# RESEARCHER-IN-THE-LOOP EVOLUTIONARY DISCOVERY OF ENTANGLEMENT-ASSISTED QUANTUM CODE FAMILIES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

LLM-guided program evolution can produce mathematical constructions, but validity, generality, and reproducibility require distinct evidence. We develop a researcher-in-the-loop discovery pipeline that combines program evolution, exact evaluation, mathematical generalization, certified nonexistence checks, and formal proof checking. We apply it to entanglement-assisted quantum error-correcting codes (EAQECCs). Starting from tasks that did not specify a family formula, evolved programs produced finite qubit constructions whose structure researchers generalized to the family $[[n, 1, n-1; n-q-1]]_q$ for prime powers $q \geq 3$ at $n \geq 2q+1$ and for $q = 2$ at odd $n \geq 5$. Lean checks the algebraic certificate underlying this family. Together with entanglement lifting, the family closes 54 gaps in archived EAQECC tables. The released evidence includes 115 verified code records, ten nonexistence results with checked DRAT proofs, family implementations, and preserved search programs for exact checks and seeded replay. In a controlled cohort of eight paired generation blocks, validation-selected iterative programs achieve success in 46.1% and 31.6% of held-out executions at unseen lengths $n = 13, 15$, compared with 19.5% and 3.1% for independently generated programs. In this setting, program evolution and researcher analysis turn finite computational discoveries into a proved family with independently checkable evidence.

## 1 INTRODUCTION

LLM-guided program search has produced mathematical constructions and algorithms (Romera-Paredes et al., 2024; Novikov et al., 2025). Their outputs can also support general mathematical results through researcher analysis. This raises three questions: *whether evolved programs produce valid codes, whether their structure can be generalized and proved to work beyond the searched cases, and whether the resulting constructions and preserved searches can be independently reproduced*. Addressing these questions requires records of the initial task, computational discoveries and subsequent mathematical analysis.

We study this process for entanglement-assisted quantum error-correcting codes (EAQECCs). Early AlphaEvolve searches produced finite qubit codes whose structure led us to an infinite qubit family. Further mathematical generalization gives the prime-power construction and proof in Theorem 1. The initial instructions suggested directions such as cyclic shifts, but did not provide this family formula. Later instructions explicitly incorporated the discovered qubit family (Appendix B.3), allowing it to guide further searches.

Our reproducibility contribution concerns these mathematical constructions and their computational evidence. The anonymous artifact supplies code matrices, family implementations, proofs, task specifications and preserved search programs. Readers can recompute code parameters, construct and verify family instances, inspect the proof and replay preserved searches. Figure 1 summarizes how researcher analysis and program evolution interact. Section 3.4 specifies which operations are reproducible. A final Lean stage checks the algebraic construction underlying Theorem 1 (Appendix D); model generation remains stochastic.



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Several improvements concern constructions published in 2020 and 2022 that remained listed in the EAQECC tables accessed on 17 July 2026 (Grassl et al., 2020; Luo et al., 2022). Our constructions and certified exclusions close nine previously tabulated qubit distance gaps (Table 1). The prior constructions predate our table access by approximately four to six years. Their dates document available constructions rather than a complete history of improvement attempts.

Our contributions are:

- **A researcher-in-the-loop discovery pipeline.** We connect LLM-guided program evolution with researcher proposals, mathematical generalization and formal proof checking for EAQECC discovery. Preserved programs, code matrices and formal certificates support reconstruction and independent verification of the resulting constructions (Sections 3 and 3.4).
- **General constructions from evolved program outputs.** Finite qubit constructions led to the general family of Theorem 1, whose algebraic construction is checked in Lean. The family and entanglement lifting close 54 listed gaps, while checked nonexistence certificates establish selected optimality and minimum-entanglement results (Section 4).
- **Program development and generalization.** We compare independent and iterative generation under shared research guidance. Eight new generation blocks yield longer-code success of 46.1% and 31.6% for iterative programs, versus 19.5% and 3.1% for independent proposals at $n = 13, 15$. Preserved lineages identify useful objective and search-policy changes (Section 5).

## 2 Background and related work

### 2.1 Entanglement-assisted quantum error-correcting codes

Quantum error correction protects quantum information sent through noisy communication links. Early stabilizer constructions imposed algebraic compatibility constraints that limited which classical codes could be reused. Brun et al. (2006) showed that entanglement shared in advance between sender and receiver relaxes these constraints, enabling a broader class of quantum codes. This is the motivation for entanglement-assisted quantum error-correcting codes (EAQECCs).

A code with parameters $[[n, k, d; c]]_q$ encodes $k$ logical qudits into $n$ transmitted $q$-level systems using $c$ shared maximally entangled qudit pairs. Each pair contains $\log_2 q$ ebits, equal to one for qubits ($q = 2$) (Grassl et al., 2022). The receiver's halves are assumed noiseless and are not counted in $n$.

The distance $d$ determines the code's guaranteed error protection. With ideal decoding, it corrects arbitrary errors on up to $\lfloor (d-1)/2 \rfloor$ transmitted qudits, or up to $d-1$ *erasures*, where the lost qudits' locations are known (Brun et al., 2006; Grassl et al., 2022). Increasing $d$ at fixed $(q, n, k, c)$ strengthens protection without increasing the transmission or entanglement budget. For example, our $[[12, 1, 11; 8]]_2$ code raises the listed distance from 9 to 11, increasing the guaranteed correction of arbitrary errors from four to five transmitted qubits with the same resources (Table 1). A verified code thus provides an encoding specification for a quantum link with given resource and error-protection requirements. Reducing $c$ at fixed $(n, k, d)$ instead lowers the entanglement needed before transmission (Luo et al., 2022).

### 2.2 Code discovery as program search

We search linear stabilizer EAQECCs represented by generator matrices over $\mathbb{F}_q$. The LLM revises a search program that proposes matrices; an exact evaluator computes their $(k, c, d)$. At fixed $(q, n, k, c)$, reference tables give a constructed lower distance bound $d_\ell$ and an upper bound $d_u$ (Grassl, 2026; Luo et al., 2022). Finding $d > d_\ell$ improves the lower bound, while reaching $d_u$ closes the gap. We also construct codes at unlisted cells and minimize entanglement, denoted $c^{\mathrm{s}}_{\min}(n, k, d)$ within our stabilizer class. Applicable bounds filter impossible targets (Grassl et al., 2022); Appendix A gives the representation, notation and comparison rules.

FunSearch evolves programs that construct mathematical objects or make algorithmic decisions, using an evaluator to select useful revisions. It produced larger cap-set constructions and im-

proved online bin-packing heuristics (Romera-Paredes et al., 2024); its executable outputs let researchers inspect and develop the underlying mathematical ideas. AlphaEvolve extends this approach to matrix-multiplication algorithms, data-center scheduling, hardware-circuit simplification and model-training kernels (Novikov et al., 2025; Georgiev et al., 2025). Related frameworks include OpenEvolve, CodeEvolve, ShinkaEvolve and TurboEvolve, which implement or extend evaluator-guided program evolution with population management and proposal-selection strategies (Sharma, 2025; Assumpção et al., 2026; Lange et al., 2025; Yang et al., 2026). We use AlphaEvolve as the evolution engine for EAQECC discovery, supplying target parameters, an exact evaluator and researcher-directed construction ideas.

Quantum-code discovery also uses genetic algorithms to evolve stabilizer representations (Webster & Browne, 2024), and reinforcement learning to optimize codes and encoders (Poulsen Nautrup et al., 2019; Su et al., 2025; Mauron et al., 2023; Olle et al., 2024). More directly, Cruz-Benito et al. (2026) evolve Python generators for bivariate bicycle codes, verify candidates, and derive structural results. Our contribution concerns EAQECC distance–entanglement targets: exact finite-field evaluation supports finite constructions, a prime-power family with a Lean-checked algebraic certificate, and certified nonexistence results. The controlled study compares independent proposals with iterative program feedback under a shared researcher proposal.

## 3 The Researcher-in-the-Loop Discovery Pipeline

Our discovery pipeline combines human-in-the-loop (HITL) research with LLM-guided program evolution and exact verification. Researchers propose construction ideas, inspect verified instances and revise the task between campaigns. Figure 1 connects these interventions to the automated search and evaluation loop. Constructed codes establish lower distance bounds; SAT refutations with checked certificates lower upper bounds. Independent verification and archived evidence support both routes. Researchers generalize the discovered constructions and develop proofs, then pass the formalized statements and proofs to the Lean checking stage.

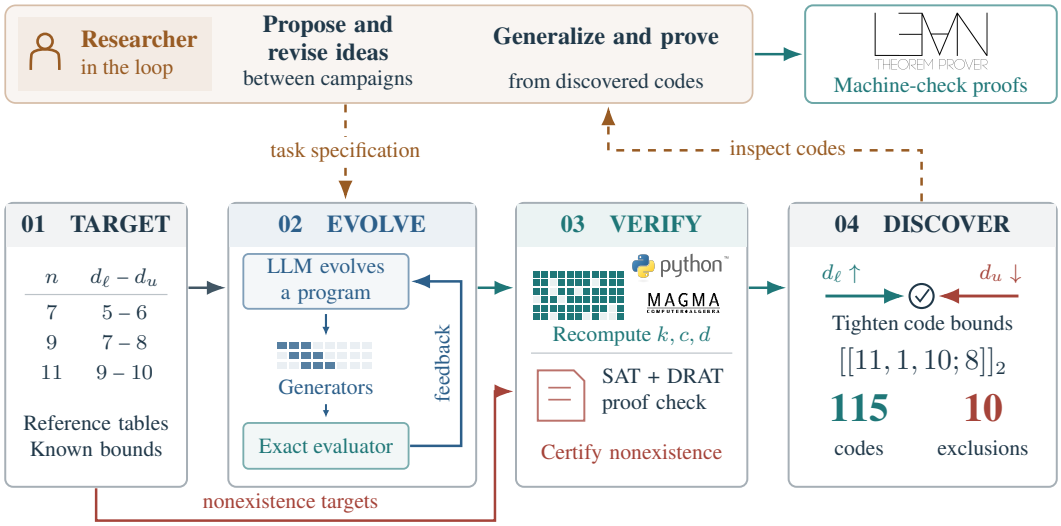



Figure 1: The researcher-in-the-loop discovery pipeline.

### 3.1 Target Selection

We compare our constructions with the bounds listed in the EAQECC tables (`codetables.de`), accessed on 17 July 2026. These reference bounds define listed-gap and absent-cell targets, filtered using the EA-Plotkin and applicable Singleton bounds. EA-Plotkin also tightens 398 upper bounds in the reference qubit table. Appendix A.3 gives the table records and filters. The artifact preserves these comparison data and uses them by default. A second copy, retrieved on 10 September 2026, contains identical intervals at all 11,640 qubit and qutrit entries.

### 3.2 LLM-GUIDED PROGRAM EVOLUTION

The qubit discovery campaigns use AlphaEvolve (Novikov et al., 2025) through Google Cloud Vertex AI. We provide a seed program, task specification and exact evaluator; AlphaEvolve generates revisions and archives their source and outcomes. An LLM modifies a designated block of the seed. Within each candidate program, an exact parameter oracle evaluates intermediate matrices; the program uses these results to choose targets, accept local moves and restart search. Algorithm 1, for example, minimizes a cost combining signature mismatch and the number of logical operators below the target distance.

After the program returns up to 16 matrices, the external evaluator recomputes their parameters and assigns the best proposal score to the program. Its reward favors a valid signature, fewer low-weight logical operators and newly achieved targets. This reward guides the next program revision; the inner cost guides matrix search within one execution. Independent code checking follows in Section 3.3.

Each candidate runs in its own subprocess with a fixed 240 s budget and a specified seed for its random-number generator. Failures receive a low score, and the campaign continues. The same interface supports fixed-program replay (Section 3.4). The four campaign logs record $401$ completed candidate searches, corresponding to $26.7$ h of allocated search budget (Appendix B.1).

The initial task specification did not contain the family proved in Theorem 1. We started with an annealing seed and instructions suggesting cyclic shifts among other construction directions. Evolved programs produced finite qubit codes; analysis of their structure led researchers to a general qubit family and its proof. The later task specifications included that family and requested further constructions, while the best program was reused as a new seed. These records distinguish the initial discovery stage from subsequent search using the mathematical result (Appendix B.3). The current prime-power theorem extends the qubit construction through further researcher-led generalization and proof.

### 3.3 INDEPENDENT VERIFICATION AND NONEXISTENCE CHECKING

The Magma export supplies only a basis of the symplectic dual $L = S^{\perp}$, where $S$ is the stabilizer space. Magma independently recomputes the dual, radical and $(k, c, d)$ (Appendix A.1). All 148 objects—115 core codes at $q = 2, 3, 4, 5$ and 33 family instances—were re-verified with zero mismatches under Magma V2.28-20. Separate $\mathbb{F}_2$ linear-algebra checks verify the two solver-refined qubit codes; all code records can also be checked in Python using the anonymous artifact. Lean checks the algebraic certificate underlying Theorem 1. A separate Comparator run checks the submitted proof against the exact formal goal and permitted axioms (Appendix D).

