Computational Search for a Hadamard Matrix of Order 668 via Legendre Pairs of Length 333:

Status Report and Roadmap

Przemyslaw Chojecki
ulam.ai

March 2026

Abstract

We report on a computational attack on the smallest open case of the Hadamard conjecture: the existence of a Hadamard matrix of order 668, equivalently a Legendre pair of length 333. We describe the theoretical framework (compression-based decomposition, mod-3 and mod-37 obstructions, macro-case enumeration), the computational infrastructure built, and the results obtained. The exhaustive 9-compression search produced 12,017,243 PSD-compatible column configurations. The 37-compression search, attacked via simulated annealing in C at 28×10^6 iterations/second, reached an L^1 PSD deviation of 236 across 18 frequencies (target: 0). We identify the precise computational bottleneck and propose concrete routes for further computation.

Contents

1 Background

A *Hadamard matrix* of order n is an $n \times n$ matrix H with entries ± 1 satisfying $HH^\top = nI$. The Hadamard conjecture asserts that such a matrix exists whenever $4 \mid n$. As of 2026, the smallest open case is $n = 668$.

A *Legendre pair* (LP) of odd length ℓ consists of two sequences $A, B \in \{+1, -1\}^\ell$ with $\sum A = \sum B = 1$ such that

$$\mathrm{PSD}(A, s) + \mathrm{PSD}(B, s) = 2\ell + 2 \quad \text{for all } s = 1, \dots, \ell - 1,$$

where $\mathrm{PSD}(X, s) = |\widehat{X}(s)|^2$ is the power spectral density at frequency s. An LP of length ℓ yields a Hadamard matrix of order $2(\ell + 1)$. For $\ell = 333$, this gives order 668.

2 Theoretical Framework

2.1 Compression

Since $333 = 9 \times 37$, we view a length-333 sequence as a 37×9 sign matrix M, with $A[9i + j] = M[i][j]$. The 9-*compression* (column sums) is a length-9 sequence where each entry is the sum of 37 values from $\{\pm 1\}$, hence an odd integer in $[-37, 37]$. The 37-*compression* (row sums) is a length-37 sequence with entries odd in $[-9, 9]$.

The p-compression preserves PSD at frequencies that are multiples of ℓ/p:

$$\mathrm{PSD}(C_p,\, t) \;=\; \mathrm{PSD}(A,\, (\ell/p) \cdot t) \quad \text{for } t = 0, 1, \dots, p - 1.$$

So the 9-compression determines PSD at the 8 multiples of 37, and the 37-compression determines PSD at the 36 multiples of 9.

2.2 Mod-3 and Mod-37 Obstructions

Theorem 1 (Mod-3 obstruction)**.** *If an H-invariant LP(333) has any multiplier $h \equiv 2 \pmod 3$, then it is impossible.*

Proof sketch. Such a multiplier forces the 3-compression (u, v, w) to satisfy $v = w$, giving $\mathrm{PSD}(A, 111) = (u - v)^2$ and similarly for B. Since u, v are odd, both PSDs are divisible by 4, yielding $r^2 + s^2 = 167$. But $167 \equiv 3 \pmod 4$, so no representation exists. □

An identical argument applies modulo 37: if a multiplier subgroup surjects onto U_{37}, the 37-compression is constant on non-identity classes, and the same sum-of-two-squares obstruction arises. Consequently, only multiplier subgroups with *proper* image modulo both 3 and 37 can support LP(333).

2.3 Macro-Cases

The LP identity at frequency 111 gives exactly 8 unordered pairs (PSD(A, 111), PSD(B, 111)) summing to 668:

$(16, 652)$	$(64, 604)$	$(76, 592)$	$(112, 556)$
$(172, 496)$	$(256, 412)$	$(268, 400)$	$(304, 364)$

These arise from the constraint that PSD at frequency 111 has the form $a^2 + 3b^2$ with $a \equiv 1 \pmod 3$ and the resulting (u, v, w) are odd integers in $[-111, 111]$.

