Evolutionary Schedule Search in a One-Source Kernel DSL, and the Continual-Improvement Loop It Anchors

Hanzo AI Research

hanzo-kernel Paper 09 — part of the *One Source, Every Backend* series

Abstract

A one-source kernel DSL exposes each operation's schedule — tile dimensions, buffering depth, vector width, padding stride, per-shape selection — as compile-time knobs, so every knob assignment is a distinct monomorphized kernel and the space of assignments is a discrete search problem. We already own the base algebra for it: a per-`(device, op, shape)` winner cache (`tune.rs`) that times a small variant set once and reuses the winner. This paper does three things. (1) It reports the campaign's manual search as the only empirical sample — roughly 15 hand-evaluated configurations — and extracts the constants a search must respect: a static-rejection filter that is *free* (register spill, LDS over the occupancy budget), a per-op fitness signal at ±0.4% and a full-bench signal at ±0.8%, strongly *non-additive* lever interaction (BK=64 lost alone; single-buffering won; the cross-product was never explored), and hard *device non-transfer* (the reference's best-occupancy 64×64 tile is the *worst* time on gfx1151). (2) It specifies a multi-fidelity evolutionary schedule search over that space — static rejection before any GPU time, bit-exactness as viability not penalty, sustained-only fitness through the honest instrument, refutation-log negative priors seeding the population, and an island model across a heterogeneous fleet — with an eval-budget analysis showing ∼300 configurations/night against ∼15 by hand. (3) It embeds the search in a continual-improvement loop: campaign transcripts distil into decision-point evaluations (the shipped 22-case `roofline` bench is the pilot) and process-supervision data; a *calibrated judge flock*, weighted by measured ground truth rather than preference, selects training signal under a Goodhart tripwire; and the whole agent fleet coordinates through a shared field rather than pairwise, which we frame as the small-N precursor of a mean-field game. The search is not yet run: its arithmetic is derived from measured constants, and the manual search is the only empirical evidence. We mark that boundary throughout.

1 The schedule is a value, and the space is searchable

In a multi-backend engine the recurring correctness hazard is "the same operation, a different hand-tuned kernel per device, therefore a different numeric behavior." The DSL removes it by writing each operation once and lowering that one source to every backend. Autotuning closes the remaining loop: the *schedule* (tile size, vector width, unroll factor, buffering, cube dimension) becomes a set of `#[comptime]` knobs on that single source. Because the compiler monomorphizes on comptime state, each knob assignment is a distinct compiled kernel with identical semantics — so exploring schedules is a search over compiled variants of one bit-exact source, never a fork.

The campaign of the companion study [1] produced the evidence that this search is worth automating: the winning schedules were *not* the ones intuition or the reference implementation

predicted, they interacted non-additively, and they did not transfer across devices. Those three facts (§3) are precisely the conditions under which a cached, per-device, measured search beats hand-tuning. We first ground the machinery in what already exists (§2), then specify the search (§4), then embed it in a learning loop (§§5–7).

2 Grounding: the existing schedule algebra

The base layer is three orthogonal values in `hanzo-kernel/src/tune.rs`, deliberately kept as pure data so the mechanism is testable without a GPU.

- `Variant` — a named schedule: a closure `f(iters) -> (output, ms)`. The same closure benchmarks (many iters) and executes (one iter), so a variant is both a candidate and a runnable kernel.
- `Tuned` — a fluent builder collecting the variants for one `(op, shape-key)`.
- `Tuner` — the cache: an in-memory `(device, op, key)` → `winner` map backed by one TSV file per device under the XDG cache dir. A cache *miss* times every variant `TUNE_ITERS=25` times and persists the fastest; a *hit* launches only the winner once. It is a plain value over an explicit root, so the select/cache/reload logic is unit-tested with pure closures — no GPU, no globals.

The result of a run is a `Pick{output, winner, from_cache, benched, timings}`; `benched` is N on a miss and 0 on a hit, making "the second call skips timing" an observable. A first-party design was chosen over `cubecl_runtime::tune` for three concrete reasons, not for their own sake: its tuner is re-exported only from `cubecl_runtime` (adopting it would force a second engine crate into a facade architected so only `Cargo.toml` names the engine); its persistent cache lives under `target/` via an MD5 checksum and `serde`, not the required XDG path; and its `TuneInputs::At` GAT plus `KeyGenerator`/`InputGenerator` traits are far more surface than "pick the fastest of N named closures" needs.