To test existence of an $[[n, k, d; c]]_q$ code, we encode the smaller subspace $L = S^{\perp}$ in symplectic normal form. Constraints enforce the target signature directly, so every satisfying assignment specifies an EAQECC. Distance constraints dominate the encoding.

For an unsatisfiable encoding, the SAT solver produces a DRAT proof, a certificate of unsatisfiability checked independently by DRAT-trim (Wetzler et al., 2014). Together with the encoding's completeness, this establishes the corresponding code nonexistence result. The qutrit encoding uses a coordinate-normalization lemma to preserve completeness. Appendix B.2 gives the proof scope and encoding, and Section 4.3 reports the certified results.

### 3.4 REPRODUCIBILITY

Readers can reconstruct two outputs: code constructions and preserved program executions. Supplied matrices support exact $(k, c, d)$ checks; family implementations generate instances, and Theorem 1 proves the construction for all admissible parameters. Task specifications and discovery logs document the progression from finite qubit codes to the researcher-derived family (Appendix B.1). Source replay concerns retained programs, including later programs that use this family. It replaces wall time with a virtual clock tied to evaluator calls and preserves the recorded random-number seeds and stopping conditions. Three programs with four seeds each reproduce all twelve stored proposal lists, with agreement between one and four workers. The anonymous artifact provides these inputs, independent parameter checks and reference-bound audits.

## 4 RESULTS

Algorithm 1 presents a later, high-scoring AlphaEvolve program whose seed already incorporated the discovered qubit family. It illustrates continued search after that generalization. Figure 2 and the shading show changes to target selection, reuse of stored proposals, construction choices and generator-row mutations. The earlier discoveries that motivated the theorem are recorded separately (Appendix B.1). Section 5 evaluates iterative feedback using newly generated programs, including Algorithm 2 in Appendix C.5.

**Algorithm 1** Search program evolved by AlphaEvolve.

**Require:** targets $\mathcal{T}$, time allowance $B$
$\Phi(r,t) = 5000|r.c - c_t| + 1000|r.k - k_t| + r.\text{offending}$
$best \leftarrow \{\}$; $b_t \leftarrow 0$ for all $t \in \mathcal{T}$
**while** time remains **do**
  $\pi_t \propto n_t^{-1.5}(1 + 10b_t)$; sample $t \sim \pi$ ▷ Use recorded progress to change target selection.
  $n \leftarrow n_t$; $s \leftarrow n_t - k_t + c_t$; $u \sim \text{Uniform}[0,1)$
  **if** $u < 0.2$, $t \in best$ and $best[t].o > 0$ **then**
    $\mathbf{G} \leftarrow \text{copy}(best[t].\mathbf{G})$ ▷ Resume from this target's stored best proposal.
  **else**
    $\mathbf{G} \leftarrow$ CONSTRUCT$(t,u)$ ▷ Expand construction choices.
  **end if**
  **if** $\mathbf{G}$ is undefined or $\text{rank}\,\mathbf{G} \neq s$ **then**
    $\mathbf{G} \leftarrow$ RANDOMSTABILIZER$(n,s)$
  **end if**
  $o \leftarrow \Phi($EVALUATE$(n,\mathbf{G},d_t),t)$; KEEPBEST$(best,t,\mathbf{G},o)$
  **if** $o = 0$ **then continue**
  $\tau \leftarrow 30$; $f \leftarrow 0$
  **while** $f < 1500$ **and** time remains **do**
    $\tau \leftarrow \max(0.1, 0.995\tau)$
    $\mathbf{G}' \leftarrow$ MIXEDMOVE$(\mathbf{G})$ ▷ Add cyclic rotation of a whole generator row.
    **if** $\text{rank}\,\mathbf{G}' \neq s$ **then** $f \leftarrow f+1$; **continue**
    $o' \leftarrow \Phi($EVALUATE$(n,\mathbf{G}',d_t),t)$; $\Delta \leftarrow o' - o$
    Set $(\mathbf{G},o) \leftarrow (\mathbf{G}',o')$ with probability $\min\{1, \exp(-\Delta/\tau)\}$
    $f \leftarrow 0$ on an accepted strict improvement, otherwise $f \leftarrow f+1$
    KEEPBEST$(best,t,\mathbf{G},o)$; **if** $o = 0$ **then break**
  **end while**
  $b_t \leftarrow$ PROGRESSBONUS$(best,t,b_t)$
**end while**
**return** up to 16 stored proposals with the smallest objective values

Shading: principal changes from the campaign seed; routine definitions are in Appendix B.1.

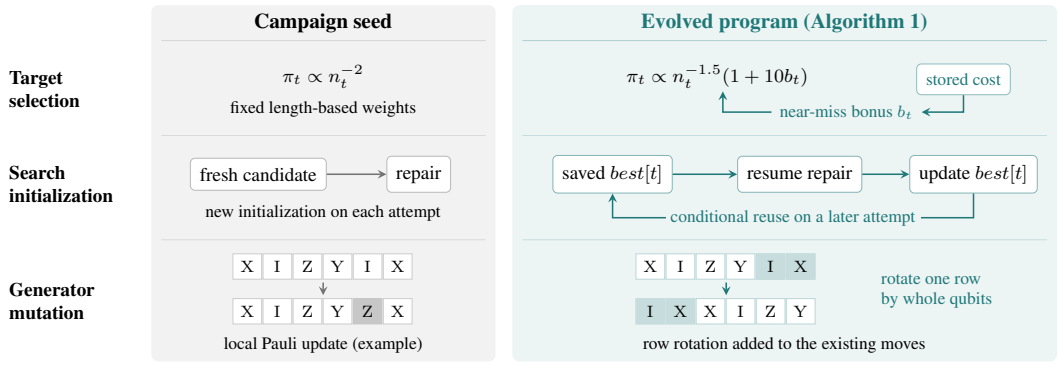



Figure 2: Program changes in Algorithm 1 (schematic).

### 4.1 DISTANCE IMPROVEMENTS

For listed cells, the tables give the July 2026 distance interval within the previous code parameters and our verified distance in Ours $d$ (2026). Bold entries attain the applicable archived, EA-Plotkin or certified upper bound (Section 4.3). References identify previous lower-bound constructions and their dates. For $q = 4, 5$, where no comparable dated entry is archived, full constructed parameters appear in a separate *New constructions* block. No historical interval is assigned where no comparison entry exists.

Table 1: Qubit EAQECCs ($q = 2$).

| Previous $[[n, k, d_\ell\text{–}d_u; c]]_q$ | Ours $d$ (2026) | References |
|---|---|---|
| $[[7, 1, 5\text{–}6; 4]]_2$ | **6** | Grassl et al. (2020) |
| $[[9, 1, 7\text{–}8; 6]]_2$ | **8** | Luo et al. (2022) |
| $[[10, 1, 7\text{–}9; 6]]_2$ | **9** | Luo et al. (2022) |
| $[[11, 1, 9\text{–}10; 8]]_2$ | **10** | Luo et al. (2022) |
| $[[12, 1, 9\text{–}11; 8]]_2$ | **11** | Luo et al. (2022) |
| $[[12, 2, 7\text{–}10; 9]]_2$ | **9** | Luo et al. (2022) |
| $[[13, 1, 9\text{–}12; 8]]_2$ | **11** | Luo et al. (2022) |
| $[[13, 1, 11\text{–}12; 10]]_2$ | **12**† | Luo et al. (2022) |
| $[[15, 1, 13\text{–}14; 12]]_2$ | **14**† | Luo et al. (2022) |

† Constructed by the evolved program at unseen code lengths.

Table 2: Qudit EAQECCs ($q = 3, 4, 5$).

| Previous $[[n, k, d_\ell\text{–}d_u; c]]_q$ | Ours $d$ (2026) | References |
|---|---|---|
| $[[9, 1, 7\text{–}8; 4]]_3$ | **8** | Luo et al. (2022) |
| $[[10, 1, 8\text{–}9; 5]]_3$ | **9** | Luo et al. (2022) |
| $[[11, 1, 9\text{–}10; 7]]_3$ | **10** | Luo et al. (2022) |
| $[[13, 1, 10\text{–}12; 8]]_3$ | **12** | Luo et al. (2022) |
| *New construction* | Ours $d$ | References |
| $[[10, 1, 9; 4]]_4$ | 9 | Ours |
| $[[12, 1, 11; 6]]_4$ | 11 | Ours |
| $[[14, 1, 13; 8]]_4$ | 13 | Ours |
| $[[12, 1, 11; 5]]_5$ | 11 | Ours |
| $[[14, 1, 13; 7]]_5$ | 13 | Ours |

The five Table 1 rows with $k = 1$ and $c = n - 3$ are covered by Theorem 1; the other four retained generator matrices were constructed by normal-form SAT. The daggered $n = 13, 15$ rows also have constructions from the evolved program at lengths withheld from training and selection (Section 5.2). These reconstructions are covered by the family theorem and are not counted again among its 54 gap closures.

Our codes and the upper bounds close all nine listed qubit gaps in Table 1. Four cells improve the previous lower bound by two: $(n, k, c) = (10, 1, 6)$, $(12, 1, 8)$, $(12, 2, 9)$ and $(13, 1, 8)$. For the last two cells, the EA-Plotkin bound and a checked DRAT refutation, respectively, give final distance intervals $[9, 9]$ and $[11, 11]$ (Section 4.3). Table 2 shows four qutrit gap closures, including $[[13, 1, 12; 8]]_3$, which raises the listed distance from 10 to 12 at fixed length and entanglement.

The previous $[[7, 1, 5; 4]]_2$ construction follows from the pure $[[11, 1, 5]]_2$ code of Calderbank et al. (1997) and the length-reduction rule of Grassl et al. (2020); we date this EAQECC construction by the 2020 rule. The other eight qubit baselines and the four displayed qutrit baselines follow from the 2022 version of Luo et al. (2022). Appendix A.4 records the source codes and propagation rules for each cell.

The 65 core qubit code records cover 61 distinct $(n, k, c)$ cells: eight listed and 53 absent cells. The 27 qutrit codes comprise 15 improvements to listed entries, 11 constructions at absent cells and one code whose distance is already listed. Multiple codes at one cell are not counted as separate table entries. Four of these listed core cells overlap the family results in Section 4.2. The additional $n = 15$ code record is kept in the controlled-study archive, separately from these core-file counts.

### 4.2 CODE FAMILIES AND REDUCED ENTANGLEMENT

Theorem 1 is the mathematical generalization of the construction identified through the evolved qubit codes. It gives $[[n, 1, n - 1; n - q - 1]]_q$ for $q \geq 3$, $n \geq 2q + 1$, and for $q = 2$ at odd $n \geq 5$. The proof in Appendix A.2 establishes the entire family; the implementations verify finite instances at $q = 2, 3, 4, 5$. Entanglement lifting increases $c$ by one at fixed $(n, k)$ without reducing distance when $c < n - k$ (Lemma 1). The qubit family closes 29 listed gaps. For $q = 3$, lifting from $c = n - 4$ to the table's $c = n - 3$ closes 25 more, for 54 in total. This family addresses finite-length

distance–entanglement tradeoffs: at fixed $q$, its rate $k/n = 1/n$ vanishes and $c/n$ tends to one. The broader archive includes 29 binary codes with $k > 1$, up to $k = 6$.

Additional constructions attain $d = n - 1$ at $c = n - q - 2$. The displayed $q = 4, 5$ examples include $[[10, 1, 9; 4]]_4$ and $[[12, 1, 11; 5]]_5$, requiring four and five shared pairs, respectively. These constructions establish achievable entanglement costs; smaller costs at $q = 4, 5$ have not been excluded. Complementary finite-field annealing searches produce 13 archived codes at $q = 4$ and 10 at $q = 5$ (Appendix B.1). Table 2 selects those using the least entanglement reached at the displayed lengths.

