ADAFLEET: AN ADAPTIVE ORCHESTRATOR–SUBAGENT COEVOLUTION FRAMEWORK FOR EFFICIENT QUANTITATIVE SIGNAL DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-improving multi-agent systems must balance task execution against changes to the prompts, policies and parameters that guide their work. In continual discovery, this choice is further complicated by a changing acceptance criterion, as each successful discovery alters which future candidates are admissible. We propose AdaFleet, an adaptive orchestrator–subagent coevolution framework that connects reflective prompt revision, dispatch-rule revision and joint fine-tuning through a shared discovery history. A pre-specified feedback rule triggers prompt revision and fine-tuning, while adoption tests compare proposed changes with incumbents. We instantiate the framework in quantitative signal discovery within an evaluation world whose six admission checks account for standalone quality and the growing pool of accepted signals. Under matched evaluator-call budgets, AdaFleet with a 4B backbone admits 6.2 times as many signals as an adapted AlphaSAGE baseline and 3.7 times as many as our re-implementation of NVIDIA's discovery pipeline. The full framework also achieves higher average admissions than adaptive dispatch across five seeds. These findings support coordinated adaptation for continual discovery under changing acceptance constraints.

1 INTRODUCTION

An agent that searches for useful outputs eventually competes with its own earlier successes. In quantitative signal discovery, a candidate can pass every standalone quality check and still fail because its daily returns track a signal that the pool already holds. Each admission changes the value of subsequent proposals. Continued discovery therefore couples two tasks: finding strong candidates and directing search toward what the growing pool still needs.

Signal-discovery systems use experience memory, pool-dependent novelty rewards and evaluator feedback to guide search (Wang et al., 2026; Chen et al., 2026; Huo et al., 2026). Complementary approaches to self-improvement revise prompts, exploration programs or model parameters (Section 2). AdaFleet connects these updates through a common discovery history that supports role-specific learning and evaluation of proposed changes.

AdaFleet runs two roles on one frozen language-model backbone (Figure 1). An orchestrator assigns signal families and refinement tasks to subagents; each role uses its own adapter. Every proposal, check outcome and adaptation result enters the history. Three mechanisms use it to improve subsequent search: reflective revision of the generation prompt, revision of the dispatch rule, and filtered supervised fine-tuning of both adapters. The subagents learn from candidate expressions, while the orchestrator learns from evaluated rule proposals. A fixed state rule requests prompt revision or fine-tuning when admissions stall relative to the run's own rate or the pool blocks a rising share of quality-check passers. Each proposed update faces its incumbent in a paired comparison on the current pool before adoption; a rejected update leaves the incumbent in place and its cost on the budget.

Under identical checks and 3,000 charged evaluator calls per run, averaged over three seeds, AdaFleet on a 4B backbone admits 3.7 times as many factors as NVIDIA's pipeline re-implemented on the same backbone, 6.2 times as many as AlphaSAGE and 9.5 times as many as FactorMiner;

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.

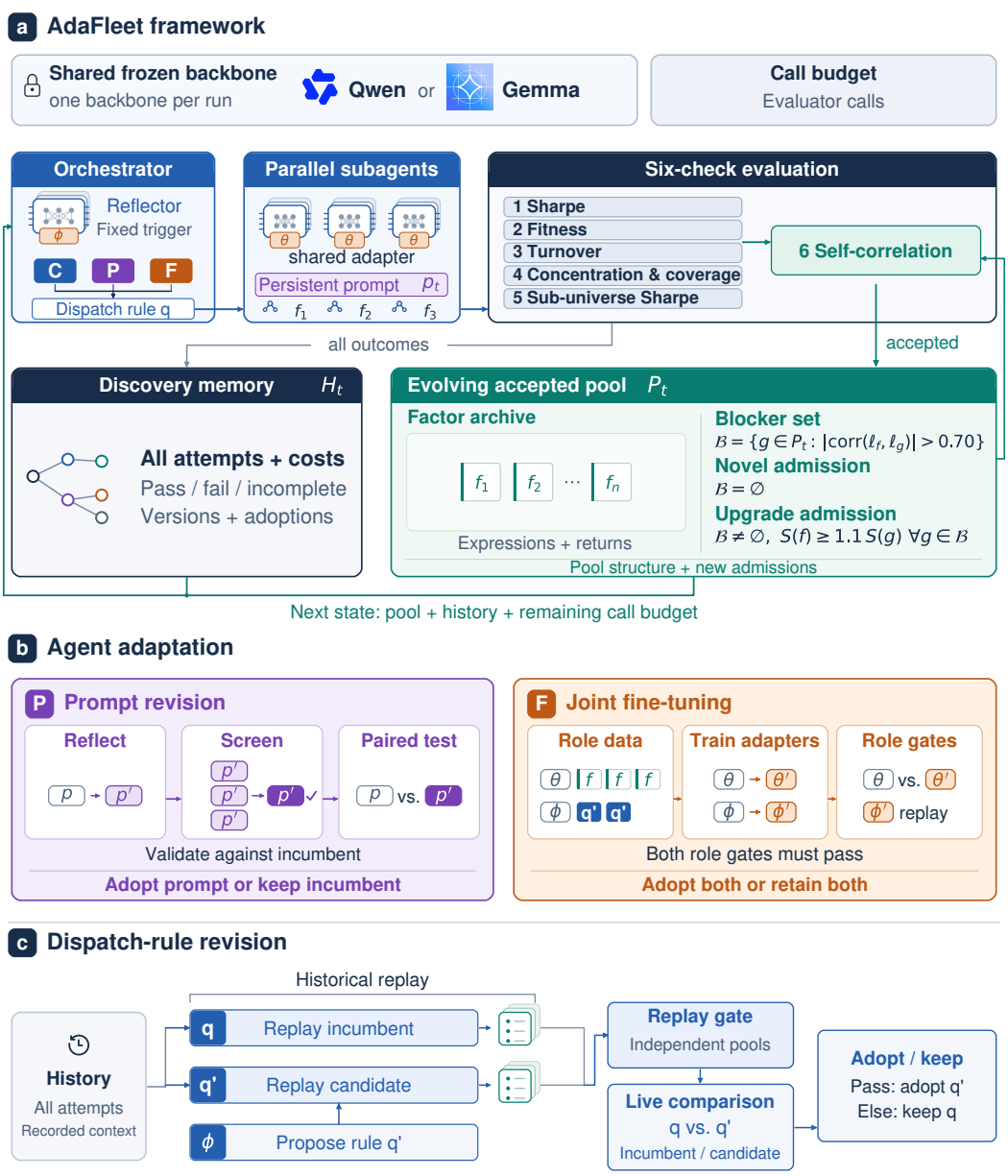

Figure 1: AdaFleet. (a) Discovery loop: dispatch rule q, subagents, six checks, history H, pool P. (b) Prompt revision and joint fine-tuning. (c) Revision of q by replay and live trial.