This layer already proves the loop on the CPU runtime: two ops (`rms_norm`, `matvec_q8_dp4a`) each expose one comptime-knobbed source with a 4-variant set; the tests show every variant bit-exact against the CPU oracle, a winner selected and cached in memory and on disk, and a fresh `Tuner` reloading the winner from disk. What is *missing* is the search itself: the variant set is hand-written and tiny, the objective is a single timing with no multi-fidelity gating, and there is no population, no cross-device migration, and no negative prior. This paper specifies those.

3 The only empirical sample: the manual search

The campaign hand-evaluated on the order of 15 schedule configurations for the two dominant prefill GEMMs on gfx1151. That sample — small, but real and hardware-verified — fixes the constants any automated search must obey.

3.1 The fidelity ladder and its costs

Evaluation is not one operation but a ladder of increasing cost and decreasing throughput (Table 1). The load-bearing fact is that the first rung is *free*: a compile plus a read of shader statistics (registers, LDS bytes, scratch) rejects a large fraction of candidates — anything that spills scratch (> 0) or exceeds the LDS occupancy budget — before a single GPU dispatch. The campaign repeatedly killed configurations at this rung: a 32 KB accumulator spill (occupancy capped at 2),

a 40 KB double-buffered K-tile (occupancy capped at 6 where registers alone allowed ∼9), an f16-accumulator variant whose VGPRs went *up* 168 → 180. None of these needed a timed run to be disqualified.

Fidelity	**Signal**	**Cost / config**	**Noise**
static (compile + shaderstats)	scratch, LDS, VGPR, occupancy	seconds, no GPU	exact (deterministic)
bit-exact oracle	`scale_rel` vs CPU reference	seconds	exact (a gate, not a score)
sustained per-op (in-engine)	GPU μs at the live shape	seconds	±0.4%
full bench (in-engine)	pp512 / tg128 vs `llama`	minutes	±0.8%

Table 1: The evaluation fidelity ladder. Static rejection and the bit-exact gate are effectively free relative to timed runs; the two timed rungs carry the ±0.4% / ±0.8% noise floors measured after the harness fix of [1]. Per-config compile time (a single-shader `glslc`/`spirv-opt` is seconds; a cubecl re-monomorphization plus engine relink is longer) is treated as an order-of-magnitude modeling constant, not a logged datapoint (§8).

3.2 Non-additivity and device non-transfer

Two properties of the sample make greedy, one-lever-at-a-time tuning provably incomplete.

Non-additivity. The schedule knobs contend for shared occupancy resources (LDS bytes and VGPRs), so their effects are not separable. `BK`=64 *lost* when changed alone (it doubled LDS and left occupancy at 6); single-buffering *won* (occupancy 6 → 8, GEMM 232 → 181 μs); the recorded losers also include `sb64` (single-buffer × BK=64) and `sb_bm64` (single-buffer × BM=64). The eventual shipping schedule was *single-buffer* ∧ *BK=32* ∧ *BM=64* ∧ *packed-f16vec2 LDS* — reached by a chain of single-lever A/Bs, with the full cross-product never explored. A greedy walk that had accepted BK=64's solo loss as final would have masked a lever that only pays in combination; a search must be able to evaluate joint assignments.

Device non-transfer. The reference implementation selects a tuned tile *family* ({64x64, 128x128, 128x256}) per GEMM shape for its target GPUs. On gfx1151 the smallest of these, the 64×64 RN4 tile, achieved the *best* occupancy of any variant (10 subgroups/SIMD) and the *worst* sustained time (1915 μs vs the 128×128 RN8 tile's 1634 μs): the larger tile's activation-reuse amortization beats occupancy on this silicon. Independently, the optimal thread count for the block-reduce MoE matvec *differs by kernel* on the same device (Q4_K at `nt`=64, Q6_K at `nt`=32). A schedule optimal on one device or one op is not merely suboptimal elsewhere — it can be the worst choice. This is the empirical justification for a winner cache keyed on `(device, op, shape)` rather than a global constant, and for running the search on the deployment hardware.