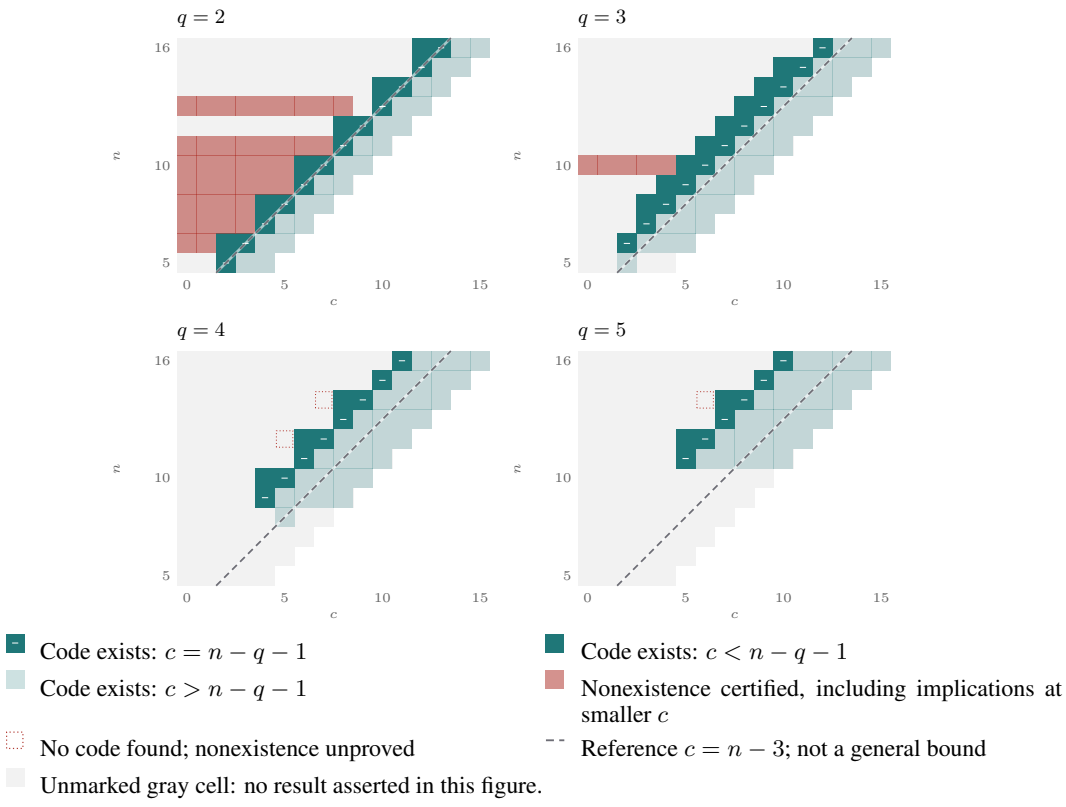



Figure 3: Linear stabilizer EAQECCs at $k = 1$, target distance $n - 1$.

Figure 3 concerns linear stabilizer EAQECCs with $k = 1$ and target distance $n - 1$. Teal at $(q, n, c)$ denotes a construction with distance at least $n - 1$. The white tick, solid fill and pale fill distinguish $c = n - q - 1$, smaller $c$ and larger $c$, respectively; these are positions relative to the family line, not claims of minimum entanglement. Pale cells correspond to explicit constructions or entanglement lifting. Red cells are excluded by released, checked certificates or by their implications at smaller $c$ through entanglement lifting. A dotted outline records an unsuccessful search, not nonexistence. Unmarked gray cells assert neither construction nor exclusion. The dashed $c = n - 3$ line is obtained by substituting $k = 1$ and $d = n - 1$ into the historical Brun–Devetak–Hsieh expression $2(d - 1) \leq n - k + c$ at equality. It is a comparison line, not a general bound in this regime (Appendix A.3).

### 4.3 Upper bounds and optimality

Solver refinement settles the two remaining qubit gaps from Table 1. Normal-form SAT finds $[[12, 2, 9; 9]]_2$, improving upon an intermediate $d = 8$ construction from our earlier search. EA-Plotkin gives $d \leq \lfloor 144/15 \rfloor = 9$, so the new code is distance-optimal. At $(n, k, c) = (13, 1, 8)$, a DRAT-certified refutation excludes $d \geq 12$; our existing $d = 11$ code is therefore optimal. Both codes are checked by separate binary linear-algebra code, and DRAT-trim replays the nonexistence certificate. We attribute these two closures to solver refinement.

Matching nonexistence results also establish minimum entanglement at selected lengths. For qubits at $k = 1$, $d = n - 1$, the minimum is $n - 3$ at $n = 7, 9, 11$ and $n - 4$ at $n = 6, 8, 10$. This finite-length parity pattern combines distinct constructions with certified exclusions; Theorem 1's binary branch covers odd lengths. For qutrits, the certified exclusion of $[[10, 1, 9; 4]]_3$ and a code at $c = 5$ give $c^{\mathrm{s}}_{\min}(10, 1, 9) = 5$. Two further exclusions, $[[5, 2, 4; 3]]_2$ and $[[6, 2, 5; 4]]_2$, were also checked by exhaustive enumeration. All ten exclusions have checked DRAT proofs (Appendix B.2).

Applying the EA-Plotkin bound of Guo & Li (2013) additionally lowers 398 listed qubit upper bounds. The auditor recomputes these corrections against the reference table data. These corrections follow from a known inequality and are reported as a separate source of upper bounds.

## 5 Ablation Study

We test whether iterative feedback helps develop a researcher-proposed change into an effective search program. Both policies receive the same initial code and research guidance; their difference is how they reuse previous programs and execution feedback.

### 5.1 Program generation

The shared proposal is to maintain and mutate a basis of $L = S^{\perp}$, converting it to $S$ for exact evaluation. For $[[9, 1, 8; 6]]_2$ and $[[11, 1, 10; 8]]_2$, this replaces a basis of 14 or 18 generators with four vectors. Symplectic complementation preserves the feasible code set but changes the variables and local mutations. A preliminary calibration compares the initial $S$-side program with a working $L$-side control and fixes a budget at which the change can produce measurable success. Neither the control's source nor generator matrices are supplied to the model.

*Independent proposals* always receive the initial program and its fixed training measurements. *Iterative evolution* receives the best-so-far program and the most recent attempt, including rejected attempts, with training feedback for both. The model can revise the representation, objective, mutation operators and restart policy. A proposal is a whole program revision; an evaluator call checks one candidate matrix produced by that program. Both use Gemini 3.5 Flash and identical tool access. There is no controller initialization: every success must be produced by the submitted program.

An initial four-block study was followed by a prospective eight-block replication using new generation and execution seeds. Each policy is allocated seven proposals per block. The replication receives 111 of 112 planned responses; one iterative pool has six proposals. All eight blocks are retained under a rule fixed before validation and testing. Training uses two seeds and 10,000 exact evaluations per target. Four new validation seeds select among the three training-best distinct programs; 32 further seeds evaluate each selection. Only training feedback enters generation. The original and new cohorts are reported separately (Appendix C).

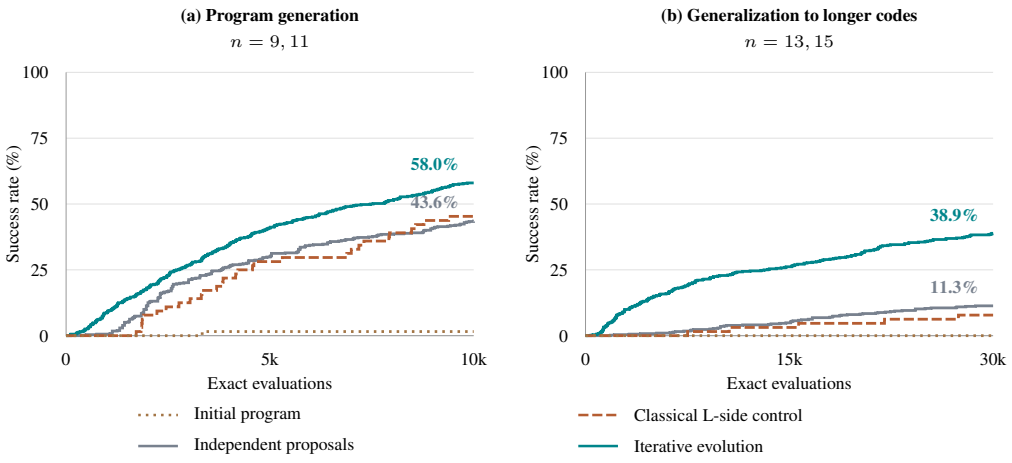



Figure 4: Prospective replication under shared research guidance.

In the new cohort, success at $n = 9, 11$ is 72.7% and 43.4% for iterative evolution, versus 60.2% and 27.0% for independent proposals. The prespecified primary endpoint is the paired block difference in mean normalized success-curve area: 0.359 versus 0.260, a gain of 0.099. Five of eight block differences are positive; a descriptive two-sided sign-flip test gives $p = 0.328$. Generation blocks, rather than execution seeds, are the units for policy inference. Restricting to the seven fully received paired blocks gives a gain of 0.150.

Figure 4 aggregates both lengths within each panel. Each policy contributes eight selected programs and 32 runs per length; each fixed control contributes 32 runs per length. Across the two generation lengths, iterative selections solve 297/512 executions, versus 223/512 for independent selections. Every saved successful code passes an independent parameter check. Appendix C.3 reports the earlier cohort and its fresh-seed repeat separately; the artifact also preserves model-use accounting for both policies.

### 5.2 Generalization to longer codes

The selected programs are applied without further modification to $[[13, 1, 12; 10]]_2$ and $[[15, 1, 14; 12]]_2$ at 30,000 calls. These lengths are withheld from generation and selection; their outcomes are secondary endpoints. In the new eight-block cohort, iterative success is 118/256 (46.1%) and 81/256 (31.6%), versus 50/256 (19.5%) and 8/256 (3.1%) for independent proposals. Two iterative selections achieve 32/32 at $n = 13$ and at least 28/32 at $n = 15$ (Appendix C.4). Their mean normalized curve areas are 0.309 and 0.193, compared with 0.099 and 0.012 for independent proposals.

A preserved lineage from the original cohort gives a source-level diagnosis of one useful edit. Adding a penalty for distance below the target—the only functional search change in that step—raises $n = 13$ success from 13/32 to 29/32 and $n = 15$ success from 3/32 to 28/32. The validation-selected child (Algorithm 2) then bundles changes to objective weights, moves and restart behavior; we therefore use the earlier step for edit-level attribution and report the complete lineage in Appendix C.5. All resulting codes pass independent linear-algebra checks. Table 1 includes the longer-code parameters alongside their prior intervals.

## 6 Limitations and conclusion

The controlled evidence is scoped to one model, one researcher proposal and binary targets. The eight generation blocks are the independent units for policy inference; repeated executions measure frozen-program variability. The policies receive equal proposal allocations but use different token totals. Exact distance verification enumerates symplectic duals, so its cost grows exponentially with their dimension. The distance guarantees assume ideal decoding and noiseless receiver-side halves; circuit costs and hardware-level error rates are not evaluated. For these controlled studies, the artifact preserves all generated sources, requests and selection records and supports fixed-program replay; stochastic model sampling is documented rather than expected to return byte-identical programs.

The central mathematical result developed from code constructions produced by evolved programs. Researchers analyzed their structure, derived a qubit family, and extended and proved the prime-power construction of Theorem 1. Later prompts reused the qubit family, showing how a result obtained through program search can guide further research. The family and entanglement lifting close 54 listed gaps. Normal-form SAT completes two final qubit closures, ten checked DRAT refutations certify selected upper bounds and minimum entanglement costs, and EA-Plotkin tightens 398 listed upper bounds.

Supplied matrices, family implementations and independent verifiers support reconstruction and checks of finite instances; preserved programs support seeded replay. The proof covers every admissible parameter. Independent Magma verification reports zero mismatches across 148 objects, and Lean checks the algebraic certificate underlying the researcher-derived theorem. The prospective eight-block cohort identifies useful program revisions and higher held-out longer-code success among iterative selections under shared research guidance. Applying the workflow to another problem still requires its own representation, evaluator and mathematical analysis.

### AI USE STATEMENT

We used Gemini models through AlphaEvolve and a local controller to propose and revise search programs under researcher guidance and exact evaluation feedback (Sections 3 and 5). LLM-based assistants, including Codex, supported implementation, mathematical reasoning and manuscript preparation. Generated codes were independently checked; Theorem 1's algebraic construction was verified in Lean. The authors reviewed AI-assisted outputs and take responsibility for the paper's claims, proofs, code and reported results.

### ETHICS STATEMENT

This work studies algebraic code constructions and search programs; it does not involve experiments on human participants or use participant-level or personal datasets. Its potential benefit is to improve methods for quantum error correction and communication, while the reported idealized parameters do not establish hardware-level reliability or operational security. Research-integrity considerations are addressed through dated sources, explicit provenance, independent verification, and separation of new parameter claims from retrospective reconstruction. Unsuccessful searches are distinguished from proved nonexistence. The controlled-study records preserve failed and excluded attempts, unavailable responses and protocol amendments, and its reviewer bundle omits account credentials and machine-local configuration. Researcher and AI contributions are disclosed, external sources are cited, and the artifact retains software licensing information. Compute allowances and model usage are documented in Appendices B.1 and C.

