Coding Agents for Coding Theory

Abraham Yeung
Stanford University
ayeung16@stanford.edu

Abstract

We describe five weeks in which an LLM coding agent, working under a protocol that specified verifiers written and tested before any search, was used on open cells of the quaternary edit-metric code table. Searching over codes invariant under a prescribed symmetry group gave $E_4(6,3) \geq 120$ (the published bound, 114, dates from 2011–2012) and, after we reimplemented a stronger local-search operator, $E_4(7,3) \geq 359$ (published 356), $E_4(6,4) \geq 30$ (published 28) and $E_4(7,4) \geq 72$ (published 65). We also describe how the work went wrong. Our search first returned 116 and recorded the remaining free class as topping out at 112; a later run of that class's orbit graph with a better operator found 116 on every seed within a minute and the 120 on two of six seeds within six minutes. Our conclusion that the method did not carry over to $n = 7$ was wrong for the same reason. In each case an intermediate result was written down, never rechecked, and then treated as a fact that ruled out further search; Section 5 describes the earlier and more expensive instance. We also report that SAT and ILP produced no constructions but every certificate, and that the calibration check we propose passed the run that under-searched the winning class, so it is necessary but not sufficient.

1 Introduction

LLM coding agents are increasingly used on open mathematical problems. Whether they are useful depends less on whether they can search than on whether what they report can be trusted. Reports of AI-assisted discovery usually describe only the path that worked, which is a small part of where the effort went. This paper describes one five-week project in full: the techniques that produced new bounds for edit-metric codes, the places where the same techniques failed, and the recorded but wrong conclusions that hid answers, one of them for three weeks. We give the failures as much space as the results because they are more useful to anyone deciding how to supervise such an agent.

The result. A quaternary code of length 6 with minimum edit distance 3, equivalently a single-edit-correcting DNA-style barcode set, can have 120 codewords. The maintained table of bounds gives 114–176 (3), the lower value from a 2012 evolutionary search (1; 2); a recent LLM-guided search reports 113 for the same single-edit-correcting problem (6). Verification ran in fresh processes: reimplementations that share no source with the search find a minimum pairwise distance of exactly 3 and zero colliding edit-balls, and the load-bearing check is on the *encoding*, the 4096-vertex confusability graph differentially tested against the definition on *all* 8,386,560 vertex pairs with zero mismatches, since a graph with missing edges would produce exactly this kind of false claim. The encoding also reproduces the table's exact published entries (`search/encoding_check.py`): $E_4(3,3) = 4$ and $E_4(4,3) = 16$ computed to optimality, and our $n = 5$ sweep topping out at the published 44. We claim no more for it than a lower bound on one cell, now $[120, 176]$. The code is invariant under the Klein four-group acting on the alphabet, which acts freely here, so invariant sizes are multiples of 4. Three further cells improved on the last day of the project (Section 4), each verified the same way: $E_4(7,3)$ to $[359, 614]$, $E_4(6,4)$ to $[30, 32]$ and $E_4(7,4)$ to $[72, 128]$.

Preprint.

2 Setup

A code $C \subseteq \Sigma^n$ over $|\Sigma| = q$ has minimum edit distance d if every two distinct codewords need at least d insertions, deletions or substitutions to interconvert; $E_q(n, d)$ is the maximum size of such a code. Maximising it is a maximum independent set (MIS) problem on 4^n vertices (4096 at $n = 6$, $392{,}358$ edges), where strong specialist solvers plateau at 113.

What the agent was. An LLM coding agent (Claude Opus 5 via Claude Code) with shell access rather than a bespoke system: it wrote the verifiers and search code and drove off-the-shelf solvers (HiGHS, CaDiCaL, SciPy/CVXPY, ReduMIS). A human chose the problem, fixed the protocol and redirected between sessions; method selection, implementation and interpretation were the agent's, except for the operator, which came from a second agent session (Section 3). The agent worked in separate sessions, and no session re-reads all earlier ones; this matters for Section 5.

Related work. The $d = 3$ cells were recently attacked by LLM-guided search (6). The checks we use are standard testing ideas (known-answer instances, metamorphic and differential testing), normally applied to a system's outputs; what cost this project three weeks was an error in intermediate state and its summaries, which a verifier-first protocol does not examine.

3 What worked

Prescribed symmetry. Require the code to be invariant under a chosen group and search orbit representatives instead of words. At $n = 6$ the natural symmetry group is $S_4 \times C_2$ (alphabet permutations $\times$ reversal, order 48); we do not claim it is the graph's full automorphism group, which we did not compute. It has 33 conjugacy classes of subgroups, 22 product-form and 11 *diagonal*, the latter containing elements that pair a permutation with reversal and so expressible neither as H nor $H \times C_2$. Three order-4 classes act freely, and in each the search collapses from 4096 words to 1021 usable orbits (those with no internal conflict). This is classical prescribed-automorphism search, in the Kramer–Mesner line (4).