it admits 7.5 times as many as fixed prompts. It leads each baseline in every seed, and NVIDIA's pipeline admits no more per call on the 30B-parameter Nemotron model of its shipped configuration. Against adaptive dispatch alone, the full configuration admits 16% more factors over five seeds, leading in three seeds overall and in the final 500 charged calls of all five. Adaptive dispatch transfers to Qwen3-8B, Qwen3.8-27B and Gemma 4 E4B, a second model family, admitting more per charged call than every baseline in these single-seed pairs.

The paper makes three contributions.

- We formulate continual signal discovery with pool-dependent admission and build an evaluation world on US equity data whose six checks include the correlation of each candidate with every accepted signal (Section 3).

- We propose AdaFleet, which coevolves the orchestrator's dispatch rule, the subagents' prompt and both roles' adapters through three guarded update mechanisms that share one discovery history and a pre-specified trigger (Section 4).
- We demonstrate higher discovery yield than adapted published pipelines under shared checks and evaluator-call budgets, and characterize dispatch and adaptation through component studies, timing controls and four backbones (Sections 5 and 6).

2 Related work

Formulaic alpha research supplies explicit signal libraries (Kakushadze, 2016) and evaluates novelty through correlation to existing alphas (Dang & Bui, 2019). Pool-aware factor search treats the accepted pool as part of the objective: AlphaGen optimizes factors for their contribution to a combined model (Yu et al., 2023), and AlphaSAGE trains a GFlowNet with a novelty reward against a changing library (Chen et al., 2026). Agent systems add memory and evaluator feedback around a frozen model, as in FactorMiner (Wang et al., 2026) and NVIDIA's signal-discovery pipeline (NVIDIA, 2026b; Huo et al., 2026), or train the generator by reinforcement learning with a novelty reward, as in AlphaAgentEvo (Tang et al., 2026); Alpha-GPT supports interactive alpha mining (Wang et al., 2025), and R&D-Agent-Quant jointly optimizes factors and models with a bandit scheduler (Li et al., 2025). Each keeps a fixed task allocation; Section 6 compares AlphaSAGE, FactorMiner and NVIDIA's pipeline under identical checks and budgets.

Self-Refine and Reflexion use generated feedback to revise outputs or guide later trials (Madaan et al., 2023; Shinn et al., 2023); OPRO and DSPy optimize prompts or language-model pipelines (Yang et al., 2024; Khattab et al., 2024). GEPA evolves prompts by reflective mutation and selection on execution feedback (Agrawal et al., 2026), and Dream-RSI scores candidate exploration policies on recorded discovery trees (Zheng et al., 2026). AdaFleet applies both to a criterion that moves with the pool (Section 4.3).

Parameter adaptation closes the loop on weights. Rejection-sampling fine-tuning learns from filtered generated examples (Yuan et al., 2023), SEAL lets the model write its own update data (Zweiger et al., 2025), and INSPO and MATPO use reinforcement learning to co-evolve an instruction population with the policy (Zhou et al., 2025) and to train planner and worker roles within one model (Mo et al., 2025). BetterTogether alternates prompt optimization with weight fine-tuning of one language-model program (Soylu et al., 2024). AdaFleet's joint fine-tuning is filtered supervised fine-tuning of two role adapters on one backbone; the orchestrator's examples are weighted by measured rule-amendment advantage, and the pair is promoted or rolled back together after each role passes its adoption tests (Section 4.3).

3 Problem and evaluation world

Published systems accept or reward each candidate on its mining period, mostly by its information coefficient (IC). NVIDIA's pipeline accepts a signal at $|\mathrm{IC}| \geq 0.02$ and $p \leq 0.05$ (NVIDIA, 2026c). FactorMiner holds every method to one rule, by default $\mathrm{IC} \geq 0.04$ and signal correlation below 0.5 with its library, lets a candidate whose IC is at least 0.10 and 1.3 times a lone blocker's IC replace that blocker, and compares equal-size sets topped up by IC with no stated candidate budget (Wang et al., 2026). AlphaSAGE rewards intrinsic quality and novelty, rather than a pool-marginal gain, which it describes as non-stationary, anneals the novelty weight against at most 50 library alphas toward zero, and runs baselines at repository defaults (Chen et al., 2026). AlphaAgentEvo matches tool-call budgets but counts success as an evolved alpha beating its seed (Tang et al., 2026). The evaluation world holds every system to identical checks, one charged-call budget and an admission rule that tightens with every acceptance, and scores the accepted pool against same-size random pools on reserved years, a test none of the four reports.

Discovery fills an append-only pool P of accepted signals, called factors once admitted. A candidate f, an expression in a fixed domain-specific language (DSL) with its simulation settings, enters P only if it passes five quality checks, on the Sharpe ratio $S(f)$, Fitness, turnover, weight concentration with coverage, and Sharpe on a smaller liquid universe, and a sixth check on pool correlation. An accepted record whose daily profit-and-loss (PnL) increments have absolute correlation above 0.70

with those of f is a blocker, and f passes only if $S(f)$ exceeds every blocker's Sharpe by at least 10%. Admissions without blockers are Novel; those with blockers are Upgrades. A settled candidate earns the discovery utility

$$r(f;P)=\begin{cases}1+\min\{S(f)-1.25,\,1\}, & \text{Novel},\\ \min\{S(f)-\max_{g\in\mathcal{B}}S(g),\,1\}, & \text{Upgrade},\\ 0, & \text{confirmed failure.}\end{cases} \tag{1}$$

where $\mathcal{B}(f,P)$ is the blocker set and confirmed failures are rejections, duplicates and invalid expressions; missing required evidence yields INCOMPLETE, counted separately. A run with a budget B of charged evaluator calls seeks to maximize admissions and the observed utility $U_{\text{obs}}(B)=\sum_{i\in\mathcal{I}_{\text{complete}}(B)}r(f_i;P_{i^-})$ over its complete evaluations, with P_{i^-} the pool at settlement. Each admission adds a correlation constraint and never removes one, so the admissible set is non-increasing in P and any fixed policy faces a moving target. Appendix A gives the check definitions and thresholds.