### REPRODUCIBILITY STATEMENT

The artifact supports reconstruction and independent checking of the mathematical outputs: generator matrices specify the finite codes, implementations generate family instances, and Appendix A.2 provides the general construction and proof of Theorem 1. The initial and later task specifications in Appendix B.3 document the transition from initial search directions to reuse of the researcher-derived qubit family. Preserved programs can be replayed with fixed seeds and evaluation budgets as described in Section 3.4 and the anonymous artifact. Appendix C and the accompanying study records supply generation requests, selections and held-out outcomes for the separate controlled comparison. Computational checks verify finite cases; general validity rests on the proof. Appendix D gives the exact formal goal and proof structure for Theorem 1. The artifact includes the Lean sources, pinned dependencies and separate kernel and Comparator verification records. Fresh LLM sampling is not part of this replay. External solver and Magma checks are reported separately from completed Python checks. The artifact is released under the MIT license and is available at `https://anonymous.4open.science/r/llm_guided_EAQECC_search-4099`.

## REFERENCES


Henrique Assumpção, Diego Ferreira, Leandro Campos, and Fabricio Murai. CodeEvolve: an open source evolutionary coding agent for algorithmic discovery and optimization. *arXiv preprint arXiv:2510.14150*, 2026.

Todd Brun, Igor Devetak, and Min-Hsiu Hsieh. Correcting quantum errors with entanglement. *Science*, 314(5798):436–439, 2006. doi: 10.1126/science.1131563.

A. R. Calderbank, E. M. Rains, P. W. Shor, and N. J. A. Sloane. Quantum error correction via codes over GF(4). arXiv:quant-ph/9608006v5, 1997. URL https://arxiv.org/abs/quant-ph/9608006v5. Version dated 10 September 1997.

Juan Cruz-Benito, Andrew W. Cross, David Kremer, and Ismael Faro. Evolutionary discovery of bivariate bicycle codes with LLM-guided search. arXiv:2606.02418, 2026. URL https://arxiv.org/abs/2606.02418.

Bogdan Georgiev, Javier Gómez-Serrano, Terence Tao, and Adam Zsolt Wagner. Mathematical exploration and discovery at scale. *arXiv preprint arXiv:2511.02864*, 2025.

Markus Grassl. Entanglement-assisted quantum communication beating the quantum Singleton bound. *Physical Review A*, 103:L020601, 2021. doi: 10.1103/PhysRevA.103.L020601.

Markus Grassl. Bounds on the parameters of various types of codes. (http://www.codetables.de), 2026. Accessed on 17 July 2026; table data preserved in the accompanying artifact.

Markus Grassl, Felix Huber, and Andreas Winter. Entropic proofs of Singleton bounds for quantum error-correcting codes. arXiv:2010.07902v1, 2020. URL https://arxiv.org/abs/2010.07902v1. First public version, 15 October 2020; Theorem 7.

Markus Grassl, Felix Huber, and Andreas Winter. Entropic proofs of Singleton bounds for quantum error-correcting codes. *IEEE Transactions on Information Theory*, 68(6):3942–3950, 2022. doi: 10.1109/TIT.2022.3149291.

Luobin Guo and Ruihu Li. Linear Plotkin bound for entanglement-assisted quantum codes. *Physical Review A*, 87:032309, 2013. doi: 10.1103/PhysRevA.87.032309.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. ShinkaEvolve: Towards open-ended and sample-efficient program evolution. *arXiv preprint arXiv:2509.19349*, 2025.

Lean Project. Comparator. https://github.com/leanprover/comparator, 2026. Revision a4f696825c583ed8a5b4060d9a0faa5b882d365b.

Gaojun Luo, Martianus Frederic Ezerman, Markus Grassl, and San Ling. How much entanglement does a quantum code need? *arXiv preprint arXiv:2207.05647*, 2022. URL https://arxiv.org/abs/2207.05647v1. First public version, 12 July 2022; Table I and Section IV.

Caroline Mauron, Terry Farrelly, and Thomas M. Stace. Optimization of tensor network codes with reinforcement learning. *arXiv preprint arXiv:2305.11470*, 2023.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Jan Olle, Remmy Zen, Matteo Puviani, and Florian Marquardt. Simultaneous discovery of quantum error correction codes and encoders with a noise-aware reinforcement learning agent. *npj Quantum Information*, 10:126, 2024.

Hendrik Poulsen Nautrup, Nicolas Delfosse, Vedran Dunjko, Hans J. Briegel, and Nicolai Friis. Optimizing quantum error correction codes with reinforcement learning. *Quantum*, 3:215, 2019. doi: 10.22331/q-2019-12-16-215.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024. doi: 10.1038/s41586-023-06924-6.

Asankhaya Sharma. OpenEvolve: an open-source evolutionary coding agent. `https://github.com/algorithmicsuperintelligence/openevolve`, 2025.

Jeonghwan Shin, Jun Heo, and Todd A. Brun. Entanglement-assisted codeword stabilized quantum codes. *Physical Review A*, 84:062321, 2011. doi: 10.1103/PhysRevA.84.062321.

Vincent Paul Su, ChunJun Cao, Hong-Ye Hu, Yariv Yanay, Charles Tahan, and Brian Swingle. Discovery of optimal quantum error correcting codes via reinforcement learning. *Physical Review Applied*, 23:034048, 2025. doi: 10.1103/PhysRevApplied.23.034048.

Mark Webster and Dan Browne. Engineering quantum error correction codes using evolutionary algorithms. *arXiv preprint arXiv:2409.13017*, 2024. URL `https://arxiv.org/abs/2409.13017`.

Nathan Wetzler, Marijn J. H. Heule, and Warren A. Hunt. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *Theory and Applications of Satisfiability Testing (SAT)*, volume 8561 of *LNCS*, pp. 422–429, 2014.

Mark M. Wilde and Todd A. Brun. Optimal entanglement formulas for entanglement-assisted quantum coding. *Physical Review A*, 77:064302, 2008. doi: 10.1103/PhysRevA.77.064302.

Yang Yang, Zining Zhong, Jindong Li, Jiemin Wu, Kaishen Yuan, Wenshuo Chen, Menglin Yang, and Yutao Yue. TurboEvolve: Towards fast and robust LLM-driven program evolution. *arXiv preprint arXiv:2604.18607*, 2026.

# A THE DOMAIN, THE TABLES, AND THE TARGETS

## A.1 CODES AS SUBSPACES OF $\mathbb{F}_q^{2n}$

For prime-power $q$, our search uses $\mathbb{F}_q$-linear stabilizer EAQECCs. Stabilizer generators are products of Pauli operators defining error-detection constraints. Entanglement assistance permits noncommuting generators on the transmitted qudits by completing them with operators on the receiver's halves into commuting constraints. Their commutation relations determine the entanglement cost (Brun et al., 2006; Wilde & Brun, 2008). Nonadditive EAQECCs form a different search space (Shin et al., 2011).

We write vectors in bold, scalar coordinates and coefficients in italic, and subspaces with italic capitals such as $S, L, R$. Modulo phase, a Pauli operator is represented by $\boldsymbol{u} = (x_1, z_1, \ldots, x_n, z_n) \in \mathbb{F}_q^{2n}$. For qubits, the pairs $(0,0)$, $(1,0)$, $(0,1)$ and $(1,1)$ represent $I, X, Z, Y$, respectively. The symplectic product with $\boldsymbol{v} = (x'_1, z'_1, \ldots, x'_n, z'_n)$ is

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_i (x_i z'_i - z_i x'_i),$$

and the symplectic weight $\mathrm{wt_s}(\boldsymbol{u})$ counts positions with $(x_i, z_i) \neq (0,0)$. For qubits, zero symplectic product means that the Pauli operators commute.

Let $S$ be the generator span and set $s = \dim S$. Its symplectic dual $S^\perp$ consists of vectors orthogonal to $S$. For any subspace $U \leq \mathbb{F}_q^{2n}$, define its radical by $\mathrm{rad}(U) := U \cap U^\perp$. Thus $S_{\mathrm{iso}} = \mathrm{rad}(S)$ contains stabilizer operations with trivial logical action. Vectors in $S^\perp \setminus S_{\mathrm{iso}}$ represent undetectable errors with nontrivial logical action. The code parameters are

$$c = \frac{s - \dim S_{\mathrm{iso}}}{2}, \qquad k = n - s + c, \qquad d = \min\{\mathrm{wt_s}(\boldsymbol{v}) : \boldsymbol{v} \in S^\perp \setminus S_{\mathrm{iso}}\}.$$

Finite-field linear algebra gives the dual and radical. With $r = \dim L = n + k - c$, the dual has $q^r$ vectors, of which $q^r - q^{n-k-c}$ are non-radical; their minimum weight gives $d$. For Theorem 1,

$r = q + 2$ is independent of length at fixed $q$. At every binary ablation length, $r = 4$: 16 dual vectors and 12 non-radical vectors. The controlled evaluator computes the radical from the Gram kernel, avoiding enumeration of $S$. The archived discovery helper instead enumerates both $S$ and $L$, totaling $2^s + 2^r$ vectors before filtering. Thus implementation and subspace dimensions determine cost, independently of the search representation. Parameters are recomputed from generators, not search-program reports.

To propagate constructions and exclusions across entanglement costs, we use the following form of the standard rule in Luo et al. (2022, Section IV, rule (4)). We include its symplectic argument for completeness.

**Lemma 1.** *An $\mathbb{F}_q$-linear stabilizer EAQECC with parameters $[[n, k, d; c]]_q$, $k \geq 1$ and $c < n - k$, gives an $\mathbb{F}_q$-linear stabilizer EAQECC with parameters $[[n, k, d'; c + 1]]_q$ and $d' \geq d$.*

*Sketch of proof.* Set $L = S^\perp$ and $R = \mathrm{rad}(L)$, with $\dim R = j = n - k - c > 0$. Choose $L = U \oplus R$. The form on $U$ is nondegenerate, since a vector of $U$ orthogonal to $U$ also lies in $R$, hence in $U \cap R = \{0\}$; hence $\dim U = 2k$. For a codimension-one subspace $R' \leq R$, put $L' = U \oplus R'$ and $S' = (L')^\perp$. Then $\mathrm{rad}(L') = R'$ and

$$L' \setminus R' \subseteq L \setminus R, \qquad \dim S' = s + 1, \qquad \dim R' = j - 1.$$

The inclusion gives $d' \geq d$, while the parameter formulas give $c' = ((s + 1) - (j - 1))/2 = c + 1$ and $k' = n - (s + 1) + c' = k$. The ambient space is unchanged, so the length remains $n$. □

Repeating the operation gives distance at least $d$ for every $c' \in \{c, \ldots, n - k\}$. Conversely, nonexistence at cost $c$ and distance at least $d$ excludes every smaller cost at the same $(n, k)$: any such code could be lifted to cost $c$. This justifies the upward construction propagation and downward exclusion propagation in Figure 3. For the qutrit family, one step reduces the radical dimension from three to two and raises $c = n - 4$ to $n - 3$. The 25 open reference cells have $n = 12, \ldots, 36$, all within Theorem 1's range; their listed upper bound $n - 1$ makes each lifted lower bound exact.

### A.2 The closed form, and why it works

Early evolved constructions $[[9, 1, 8; 6]]_2$ and $[[11, 1, 10; 8]]_2$ (Appendix B.1) motivated the odd-length qubit family recorded in later tasks. The construction below generalizes it to prime powers. Its algebraic properties have also been independently rechecked with the Lean kernel; Appendix D gives the formal goal, proof and correspondence with this construction.

Fix a prime power $q = p^e$, a length $n \geq 2q + 1$, and an enumeration $u_1, \ldots, u_q$ of $\mathbb{F}_q$. On qudits $1, \ldots, 2q$ take $q$ disjoint pairs and let $\boldsymbol{\rho}_i$ be $Z$-type on pair $i$ with $z$-values $(1, -1)$. Juxtaposition and powers denote concatenation of coordinate pairs; $(0, 1)^0$ is empty, so the tail at $n = 2q + 1$ contains only $(0, \lambda)$. Put

$$\boldsymbol{a} = (1, 0)^n, \qquad \boldsymbol{b} = (u_1, 1)(u_1, z_2) \,\big|\, \cdots \,\big|\, (u_q, 1)(u_q, z_2) \,\big|\, (0, 1)^{n-2q-1}(0, \lambda),$$

with $z_2 = 1$ for odd $p$ and $z_2 = 0$ for $p = 2$, and $L = \mathrm{span}\{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_q\}$.