2.4 Parseval Constraints

By Parseval's theorem applied to the compressions:

$$\sum_{j=0}^{8} c_{A,j}^2 + \sum_{j=0}^{8} c_{B,j}^2 = 594 \quad \text{(9-compression)}, \tag{1}$$

$$\sum_{j=0}^{36} d_{A,j}^2 + \sum_{j=0}^{36} d_{B,j}^2 = 650 \quad \text{(37-compression)}. \tag{2}$$

Remark 2. The 3-compression of the 37-compression is *not* the 3-compression of the original sequence, since grouping positions by j mod 3 in $\{0, \dots, 36\}$ is different from grouping k mod 3 in $\{0, \dots, 332\}$. The two compressions are "orthogonal" via the CRT isomorphism $\mathbb{Z}/333\mathbb{Z} \cong \mathbb{Z}/9\mathbb{Z} \times \mathbb{Z}/37\mathbb{Z}$.

3 Computational Infrastructure

All code is in the `hadamard/` directory. The main components are:

File	Purpose
Core framework (Python)	
`core.py`	PSD, LP verification, LP→Hadamard construction
`compression.py`	p-compression, macro-case enumeration, obstruction checks
`verify.py`	Verification of all mathematical claims
`test_pipeline.py`	End-to-end test on LP(9) → $H(20)$
9-compression search (Python)	
`search9.py`	Exhaustive PSD catalog via vectorized DFT group decomposition
`run_all_cases.py`	Batch runner for all 8 macro-cases
37-compression search (Python + C)	
`search37_fast.py`	SA for 37-compression in Python
`search37_basin.py`	Basin-hopping SA with reheating
`search37_ap.py`	Alternating projection (Gerchberg–Saxton) + SA hybrid
`sa37.c`	SA in C (28×10^6 iter/s), incremental DFT updates
`descent37.c`	Exhaustive steepest descent (2/3/4-entry neighborhoods)
`phase_search.c`	Phase-space SA (continuous DFT parameterization)
Full-length search	
`search_full.py`	SA on full 333-bit sequences (column-swap moves)
`direct_sat.py`	Direct matrix local search with fixed column sums
Intersection and decompression	
`marginals.py`	Gale–Ryser compatibility check for row/column sums
`sat_complete.py`	SAT encoder for 37×9 matrix decompression
`reconstruct.py`	PSD match → sequences → marginals → SAT → verify
`pipeline.py`	Full automated pipeline driver
`run.py`	CLI driver for individual stages

4 Results Obtained

4.1 9-Compression Search (Complete)

For each of the 8 macro-cases, we exhaustively enumerated all achievable PSD triples (PSD(1), PSD(2), PSD(4)) for the 9-compressed sequences, using the DFT group decomposition ($9 = 3 \times 3$, three groups of ~ 1000 triples each). The search ran in ~ 100 minutes total.

Case	($\text{PSD}_A(111)$, $\text{PSD}_B(111)$)	Matching configurations
0	$(16, 652)$	1,266,213
1	$(64, 604)$	1,317,923
2	$(76, 592)$	1,329,962
3	$(112, 556)$	1,327,949
4	$(172, 496)$	1,401,576
5	$(256, 412)$	1,317,121
6	$(268, 400)$	1,315,572
7	$(304, 364)$	2,740,927
	Total	12,017,243

Each configuration specifies $(\text{PSD}_A(k), \text{PSD}_B(k))$ at $k = 1, 2, 4$ of the length-9 compressed pair. Results are stored in `results/psd9_case{0--7}.json`.

4.2 37-Compression Search (Partial)

We searched for length-37 compressed pairs (d_A, d_B) satisfying

$$\text{PSD}(d_A, k) + \text{PSD}(d_B, k) = 668 \quad \text{for } k = 1, \ldots, 18$$

using several methods:

Method	Iterations	Speed (iter/s)	Best L^1 energy
Python SA	20×10^6	65,000	434
Python AP+SA hybrid	20×10^6	65,000	321
C SA (`sa37.c`)	2×10^9	28×10^6	277
C SA, 100 trials	2×10^9	28×10^6	**236**