For comparison, ReduMIS on the raw 4096-vertex graph reached 113 on three of four seeds, each stopping on its own after 4.6 to 6.1 minutes (21.7 core-minutes in all).

What our search missed. Our sweep gave each class four seeds. Two of the three free-acting classes returned 116; the third returned 112 on all four, and we recorded 112 as its maximum. Both parts of that were wrong. Re-running the same class with our operator at ten seeds returns 116 on five of them. A later run of the same orbit graph by an independent review of our repository (a second agent session with no shared context), using a stronger local-search operator (the (1,2)-swaps and plateau moves of Andrade, Resende and Werneck (5)), reached 116 on every seed within a minute and 120 on two of six seeds within six minutes. We then reimplemented that operator in C (`search/mis_ils.c`, about 10^3 times more iterations per second than our Python). It reproduces the 120 and finds 120s in both C_4 classes, so all three free classes contain one. Whether a seed reaches 120 is decided early: fifteen ten-minute seeds on the free classes all returned 116, while thirty one-minute seeds returned 120 four times, so more seeds beat longer ones. At that per-seed rate, four seeds miss the 120 with probability about 0.57: the four-seed result was a coin toss recorded as a fact. Nothing reached 124; the Lovász theta bound on the orbit graph is 45 orbits (180 words), too loose to say whether a 124 exists.

The instance reduction is real: our operator gets 111 on the raw graph at 150 core-minutes and 116 on the orbit graph at 20, and the stronger operator gets 120 on the orbit graph while both get at most 114 on the raw one. But what a restricted search returns depends on the operator and the seed budget as much as on the instance, and we read a four-seed result as a property of the class.

Orbit arithmetic. An invariant code is a union of orbits, so its size is a subset sum of the usable orbit sizes; where the group acts freely, sizes are multiples of 4, so nothing lies between 116 and 120.

Calibration cells. Verification tells you an artifact is valid. It does not tell you whether a search was strong enough for a negative result to mean anything, and most of a project's output is negative results. Include in every batch one cell with an independent reference; ours is the unrestricted problem,

whose exact value is open, with ReduMIS's 113 as reference. Table 1 shows why: a sweep over 15 previously unexamined cells returned a plausible spread of values up to 100, while the same solver scored 99 on the unrestricted problem, where 113 is attainable. The batch was measuring the solver rather than the cells, so we discarded it. The check is not sufficient: the sweep that under-searched the winning class passed its calibration cell. It detects a broken solver but not an under-budgeted one.

Table 1: Calibration across three runs, and (right) the compute-response of the repaired operator, nearly flat over 6–60 core-min; the 150 row is our earlier raw-graph local search, for reference only.

	the three runs				operator vs. budget	
	run	core-min	best	calib.	budget	calib.
1	original operator	6	100	99	6 (broken op.)	99
2	repaired operator	60	112	110	6	109
3	later sweep, same op.	20	**116**	109	20	109
					60	110
					150 *(other solver)*	111

The check needs a decision rule, and an absolute threshold is the wrong one. Run 3, which produced the 116, scored 109, *below* the 110 that licensed trusting the operator in run 2, so a fixed cutoff would have discarded the run that found the bound. Normalising by compute resolves it. The right-hand columns plot the repaired operator against budget: 109 at six core-minutes and 110 at sixty, a $10\times$ increase buying one codeword. The curve has four points (the one at 15 core-minutes is omitted from the table); two share their configuration with runs 2 and 3, so they are not independent of what is being judged. The curve is close to flat, and against it run 1's 99 at six core-minutes is ten low. The rule is *distance from the compute-response curve*, not distance from a number; the curve costs four short runs on a known cell.

A cheaper variant needs no known answer. Conjugate subgroups give isomorphic problems, so cells within one conjugacy class must report equal optima, and any spread shows the solver is malfunctioning. Run 1 returned $88, 87, 86, 84, 88, 83$ inside a single class.