**Theorem 1.** *Let $q$ be a prime power. For $q \geq 3$ with $n \geq 2q + 1$, or $q = 2$ with odd $n \geq 5$, one can choose $\lambda \in \mathbb{F}_q^\times$ so that $L$ is the normalizer side of an $[[n, 1, n - 1; n - q - 1]]_q$ EAQECC. For $q \geq 3$, its entanglement cost $c$ is $q - 2$ shared pairs below the threshold in the BDH expression; for $q = 2$, it meets that threshold.*

*Proof.* The two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}_q^{2n}$ and the $q$ radical candidates $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_q$ are defined above. Write $R_0 = \mathrm{span}\{\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_q\}$. Each $\boldsymbol{\rho}_i$ has two nonzero $Z$ components on a distinct pair of qudits, so these $q$ vectors are linearly independent. Moreover, $\boldsymbol{a}$ and $\boldsymbol{b}$ have equal $X$ components within each pair. Their symplectic products with $\boldsymbol{\rho}_i$ therefore vanish. The radical candidates are mutually orthogonal, since they are all $Z$-type.

The symplectic product of the two vectors is the sum of the $Z$ components of $\boldsymbol{b}$:

$$\delta := \langle \boldsymbol{a}, \boldsymbol{b} \rangle = q(1 + z_2) + (n - 2q - 1)\, 1_{\mathbb{F}_q} + \lambda = (n - 1)\, 1_{\mathbb{F}_q} + \lambda.$$

Here integers are interpreted in $\mathbb{F}_q$, and $q\, 1_{\mathbb{F}_q} = 0$ because its characteristic divides $q$. If $q \geq 3$, at most one of the $q - 1 \geq 2$ nonzero choices of $\lambda$ makes $\delta = 0$; choose any other nonzero value.

If $q = 2$ and $n$ is odd, the only choice $\lambda = 1$ gives $\delta = 1$. For even binary $n$, it gives $\delta = 0$: this particular construction loses its nondegenerate pair. Thus $\delta \neq 0$ in every case covered by the theorem.

To establish independence of the full generating set, suppose $\alpha\boldsymbol{a} + \beta\boldsymbol{b} + \sum_i t_i\boldsymbol{\rho}_i = \mathbf{0}$. Since $n \geq 2q + 1$, the final qudit is outside the support of every $\boldsymbol{\rho}_i$. Its two components are $(\alpha, \lambda\beta)$, so $\alpha = \beta = 0$. Independence of the radical candidates then implies $t_i = 0$ for every $i$. Consequently $\dim L = q + 2$. The restriction of the symplectic form to $L$, in this basis, has one nondegenerate $2 \times 2$ block $\left(\begin{smallmatrix} 0 & \delta \\ -\delta & 0 \end{smallmatrix}\right)$ and a $q \times q$ zero block. Hence its rank is two and $\mathrm{rad}(L) = R_0$.

Every vector outside this radical has the unique form

$$\boldsymbol{w} = \alpha\boldsymbol{a} + \beta\boldsymbol{b} + \sum_{i=1}^{q} t_i\boldsymbol{\rho}_i, \qquad (\alpha, \beta) \neq (0, 0).$$

All $n - 2q$ tail qudits are nonzero: their components are $(\alpha, \beta)$, except for the final $(\alpha, \lambda\beta)$. On pair $i$, the two coordinate pairs are

$$(\alpha + \beta u_i,\ \beta + t_i), \qquad (\alpha + \beta u_i,\ \beta z_2 - t_i).$$

If $\beta = 0$, then $\alpha \neq 0$ and all $n$ qudits are nonzero. If $\beta \neq 0$, exactly one index $i$ satisfies $u_i = -\alpha/\beta$, since $u_1, \ldots, u_q$ enumerate all $q$ elements of the finite field $\mathbb{F}_q$, and $-\alpha/\beta \in \mathbb{F}_q$. Every other pair has two nonzero $X$ components. On the exceptional pair, both $Z$ components could vanish only if $t_i = -\beta = \beta z_2$. This is impossible because $1 + z_2 \neq 0$: it equals 2 in odd characteristic and 1 in characteristic two. At most one qudit can therefore be zero, giving $\mathrm{wt_s}(\boldsymbol{w}) \geq n - 1$.

This lower bound is attained. Fix any index $i$, choose $\beta = 1$, $\alpha = -u_i$, $t_i = -1$, and $t_h = 0$ for $h \neq i$. The first qudit of pair $i$ is zero and its second has nonzero $Z$ component $1 + z_2$. All other pairs have nonzero $X$ components, and the tail remains nonzero. This vector lies outside $R_0$ and has weight exactly $n - 1$.

Finally, set $S = L^{\perp}$. Nondegeneracy of the ambient symplectic form gives $S^{\perp} = L$ and $\dim S = 2n - q - 2$. Also $S \cap S^{\perp} = L^{\perp} \cap L = R_0$, whose dimension is $q$. The parameter formulas in Appendix A.1 now give

$$c = \frac{2n - q - 2 - q}{2} = n - q - 1, \qquad k = n - (2n - q - 2) + c = 1, \qquad d = n - 1.$$

For these parameters, $2(d - 1) - (n - k + c) = (2n - 4) - (2n - q - 2) = q - 2$. Thus the BDH threshold exceeds $c$ by $q - 2$ shared pairs for $q \geq 3$ and met with equality for $q = 2$, as stated. □

The auditor instantiates this construction and verifies every member exactly for $q \in \{2, 3, 4, 5\}$ and $2q + 1 \leq n \leq 2q + 8$ (claim **`families`**); the theorem covers every prime power.

### A.3 Reference bounds and target selection

The reference tables record, for each parameter cell $(q, n, k, c)$, the listed lower and upper distance bounds and their construction sources. Our comparison uses the bounds accessed on 17 July 2026. The reference qubit table contains 563 entries whose distance bounds differ by one. The artifact preserves these table data and a second copy retrieved on 10 September 2026; both cover qubit codes with $n \leq 64$ and qutrit codes with $n \leq 36$. The reproduction command uses the July reference data unless another retrieval date is explicitly selected. Schema checks reject malformed records before classifying improvements or previously unlisted cells.

An entry with distance interval $[d_\ell, d_u]$ closes when construction and nonexistence bounds meet. For $(q, n, k, c) = (2, 7, 1, 4)$, a verified $[[7, 1, 6; 4]]_2$ code closes the listed $5 \leq d \leq 6$ gap. Cells absent from the reference tables are called *record targets* in the archive; absence alone does not establish novelty relative to all published constructions.

The familiar BDH Singleton expression $2(d - 1) \leq n - k + c$ is not a general bound in the large-distance regime (Grassl, 2021). Our principal family has $d = n - 1$, so we use the applicable generalized Singleton bounds of Grassl et al. (2022). The $[[12, 1, 11; 5]]_5$ code, for example, uses four fewer pairs than substitution into the BDH expression would suggest; it does not contradict a valid upper bound.

For qubit ($q = 2$) linear stabilizer EAQECCs with $k \geq 1$, the EA-Plotkin bound (Guo & Li, 2013) gives $d \leq \lfloor 3 \cdot 4^{k-1} n/(4^k - 1) \rfloor$. It lowers 398 upper bounds in the reference *qubit* table. The qubit target list retains cells not excluded by this inequality, the applicable Singleton bounds, or our certified refutations.

### A.4 Sources and dates of previous bounds

The dates in Tables 1 and 2 refer to verified versions of public sources, not necessarily to the first discovery of each EAQECC parameter set. The reference qubit table fixes which lower and upper bounds we compare against. For $(7, 1, 4)$, the underlying pure $[[11, 1, 5]]_2$ code is recorded by Calderbank et al. (1997); applying Theorem 7 of Grassl et al. (2020) with $\ell = 4$ establishes the EAQECC lower bound by 15 October 2020. The elapsed time to 17 July 2026 is 5.8 years when rounded to one decimal.

The other eight qubit rows use the July 12, 2022 version of Luo et al. (2022). For odd $n \in \{9, 11, 13, 15\}$, Table I contains $[[n-1, 0, n-1; n-3]]_2$. Quantum shortening gives $[[n-2, 1, n-2; n-3]]_2$, and two length extensions yield the previous $[[n, 1, n-2; n-3]]_2$ bound. The two even-length rows arise similarly:

$$[[8, 0, 8; 6]]_2 \longrightarrow [[7, 1, 7; 6]]_2 \longrightarrow [[10, 1, 7; 6]]_2,$$
$$[[10, 0, 10; 8]]_2 \longrightarrow [[9, 1, 9; 8]]_2 \longrightarrow [[12, 1, 9; 8]]_2.$$

A fourth length extension of $[[9, 1, 9; 8]]_2$ also gives the previous $[[13, 1, 9; 8]]_2$ bound. For the $k = 2$ row, the source's $[[17, 0, 12; 9]]_2$ code gives

$$[[17, 0, 12; 9]]_2 \longrightarrow [[13, 4, 8; 9]]_2 \longrightarrow [[13, 2, 8; 9]]_2 \longrightarrow [[12, 2, 7; 9]]_2$$

by four quantum shortenings, taking a subcode, and puncturing. The rules and source codes are both present in the 2022 version, giving 4.0 years between publication and our table access date. These derivations match the provenance strings in the archived EAQECC table. They establish that the baselines were available by the stated dates and still listed in 2026; they do not establish a complete chronology of intervening work.

For the displayed qutrit rows, Table II of the same 2022 source contains $[[10, 0, 8; 4]]_3$, $[[10, 0, 9; 6]]_3$, and $[[13, 1, 10; 8]]_3$. Quantum shortening of $[[10, 0, 8; 4]]_3$ gives $[[9, 1, 7; 4]]_3$. Separately, increasing the dimension of the same $[[10, 0, 8; 4]]_3$ source code using one additional shared pair gives $[[10, 1, 8; 5]]_3$ (Luo et al., 2022, Section IV, rule (6)). Applying this dimension-increase rule to $[[10, 0, 9; 6]]_3$ and then extending its length yields $[[11, 1, 9; 7]]_3$. The fourth baseline is the listed $[[13, 1, 10; 8]]_3$ code itself. The $q = 4, 5$ rows have no prior entry in the archived tables and are not included in these historical comparisons.

## B The discovery pipeline in detail

### B.1 Campaign configuration and verification

Across the four discovery campaigns, Google Cloud Vertex AI's AlphaEvolve service sampled `gemini-3.5-flash` and `gemini-3.1-pro-preview` equally, with four generation and four evaluation workers. The four campaigns requested 120, 120, 80 and 80 programs; worker logs record 120, 121, 80 and 80 completed searches (401 total). Service counters are 121, 122, 81 and 81, each including an initial program with a supplied score. Completed searches determine the nominal allowance: $401 \times 240$ s, or 26.73 h. Archived records report 1.87 M input and 3.74 M output tokens. Campaigns 1–2 share one task specification and campaigns 3–4 another (Appendix B.3).

Campaign 1 records 120 completed searches. The first new $[[9, 1, 8; 6]]_2$ and $[[11, 1, 10; 8]]_2$ codes precede completions 4 and 119, respectively: 0.267 h and 7.933 h of allocated search budget at 240 s per completed search. The logs do not provide elapsed discovery times. A later task records $[[7, 1, 6; 4]]_2$ as an earlier construction without a first-hit budget. The $[[13, 1, 12; 10]]_2$ record instead cites the closed-form theorem as its source. Initial and later tasks document the transition from finite constructions to the researcher-derived family. Algorithm 1 is a later retained program; the logs do not identify a surviving source for every early discovery. Section 5 uses separately generated, preserved programs and its own protocol.

For the complementary finite-field searches at $q = 4, 5$, simulated annealing searches $L = \mathrm{span}\{\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_j, \boldsymbol{a}, \boldsymbol{b}\}$ with $j = n - 1 - c$ fixed, disjoint $Z$-pair rows, each with entries $(1, 1)$. Moves resample up to three entries of $\boldsymbol{a}$ or $\boldsymbol{b}$. The objective counts non-radical span vectors below distance $n-1$ by exact enumeration and Gram tests. Full-rank and $\langle \boldsymbol{a}, \boldsymbol{b} \rangle_s \neq 0$ checks prune proposals; Metropolis acceptance, cooling and restarts guide search. Before saving, a separate parameter calculation checks $(k, c, d) = (1, c, n - 1)$. All 23 archived matrices have this form and pass the independent Magma check. The anonymous artifact supplies the routines and code-to-record mappings.

The independent Magma check receives generators only and re-derives every quantity we report. Symplectic weight is obtained by embedding $(x_i, z_i) \mapsto x_i + \omega z_i$ into $\mathrm{GF}(q^2)$, choosing $\omega \notin \mathbb{F}_q$. Then $\{1, \omega\}$ is an $\mathbb{F}_q$-basis, so the image is zero exactly when $(x_i, z_i) = (0, 0)$. Magma's own `Weight()` therefore counts nonzero coordinate pairs from the supplied generators.