4 Multi-fidelity evolutionary schedule search

The search treats a schedule as a genome and the fidelity ladder as a cost-ordered filter, so that GPU time is spent almost entirely on candidates that have already survived free static analysis and the correctness gate.

Genome. A schedule is the tuple

$$g \;=\; \big(\underbrace{\mathrm{BM},\mathrm{BN},\mathrm{BK}}_{\text{tile}},\; \underbrace{\beta}_{\text{buffering}\in\{1,2\}},\; \underbrace{v}_{\text{vec width}\in\{\text{u16,f16}\times 2,\text{uvec4}\}},\; \underbrace{\sigma}_{\text{pad stride}},\; \underbrace{\pi}_{\text{per-shape selector}}\big),$$

each coordinate a comptime knob, so g names one monomorphized, bit-exact kernel. Selection is elitist with tournaments; recombination is uniform crossover over coordinates; mutation perturbs one coordinate within its legal set. The refutation log contributes two things to the genetics: it *seeds* the initial population away from known-dead regions, and it *masks* operators that only produce dead offspring (e.g. the coopmat-with-f16-accumulator direction, or BM≥128 on this device).

Algorithm 1 EvoSchedule — multi-fidelity search over comptime schedules

Require: op o, shape s, device d, budget B, log L, population size μ, offspring λ

1: $P \leftarrow \textsc{SeedPopulation}(o, d, L)$ ▷ negative priors; mask dead operators
2: $best \leftarrow \textsc{Incumbent}(o, s, d)$
3: **while** B not exhausted **do**
4: $Q \leftarrow \varnothing$
5: **for** $i \leftarrow 1 \ldots \lambda$ **do**
6: $(a, b) \leftarrow \textsc{Tournament}(P),\ \textsc{Tournament}(P)$
7: $c \leftarrow \textsc{Mutate}(\textsc{UniformCrossover}(a, b))$ ▷ or $\textsc{LlmMutate}(a, L)$, §4.2
8: **if** $\textsc{Spill}(c, d) > 0$ **or** $\textsc{Lds}(c, d) > \textsc{OccBudget}(d)$ **then**
9: **continue** ▷ FREE static reject; no GPU time
10: **end if**
11: $Q \leftarrow Q \cup \{c\}$
12: **end for**
13: **for** $c \in Q$ **do** ▷ only static survivors reach the GPU
14: **if** $\textsc{ScaleRel}(c, \textsc{Oracle}(s)) > \tau$ **then**
15: $c.fit \leftarrow \infty$ ▷ infeasible: viability, not a penalty
16: **else**
17: $c.fit \leftarrow \textsc{SustainedPerOp}(c, s, d)$ ▷ honest instrument, ±0.4%
18: **end if**
19: **end for**
20: $P \leftarrow \textsc{ElitistSelect}(P \cup Q,\ \mu)$
21: $best \leftarrow \arg\min_c c.fit$
22: $\textsc{Migrate}(d, best)$ ▷ island model: publish winner to the fleet, §4.1
23: **end while**
24: **assert** $\textsc{FullBenchWin}(best)$ vs `llama`, same run, ±0.8% ▷ final gate
25: **return** *best*

Three design choices distinguish this from a generic evolutionary tuner. Line 9 makes static rejection *free and first*: no fitness is defined for a candidate that spills or overflows LDS, so those never consume GPU time. Line 15 treats bit-exactness as *viability, not a soft penalty* — an incorrect kernel does not get a discounted fitness, it does not reproduce. Line 17 scores on sustained in-engine per-op time, never a cache-warm microbenchmark, because for memory-bound kernels the warm number systematically overstates the real regime (the campaign burned several otherwise-correct kernels on exactly that lie).

4.1 Island model over a heterogeneous fleet

Because fitness is device-specific (§3.2), the search is an island model: one deme per box (evo/gfx1151, spark/GB10, dbc/M4 Max), each evolving against its own hardware. Migration carries a winning *genome*, not a fitness — an immigrant is always re-evaluated locally, because a schedule that tops one island can be the worst on another. Migration is therefore a source of good *structure* to recombine, not a claim of cross-device optimality. This is the fleet-level view whose coordination we formalize in §7.