4 Where it did not work

$n = 7$: first a negative result, then not. Applied unchanged to $E_4(7, 3)$, whose record is 356, the best over 30 of the 33 invariance classes was 355. (Three classes our first enumeration missed, which need three generators, were searched afterwards and reach 340.) An earlier draft reported this as "the method did not transfer," then qualified it as a statement about our search rather than about the cell, since the same operator had under-searched the $n = 6$ winning class. The qualification was correct. The C operator, on the same orbit graphs, returned 358 on both of its ten-minute seeds in the order-6 class that had given 355, and 359 in the reversal-only class (6844 orbits); that code consists of 164 reversal pairs and 31 palindromes. Each passed all four verifiers (two for $d = 3$, two for general d), a from-scratch minimum distance of exactly 3, and a stabiliser check. Three further fifteen-minute seeds on the order-6 class all gave 355. We did not attempt $n = 8$.

Other distances. Nothing in the method is specific to $d = 3$: symbol permutations and reversal preserve edit distance, so the same 33 classes act on every conflict graph "$\mathrm{lev} \leq d - 1$." We built a general-d version (a C graph builder and two independent verifiers with a 3463-check accept/reject suite) and first checked it on the table's exact cells, which it must reproduce and must not exceed: $E_4(5, 3) = 44$, $E_4(5, 4) = 11$, $E_4(6, 5) = 8$ and $E_4(6, 6) = 4$ all came back exactly. $E_4(6, 4)$, tabled 28–32, then gave 30 in three classes, and a MaxSAT solve of one of those orbit graphs shows 30 is the optimum within that class; $E_4(7, 4)$, tabled 65–128, gave 72 in two order-8 classes. Each of these took about an hour.

SAT and ILP gave no constructions and all of the certificates. The protocol prescribed an escalation order ending in SAT and ILP. For constructions they returned nothing usable: SAT reported `UNKNOWN` after 462 s and 389 s on instances known to be feasible (binary deletion codes at $n = 11$, from the same project), and the ILP returned unconverged incumbents on the cells here. For bounds the same tools produced every certificate the project has: `drat-trim`-verified UNSAT certificates,

including a proof that the A_4-invariant optimum is exactly 112, and LP duality reproducing 24 of 24 published *LP relaxation values* as exact rational certificates.

5 The most expensive error

Three weeks before the result, a sweep reported "best invariant code 112, proven optimal in the A_4 class" over what it described as all subgroups. Four defects were behind that sentence. The parameterisation expressed invariance as "$H \leq S_4$, optionally with reversal," reaching only the 22 product-form classes. One of the 116-word codes found later lives in a diagonal class it could not express. And a size cap skipped 15 of the 60 (subgroup, reversal) cells of that parameterisation, precisely the weakest-symmetry ones.

Past the cap the solver returned unproven incumbents, and the summary dropped their status. Those two defects are the instructive ones. Three conjugate cells, isomorphic problems that must share an optimum, were reported as 100, 92 and 8 for a quantity now known to be ≥ 120. The results file *did* record `proven_optimal_invariant: false` for all three, and the conjugacy check of Section 3 would have flagged the spread. (The A_4 optimum really is 112; the error was "all subgroups.") The flag was present in the data and absent from the summary, and the summary is what a human collaborator reads. None of the four raised an error; all produced plausible numbers, so a protocol that checks outputs never questioned them. The protocol had the same gap: the ground-truth verifier did not cover this problem family until ten days after the record was found. We keep the description "verifier-first" because that is what the protocol specified, not what the main artifact received.

An agent writes down conclusions faster than any collaborator re-reads them, and those conclusions narrow the search without anyone noticing: no search that trusted the recorded summary would have revisited the class that held the bound. The fix is cheap: recheck settled conclusions on a schedule, and record unproven solver output as `unknown` rather than as a value. Our own first correction was itself incomplete and was caught only by an outside review.

6 Limitations

This is a single case study: one project, one problem family, one human supervisor, and no comparison across projects. The 120 came from a second agent session rather than our main search: that session ran our orbit graph with a stronger operator and reached 120 where we had reached 116, and the later bounds came from our reimplementation of that operator. The only control we ran is the operator comparison of Section 3; we did not run ReduMIS on the orbit graphs, which is the comparison a reader would most want. Every improvement is a lower bound on one cell, and the incumbents are unverified heuristic claims whose codeword lists were never published, so "standing since 2012" may partly reflect how few have attacked them; the $d = 4$ cells in particular have, to our knowledge, never been attacked with symmetry, and their upper bounds (32, 128) leave room we have not explored. Our classification of the failures is our own reading of our own log. The logs are self-reported by the system under study, so we release them: the supplement ships the codes, all four verifiers, the defective sweep, the protocol document and every run record, and one script re-verifies every claim in minutes.