Data and admission rules stay fixed as P grows. CRSP prices and returns, Compustat accounts, I/B/E/S estimates and OptionMetrics volatility enter through Wharton Research Data Services (WRDS) (WRDS, 2026b). Ordinary shares enter at a prior 20-day dollar-volume rank up to 3,000, stay eligible through rank 3,300 and clear a USD 3 million liquidity floor, about 2,720 names per day in 2019–2023. Portfolios rebalance daily with a one-day signal delay and sub-industry neutralization; a candidate that fails the default screen above fixed Sharpe and Fitness floors is re-simulated over decay and neutralization settings that remain attached to its record. Runs score 2019–2023; 2024 and 2025 serve validation and later reporting (Section 5). WorldQuant BRAIN (WorldQuant, 2022) motivates the simulation and admission conventions. Admission and the main tables use gross returns (Appendix B). At default settings, six selected probes matched the platform's admission decisions, five rejections and one admission, with mean absolute Sharpe and Fitness differences of 0.170 and 0.175; one- and two-day reversal signals still disagree.

4 AdaFleet

4.1 Roles and persistent state

AdaFleet runs one frozen language-model backbone with two low-rank adapters (Hu et al., 2022): ϕ for the orchestrator, which also serves as the reflection model for prompt revision, and a shared θ for the subagents. The orchestrator owns a dispatch rule q that turns the history into blocks of tasks. Subagents answer each task with a DSL expression under the generation prompt $\mathbf{p}$, the evaluator applies the six checks against the current pool, and settlement appends admissions to P in a fixed order. The history H records every attempted candidate with its task context, check outcomes, versions in force, costs and adaptation outcomes.

The persistent state is $(P,H,\mathbf{p},q,\phi,\theta)$ and the remaining charged-call budget. Three mechanisms change it beyond the pool: prompt revision (P) rewrites $\mathbf{p}$, dispatch-rule revision amends q, and joint fine-tuning (F) replaces (ϕ,θ); between updates the run continues search (C) under the current state. Evaluator thresholds and market-data access stay outside all three interfaces. Figure 1 shows the loop and the mechanisms, Algorithm 1 their order within a run and Section 4.3 their adoption evidence.

4.2 Pool-conditioned dispatch

The dispatch rule plans each search block from the history H instead of a fixed batch of random family hints. Every settled proposal in H is typed as admitted, blocked by the pool, near miss, far rejection, illegal or incomplete and attributed to its family and task; a near miss fails at most one quality check or comes close to the Sharpe bar. From these counts, recent admissions and the remaining budget, the rule sets three quantities. The block size shrinks after a window that admitted nothing and mostly repeated earlier expressions, and grows back after admissions. Exploration tasks go to families by an upper-confidence estimate of each family's recent shaped score, a bounded Sharpe-scaled score per proposal, and its admissions; weak or duplicate-prone families receive fewer tasks but are never removed. The refinement share returns near misses to subagents as parents to

Algorithm 1 AdaFleet: discovery and adaptation

Require: adapters (ϕ, θ) on a frozen backbone, prompt $\mathbf{p}$, rule q, empty P and H, budget B
while charged calls remain **do**
  q plans a block of exploration, refinement and repair tasks from H, P and the remaining budget
  subagents generate expressions with $(\theta, \mathbf{p})$; the evaluator runs the six checks against P
  settle admissions into P in order; append every outcome to H; charge the calls to B
  advance a running live rule trial; at its end adopt the challenger q' only if it wins
  if a rule round is due and no trial is running **then**
    ϕ proposes q'; screen by replay on H; compare a survivor with the incumbent in live search
  end if
  if the trigger fires at a check point, or the F floor is reached **then**
    next feasible P: reflect on H; adopt $\mathbf{p}'$ only if it passes the paired test on a frozen P
    next feasible F: train (ϕ', θ') by Equation 2; adopt both only if both adoption tests pass
  end if
end while

edit; a candidate blocked by the pool rotates its family and receives a decorrelation edit. Because the outcome types depend on P, the same rule allocates effort differently as the pool grows. Appendix C gives the constants.

4.3 Updating prompts, dispatch rules and role adapters

Each mechanism constructs a challenger from the discovery history and evaluates it against the incumbent. Prompt and subagent-adapter tests use paired tasks on frozen copies of the current pool. They require nonnegative discovery-utility gain (Equation 1) and bounded regressions in legal rate and expression-skeleton entropy. Orchestrator-adapter tests assess rule parseability and replay score; dispatch-rule revisions combine replay with live search. Evaluator calls count toward the run's budget; validation on frozen pool copies leaves P unchanged. Each component is replaced only when its adoption tests pass; otherwise the incumbent continues.

Reflective prompt revision. In P, the reflection model reads from H which tasks produced illegal, blocked or near-miss expressions under the incumbent prompt and why, and proposes revisions within a bounded search. Candidates are scored on validation tasks by a dense screening score, with bounded credit for useful near misses and none for duplicates or pool copies. The selected candidate then meets the incumbent on disjoint holdout tasks in deployment form, the task text and prompt layout used during search. The paired test requires nonnegative gains in dense score and discovery utility, and a win rate of at least 0.5.

Dispatch-rule revision. The orchestrator proposes amendments to q through a bounded interface for block size, task shares and family weighting, with evaluator and trigger rules held fixed. Replay screens each amendment on recorded candidates in H, reusing evaluator outputs and recomputing pool-dependent settlement where context changes. A surviving challenger then guides new proposals in a live comparison with the incumbent and is adopted if it wins. The amendments, replay scores and live outcomes become training examples for the orchestrator; Appendix C gives the round schedule, comparison order and block counts.