4.2 LLM-guided mutation as an optional operator

The mutation operator can be replaced or augmented by a language model that proposes a new genome from the parent plus the refutation log — the pattern of program-search systems such as FunSearch [5] and AlphaEvolve [6], where an LLM proposes and an external evaluator selects. Here the selector is exactly the gate ladder of [1]: an LLM proposal enjoys no privilege, it must pass static rejection, the bit-exact oracle, and sustained timing like any other candidate. The LLM supplies search *bias* (informed guesses about which joint assignments to try, drawn from the log's mechanisms); the hardware supplies *truth*. This keeps the creative operator unfalsifiable-proof: it can only ever propose, never accept.

4.3 Eval-budget arithmetic

The point of cost-ordering is a favorable generation budget. Take $\lambda = 16$ offspring per generation. The campaign's static-rejection experience (spills and LDS-overflows were common among hand-tried variants) puts the free-reject rate near 50%, so $\sim$8 candidates per generation reach the GPU. Each GPU evaluation is a bit-exact check plus a 25-iteration sustained per-op timing — seconds of GPU time — plus a monomorphization/relink amortized across the batch. At an effective few minutes per surviving candidate, a single box clears on the order of a few hundred configurations in an overnight window; against the $\sim$15 the campaign evaluated by hand over weeks, that is a 10–20$\times$ throughput increase in schedules examined, with the full-bench rung (minutes, $\pm$0.8%) reserved for the handful of per-op finalists. The arithmetic is a projection from measured constants, not a measured result (§8); its purpose is to show the search is budget-feasible on existing hardware, not to promise a specific speedup.

5 The continual-improvement loop

The search consumes agent-run optimization campaigns and, in turn, those campaigns produce data that improves the agents. The loop has four stages.

1. **Run** campaigns (human- or agent-driven) under the gate ladder, logging every decision point: the evidence on the table, the action space, and the measured outcome.
2. **Distil** each transcript into *decision-point* triples ⟨evidence-snapshot, action-space, measured-outcome⟩. These become (a) new evaluation cases and (b) process-supervision targets.
3. **Select** training signal with an evaluation that rewards the right *process*, not only the right answer (trap-resistance and an overthink penalty as auxiliary rewards).
4. **Improve** the models, which become better mutation operators (§4.2) and better campaign agents — closing the loop.

5.1 Decision-point distillation and its schema

The pipeline's pilot already ships: the 22-case `roofline` benchmark is decision-point distillation run by hand over the campaign of [1]. Each case is one record:

```
{id, category, trap, question:{evidence, choices}, answer, rationale}
```

where `answer` is the hardware-verified outcome and the distractors are the campaign's *actual* refuted theories. Categories span `diagnosis`, `experiment`, `trap`, `update`, `calculation`, `discipline`, and `synthesis`; 6 of 22 are traps (the prompt asserts a false premise the evidence contradicts). Ground truth is unpublished, so the set is uncontaminated by pretraining. Crucially, the standard results JSONL already logs `usages.completion_tokens` per item, so two post-hoc metrics need no new instrumentation: the *overthink index* (mean completion tokens on *correct* answers) and *trap resistance* (accuracy on the trap subset).

5.2 Reward shaping

Selection optimizes a shaped reward rather than raw accuracy, so that a model is rewarded for reasoning *to* the measurement and penalized for reasoning *past* it:

$$R \;=\; \underbrace{\mathbb{1}[\hat{a} = a^\star]}_{\text{accuracy}} + \lambda_{\text{trap}} \cdot \mathbb{1}[\text{trap} \wedge \hat{a} = a^\star] \;-\; \lambda_{\text{ot}} \cdot \frac{\text{tokens\_on\_correct}}{T_{\text{ref}}}.$$

The trap term overweights the cases where the prompt's framing is adversarial (verify-before-comply); the overthink term, normalized by a reference length T_{ref}, taxes the models that talk themselves back into a refuted theory. The `rationale` field supplies process-supervision targets: the reward is not only on the boxed letter but on recovering the decisive experiment. This is implementable directly against the existing harness — the arms are the answer letters, the results JSONL carries the token usages, and the golds are the hardware-verified answers.