7 Conclusion

Prescribed symmetry produced the bounds by giving a local-search operator a $4\times$ smaller instance; the orbit arithmetic only fixed the step size. Three times in this project a conclusion was recorded, never rechecked, and turned out to be hiding the answer: a sweep whose summary dropped the solver's "unproven" status, a four-seed result read as a property of a class, and a "does not transfer" verdict that was a property of the operator. Verifier-first protocols check outputs and not intermediate state; as agents run for longer, we expect that gap to be the main source of unnoticed error.

References

[1] D. A. Ashlock, S. K. Houghten, J. A. Brown, J. Orth. On the synthesis of DNA error correcting codes. *BioSystems* 110(1):1–8, 2012. `https://doi.org/10.1016/j.biosystems.2012.06.005`

[2] J. Orth. The Salmon Algorithm: A New Population Based Search Metaheuristic. MSc thesis, Brock University, St. Catharines, December 2011. Table 9.5 records $E_4(6,3)$: previous best 108, new best 114.

[3] S. Houghten. A Table of Bounds on Optimal Fixed-Length Quaternary Edit-Metric Codes. Brock University, `https://www.cosc.brocku.ca/~houghten/quaternaryedit.html`; cells $(6,3)$, $(7,3)$, $(6,4)$, $(7,4)$ read 114–176, 356–614, 28–32, 65–128. Automated fetches have returned 403 since 2026-08-26, so the 2026-09-03 check was a manual browser load; our own 2026-08-09 fetch and the 2025-08-22 Wayback snapshot agree, and the 2011 snapshot reads 111–176.

[4] E. S. Kramer and D. M. Mesner. t-designs on hypergraphs. *Discrete Mathematics* 15(3):263–296, 1976. `https://doi.org/10.1016/0012-365X(76)90030-3`

[5] D. V. Andrade, M. G. C. Resende, R. F. Werneck. Fast local search for the maximum independent set problem. *Journal of Heuristics* 18(4):525–547, 2012. `https://doi.org/10.1007/s10732-012-9196-4`

[6] F. Weindel and R. Heckel. LLM-Guided Search for Deletion-Correcting Codes. arXiv:2504.00613, 2025 (v2, 2026). Table 3 reports 113 at $n = 6$, 352 at $n = 7$, 1130 at $n = 8$.

A The four codes

Each list is the complete code; sha256 is over the sorted words, one per line, each line newline-terminated. Each was re-verified from the definition in a fresh process (pairwise Levenshtein distance $\geq d$).

$E_4(6,3) \geq 120$**, 120 words.** Stabiliser in $S_4 \times C_2$: Klein four-group $V_4 = \{\mathrm{id}, (0\,1)(2\,3), (0\,2)(1\,3), (0\,3)(1\,2)\}$ on symbols, no reversal. sha256 `adeaa3604074f5f2b42ece623e45df96f9521650b98fb11a7182321fbc5c3576`.

```
000323 001231 001302 002011 002103 003000 003222 010022 010110 011223 011311 012132
012300 013033 013201 020312 021021 021113 022030 022202 023131 023303 030013 030220
031032 031101 032233 032321 033112 033330 100200 100332 101001 101133 102122 102310
103023 103211 110213 110320 111232 112111 112333 113012 113100 120010 120123 121102
121331 122003 122221 123230 123322 130002 130130 131203 132020 132212 133121 133313
200020 200212 201121 201313 202130 203203 203331 210011 210103 211112 211330 212002
212231 213210 213323 220233 220321 221000 221222 222101 223013 223120 230122 230310
231023 231211 232200 232332 233001 233133 300003 300221 301012 301100 302232 302301
303113 303320 310030 310202 311131 311303 312220 312312 313021 320132 320300 321033
321201 322022 322110 323223 323311 330111 330333 331230 331322 332031 332102 333010
```