Joint role fine-tuning. F trains challenger adapters (ϕ', θ') by filtered supervised fine-tuning on the experience recorded before the request. The subagent set D_θ holds admitted expressions and near misses that pass the five quality checks and are blocked only by pool correlation, each with its recorded task turn; deduplication removes repeated outputs, sign copies and blocked variants of admitted skeletons. The orchestrator set D_ϕ holds rule amendments with their measured live or replay advantages, which set the weights a. On the frozen sets,

$$\begin{aligned} \mathcal{L}_\theta &= -\sum_{(x,y)\in D_\theta} \frac{1}{|y|} \log \pi_\theta(y \mid x, \mathbf{p}), \\ \mathcal{L}_\phi &= -\sum_{(h,e,a)\in D_\phi} \frac{a}{|e|} \log \mu_\phi(e \mid h), \end{aligned} \tag{2}$$

where y is an expression with task context x, e is a rule proposal with context h, and $|y|$ and $|e|$ count target tokens. The subagent challenger must pass the paired test with an added bound on

reference-score degradation, the orchestrator challenger a parseability and replay-score check, and joint adoption promotes or rolls back both adapters together. With too few orchestrator examples, the attempt may use a subagent-only fallback, recorded as its own update type.

4.4 When updates are requested

A state rule fixed before any run decides when P and F are requested. Every 50 main-search records after a warm-up, it applies two tests to H. The admission-gap test fires when the records since the last admission exceed the geometric threshold at level 0.05 under the run's own admission rate. The crowding test fires when the share of five-check passers blocked by the pool in the recent window exceeds its earlier baseline in a one-sided binomial test at level 0.05.

Requests alternate between P and F, starting with P, with at most three attempts of each type per run; a waiting period follows every request and lengthens when repeated requests yield no admission. A floor requests one F when none has executed by 75% of the charged-call budget. Appendix C gives the constants and the timing controls.

5 Experimental setup

Qwen3-4B-Instruct-2507 (Yang et al., 2025) is the frozen backbone of the seeded comparison; every run starts from an empty pool and an empty history. Fixed prompts draw random family hints without feedback. Memory with deduplication retrieves earlier experience and resamples repeats. Adaptive dispatch changes block size, task families and refinement share from observed outcomes (Section 4.2). Adaptive dispatch with prompt revision adds P under the state rule; the full configuration adds dispatch-rule revision and F. Two removals drop P (the full configuration without P) or restrict F to the subagent adapter (subagent-only fine-tuning). Two timing controls replace the state rule by a fixed schedule, requests at 20, 40, 60 and 80% of the budget, or by random timing, four points drawn before the runs. Qwen3-8B, Qwen3.8-27B and Gemma 4 E4B, a second model family, add adaptive-dispatch and full-configuration pairs.

NVIDIA's signal-discovery pipeline (NVIDIA, 2026b; Huo et al., 2026) enters through three routes: its generate, evaluate and advise loop re-implemented on Qwen3-4B with five signals per round, our DSL parser replacing its code agent; the shipped configuration (NVIDIA, 2026c) on Nemotron-3.5-Lightning-30B-A3B, thinking disabled, two signals per round, three iterations; and its original code and agents on those weights. AlphaSAGE keeps its GFlowNet generator (Chen et al., 2026) with our discovery utility as its quality term and no language model. FactorMiner (Wang et al., 2026), rebuilt from the paper without public code, keeps its experience memory with consolidation and retrieval, batch generation with within-batch deduplication and incumbent replacement on frozen Qwen3-4B; our six checks decide admission and replacement. Appendix C lists every adaptation.

Each run has a budget of 3,000 charged evaluator calls, 2,000 for the Qwen3.8-27B pair. One call is charged per proposal, and the validation calls of P and F attempts are charged whether or not the attempt is adopted; generation, training and replay computation are recorded separately (Appendix C.5). One full-budget run per arm and seed is reported; seeds 3 and 4 add to adaptive dispatch and the full configuration. Admissions are counted per 1,000 charged calls, repeats and illegal proposals included, and per 1,000 distinct legal proposals; comparisons are descriptive and paired by seed.

The out-of-sample (OOS) percentile ranks the admitted pool's equal-weight Sharpe in 2024 and 2025, at frozen in-sample settings, among 1,000 same-size random pools drawn from 200 distinct legal proposals of the same run, signs as proposed, as the share of random pools with strictly lower Sharpe (Appendix D.3). The full configuration counts as better than adaptive dispatch only if it admits more factors per 1,000 charged calls in each of seeds 0–2 and its mean percentile over these seeds is at least 50 in both years. Both conditions were fixed before any Qwen3-4B full-configuration or adaptive-dispatch run of the run matrix had completed; the Qwen3.8-27B pair had already been observed. Appendix D.2 gives the timeline, prior inputs and later execution details.

Table 1: AdaFleet, adapted published systems and fixed prompts under identical checks and 3,000 charged calls per run; seed means. Ratio: the full configuration's mean rate divided by the row's.

			Admissions per 1,000			Final 500	OOS percentile		
System	Backbone	Seed	calls	min–max	distinct legal	calls	2024	2025	Ratio
AdaFleet, full configuration	Qwen3-4B	0–2	15.9	9.7–22.7	20.5	8.3	44	100	1.0
AdaFleet, adaptive dispatch	Qwen3-4B	0–2	12.7	12.0–13.3	13.7	3.3	57	74	1.3
NVIDIA re-implementation	Qwen3-4B	0–2	4.3	4.0–4.7	11.2	1.3	42	87	3.7
NVIDIA shipped configuration	Nemotron	0	4.3	4.3	8.9	2.0	64	73	3.7
NVIDIA original code	Nemotron	0	1.3	1.3	2.6	0.0	100	100	11.9
AlphaSAGE	GFlowNet	0–2	2.6	1.7–3.3	3.1	0.3	99	99	6.2
FactorMiner	Qwen3-4B	0–2	1.7	0.7–3.7	7.3	0.0	63	65	9.5
Fixed prompts	Qwen3-4B	0–2	2.1	1.7–2.3	6.8	0.0	43	63	7.5

Table 2: AdaFleet configurations and timing controls, seeds 0–2; final 500 and OOS are seed means.