5.3 Calibrated judge flocks

Selecting training signal at scale needs judges, and LLM-as-judge pipelines are usually *preference-only*: they rank outputs by what a model prefers, inheriting position, verbosity, and self-preference biases [14]. This stack has a rarer asset — *anchored calibration*: subsets with measured ground truth against which a judge's reliability is a fact, not an opinion. Four anchors exist today: the 22 hardware-verified `roofline` answers (6 of them traps); the router corpus of **2,981** counterfactual outcomes (373 prompts × 8 model arms); the bit-exact kernel gates; and the enso-bench golds. The router corpus is concrete evidence that calibration is separable from preference: a first learned routing head scored AIQ = 0.6248 against a 0.7427 heuristic and was *not* shipped; after a train/serve featurizer skew was fixed (it had trained without prompt text), the head reached held-out decision quality 0.9215 ± 0.011 versus the rule's 0.9162 ± 0.007, with an oracle ceiling of 0.9402. The anchored eval, not a preference signal, is what exposed the skew.

Five properties are load-bearing. **Diversity is not decoration**: same-family judges share correlated blind spots that vote aggregation cannot remove, so the flock is drawn across model families — consistent with the Panel-of-LLM-evaluators result that a small, diverse jury beats a single frontier judge on both bias and cost [11]. **Reliability is per-domain**: a judge trustworthy on kernel-perf reasoning is not automatically trustworthy on prose, so weights are conditioned on the domain of the item. **Disagreement is a hardness detector** (line 8): high inter-judge variance marks the items worth escalating to a debate round [15] or a human — averaging them

Algorithm 2 CALIBRATEDJUDGEFLOCK — anchored, diverse, disagreement-aware judging

Require: judges J (family-diverse), anchored sets $A_{1..m}$ with golds, candidate pool C, student θ

1: **for** $j \in J$, domain k **do** ▷ (1) reliability from measured truth, per domain
2: $w_{j,k} \leftarrow \textsc{Reliability}(j \text{ on } A_k)$ ▷ accuracy vs golds; also tokens/correct-verdict
3: **end for**
4: **for** $x \in C$ **do**
5: $\{s_j\}_{j \in J} \leftarrow \textsc{Verdicts}(J, x)$
6: $\bar{s}(x) \leftarrow \dfrac{\sum_j w_{j,\mathrm{dom}(x)}\, s_j}{\sum_j w_{j,\mathrm{dom}(x)}}$ ▷ (3) reliability-weighted aggregation
7: **if** $\textsc{Var}_j(s_j) > \eta$ **then** ▷ (4) disagreement-as-curriculum
8: escalate x to a debate round or human; route to the high-value training pool
9: **end if**
10: **end for**
11: $\theta' \leftarrow \textsc{Finetune}_{\mathrm{DPO/RLAIF}}(\theta,\ \bar{s},\ \text{anchored items overweighted})$ ▷ (5)
12: **if** $\textsc{Anchored}(\theta') < \textsc{Anchored}(\theta)\ \wedge\ \textsc{FlockPref}(\theta') > \textsc{FlockPref}(\theta)$ **then**
13: **halt**; recalibrate ▷ (6) Goodhart tripwire: judge-hacking detected
14: **end if**
15: **re-measure** $w_{j,k}$ next cycle; version the flock ▷ (7) a frozen flock is the student's ceiling
16: **return** θ'

away discards the highest-value training data. **The Goodhart tripwire** (line 13) is the anchored holdout's job: a student that improves on flock *preference* while regressing on measured *truth* is hacking the judges, and the cycle halts. **The flock is versioned and re-measured** each cycle, because a frozen flock is a hard ceiling on the student it trains. Judges are themselves scored on tokens-per-correct-verdict — the overthink index of §5.1 applies to judges too, keeping the panel cheap.

This training-time flock is the mirror of a shipped inference-time one: the `enso-ultra` product line's verify-then-select over a diverse candidate set *is* a flock at inference. The contribution here is to run the same diverse-panel discipline at training time, and to anchor it — because we can — on measured ground truth rather than preference alone. The base techniques are external and cited plainly: learning from AI feedback [12], preference optimization [13], and the judge-bias and debate literatures [14, 15]; the anchoring and the Goodhart tripwire are what the measured corpora add.