The best solution (energy 236.2) has $\sum d_A^2 + \sum d_B^2 = 650$ (Parseval exact). Its per-frequency breakdown:

k	PSD_A	PSD_B	Sum	Dev	k	PSD_A	PSD_B	Sum	Dev
1	414.9	278.0	692.9	+24.9	10	346.9	321.7	668.5	+0.5
2	52.6	628.9	681.5	+13.5	11	247.7	388.0	635.7	−32.3
3	3.4	660.2	663.6	−4.4	12	12.1	646.1	658.3	−9.7
4	76.0	606.6	682.6	+14.6	13	176.7	501.7	678.3	+10.3
5	369.5	277.2	646.6	−21.4	14	397.0	291.9	688.9	+20.9
6	353.6	285.1	638.7	−29.3	15	68.5	620.1	688.6	+20.6
7	145.5	519.8	665.3	−2.7	16	427.2	237.2	664.4	−3.6
8	353.9	307.5	661.4	−6.6	17	157.5	503.2	660.6	−7.4
9	443.3	237.5	680.8	+12.8	18	41.7	625.6	667.3	−0.7

Of the 18 frequencies, 5 are within 5 and 8 within 10 of the target; the worst offenders are $k = 11$ (-32.3) and $k = 6$ (-29.3).

We also tried several other approaches:

- **Alternating projections (Gerchberg–Saxton style):** project between the PSD constraint surface and the integer constraint set. Best energy ~ 880; useful as a warm-start for SA.
- **Basin-hopping SA:** periodic reheating with large random perturbations. Best energy 413.
- **Direct full-length SA:** SA on the full 333-bit sequences with column-preserving swaps in the 37×9 matrix. Best energy 16,590 (average 100 per frequency), confirming that the compression approach is essential.
- **Cascade descent** (`descent37.c`): exhaustive steepest descent over all 2-, 3-, and 4-entry neighborhoods. Confirmed that *every* SA solution is already a true local minimum: zero improving moves exist in neighborhoods of size up to 4.

4.3 LP→Hadamard Construction

The LP-to-Hadamard construction was verified on $\text{LP}(3) \to H(8)$, $\text{LP}(5) \to H(12)$, and $\text{LP}(9) \to H(20)$.

5 The Bottleneck

The computational bottleneck is the **37-compression search**: finding two length-37 odd-integer sequences whose PSD values sum to 668 at all 18 independent frequencies. The key difficulties are:

(i) **Isolated local minima.** The L^1 PSD energy landscape has deep, isolated basins. SA converges to the basin bottom in $\sim 10^9$ iterations, but the basin energies cluster around 300–400. Only $\sim 1\%$ of basins reach below 280.

(ii) **Rigid local structure.** Every SA solution is a true local minimum with respect to changing up to 4 entries simultaneously. Escaping requires coordinated changes of 5+ entries.

(iii) **Algebraic integrality.** PSD values are norms of elements in $\mathbb{Z}[\zeta_{37}]$. The constraint $\mathrm{PSD}_A(k) + \mathrm{PSD}_B(k) = 668$ requires both norms to be non-negative integers summing to 668. This integrality condition creates a discrete lattice of valid PSD spectra, and the continuous SA navigates a landscape that approximates but doesn't respect this lattice.

(iv) **Coupling.** All 18 PSD constraints must be simultaneously satisfied. Improving one frequency typically worsens others. The joint search over (d_A, d_B) has 74 integer variables and 38 constraints (18 PSD + 2 sums + 18 conjugacy), leaving 36 effective degrees of freedom.

6 Proposed Routes for Large-Scale Computation

We identify four independent routes, ordered by expected impact.

6.1 Route 1: Massive Multi-Trial SA

Rationale. Our 100-trial run found $E = 236$ in trial 7. The best energy scales roughly as the minimum of N draws from a distribution with tail $\Pr[E < x] \approx (x/400)^2$ for $x < 300$. To reach $E < 100$, we estimate needing $\sim 10^4$–10^5 trials.

Compute requirements. Each trial: 2×10^9 iterations at 28×10^6/s $\approx$ 72s. For 10^5 trials: $\approx$ 2000 CPU-hours ($\approx$ 80 CPU-days, or $\sim$ 3 days on a 32-core machine).

How to run.

```
# Compile
cd hadamard && cc -O3 -march=native -o sa37 sa37.c -lm

# Run 100k trials, 2B iterations each, seed 1
# On a 32-core machine, split into 32 jobs:
for i in $(seq 0 31); do
  ./sa37 3125 2000000000 $((i*3125)) \
    >results/sa37_batch${i}.txt 2>&1 &
done
wait
# Collect best:
grep "^ENERGY" results/sa37_batch*.txt | sort -t= -k2 -n | head
```

Expected runtime: $\sim$ 3 days on 32 cores. If $E = 0$ is found, the output gives (d_A, d_B) directly.