	Admissions per 1,000 calls				Final 500	OOS Sharpe (percentile)	
Configuration	Seed 0	1	2	Mean	calls	2024	2025
Fixed prompts	1.7	2.3	2.3	2.1	0.0	1.09 (43)	0.61 (63)
Memory with deduplication	8.0	15.7	6.3	10.0	5.0	1.32 (22)	0.86 (62)
Adaptive dispatch	12.0	13.3	12.7	12.7	3.3	2.09 (57)	0.83 (74)
Adaptive dispatch with prompt revision	12.0	9.7	9.7	10.4	4.7	1.64 (34)	1.12 (73)
Full configuration	22.7	15.3	9.7	15.9	8.3	2.13 (44)	1.16 (100)
Full configuration, fixed schedule	14.7	9.0	14.7	12.8	7.0	1.71 (22)	0.41 (86)
Full configuration, random timing	15.3	21.0	7.7	14.7	6.0	2.31 (37)	1.08 (91)

6 Results

6.1 Same-world comparison with published pipelines

AdaFleet admits more factors than fixed prompts and all three adapted published pipelines in every seed under the same six checks and 3,000 charged calls (Table 1; verification in Appendix D.1). The full configuration's mean rate is 15.9 admissions per 1,000 calls, compared with 12.7 for adaptive dispatch. On NVIDIA's shipped Nemotron-3.5-Lightning-30B-A3B (30B total, 3B active parameters) (NVIDIA, 2026a), the re-implementation admits 13 factors at seed 0, the same 4.3 per 1,000 calls as on Qwen3-4B, and NVIDIA's original code, on its price and volume fields, admits 4, the last at call 935. Repeated proposals, charged and answered from the cache, account for about 68% of fixed-prompt calls, 52% of NVIDIA re-implementation calls and 71% to 87% of FactorMiner calls; per 1,000 distinct legal proposals, the full configuration still leads, admitting 20.5 factors against FactorMiner's 7.3 over seeds 0–2.

6.2 Where the gain comes from

Memory with deduplication lifts fixed prompts from 2.1 to 10.0 admissions per 1,000 calls and adaptive dispatch reaches 12.7, most of the gain (Table 2); adding P alone yields 10.4; all seven prompt attempts were rejected and left their validation cost. The full configuration reaches 15.9 with 496 to 624 calls of each run spent on prompt and adapter validation (496 to 800 over the five seeds), and per distinct legal proposal it leads adaptive dispatch in seeds 0 and 1 and trails in seed 2 (Appendix D.1). The seed-0 removals yield 52 admissions without P and 66 with subagent-only fine-tuning, against 68 for the full configuration (Table 3); the configuration without P adopted one of its two F updates, as a joint update, and subagent-only fine-tuning adopted its one P attempt and rejected its one F attempt.

6.3 Discovery under a growing pool

In Figure 2a, adaptive dispatch leads early, with 21.3 mean cumulative admissions at call 1,000 against 16.0 for the full configuration, whose prompt attempts begin in this phase; from call 2,000 the order reverses, 33.7 against 30.0, and the gap widens to the end of the budget. Over five seeds the full configuration admits 45.4 factors per run against 39.0 for adaptive dispatch, 16% more. The per-seed counts are 68 versus 36, 46 versus 40, 29 versus 38, 36 versus 49 and 48 versus 32, a lead

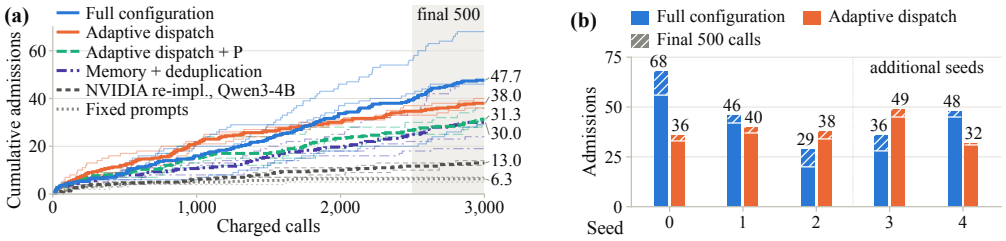

Figure 2: (a) Cumulative admissions against charged calls, seeds 0–2 and mean. (b) Admissions by seed, full configuration versus adaptive dispatch. Final 500 calls shaded and hatched.

Table 3: Seed 0 runs; 3,000 charged calls each, 2,000 for the Qwen3.8-27B pair.

	Admissions			OOS Sharpe (percentile)	
Configuration	Total	Per 1,000 charged calls	Per 1,000 distinct legal proposals	2024	2025
Subagent-only fine-tuning, Qwen3-4B	66	22.0	27.4	2.88 (2)	0.72 (96)
Full configuration without P, Qwen3-4B	52	17.3	20.6	2.08 (83)	0.67 (99)
Qwen3-8B, adaptive dispatch	33	11.0	13.2	1.27 (100)	1.02 (93)
Qwen3-8B, full configuration	31	10.3	14.9	1.04 (69)	0.38 (89)
Qwen3.8-27B, adaptive dispatch	20	10.0	10.1	0.86 (34)	0.60 (91)
Qwen3.8-27B, full configuration	17	8.5	9.3	1.23 (62)	0.55 (64)
Gemma 4 E4B, adaptive dispatch	24	8.0	8.3	1.98 (100)	0.73 (90)
Gemma 4 E4B, full configuration	21	7.0	10.9	1.91 (84)	1.18 (93)

in seeds 0, 1 and 4 (Figure 2b). Seeds 3 and 4, outside Table 2, have adaptive dispatch at 16.3 and 10.7 admissions per 1,000 calls and the full configuration at 12.0 and 16.0. Novel admissions alone give the same winner in every seed.

The full configuration leads in the final 500 charged calls of all five seeds, 36 admissions against 15 in total, and retains 95% of its run-average admission rate in this window against 46% for adaptive dispatch; its final-500 mean over seeds 0–2 is 8.3 admissions, and no baseline row of Table 1 exceeds 2.0. Over the final 1,000 calls it leads in three of five seeds.

6.4 When and what to adapt

In the five state-rule runs of the full configuration, six P attempts yielded zero adoptions and seven F attempts three joint adoptions, finishing at calls 2,464, 2,472 and 2,984; four of the F requests came from the budget floor (per seed, executed/adopted: P 1/0, 1/0, 1/0, 2/0 and 1/0; F 1/0, 2/1, 1/1, 2/1 and 1/0, every F attempt joint). Dispatch-rule revisions were adopted in each of these runs, 12 of 40 live trials in total. The adoption tests reject most challengers, and every rejected attempt keeps its cost. Seeds 0 and 4 achieved their final-500 lead with revised dispatch rules and unchanged prompts and adapters; seeds 1–3 also adopted a joint update after call 2,400.