6 Generalization: the oracle-substitution rule

The architecture — cheap-reject ladder, honest instrument, refutation-log prior, evolutionary search under a viability gate, distillation into decision-point evals — is not specific to kernels. It ports wherever two things exist: a *trustworthy fitness oracle* and a *cheap rejection ladder* beneath it. Only the oracle changes.

- **Coding agents.** Fitness oracle: the test suite, the type checker, the runtime verifier. Cheap rejects: compile failure and static analysis stand in for shaderstats; a failing test stands in for a bit-exactness miss. Verified-campaign transcripts distil into decision-point evals exactly as here, and the process-supervision target is the sequence of decisive checks, not just the final diff.
- **Marketing agents.** Fitness oracle: an A/B conversion metric. The *same* disciplines transfer unchanged — contended-measurement refusal (a concurrent campaign contaminates the lift the way a foreign GPU job contaminates a kernel A/B), a noise floor below which a lift is

unreadable, and a refutation log of dead creative directions. What changes is only the oracle: a *statistical-significance gate* replaces bit-exactness. Viability becomes "the lift clears the significance threshold," not "the bits match."

The rule has a hard boundary, which we state rather than paper over: the architecture works wherever fitness is *measurable and cheap enough to gate on*, and it fails where fitness is not. A task whose only signal is diffuse long-horizon human judgment, with no anchor and no cheap reject, has no honest instrument for the ladder to escalate through — and there the search degenerates into preference-chasing, precisely the failure mode the judge flock (§5.3) is built to detect. The method is as strong as its weakest oracle.

7 Mean-field coordination of the agent fleet

The search (§4.1), the judge flock (§5.3), and the campaigns themselves are run by a fleet of agents sharing scarce resources. The campaign wrote its coordination laws by hand; we formalize them here as the small-N precursor of a mean-field game, and are explicit about where the analogy holds and where it does not.

7.1 The observed game

N agents contend for shared resources: GPUs (a hard "one agent per box" law), the `git main` version queue (publish sequencing), `cargo` locks, and the search-space basins themselves. The log records two failures of *pairwise* coordination and one success of *field* coordination.

- **Failure — the patch-line race.** An agent's local `[patch.crates-io]` redirect raced into a peer's release commit; a pre-commit grep passed, then the line was re-added before the commit landed, so the tagged release did not build off-box until a follow-up fixed it. A pairwise "check then act" lost to a concurrent writer.
- **Failure — branch-delete before merge.** An agent deleted a remote feature branch before verifying the merge had landed; the fast-forward had aborted because `main` moved under it, and the commits had to be rescued from the local object store and re-landed. Again: the shared state moved between one agent's fetch and its act.
- **Success — the front map.** Publishing a *population-level* summary — which fronts had explorers, which trees were owned — caused a peer agent to *stand down* of its own accord, twice, leaving its checkout clean and detached. The peer best-responded to the field (the occupancy of fronts), not to any individual agent.

The pattern is exactly the mean-field intuition: pairwise coordination is $O(N^2)$ interactions and races on shared mutable state; field coordination is each agent best-responding to an aggregate, at $O(N)$ communication.

7.2 The formalism

Let μ be the *field*: a distribution over resource states — per-box GPU occupancy, per-tree ownership density, the version queue, and per-basin explorer density in schedule space. Each agent solves a best-response

$$\pi^\star \;=\; \arg\max_{\pi}\; \mathbb{E}\Big[\; \underbrace{V(\pi)}_{\text{lever value}} \;-\; \underbrace{c\big(\mu(\pi)\big)}_{\text{congestion}} \;-\; \underbrace{\kappa\cdot\mathrm{switch}(\pi)}_{\text{switching cost}} \Big],$$

and an equilibrium is a field $\mu^\star$ induced by the agents' best responses to $\mu^\star$ — a fixed point, in the sense of the mean-field game limit of Lasry and Lions [16] and the Nash-certainty-equivalence of Huang, Malhamé and Caines [17]. The solver the campaign already ran is *fictitious play*: publish the current field, let agents best-respond, republish. Convergence is expected under monotone congestion costs (the Lasry–Lions monotonicity condition): a second agent on a box, a second explorer in a basin, is strictly worse for everyone there, which is the "one agent per box" and anti-crowding rules made quantitative.