Algorithm 1 summarizes the archived evolved construction program; its preserved source is available in the anonymous artifact. The generator matrix $\mathbf{G}$ spans $S$, and $best[t]$ stores the lowest-cost proposal seen for target $t$. The objective remains $5000|r.c - c_t| + 1000|r.k - k_t| + r.\mathrm{offending}$, with cost $10^9$ for an evaluator error. Here $r.\mathrm{offending}$ counts $\boldsymbol{v} \in S^{\perp} \setminus S_{\mathrm{iso}}$ with $\mathrm{wt_s}(\boldsymbol{v}) < d_t$; it is an integer count. After return, the external evaluator instead maximizes $1000 - \mathrm{offending}$ for matching $(k, c)$, adding $10^6$ for a newly achieved archive target. A signature mismatch scores $-5000|r.c - c_t| - 1000$; the best proposal sets the program score. Both calls use the exact parameter routine; Magma supplies the separate implementation check. The campaign seed used fixed target weights $n_t^{-2}$. The evolved program uses normalized weights proportional to $n_t^{-1.5}(1 + 10b_t)$, initially with $b_t = 0$. After a repair attempt, PROGRESSBONUS uses the stored best cost $o_t^{\star}$:

$$b_t \leftarrow \begin{cases} (1 + o_t^{\star})^{-1}, & 0 < o_t^{\star} < 100, \\ 0, & o_t^{\star} = 0, \\ b_t, & \text{otherwise.} \end{cases}$$

If no proposal was recorded, take $o_t^{\star} = 10^9$. A proposal with zero cost at its initial evaluation takes the earlier **continue** branch and skips this bonus update, exactly as in the source.

For a new attempt, the program samples one scalar $u$ and tests branches in order. If $u < 0.2$ and a stored positive-cost proposal exists, it resumes from that matrix. Otherwise, CONSTRUCT uses a cyclic construction when $u < 0.4$ and $s \leq 2n$, a divisor-stride construction when $u < 0.6$ and $s \leq 2n$, or a block construction when $u < 0.8$. If none applies, or the resulting rank is not $s$, it uses a random stabilizer. The construction branches and their validity checks remain subject to the same exact evaluator as the seed.

The cyclic branches use one or two nonzero base vectors. Row $i$ takes base $i \bmod m$ and rotates it by $2h\lfloor i/m \rfloor \bmod 2n$ bits, where $m$ is the number of bases. The ordinary cyclic branch sets $h = 1$; the added branch samples a positive proper divisor of $n$. The block branch extends the allowed dual dimension from 8 to 12 and block sizes from $\{2, 4, 6\}$ to $\{2, 4, 6, 8\}$.

During repair, MIXEDMOVE samples $v \in [0, 1)$: if $v < 0.2$ and $s \geq 2$, XOR two distinct rows; otherwise, if $v < 0.4$, rotate one row by $1, \ldots, n-1$ qubits uniformly; otherwise, apply a weight-1–3 Pauli perturbation to one row. The rotation changes one generator, not all coordinates of the code. Rank-deficient proposals increment $f$ without evaluation. The inherited Metropolis loop cools by 0.995 to 0.1 and restarts after 1,500 attempts without an accepted strict improvement. It retains each target's best accepted proposal and returns at most 16. Figure 2 shows source changes, not their individual performance effects.

### B.2 THE REFUTATION ENCODING

Encode $L = S^{\perp}$ using $j = n - k - c$ radical generators $\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_j$ and $k$ hyperbolic pairs $(\boldsymbol{u}_i, \boldsymbol{v}_i)$. Gram constraints enforce rank $2k$ and pair independence. For qubits, all $2^{2k+j} - 2^j$ non-radical combinations must have weight at least $d$. Gray-code chaining changes one generator contribution at a time; Sinz counters enforce weight. When $j \geq 1$, local symplectic maps and coordinate permutations make $\boldsymbol{\rho}_1$ a nonzero $Z$-prefix. For its bits $(x_i, z_i)$, the CNF sets $x_i = 0$, $z_1 = 1$ and adds $(z_i \vee \neg z_{i+1})$ for $i < n$. These clauses allow every nonempty prefix in one CNF, without fixing its weight or making the other radical rows $Z$-only.

For qutrits, each field element is one-hot encoded. The radical basis is in reduced row echelon form, and modular state chains impose its Gram constraints. For $k = 1$, it suffices to test the four projective logical representatives $\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{u} + \boldsymbol{v}, \boldsymbol{u} + 2\boldsymbol{v}$, each shifted by every radical vector. Their nonzero scalar multiples have the same weights. At $d = n - 1$, pairwise clauses prevent a representative from vanishing at two coordinates. The following lemma permits an additional normalization.

**Lemma 2** (Coordinate normalization)**.** *Let $L \leq \mathbb{F}_q^{2n}$ have radical $R$ with $\dim L = \dim R + 2$ and $\dim R > 2$. If every vector in $L \setminus R$ has symplectic weight at least $n - 1$, then local symplectic transformations can make $R$ entirely $Z$-type.*

*Sketch of proof.* By definition, $R = \mathrm{rad}(L) = L \cap L^{\perp}$. Let $\pi_i : \mathbb{F}_q^{2n} \to \mathbb{F}_q^2$ be $\pi_i(\boldsymbol{v}) = (x_i, z_i)$, and $\pi_{ih} = (\pi_i, \pi_h) : \mathbb{F}_q^{2n} \to \mathbb{F}_q^4$ for $i \neq h$. Writing $K_{ih} = \ker(\pi_{ih}|_L)$, the distance hypothesis gives

$$K_{ih} \subseteq \{\boldsymbol{v} \in L : \mathrm{wt_s}(\boldsymbol{v}) \leq n - 2\} \subseteq R, \qquad K_{ih} = \ker(\pi_{ih}|_R).$$

Rank–nullity therefore yields

$$\dim \pi_{ih}(L) - \dim \pi_{ih}(R) = \dim L - \dim R = 2, \qquad \dim \pi_{ih}(R) \leq 2.$$

Suppose $\dim \pi_i(R) = 2$. For every $h \neq i$, $2 = \dim \pi_i(R) \leq \dim \pi_{ih}(R) \leq 2$. Hence the inclusion $\ker(\pi_{ih}|_R) \subseteq \ker(\pi_i|_R)$ is an equality, since both kernels have dimension $\dim R - 2$. Consequently

$$\ker(\pi_i|_R) = \bigcap_{h \neq i} \ker(\pi_{ih}|_R) = \{\mathbf{0}\},$$

which implies $\dim R = \dim \pi_i(R) = 2$, a contradiction. Thus $\dim P_i \leq 1$ for every $P_i := \pi_i(R)$.

If $P_i = \{\mathbf{0}\}$, take $M_i = I_2$. Otherwise write $P_i = \mathrm{span}\{\boldsymbol{w}_i\}$ with $\boldsymbol{w}_i = (a_i, b_i)^{\mathsf{T}} \neq \mathbf{0}$, and set

$$M_i = \begin{cases} \begin{pmatrix} b_i & -a_i \\ a_i^{-1} & 0 \end{pmatrix}, & a_i \neq 0, \\ \begin{pmatrix} b_i & 0 \\ 0 & b_i^{-1} \end{pmatrix}, & a_i = 0. \end{cases}$$

In both cases $\det M_i = 1$ and $M_i \boldsymbol{w}_i = (0, 1)^{\mathsf{T}}$. For $J = \left(\begin{smallmatrix} 0 & 1 \\ -1 & 0 \end{smallmatrix}\right)$, $M_i^{\mathsf{T}} J M_i = J$. Therefore $T = \mathrm{diag}(M_1, \ldots, M_n)$ is symplectic and

$$T(R) \subseteq \{(0, z_1, \ldots, 0, z_n) : z_i \in \mathbb{F}_q\}, \qquad \mathrm{wt_s}(T\boldsymbol{v}) = \mathrm{wt_s}(\boldsymbol{v}) \quad (\boldsymbol{v} \in \mathbb{F}_q^{2n}).$$

The weight identity follows because every $M_i$ is invertible. Thus $T$ gives the required local normalization. □

The lemma also applies to binary targets with $k = 1$, $d \geq n-1$ and $j > 2$. Adding all radical-$X$ zero constraints to the seven binary exclusions gives seven further UNSAT instances with checked DRAT proofs. A feasible $[[6, 1, 5; 2]]_2$ control remains SAT and its decoded generators pass independent parameter checks. The artifact preserves these normalized certificates alongside the original ones.

For $[[10, 1, 9; 4]]_3$, $j = 5$. Row reduction preserves the zero $X$ components, so the normalized encoding retains every possible code. Adding $jn = 50$ unit clauses gives 33,033 variables and 476,271 clauses. The lemma establishes completeness; DRAT certifies unsatisfiability. As a positive control, the same normalized encoder is SAT when fixed to the archived $[[10, 1, 9; 5]]_3$ code. The anonymous artifact supplies the CNF, proof, normalization metadata and checks of the added clauses.

For certificate verification, the original CNFs are summarized in Table 3. Solver time measures a fresh UNSAT decision; for $[[10, 1, 9; 5]]_2$, it also includes proof generation. The check time uses DRAT-trim. Peak is the larger solver/checker resident set. Concurrent workloads preclude isolated timing comparisons. Execution conditions and logs are in the anonymous artifact.

Table 3: Original SAT refutation resources.

| Excluded code | Variables | Clauses | Proof (MiB) | Solver (s) | Check (s) | Peak (MiB) |
|---|---|---|---|---|---|---|
| $[[5, 2, 4; 3]]_2$ | 414 | 1,275 | 0.03 | 0.02 | 0.46 | 48.2 |
| $[[6, 1, 5; 1]]_2$ | 1,881 | 6,425 | 22.86 | 1.88 | 1.88 | 68.1 |
| $[[6, 2, 5; 4]]_2$ | 498 | 1,540 | 0.04 | 0.02 | 0.04 | 48.1 |
| $[[7, 1, 6; 3]]_2$ | 1,110 | 3,666 | 7.66 | 0.61 | 0.51 | 57.2 |
| $[[8, 1, 7; 3]]_2$ | 2,513 | 8,609 | 172.27 | 16.94 | 13.58 | 195.0 |
| $[[9, 1, 8; 5]]_2$ | 1,430 | 4,734 | 11.86 | 0.89 | 0.73 | 60.2 |
| $[[10, 1, 9; 5]]_2$ | 3,145 | 10,793 | 234.91 | 66.33 | 58.76 | 491.0 |
| $[[11, 1, 10; 7]]_2$ | 1,750 | 5,802 | 16.58 | 1.23 | 0.89 | 63.1 |
| $[[13, 1, 12; 8]]_2$ | 4,093 | 14,069 | 43.48 | 3.46 | 2.48 | 81.3 |
| $[[10, 1, 9; 4]]_3$ | 33,033 | 476,271 | 28.93 | 11.90 | 62.25 | 239.9 |

### B.3 Researcher proposals and task specifications

These structured summaries separate researcher input from the task given to the model. Shaded rows mark researcher proposals or findings. Full specifications are preserved verbatim in the anonymous artifact. The controlled study is separate from the discovery campaigns.

**Discovery campaigns 1–2** — Initial search

| | |
|---|---|
| **Researcher proposal** | Explore cyclic qubit shifts, constructive warm starts, $\mathrm{GF}(4)$-additive constructions and coupled generator moves. Reuse structure between targets sharing $(n, c)$. |
| **Task to the model** | Construct binary EAQECCs for 68 open table entries. Each target $[[n, k, d; c]]_2$ has listed lower distance bound $d - 1$ and upper bound $d$. |
| **Exact feedback** | Match the requested $(k, c)$ and reach distance $d$. The evaluator checks the symplectic constraints and reports the number of nontrivial logical operators below the target distance. |
| **Knowledge supplied** | Stabilizer representation, evaluator access and construction directions. The family formula derived later was not supplied. |

Source: archived task specifications for campaigns 1 and 2.

**Discovery campaigns 3–4** — After generalization

| | |
|---|---|
| **Researcher result** | The recorded qubit constructions $[[7, 1, 6; 4]]_2$, $[[9, 1, 8; 6]]_2$ and $[[11, 1, 10; 8]]_2$ were generalized by researchers to the odd-length family $[[n, 1, n - 1; n - 3]]_2$. |
| **Researcher proposal** | Develop further constructions using other shift steps, quasi-cyclic orbits, multiple radical blocks, $\mathrm{GF}(4)$-additive constructions, and extension, shortening or puncturing of the known family. |
| **Task to the model** | Search 329 qubit parameter targets with $5 \leq n \leq 13$, including listed gaps and previously unlisted parameter cells. The task now explicitly includes the discovered qubit family. |
| **Exact feedback** | Use the same parameter and distance checks. The specification also permits constructing the smaller dual space and converting it to stabilizer generators for evaluation. |

Source: archived task specifications for campaigns 3 and 4.

| **Controlled study** | Shared proposal for both policies |
|---|---|
| **Researcher proposal** | Maintain and mutate a basis of $L = S^{\perp}$ throughout search. Convert each candidate $L$ to $S$ for exact evaluation. For the training targets, $L$ has only four basis vectors. |
| **Task to the model** | Implement `search(target, max_evals)` under a separate quota for each target and seed. Independent proposals and iterative evolution receive the same research suggestion. |
| **Information withheld** | No successful code matrix, family formula or working $L$-side search program is supplied. There is no controller-provided initialization. |

Source: archived controlled-study task; this is separate from the discovery campaigns.