The state rule achieves the highest mean admission rate among the three request schedules: 15.9 per 1,000 calls over seeds 0–2, against 12.8 for the fixed schedule and 14.7 for random timing (Table 2). Both controls executed two P and two F attempts in every seed, more than the state rule, so the comparison combines timing and attempt-count differences. The fixed schedule adopted one of its two P attempts in every seed and 2, 1 and 0 of its F attempts (joint 1/1, 1/1 and 2/0). Random timing adopted 0, 1 and 1 P attempts and 0, 1 and 0 F attempts (joint 1/0, 1/1 and 1/0). Across both controls, 5 of 12 prompts and 4 of 12 adapter updates were adopted, three of them joint, and every adopted prompt came from the first P request of its run.

6.5 Out-of-sample pool quality

The full configuration's admitted pools rank at percentiles 100, 100 and 99.6 among same-run random pools in 2025, and at 31, 72 and 29 in 2024, mean 44. The pre-specified criterion (Section 5) requires a mean percentile of at least 50, met in 2025 and missed in 2024; its admission requirement is met in seeds 0 and 1 and missed in seed 2, so the combined criterion is not met.

In 2025 every percentile in Tables 1 and 2 exceeds 60 and the full configuration is the only AdaFleet configuration at 100; in 2024 adaptive dispatch, at 57, is the only AdaFleet configuration above 50 (Table 3 gives the seed-0 runs). The smallest pools rank highest: AlphaSAGE's pools of 5 to 10 price–volume factors and the 4 factors of NVIDIA's original code reach the 98th percentile or higher in every seed and year, and the 2-factor pools of FactorMiner in seeds 0 and 2 have wide random-pool bands and coarse percentiles.

6.6 Backbones from 4B to 27B parameters in two model families

At seed 0, adaptive dispatch admits 33 factors on Qwen3-8B, 20 on Qwen3.8-27B and 24 on Gemma 4 E4B, against 31, 17 and 21 for the full configuration (Table 3); its 11.0, 10.0 and 8.0 admissions per 1,000 calls exceed the highest baseline mean rate, 4.3 for the NVIDIA re-implementation (Table 1). The full configuration spends 160 to 920 calls on validation and leads per distinct legal proposal on Qwen3-8B and Gemma 4 E4B and trails on Qwen3.8-27B. Both P attempts on Qwen3-8B and both joint F attempts on Gemma 4 E4B were adopted; the Qwen3.8-27B F attempt was rejected. At these budgets and one seed each, the gain from the update mechanisms on these backbones is at most equal to their validation cost; on Qwen3-4B the lead appears in the final 500 calls.

7 Limitations and conclusion

The evaluation covers three to five seeds per core arm and one per component removal and additional backbone pair. Efficiency is measured by charged evaluator calls, with other computation recorded separately (Appendix C.5); admission and portfolio results use gross returns. Both later years were inspected before the criterion was fixed. In 2024 the full configuration's admitted pools average the 44th percentile of same-run random pools, so the pre-specified criterion is not met. Gross returns establish no profitability after trading costs or execution delay. On Qwen3-4B, no prompt revision was adopted under the state rule, and every adopted joint update finished after charged call 2,400. The evaluation world retains missing-return and data-vintage assumptions, and short-horizon reversal signals disagree with the external platform.

AdaFleet connects task allocation, prompt revision and role-specific learning through a shared discovery history. Under identical checks and budgets, the full configuration on the 4B backbone admits 3.7 to 9.5 times as many factors as the three adapted pipelines over seeds 0–2. Pool-conditioned dispatch supplies most of that gain; the update mechanisms add 16% over five seeds, and their lead is concentrated in the final 500 calls, after their validation calls have been paid. Whether this ordering holds at larger budgets, on more seeds of the larger backbones and under trading costs remains open.

AI use statement

Generative AI tools were used for literature exploration, manuscript drafting and revision, and textual consistency checks. The authors remain responsible for verifying the final text, references, and scientific claims.

Ethics statement

This work uses historical market and company data accessed through WRDS and does not recruit human participants. Access and redistribution are subject to the source datasets' licenses. The evaluation studies automated factor discovery with gross backtests; application to live trading is outside the reported evaluation.

Reproducibility statement

Section 4 specifies the implemented roles and update mechanisms. Appendices A–C give admission, world-construction and configuration details. Appendix D describes verification from proposal, admission and adaptation records and the out-of-sample evaluation, including run selection, prior observations and evaluation coverage. An anonymous interactive companion page, hosted without visitor tracking, is at https://elegant-strudel-53c0bc.netlify.app.

REFERENCES

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, pp. 8479–8565, 2026.

Binqi Chen, Hongjun Ding, Ning Shen, Taian Guo, Jinsheng Huang, Luchen Liu, and Ming Zhang. AlphaSAGE: Structure-aware alpha mining via GFlowNets for robust exploration. In *International Conference on Learning Representations*, pp. 2955–2982, 2026.

CRSP. CRSP US stock databases. Database documentation, 2026. URL https://indexes.morningstar.com/research-data-products/crsp-us-stock-databases. Accessed September 25, 2026.

Chinh Dang and Crispin Bui. Alpha correlation. In Igor Tulchinsky (ed.), *Finding Alphas: A Quantitative Approach to Building Trading Strategies*, chapter 8, pp. 61–68. Wiley, second edition, 2019. doi: 10.1002/9781119571278.ch8.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Peihan Huo, Yang Shen, Hanyue He, and Ioana Boier. Automating and optimizing financial signal discovery with multi-agent systems. NVIDIA Technical Blog, May 21, 2026, https://developer.nvidia.com/blog/automating-and-optimizing-financial-signal-discovery-with-multi-agent-systems/, 2026.

Zura Kakushadze. 101 formulaic alphas. *Wilmott*, 2016(84):72–81, 2016. doi: 10.1002/wilm.10525.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *International Conference on Learning Representations*, pp. 54928–54958, 2024. URL https://openreview.net/forum?id=sY5N0zY5Od.

Yuante Li, Xu Yang, Xiao Yang, Xisen Wang, Weiqing Liu, and Jiang Bian. R&D-Agent-Quant: A multi-agent framework for data-centric factors and model joint optimization. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3967.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023. doi: 10.52202/075280-2019.