Algorithm 3 MeanFieldFleetCoordinator — interact through the field, not pairwise

Require: agents $\{1..N\}$, resources $\mathcal{R}$ (boxes, trees, version queue, basins)

loop ▷ fictitious play: the publish/respond cycle the campaign ran
  $\mu \leftarrow \textsc{Field}(\mathcal{R})$: {GPU busy, tree-ownership density, version queue, basin explorer density}
  for agent a **in parallel do** ▷ $O(N)$ communication: each reads μ, none reads peers
    $a.\pi \leftarrow \arg\max_\pi \; V_a(\pi) - c(\mu(\pi)) - \kappa\,\mathrm{switch}_a(\pi)$
    if $\mu(\mathrm{box}(\pi)) > 0$ **then** best-respond elsewhere ▷ one agent per box
  end for
  $\textsc{Publish}(\mu')$ ▷ next field; version-queue writes take the next free patch, index-verified
end loop

7.3 The router is a textbook congestion game

The same structure appears one level down, at inference routing. A population of requests routed to a set of model arms degrades those arms' latency and cost as load rises — a congestion game whose equilibrium is Wardrop routing [18]: traffic distributes so that no request can lower its own cost by switching arms, i.e. each request best-responds to the *arm-load field*. This is directly implementable on the existing seam: the `hanzo-router` replica layer already tracks per-arm load and prefix affinity, so the learned routing head of §5.3 need only take expected-load arguments in its SLO terms to route against the field rather than against a static cost table.

7.4 Search and judges as low-density equilibria

Two earlier constructions are special cases. The island search (§4.1) becomes an *anti-crowding* field over schedule-space basins — explorers are pushed toward under-occupied basins — of which the refutation log is the static, infinite-density limit (a killed basin has permanent congestion cost ∞, so no agent ever re-enters). The judge flock (§5.3) is a low-density equilibrium in *error-correlation* space: the diversity premium is exactly a congestion cost that penalizes adding a judge whose blind spots correlate with the panel's, driving the flock toward independent error.

7.5 Honest boundaries

Two caveats are mandatory. First, this is *mean-field control* (MFC), not a true mean-field *game*: we own every agent's policy, so the fleet is a cooperative planner minimizing a shared cost, not a population of selfish players. The genuine-game regime — selfish agents, external model arms, multi-tenant SaaS pricing — is where the Nash rather than the social-optimum solution matters, and we do not claim the equilibrium analysis there. Second, the fleet is *small*: $N \approx 3$–6. Mean-field approximation error scales as $O(1/\sqrt{N})$, so at this scale the equilibrium *guarantees* are asymptotic and not in force. What *is* validated at small N is the *architecture*: interact through a published field, $O(N)$ communication, no pairwise races — which turned the two $O(N^2)$ coordination failures

above into the one $O(N)$ success. The asymptotics are the reason to keep the architecture as the fleet grows, not a claim about today's fleet.

8 Limitations

- **The search is designed, and its first hunt is only now in progress.** Algorithm 1 and its budget (§4.3) are derived from measured constants (the fidelity noise floors, the static-reject experience, the non-additivity and non-transfer of the sample); no completed evolutionary run exists yet, and its result against the hand-search is exactly the controlled experiment that will confirm or refute the loop's value over an expert ([1], velocity section). The ∼15-config manual search is the only empirical schedule-search data in this paper, and it is a convenience sample from one campaign, not a controlled sweep.
- **The compile-cost constant is modeled.** The per-config compile time used in the budget is an order-of-magnitude estimate from build observations, not a logged measurement; the ±0.4%/±0.8% noise floors and the "shaderstats is free" rung are the sourced constants.
- **The LLM operator is unproven here.** We adopt the propose-and-select pattern on external evidence (FunSearch [5], AlphaEvolve [6]); we have not measured an LLM mutation operator against random mutation on our own search.
- **The judge flock is specified, partially anchored.** The four anchor corpora are real and sized (notably the 2,981-event router corpus), but the full flock loop — disagreement escalation, the Goodhart tripwire, multi-cycle reliability re-measurement — is specified, not yet run end-to-end.
- **Mean-field is small-N.** As §7.5 states, the equilibrium guarantees are asymptotic; at N=3–6 only the architecture is validated, by three logged anecdotes (two failures, one success), which is illustration, not proof.

