model full distribution, not just binary
- **Correlated noise:** Model temporal correlations in $p(t)$
- **Hierarchical codes:** Include concatenated and surface codes
- **Batch decisions:** Choose codes for blocks of qubits jointly

Conclusion

Summary:

- Formulated adaptive QEC as a stochastic multi-armed bandit
- Defined reward $r_t = f_t - \lambda C_{k_t}$ trading fidelity vs cost
- Presented three strategies: Greedy, Parametric UCB, Model-Free UCB
- Connected to our heuristic: greedy policy on estimated $\hat{p}_t$
- Extended to non-stationary channels via sliding windows

Key insight:

Online learning naturally balances exploration of uncertain codes with exploitation of codes we believe are optimal

Impact: Principled framework for resource-efficient quantum communication

Thank You!

Questions?

Kollapudi Jithin
Krunal Vaghela
Yaswanth R Kumar