$E_4(7,3) \geq 359$**, 359 words.** Stabiliser in $S_4 \times C_2$: word reversal only. sha256 `c45d36c5cab37fff1947853beb5749dc609963fa73d68dffd462fdd8243e6012`.

```
0000132 0000201 0001222 0002103 0002312 0003110 0003323 0011233 0011310 0012032
0012211 0013122 0013301 0020121 0020303 0021020 0022331 0023213 0031003 0031112
0031321 0032220 0033332 0100233 0100321 0101010 0102001 0103102 0110022 0110330
0111121 0112110 0112302 0113000 0120113 0121201 0121333 0122030 0123322 0130011
0130202 0131100 0132132 0133023 0133310 0200123 0201200 0201311 0202130 0203020
0203232 0210003 0210112 0211031 0212313 0213221 0220032 0220220 0221212 0222023
0222300 0223011 0231022 0232233 0233131 0233303 0300030 0300313 0301302 0302122
0302210 0310231 0311013 0312020 0313130 0313203 0313312 0320001 0321111 0321230
0322012 0322221 0323033 0330110 0330322 0331223 0332301 0333002 1000113 1000230
1000322 1001333 1002010 1002131 1003203 1003311 1010012 1011023 1011101 1012222
1012303 1013231 1020000 1021210 1021302 1022033 1022201 1023112 1030301 1031011
1031232 1032002 1032213 1032330 1033100 1100002 1100310 1101301 1102123 1103220
1110103 1111230 1111322 1112331 1113032 1113213 1120023 1120211 1120332 1121012
1121131 1122100 1130122 1131020 1131113 1132221 1133001 1133333 1200033 1201223
1202021 1203012 1210200 1210323 1211002 1211110 1212132 1220102 1222230 1222311
1223003 1223120 1230010 1230231 1231300 1232103 1233321 1300212 1301003 1301120
1302332 1303133 1310302 1311211 1312001 1312233 1313320 1320130 1320203 1320321
1321222 1321313 1323101 1330032 1331102 1332111 1332200 1332323 1333013 2000011
2001002 2001121 2002223 2002301 2003212 2003330 2010221 2011113 2011331 2012102
2013010 2020202 2020310 2021132 2022322 2023123 2030023 2030131 2031030 2031201
2032110 2033313 2100101 2101211 2101332 2102012 2102230 2103021 2103303 2110013
2110120 2110232 2111300 2112133 2113112 2113201 2120031 2121103 2121220 2123002
```

2123233 2130312 2131323 2132000 2133130 2133222 2200022 2200110 2200203 2201013
2201320 2202333 2210311 2211122 2212030 2213023 2213100 2220133 2221000 2221231
2222101 2222213 2223312 2230001 2230330 2231111 2232202 2233032 2233210 2300220
2300331 2301233 2302100 2303111 2303322 2310000 2311202 2312121 2312310 2313333
2320112 2321301 2322003 2322232 2323020 2330103 2330211 2331012 2332031 2333300
3000003 3000120 3001031 3001300 3002232 3003221 3010111 3010332 3011212 3012000
3012321 3013103 3020022 3020231 3021203 3022113 3023001 3023130 3030200 3031123
3032101 3033012 3033233 3033320 3100112 3101022 3101130 3103013 3103331 3110001
3110210 3111033 3111102 3111311 3112203 3113323 3122222 3122301 3123111 3123200
3130030 3130223 3131231 3132120 3133302 3200211 3200302 3201101 3202220 3203122
3203310 3210020 3210233 3211303 3212011 3212123 3213202 3220313 3221021 3221330
3222002 3223223 3230121 3231312 3232331 3233000 3233113 3300021 3301113 3302033
3302201 3303230 3310222 3312112 3313031 3320010 3320123 3321032 3321100 3322131
3322320 3323212 3323303 3331001 3331210 3331333 3332022 3333132 3333311

$E_4(6,4) \geq 30$**, 30 words.** Stabiliser in $S_4 \times C_2$: $\{\mathrm{id},\ (0\,1)(2\,3),\ ((2\,3),\mathrm{rev}),\ ((0\,1),\mathrm{rev})\}$ (diagonal, order 4). sha256 `fc6fb24268384a77f148646c7760bb3049b57258d28386310f392f89675a270f`.

000333 001011 003122 013210 020202 021130 032231 033003 102301 110100 111222 112033
122112 123320 130021 131313 200220 201103 212121 220311 222000 223233 230132 303030
310012 311331 321023 331200 332322 333111

$E_4(7,4) \geq 72$**, 72 words.** Stabiliser in $S_4 \times C_2$: the order-8 diagonal group generated by $((0\,1),\mathrm{rev})$, $((2\,3),\mathrm{rev})$ and $(0\,2)(1\,3)$. sha256 `85cb1ceb68de50ccc1566cddddc8240d28b328cbb2b688e8cc0520bc74aeb391`.

0000022 0001210 0012221 0020300 0022132 0033333 0110212 0120033 0131000 0203223
0210130 0221103 0222011 0233112 0303302 0311313 0323231 0330220 1001303 1020111
1031122 1103330 1110301 1111133 1122222 1131211 1133023 1200202 1212213 1221331
1232320 1301021 1312332 1322003 1330012 1333100 2000233 2003321 2011330 2021001
2032312 2101013 2112002 2121120 2133131 2200310 2202122 2211111 2222200 2223032
2230003 2302211 2313222 2332030 3003113 3010102 3022020 3030031 3100221 3111322
3112230 3123203 3130110 3202333 3213300 3223121 3300000 3311201 3313033 3321112
3332123 3333311