6.2 Route 2: Multiplier-Orbit Structured Search

Rationale. The unrestricted search has 10^{37} configurations per sequence. A multiplier subgroup $H \leq (\mathbb{Z}/333\mathbb{Z})^*$ of order d partitions positions into orbits of size d, reducing the search to $\sim 10^{37/d}$ orbit-value assignments. For $d = 18$, this gives $\sim 10^2$ orbits $\times$ 10 values each $= 10^{20}$, still large but potentially tractable with SA + symmetry-aware moves.

Constraint. Only subgroups with $H \leq \ker(\mathbb{Z}/333\mathbb{Z}^* \to (\mathbb{Z}/3\mathbb{Z})^*)$ survive the mod-3 obstruction. Since $(\mathbb{Z}/333\mathbb{Z})^* \cong (\mathbb{Z}/9\mathbb{Z})^* \times (\mathbb{Z}/37\mathbb{Z})^* \cong C_6 \times C_{36}$, the kernel of reduction mod 3 is $C_2 \times C_{36}$ (order 72). We seek subgroups of this kernel.

Implementation plan.

1. Enumerate subgroups of order 6, 9, 12, 18 in $\ker \subset (\mathbb{Z}/333\mathbb{Z})^*$.
2. For each, compute the orbit partition of $\{0, \dots, 332\}$.
3. Run SA on the orbit-collapsed sequence (one value per orbit, $\sim$ 18–55 variables instead of 37).
4. Verify any $E = 0$ solution by expanding to full length-333 and checking LP.

How to run.

```
# Step 1: enumerate subgroups (SageMath)
sage: G = Integers(333).unit_group()
sage: # Find subgroups of the mod-3 kernel
sage: # Output orbit structure for each

# Step 2: for each subgroup, modify sa37.c to search over
# orbit values instead of individual entries.
# (Need custom C code per subgroup structure.)
```

6.3 Route 3: SAT/SMT Encoding of Integer PSD

Rationale. The PSD constraint $\mathrm{PSD}(d, k) = |\sum_j d_j \omega^{jk}|^2$ is quadratic in the entries d_j. For entries in $\{-9, -7, \dots, 9\}$, each d_j can be encoded with 4 bits. The full system (37 entries $\times$ 4 bits = 148 bits, 18 quadratic constraints) is within reach of modern SMT solvers (e.g., Z3, CVC5) or pseudo-Boolean solvers.

Implementation plan.

1. Encode $d_j = 2v_j - 10 + 1$ where $v_j \in \{0, \dots, 9\}$ (binary representation).
2. Express $\mathrm{Re}(\widehat{d}(k))$ and $\mathrm{Im}(\widehat{d}(k))$ as linear combinations of d_j with fixed (irrational) coefficients. Approximate by rationals with sufficient precision (the PSD values are integers, so exact arithmetic is possible using the algebraic structure of $\mathbb{Z}[\zeta_{37}]$).
3. Encode $\mathrm{PSD}_A(k) + \mathrm{PSD}_B(k) = 668$ as a system of integer quadratic equations.
4. Feed to Z3 or use Gröbner basis methods.

How to run.

```
# Z3/Python approach:
pip install z3-solver
python3 -m hadamard.smt_search  # (to be written)

# Alternatively, use the SCIP or Gurobi MIQP solver
# for the quadratic integer program.
```

6.4 Route 4: Lattice-Based Spectral Synthesis

Rationale. Given a target PSD spectrum for one sequence (say $\mathrm{PSD}_A(k) = t_k$), finding a length-37 odd-integer sequence realizing it is equivalent to finding a short vector in a lattice. Specifically, the DFT constraint $|\widehat{d}(k)|^2 = t_k$ determines the magnitudes; the phases are free. The integrality constraint $d_j \in \{-9, \dots, 9\}$ odd is a short-vector condition in the lattice generated by the inverse DFT.

Implementation plan.

1. For each of the 12M 9-compression configurations, compute the implied PSD_A at frequency 111 (fixes the macro-case).
2. Enumerate candidate PSD spectra $t_1, \ldots, t_{18}$ with $\sum t_k$ satisfying Parseval and each t_k a valid norm in $\mathbb{Z}[\zeta_{37}]$.
3. For each candidate spectrum, use the Fincke–Pohst algorithm or LLL lattice reduction to find a short vector in the inverse DFT lattice whose entries are odd integers in $[-9, 9]$.
4. Check LP compatibility.