Zhanfeng Mo, Xingxuan Li, Yuntao Chen, and Lidong Bing. Multi-agent tool-integrated policy optimization. *arXiv preprint arXiv:2510.04678*, 2025.

MSCI. The global industry classification standard (GICS). Index methodology, 2026. URL https://www.msci.com/indexes/index-resources/gics. Accessed September 25, 2026.

NVIDIA. NVIDIA-Nemotron-3.5-Lightning-30B-A3B-BF16. Hugging Face model card, https://huggingface.co/nvidia/NVIDIA-Nemotron-3.5-Lightning-30B-A3B-BF16, 2026a. Released August 11, 2026. Accessed September 25, 2026.

NVIDIA. Quantitative signal discovery agent. NVIDIA AI Blueprint, https://build.nvidia.com/nvidia/quantitative-signal-discovery-agent, 2026b. Accessed September 25, 2026.

NVIDIA. Quantitative signal discovery agent: blueprint source repository. GitHub, https://github.com/NVIDIA-AI-Blueprints/quantitative-signal-discovery-agent, 2026c. Commit 9a89d24 (August 25, 2026), Apache-2.0 license. Accessed September 25, 2026.

OptionMetrics. IvyDB US. Database documentation, 2026. URL https://optionmetrics.com/united-states/. Accessed September 25, 2026.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652, 2023. doi: 10.52202/075280-0377.

Dilara Soylu, Christopher Potts, and Omar Khattab. Fine-tuning and prompt optimization: Two great steps that work better together. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 10696–10710. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.597.

S&P Global Market Intelligence. Compustat financials. Database documentation, 2026. URL https://www.marketplace.spglobal.com/en/datasets/compustat-financials-(8). Accessed September 25, 2026.

Ziyi Tang, Xuexiong Yin, Weixing Chen, Zechuan Chen, Yongsen Zheng, Wenxuan Ye, Keze Wang, and Liang Lin. AlphaAgentEvo: Evolution-oriented alpha mining via self-evolving agentic reinforcement learning. In *International Conference on Learning Representations*, pp. 71206–71223, 2026.

Saizhuo Wang, Hang Yuan, Leon Zhou, Lionel Ni, Heung-Yeung Shum, and Jian Guo. AlphaGPT: Human-AI interactive alpha mining for quantitative investment. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 196–206. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-demos.14.

Yanlong Wang, Jian Xu, Hongkang Zhang, Shao-Lun Huang, Danny Dongning Sun, and Xiao-Ping Zhang. FactorMiner: A self-evolving agent with skills and experience memory for financial alpha discovery. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '26)*, pp. 12327–12338, 2026. doi: 10.1145/3770855.3818978.

WorldQuant. WorldQuant launches BRAIN platform and inaugural global alphathon competition, August 2022. URL https://www.worldquant.com/ideas/worldquant-launches-brain-platform-and-inaugural-global-alphathon-competition/. Published August 2, 2022. Accessed September 25, 2026.

WorldQuant. 104 – alpha examples by data category: Part 1. WorldQuant BRAIN, Learn to Quant, 2026. URL https://worldquantbrain.com/alpha-examples. Accessed September 25, 2026.

WRDS. I/B/E/S overview. Database documentation, Wharton Research Data Services, 2026a. URL https://wrds-www.wharton.upenn.edu/pages/grid-items/overview-ibes-demo/. Accessed September 25, 2026.

WRDS. What is WRDS? Wharton Research Data Services, 2026b. URL https://wrds-www.wharton.upenn.edu/pages/about/what-wrds/. Accessed September 25, 2026.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *International Conference on Learning Representations*, pp. 12028–12068, 2024. URL https://openreview.net/forum?id=Bb4VGOWELI.

Shuo Yu, Hongyan Xue, Xiang Ao, Feiyang Pan, Jia He, Dandan Tu, and Qing He. Generating synergistic formulaic alpha collections via reinforcement learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 5476–5486, 2023. doi: 10.1145/3580305.3599831.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023. Version 2.

Tong Zheng, Xidong Wu, Zheng Zhang, Zhankui He, Chaoyi Zhang, Benjamin Coleman, Ruoqiao Wei, Di Bai, Haolin Liu, Rui Liu, Xue Wang, Yue Zhuan, Wang-Cheng Kang, Renkai Xiang, Heng Huang, Xinwu Cheng, and Yunsong Guo. Dream-RSI: Recursive self-improvement through evolving worlds. *arXiv preprint arXiv:2609.14858*, 2026. Version 1.

Han Zhou, Xingchen Wan, Ivan Vulić, and Anna Korhonen. Agentic policy optimization via instruction-policy co-evolution. *arXiv preprint arXiv:2512.01945*, 2025.

Adam Zweiger, Jyothish Pari, Han Guo, Yoon Kim, and Pulkit Agrawal. Self-adapting language models. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.522 02/085713-2483.

A Admission statistics and protocol details

This appendix defines the statistics, thresholds and settlement order behind the six checks of Section 3. Let $x_{f,t}$ be the daily gross return of factor f per unit of return capital, ν the annualization count, $V^{\mathrm{trade}}_{f,t}$ the absolute rebalancing trade amount on day t, $B^{\mathrm{book}}_{f,t}$ the gross book size, and n the number of days with a valid return. The admission profile uses gross profit-and-loss (PnL) and return capital equal to one half of the gross book; turnover retains the full gross-book denominator. The per-factor statistics are

$$\begin{aligned} S(f) &= \sqrt{\nu}\,\frac{\bar{x}_f}{s(x_f)}, \qquad R(f) = \nu\,\bar{x}_f, \\ T(f) &= \frac{1}{n}\sum_t \frac{V^{\mathrm{trade}}_{f,t}}{B^{\mathrm{book}}_{f,t}}, \quad \mathrm{Fit}(f) = S(f)\sqrt{\frac{|R(f)|}{\max\{T(f), 0.125\}}}. \end{aligned} \tag{3}$$

The fourth check requires, on every day of the window, at least 20 held names, a single-name weight of at most 0.10 of the gross book, which portfolio construction enforces at every rebalance, and an absolute net exposure of at most 0.01 of the gross book. The check is unavailable, and the candidate INCOMPLETE, when a held return is unknown or invalid-signal days exceed 5% of the window. Book size, return capital and the sample-size rule are given in Appendix B.2.