The controlled-study protocol and calibration are specified in Appendix C.1.

# C Research-guided ablation details

## C.1 Shared research guidance and evaluation calibration

The controlled study uses Gemini 3.5 Flash through Vertex `generateContent`, with a local program-development controller. It is distinct from the managed AlphaEvolve discovery campaigns. The two policies receive identical instructions to maintain and mutate $L = S^{\perp}$ throughout search and convert candidate bases to $S$ using binary nullspace and symplectic-swap helpers. The initial code is the same $S$-side annealing callback in both policies. There is no protected initializer. Programs may change representations, objectives, mutation operators and restart schedules; the working $L$-side calibration code, archived codes and family formula are not provided to either policy.

Before model generation, four calibration seeds compare the initial callback with an interface-adapted, researcher-assisted $L$-side control at up to $10^5$ evaluator calls. The controls depend on the call limit only through the remaining allowance, so their first-hit records give success curves at 1,000, 3,000, 10,000, 30,000 and 100,000 calls. At $n = 9$, the rule selects the smallest grid budget with initial success between 0.1 and 0.5, control success at least 0.5 and a difference of at least 0.25. A second cell may have zero initial success if the control succeeds in at least half the executions. The selected budget is 10,000: initial/control success is 1/4 versus 2/4 at $n = 9$, and 0/4 versus 2/4 at $n = 11$. Calibration is a test of whether the evaluator can reward a useful edit, not a measurement of evolutionary contribution. Independently verified codes establish feasibility at all training and transfer parameters before generation.

## C.2 Generation, interruption and selection

Four blocks each plan 12 proposals per policy. Independent proposals receive the fixed initial source and its fixed training feedback. Iterative evolution receives the best-so-far source and the latest attempt, including rejected attempts, with their training feedback. Training uses two fixed execution seeds at $n = 9, 11$. Each proposal is ranked lexicographically by mean normalized success-curve area, success fraction and clipped distance ratio; earlier proposals break remaining ties. For first-hit call $h$ under allowance $B$, the primary utility is

$$u(h; B) = \begin{cases} 1 - h/B, & \text{if the target is solved by call } B, \\ 0, & \text{otherwise.} \end{cases}$$

This equals the normalized integral of the success indicator over the available evaluator-call budget. The initial program remains eligible for selection. Proposal counts match; token and execution costs need not.

The planned 12-proposal runs received 58 model responses before unavailable responses interrupted generation. Before any validation or test observations, a completion-only rule retained the longest fully received prefix shared by every policy and block: seven proposals each, 56 total. The two additional responses and all request records are preserved separately. Earlier exploratory studies informed calibration and guidance; their measurements are not pooled into this comparison.

For the included 28 requests per policy, recorded input/output/reasoning token totals are 50,344/22,891/22,780 for independent proposals and 86,501/26,145/32,128 for iterative evolution. Mean request latency is 8.77 s and 11.93 s, respectively. The additional source and feedback in iterative prompts therefore incur greater model usage despite equal proposal counts. Receipt-level accounting is preserved in the artifact; service latency is separate from search time.

After generation ends, four new validation seeds at the training parameters select among each policy/block's three training-best distinct programs, using the same metric order and then training rank. The eight selected programs are distinct. Test uses eight further seeds at each training parameter and 10,000 calls per execution. The separate length transfer uses $n = 13, 15$, four seeds and 30,000 calls. Transfer measurements never enter selection. All seed lists, requests, source hashes, parent relationships, candidate outcomes and selections are retained in `experiments/hitl_ablation/`.

### C.3 Complete policy outcomes

Table 4 reports the original test cohort and the 32-seed repeat of the same validation-selected programs. Displayed values are rounded; differences use unrounded values. The new seeds were fixed before execution and did not change any program or selection. The original paired mean curve-area difference is $+0.1188$ (two positive blocks), versus $+0.0044$ in the repeat (one positive block). Descriptive two-sided sign-flip values over the four block differences are 0.50 and 1.00. These calculations use four generation blocks; execution seeds within one program quantify its run-to-run variability. Excluding the block affected by later unavailable responses gives an original mean difference of $+0.0819$. All included proposals precede those responses.

| | Original /16 | | Repeat /64 | | Curve-area difference | |
|---|---|---|---|---|---|---|
| Block | Independent | Iterative | Independent | Iterative | Original | Repeat |
| 1 | 4 | 3 | 38 | 7 | $-0.114$ | $-0.201$ |
| 2 | 4 | 10 | 27 | 26 | $+0.229$ | $-0.014$ |
| 3 | 5 | 16 | 34 | 49 | $+0.395$ | $+0.287$ |
| 4 | 8 | 9 | 36 | 33 | $-0.036$ | $-0.055$ |
| Total/mean | 21/64 | 38/64 | 135/256 | 115/256 | $+0.119$ | $+0.004$ |

Table 4: Test outcomes by generation block.

Iterative selections are proposals 2, 1, 3 and 1 in blocks 1–4, respectively (proposal numbers start at one). Thus two selections are first proposals and two incorporate feedback from earlier generated programs. The classical $L$-side control uses the shared proposed representation with the preserved annealing objective, moves and restart policy; its source is unchanged from calibration.

The repeat covers 32 new seeds at each of $n = 9, 11, 13, 15$, at 10,000 calls for the first two lengths and 30,000 for the latter two. It includes all eight selected programs, both fixed controls, and the first two programs in the block-3 iterative lineage. Together with 24 executions of the classical control on the original test and longer-code seeds, this gives 1,560 executions and 545 independently verified successful code records, with no process errors. All records, program hashes and the frozen protocol are in `experiments/hitl_followup/`.

At $n = 9, 11$, the repeat gives 96/128 and 39/128 successes for independent proposals, versus 81/128 and 34/128 for iterative evolution. The classical control gives 26/32 and 10/32, and the initial program 1/32 and 0/32. Figure 5 shows the longer-code curves; the artifact also retains the complete four-length plot.

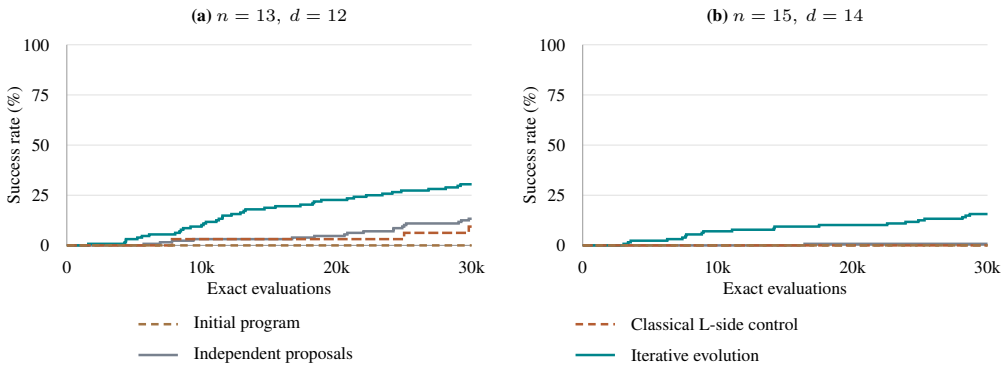



Figure 5: Longer-code replay on fresh execution seeds.

Helper computations remain outside the evaluator-call count but inside the common resource limits. Each target/seed runs in a separate sandbox with 120/125 s soft/hard CPU limits, 135 s wall time and 1 GiB resident memory. Every explicit or returned-candidate verification consumes the same call quota. Execution times and model request latencies are recorded separately in the artifact.

The artifact audit reconstructs the original proposal inputs and parent relationships, checks validation selections and independently verifies saved code records. The complete inputs and replay commands are available in the anonymous artifact.

### C.4 Prospective generation replication

The new cohort fixes eight paired blocks and seven proposal slots per policy before generation. Model, shared task, starting program, representation helpers, ranking and validation selection match the original study. New model seeds and disjoint training, validation, test and longer-code execution seeds separate the cohorts. Each block uses two training seeds, four validation seeds and 32 held-out seeds per length. The test budget is 10,000 calls at $n = 9, 11$; longer-code performance at 30,000 calls is secondary.

Of 112 allocated slots, 111 return programs: 56 independent and 55 iterative. One response in block 8 is unavailable. Before validation or testing, the recorded rule retained all eight blocks, using received programs without replacing the missing proposal. Subsequent iterative requests use the most recent received source and feedback. The seven fully received paired blocks provide a sensitivity analysis. No program is reselected using test or longer-code outcomes.

Table 5 reports all blocks; each longer-length entry has 32 executions per policy and block. The mean primary curve-area difference is $+0.0987$; five differences are positive and the descriptive two-sided sign-flip value is $p = 0.3281$. On the seven complete blocks, the mean difference is $+0.1504$. These are eight paired generation observations, not 512 independent generation trials.

| Block | Independent | Iterative | $\Delta$ area | $n = 13$ | $n = 15$ |
|---|---|---|---|---|---|
| | Test successes /64 | | | Independent / iterative | |
| 1 | 25 | 19 | -0.063 | 3 / 5 | 0 / 2 |
| 2 | 25 | 30 | +0.007 | 3 / 10 | 0 / 5 |
| 3 | 24 | 35 | -0.001 | 2 / 15 | 1 / 4 |
| 4 | 39 | 45 | +0.021 | 17 / 16 | 5 / 7 |
| 5 | 26 | 64 | +0.578 | 6 / 32 | 2 / 32 |
| 6 | 34 | 40 | +0.095 | 15 / 8 | 0 / 3 |
| 7 | 25 | 62 | +0.417 | 3 / 32 | 0 / 28 |
| 8 | 25 | 2 | -0.264 | 1 / 0 | 0 / 0 |
| Total/mean | 223/512 | 297/512 | +0.099 | 50 / 118 | 8 / 81 |

Table 5: New generation blocks.

All 16 selected sources are distinct. Source inspection finds sustained search in $L$, with exact distance feedback obtained through the common charged evaluator. Rank checks and basis conversions do not compute distance privately. One program performs uncharged basis-row additions that leave the subspace unchanged; other proposals evaluate their candidate subspaces. Reads of the incumbent before search return no matrix because controller initialization is disabled. The artifact audit reconstructs all 111 prompts and parent choices, reproduces validation selection, checks evaluation quotas and independently verifies saved codes.

The received independent requests use 100,632 input, 41,801 output and 58,067 reported reasoning tokens; iterative requests use 165,819, 49,296 and 64,426, respectively. Equal proposal allocations therefore do not imply equal token costs. Full transport records and the missing-response rule are preserved in the artifact.

For Figure 4, the original initial program and classical $L$-side control are also run on these execution seeds. This fixed-control replay was added after the new policy results were available; neither control was modified, tuned or selected using them. Each control has 32 runs per length, and is separate from the eight paired policy observations. These runs contextualize the success curves; the prespecified policy endpoint and its analysis remain unchanged.

### C.5 Preserved program and generalization to longer codes

In the original four-block cohort, the program selected in block 3 is `b02_evolution_g02`, whose preserved parent is `b02_evolution_g01`; generation indices start at zero. The parent already searches $L$ and penalizes distance shortfall. The child increases the ebit-mismatch weight from 5,000 to 10,000 and the offending weight from 1 to 100. It also mixes row replacement, Pauli perturbations and basis-row mixing, scales mutation size with length, and restores a saved local best with a capped temperature adjustment after stagnation. Tinted expressions in Algorithm 2 mark these edits relative to this immediate parent; the inherited $L$-side representation and distance-shortfall term are not tinted. These changes are observed in the preserved source, rather than inferred from performance alone. The repeat below measures the combined changes in this final step.

The program cost in Algorithm 2 is

$$\Phi(R) = 10^4 |R.c - c_t| + 100\, R.\mathrm{offending} + 10^3 \Delta_d(R),$$

where $\Delta_d(R) = \max(0, d_t - R.d)$, or $d_t$ if distance is unavailable. If the offending count or entanglement cost is missing, the score is $10^9$. These definitions match the preserved source.

In Algorithm 2, a random Pauli perturbation has weight uniformly sampled from $1, \ldots, \max(2, \lfloor n/2 \rfloor) - 1$. MIXEDMOVE chooses a row uniformly: with probability 0.3 it replaces the row by such a perturbation, with probability 0.3 it XORs the perturbation into the row, and otherwise it samples a second row. If that row is different, it XORs the two rows and, with probability 0.5, adds a Pauli perturbation. A rank-deficient proposal increments $f$ and skips evaluation. Otherwise, $f$ resets only on an accepted improvement over the current cost, and increments on all other moves. The counter $g$ resets on a new restart-local best and increments on other evaluated moves. At $g > 80$, the program restores that best, sets $g = 0$ and applies $\tau \leftarrow \min(30, 1.5\tau)$. This adjustment can lower a temperature still above 30; it is not always a temperature increase. The source and its budget behavior remain unchanged.