How to run.

```
# SageMath for lattice enumeration:
sage: K.<z> = CyclotomicField(37)
sage: # Enumerate elements of Z[z] with given norm
sage: # Use LLL/BKZ for short vector search
```

7 Recommended Compute Plan

7.1 Immediate (single workstation, 1–3 days)

1. Run Route 1 (massive SA) with 10,000+ trials:

```
cd hadamard
cc -O3 -march=native -o sa37 sa37.c -lm
# 8 parallel jobs, 1250 trials each:
for i in $(seq 0 7); do
  ./sa37 1250 2000000000 $((i*1250)) \
    >results/sa37_run${i}.txt 2>&1 &
done
```

2. Post-process: extract trials with $E < 200$, examine their PSD spectra for patterns.

7.2 Medium-term (cluster, 1–2 weeks)

1. Run Route 1 at scale: 10^5 trials across ~ 100 cores.
2. In parallel, implement Route 2 (multiplier orbits) for subgroups of order 9, 12, 18.
3. Implement Route 3 (SMT) as an independent verification channel.

7.3 Long-term (if LP(333) not found)

1. Investigate whether LP(333) is obstructed by a deeper algebraic constraint (e.g., class-number conditions in $\mathbb{Q}(\zeta_{37})$, Galois PSD tests from the quaternary LP literature).
2. If LP(333) is likely non-existent, search for alternative Hadamard constructions for order 668 (e.g., Williamson-type, Turyn-type, or cocyclic constructions).
3. Consider LP lengths $\ell = 333$ with different normalizations ($\sum A = \sum B = -1$) or non-binary generalizations.

8 Key Scripts and Settings Reference

8.1 Quick verification of all mathematical claims

```
cd /path/to/math-tests
python3 -m hadamard.verify
```

8.2 Reproduce the 9-compression search

```
python3 -m hadamard.run_all_cases
# Output: results/psd9_case{0-7}.json
# Runtime: ~100 minutes, single core
```

8.3 Run C SA for 37-compression

```
cd hadamard
cc -O3 -march=native -o sa37 sa37.c -lm
./sa37 <n_trials> <iterations> <seed>
# Example: 100 trials, 2B iterations
./sa37 100 2000000000 42
```

8.4 Run cascade descent from a known solution

```
cc -O3 -march=native -o descent37 descent37.c -lm
./descent37 1 100 42 fixed
# Runs 2/3/4-entry exhaustive descent from hardcoded best
```

8.5 Full pipeline (if E=0 solution found)

Edit `pipeline.py` to insert the (d_A, d_B) solution, then:

```
python3 -m hadamard.pipeline
# Intersects with 9-compression results
# Checks Gale-Ryser
# Attempts SAT decompression
# Verifies LP and constructs H(668)
```

9 Conclusion

We have built a complete computational infrastructure for the LP(333) $\to H(668)$ search, verified all theoretical obstructions, exhaustively solved the 9-compression subproblem, and pushed the 37-compression SA to within $236/668 \approx 3.5\%$ of the target. The cascade descent analysis proves that the $\sim$ 236–400 energy basins are genuine local minima robust to 4-entry perturbations.

The most promising path forward is massive parallelization of the SA (Route 1, $\sim 10^4$–10^5 trials), complemented by multiplier-orbit reduction (Route 2) and SMT encoding (Route 3). If LP(333) exists, these approaches should find it with $O(10^3)$–$O(10^4)$ CPU-hours of computation.

A Energy Progression Through the Search

The following table records the improvement in best L^1 energy as methods were developed and refined during the project.

Step	Method	Trials	Iter/trial	Best E	Notes
1	Python SA	10	20×10^6	434	Baseline
2	Python basin-hop SA	5	50×10^6	413	Reheating helps
3	Python AP+SA hybrid	5	10×10^6	371	AP warm-start
4	Python AP+SA hybrid	20	20×10^6	321	More restarts
5	C SA (`sa37.c`)	10	500×10^6	356	430× speedup
6	C SA	13	2×10^9	277	Longer runs
7	C SA	100	2×10^9	**236**	Many restarts
8	C SA + cascade descent	20	2×10^9	329	All solutions are $\leq$4-entry local minima