9 Conclusion

The schedule is a value, the space of schedules is searchable, and the campaign proved the search worth automating by showing its winners were non-obvious, non-additive, and non-transferable. We specified a multi-fidelity evolutionary search that spends GPU time only on candidates already past a free static filter and a correctness gate, seeded by a log of what is already known to be dead. We embedded it in a loop that turns campaign transcripts into decision-point evaluations and process supervision, selects training signal through a judge flock anchored on measured ground truth under a Goodhart tripwire, and coordinates the whole agent fleet through a shared field rather than pairwise. The through-line is a single discipline: build the honest instrument first, let it — not intuition, not the reference, not a preference model — decide, and write down every refutation so it is never bought twice. The search and the flock are that discipline made mechanical; the mean-field frame is that discipline made to scale.

References

[1] Hanzo AI Research. *Refutation-Driven Performance Engineering: An Empirical Study of a Multi-Week GPU Kernel Campaign.* hanzo-kernel Paper 08 (companion).

[2] Hanzo AI Research. *The Schedule Is a Value: Comptime-Specialized Autotuning.* hanzo-kernel Paper 07.2. The `Variant`/`Tuned`/`Tuner` algebra of §2.

[3] Tracel AI. *CubeCL: multi-platform GPU compute in Rust.* https://github.com/tracel-ai/cubecl. Comptime monomorphization; the studied `cubecl_runtime::tune`.

[4] G. Gerganov and the `ggml`/`llama.cpp` contributors. *`llama.cpp`.* https://github.com/ggml-org/llama.cpp. The same-box baseline and tile-family reference.

[5] B. Romera-Paredes, M. Barekatain, A. Novikov, et al. *Mathematical discoveries from program search with large language models.* Nature **625**, 468–475 (2024). LLM-proposes / evaluator-selects.

[6] Google DeepMind. *AlphaEvolve: A coding agent for scientific and algorithmic discovery.* Technical report (2025). Evolutionary program search with an LLM mutation operator.

[7] L. Zheng, C. Jia, M. Sun, et al. *Ansor: Generating High-Performance Tensor Programs for Deep Learning.* OSDI (2020). Schedule search over tensor programs.

[8] T. Chen, T. Moreau, Z. Jiang, et al. *TVM: An Automated End-to-End Optimizing Compiler for Deep Learning.* OSDI (2018).

[9] J. Ansel, S. Kamil, K. Veeramachaneni, et al. *OpenTuner: An Extensible Framework for Program Autotuning.* PACT (2014). Ensemble search over program-configuration spaces.

[10] R. C. Whaley and J. Dongarra. *Automatically Tuned Linear Algebra Software (ATLAS).* Supercomputing (1998). Empirical per-machine tuning of a numeric kernel library.

[11] P. Verga, S. Hofstatter, S. Althammer, et al. *Replacing Judges with Juries: Evaluating LLM Generations with a Panel of Diverse Models (PoLL).* Cohere (2024). Small diverse juries beat a single frontier judge on bias and cost.

[12] Y. Bai, S. Kadavath, S. Kundu, et al. *Constitutional AI: Harmlessness from AI Feedback.* Anthropic (2022). Learning from model feedback (RLAIF).

[13] R. Rafailov, A. Sharma, E. Mitchell, et al. *Direct Preference Optimization.* NeurIPS (2023).

[14] L. Zheng, W.-L. Chiang, Y. Sheng, et al. *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena.* NeurIPS (2023). Documents position, verbosity, and self-preference bias in LLM judges.

[15] G. Irving, P. Christiano, and D. Amodei. *AI Safety via Debate.* arXiv:1805.00899 (2018). Escalation of hard items to a debate round.

[16] J.-M. Lasry and P.-L. Lions. *Mean field games.* Japanese Journal of Mathematics **2**, 229–260 (2007).

[17] M. Huang, R. P. Malhamé, and P. E. Caines. *Large-population stochastic dynamic games: closed-loop McKean–Vlasov systems and the Nash certainty equivalence principle.* Communications in Information and Systems **6**(3), 221–252 (2006).

[18] J. G. Wardrop. *Some theoretical aspects of road traffic research.* Proceedings of the Institution of Civil Engineers **1**(3), 325–362 (1952). Equilibrium routing under congestion.