**Algorithm 2** Evolved search in the symplectic dual.

**Require:** target $(n, k_t, c_t, d_t)$, exact-evaluation budget $N$

$\Phi(R) = 10^4|R.c - c_t| + 100\,R.\text{offending} + 10^3\Delta_d(R)$ ▷ Increase penalties for entanglement mismatch and low-weight violations.

$r \leftarrow n + k_t - c_t$
**while** budget remains **do**
  $L \leftarrow$ a random independent list of $r$ Pauli vectors
  $C \leftarrow \Phi(\textsc{Eval}(L^{\perp}))$; $(L_*, C_*) \leftarrow (L, C)$ ▷ Retain the best state within each restart.
  $\tau \leftarrow 50$; $f \leftarrow 0$; $g \leftarrow 0$
  **while** $f < 600$ **and** budget remains **do**
    $\tau \leftarrow \max(0.01, 0.995\tau)$
    $L' \leftarrow \textsc{MixedMove}(L)$ ▷ Add row mixing and allow larger Pauli changes at longer lengths.
    **if** $\operatorname{rank} L' \neq r$ **then** $f \leftarrow f + 1$; **continue**
    $C' \leftarrow \Phi(\textsc{Eval}((L')^{\perp}))$; $\delta \leftarrow C' - C$
    **if** $\delta \leq 0$ **or** $U(0,1) < \exp(-\delta/\tau)$ **then**
      $(L, C) \leftarrow (L', C')$
      **if** $C < C_*$ **then**
        $(L_*, C_*) \leftarrow (L, C)$; $g \leftarrow 0$
      **else**
        $g \leftarrow g + 1$
      **end if**
      **if** $\delta < 0$ **then** $f \leftarrow 0$ **else** $f \leftarrow f + 1$
    **else**
      $f \leftarrow f + 1$; $g \leftarrow g + 1$
    **end if**
    **if** $g > 80$ **then** ▷ On stagnation, return to the best state and adjust the temperature.
      $(L, C) \leftarrow (L_*, C_*)$; $g \leftarrow 0$
      $\tau \leftarrow \min(30, 1.5\tau)$
    **end if**
  **end while**
**end while**
**return** the evaluator's best verified record, if any

Shading: changes from the immediate parent; exact definitions in Appendix C.5.

The original child achieves 16/16 primary-test successes and 3/4 at each longer length. Its successful longer-code executions use 3,260–9,755 calls at $n = 13$ and 7,125–16,756 at $n = 15$. Table 6 reports every preserved lineage stage on the 32 new seeds; the original validation selection is unchanged. Between the first proposal and parent, the only functional search change is the distance-shortfall penalty. Between parent and selected child, several policies change together. This lineage was chosen for diagnosis after its original performance, so the table describes that program's development separately from the four-block policy comparison.

| Program | $n = 9$ | $n = 11$ | $n = 13$ | $n = 15$ |
|---|---|---|---|---|
| First proposal | 27/32 | 19/32 | 13/32 | 3/32 |
| Parent: add distance penalty | 28/32 | 23/32 | 29/32 | 28/32 |
| Validation-selected child | 30/32 | 19/32 | 30/32 | 20/32 |

Table 6: Preserved lineage on fresh execution seeds.

All eight selected programs implement sustained $L$-side repair. Each passes candidate matrices to the common evaluator throughout search; none substitutes a private distance test that filters solutions before their first charged evaluation.

# D LEAN VERIFICATION OF THEOREM 1

The formal artifact verifies the algebraic construction underlying Theorem 1. Its shared definitions import Mathlib; the construction proof is a separate module. The challenge imports only those definitions, and the submitted solution imports the construction proof. The sources and pinned project are in the anonymous artifact.

## D.1 EXACT GOAL AND MATHEMATICAL INTERPRETATION

Use the field $\mathbb{F}_q$ and enumeration $u_1, \dots, u_q$ from Appendix A.2. Lean denotes this field by $\mathbb{F}$ and its cardinality by `Fintype.card` $\mathbb{F}$, which equals $q$. The natural number `tail` is $t = n - 2q > 0$; hence $n = 2q + t$. The disjunction below specifies the same $q \geq 3$ and odd-length binary cases as Theorem 1. Within namespace `ICLR2027EAQECC.Comparator`, the submitted declaration is:

```
theorem theorem1_exact_goal
    {𝔽 : Type u} [Field 𝔽] [Fintype 𝔽] [DecidableEq 𝔽]
    {tail : ℕ} [NeZero tail]
    (hcase : 3 ≤ Fintype.card 𝔽 ∨
      (Fintype.card 𝔽 = 2 ∧ Odd (2 * Fintype.card 𝔽 + tail))) :
    ∃ z₂ lam : 𝔽, lam ≠ 0 ∧ 1 + z₂ ≠ 0 ∧
      Nonempty (NormalizerSideCertificate 𝔽 tail z₂ lam) := by
  exact theorem1_exact hcase
```

Lean represents the radical coefficients by the function space

$$\mathcal{T} = \{f : \mathbb{F}_q \to \mathbb{F}_q\}.$$

A function $f$ assigns one scalar to each field element; $f$ need not be linear. Addition and scalar multiplication are pointwise, $(f + g)(u) = f(u) + g(u)$ and $(af)(u) = af(u)$ for $a \in \mathbb{F}_q$. The enumeration $u_1, \dots, u_q$ identifies $f$ with the vector $\boldsymbol{\tau} = (f(u_1), \dots, f(u_q)) \in \mathbb{F}_q^q$. This evaluation map is a linear isomorphism, so $\dim_{\mathbb{F}_q} \mathcal{T} = q$. Thus Lean's `ClosedCoeff` type is $\mathcal{C} = \mathbb{F}_q^2 \oplus \mathcal{T} \cong \mathbb{F}_q^{q+2}$. Write $\boldsymbol{\eta} = (\alpha, \beta, f)$ and $\tau_i = f(u_i)$, matching the coefficients of $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\rho}_i$ in Appendix A.2. Relabeling the paired coordinates by $i = 1, \dots, q$, the linear map $\Phi =$ `closedWord` is

$$\begin{aligned}
\Phi(\boldsymbol{\eta})_{(i,0)} &= (\alpha + \beta u_i,\ \beta + \tau_i),\\
\Phi(\boldsymbol{\eta})_{(i,1)} &= (\alpha + \beta u_i,\ \beta z_2 - \tau_i),\\
\Phi(\boldsymbol{\eta})_{\mathrm{tail},j} &= (\alpha,\ \beta\lambda_j), \qquad 0 \leq j < t,
\end{aligned}$$

where $\lambda_0 = \lambda$ and $\lambda_j = 1$ for $j > 0$. Thus $\Phi(\boldsymbol{\eta}) = \alpha\boldsymbol{a} + \beta\boldsymbol{b} + \sum_{i=1}^{q} \tau_i \boldsymbol{\rho}_i$ in the formal coordinate order. Let $L = \operatorname{im}\Phi$, $\mathcal{C}_0 = \{(0, 0, f) : f \in \mathcal{T}\}$, and $\delta = \langle \Phi(1, 0, 0_{\mathcal{T}}), \Phi(0, 1, 0_{\mathcal{T}}) \rangle_s$, where $0_{\mathcal{T}}$ is the zero function. The certificate constructed by `theorem1_exact` proves:

(1) `encodeInjective` and `deltaNonzero`: $\Phi$ is injective and $\delta \neq 0$.

(2) `gramFormula` and `radicalIff`: for all $\boldsymbol{\eta} = (\alpha, \beta, f)$ and $\boldsymbol{\eta}' = (\alpha', \beta', g)$,

$$\langle \Phi(\boldsymbol{\eta}), \Phi(\boldsymbol{\eta}') \rangle_s = (\alpha\beta' - \beta\alpha')\delta, \qquad \Phi(\boldsymbol{\eta}) \in \operatorname{rad}(L) \iff \boldsymbol{\eta} \in \mathcal{C}_0.$$

(3) `normalizerFinrank` and `radicalFinrank`: $\dim L = q + 2$ and $\dim \mathcal{C}_0 = q$.

(4) `distanceLower` and `distanceAttained`:

$$\forall \boldsymbol{\eta} \notin \mathcal{C}_0, \quad \mathrm{wt}_{\mathrm{s}}(\Phi(\boldsymbol{\eta})) \geq n - 1, \qquad \exists \boldsymbol{\eta} \notin \mathcal{C}_0, \quad \mathrm{wt}_{\mathrm{s}}(\Phi(\boldsymbol{\eta})) = n - 1.$$

Injectivity identifies $\Phi(\mathcal{C}_0)$ with the $q$-dimensional radical of $L$. Taking $S = L^{\perp}$ gives $\dim S = 2n - q - 2$. The standard stabilizer correspondence (Appendix A.1) therefore yields $c = (\dim S - q)/2 = n - q - 1$, $k = n - \dim S + c = 1$ and $d = n - 1$. These are the code parameters recorded by the certificate; the proof-bearing fields above certify their underlying symplectic construction.

For the coordinate convention, Lean's $(u_i, \mathrm{false})$ and $(u_i, \mathrm{true})$ are our $(i, 0)$ and $(i, 1)$. Its tail puts $\lambda$ first; Appendix A.2 puts it last. This permutation preserves the symplectic product and weight. The proof chooses $z_2 = 0$ in characteristic two and $z_2 = 1$ otherwise.

Figure 6 groups the certificate's mathematical dependencies. The dashed arrow is the standard stabilizer interpretation used in the paper; it is separate from the Lean-checked algebraic fields.

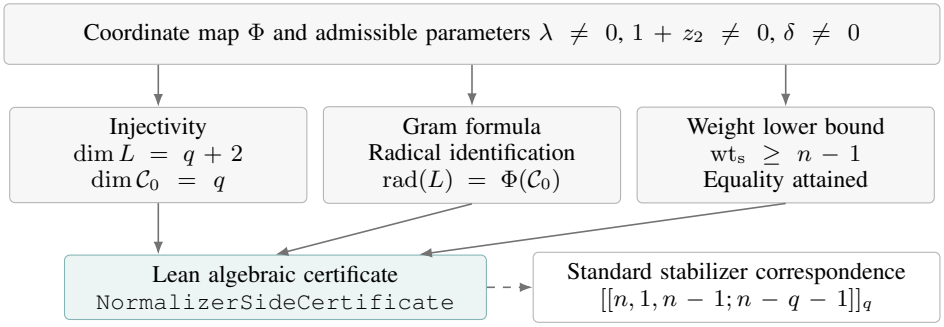



Figure 6: Dependencies of the algebraic certificate.

### D.2 Solution and formal verification

The solution above invokes `theorem1_exact`. Its proof body is reproduced verbatim below. For $q \geq 3$, it chooses $\lambda$ outside two forbidden values. In the binary branch, odd length permits $\lambda = 1$. Both branches invoke `theorem1_closed_form`, which assembles the certificate from the injectivity, Gram, radical, dimension and distance lemmas in the same source module.

```
let z₂ : 𝔽 := if (2 : 𝔽) = 0 then 0 else 1
have hz₂ : 1 + z₂ ≠ 0 := by
  dsimp [z₂]
  split_ifs with htwo
  · simp
  · simpa [one_add_one_eq_two] using htwo
rcases hcase with hlarge | ⟨hcard, hodd⟩
· obtain ⟨lam, hlam, hδbase⟩ := exists_nonzero_add_ne_zero hlarge
      (((2 * Fintype.card 𝔽 + tail - 1 : ℕ) : 𝔽))
  have hδ : closedDelta (tail := tail) z₂ lam ≠ 0 := by
    rw [closedDelta_eq_cast_length_sub_one_add]
    exact hδbase
  exact ⟨z₂, lam, hlam, hz₂, theorem1_closed_form z₂ lam hlam hz₂ hδ⟩
· let lam : 𝔽 := 1
  have hlam : lam ≠ 0 := one_ne_zero
  have hδ : closedDelta (tail := tail) z₂ lam ≠ 0 := by
    rw [closedDelta_eq_cast_length_sub_one_add]
    exact binary_odd_delta_ne_zero hcard hodd
  exact ⟨z₂, lam, hlam, hz₂, theorem1_closed_form z₂ lam hlam hz₂ hδ⟩
```

The proof and submitted solution use only `propext`, `Quot.sound` and `Classical.choice`. Comparator confirms that the solution proves the specified goal under this axiom policy, and Lean's kernel accepts the proof (Lean Project, 2026). The anonymous artifact provides the pinned project and reproducible verification records.