For the sixth check, $c(f, g)$ is the absolute Pearson correlation between the daily returns of f and of pool record g over their common days within the last 756 trading days of the window, at least 252 of them; every g with $c(f, g) > 0.70$ is a blocker, and f passes only if $S(f) \geq 1.10\,S(g)$ for every blocker.

Decision rule. All six checks are computed for every legal candidate. A candidate whose required evidence is missing for any check is INCOMPLETE; otherwise any failed check gives REJECTED, and every failed check is recorded.

Table 4: The six admission checks.

No.	Check	Local rule	Note
1	Sharpe	$S(f) \geq 1.25$	gross, annualized
2	Fitness	$\mathrm{Fit}(f) \geq 1.00$	Equation 3
3	Turnover	$0.01 \leq T(f) \leq 0.70$	
4	Weight concentration and coverage	at least 20 names, weight at most 0.10, net at most 0.01 of the gross book	text above
5	Sub-universe Sharpe	$S_{\mathrm{sub}}(f) \geq 0.433\,S(f)$	$0.75\sqrt{1/3}$; re-evaluated inside the 1,000 most liquid members
6	Self-correlation and upgrade	every blocker decorrelated or dominated	text above

Sixth check. The pool stores, for each admitted expression, its evaluation context, daily returns, Sharpe ratio, admission type and settings; insufficient overlap makes the check unavailable rather than imputing a zero correlation, and the absolute value also blocks sign-flipped copies.

Duplicate, REJECTED, INCOMPLETE, illegal. A candidate whose signal values or positions over the window coincide with an admitted record's is a duplicate and is rejected; equivalents with different values are left to the correlation check. A repeated expression reuses its cached evaluation but is charged. INCOMPLETE records insufficient required evidence and the missing reason; the candidate counts as neither a success nor a low-quality failure. An expression the parser or evaluator rejects is illegal; its text, error message and charged call are recorded.

Same-batch settlement. Pool-independent statistics are computed for the whole block in parallel; candidates then settle one at a time in task order, the same order in every arm and in validation, each checked against the pool as it stands after the previous candidate, so a candidate can be blocked by one admitted earlier in the same block.

B EVALUATION-WORLD DETAILS

B.1 DATA, UNIVERSE AND VALUATION

Prices, volume, total returns and security events come from the CRSP daily panel (CRSP, 2026), accounts from Compustat quarterly (S&P Global Market Intelligence, 2026), industry labels from the historical Global Industry Classification Standard (GICS) records over their validity intervals (MSCI, 2026), analyst estimates, recommendations, actual earnings and price targets from I/B/E/S (WRDS, 2026a) and implied and historical volatility from OptionMetrics (OptionMetrics, 2026); the availability date is the first date on which the expression evaluator may read a value, and portfolio formation applies its signal delay afterward. Compustat quarterly fields become usable on the first trading day after the report date, or fiscal period end plus 90 calendar days when the report date is missing and never before period end, and expire 400 calendar days after the last valid value. I/B/E/S estimates and recommendations become usable on the third trading day after the summary date for 60 calendar days, actual quarterly earnings per share on the first trading day after announcement for 120 days, and USD 12-month price targets on the first trading day after announcement as a rolling 63-trading-day mean. OptionMetrics volatility is usable on its observation date and carried at most 20 (implied) or 5 (historical) trading days. Per-share accounting fields and analyst values use the panel's price-share basis before being combined with prices, so later stock splits do not enter historical ratios. Availability lags, share-basis conversion and expiry address these sources of leakage; vendor extracts are used as delivered.

The universe admits ordinary shares including real-estate investment trusts and foreign-incorporated ordinary shares, excludes American depositary receipts, uses no exclusion list, and links company fundamentals and GICS labels to every supported share class, with company capitalization summed over the classes linked by the CRSP company identifier.

A held position uses its available daily return and applies a terminal return once before retiring to cash, falling back in order to the CRSP delisting return, a delisting value relative to the last price, a total loss for a for-cause delisting and otherwise the last known price. A listed security without a usable price or return is marked at its last valid price; a priced row that reopens a gap without a CRSP return books zero return (560 rows in the panel, eight of them held in 2019–2023); and a gap still open at the window end is valued at the last price. Scans of every held position in 2019–2023, 2024 and 2025 found no position with an unknown return.

Among the six calibration probes of Section 3, on three of the five probes rejected on both sides, the two sides fail different bars; probes with horizons of five days and longer and fundamental probes agree with the platform in sign and roughly in size.

B.2 SIMULATION, SETTINGS AND ADMISSION

The default simulation ranks finite signals across eligible securities, applies linear decay and demeans within GICS sub-industries, following the platform's public conventions for signal delay, decay and industry neutralization (WorldQuant, 2026). Missing industry labels form a separate group, and decay renormalizes its weights over finite past inputs given a finite current signal. The portfolio rebalances daily; a signal available at the close of day $t-1$ determines holdings earning total return

on day t. On a day without a usable signal the portfolio keeps its previous holdings without trading, and the day counts as invalid; unavailable required return evidence yields INCOMPLETE.

The admission profile uses a USD 20 million gross book and a USD 10 million return-capital denominator, the two denominators of Equation 3. Sharpe uses 252 trading days per year, sample standard deviation and at least 252 observations.

The settings search first evaluates the default of decay one and GICS sub-industry neutralization. If a preliminary quality screen fails but Sharpe is at least 0.8 or Fitness is at least 0.5, it tries decay windows of $\{1, 4, 8\}$ crossed with sub-industry, industry and market neutralization. It selects the highest-Sharpe variant passing that screen; if none passes, it retains the default record.

The screen applies the Sharpe, Fitness, turnover, maximum-weight and sub-universe checks; settlement then applies all six, so a selected variant can still fail.

C Implementation and run matrix

C.1 Trigger and feasibility

The state rule checks every 50 settled main-loop records. Normal eligibility requires more than 300 such records and more than 10% of the charged-call budget. After at least three admissions, an admission-gap condition compares time since the last admission with the geometric-gap threshold $\log(0.05)/\log(1-\widehat{p})$, where $\widehat{p}$ is the cumulative admission rate per main-loop record. Before three admissions it uses a 300-record gap. The crowding condition compares the blocked share among new candidates passing the first five checks in the last 300 records with an earlier baseline that excludes warm-up. It requires at least five window passers and 10 baseline passers, with a pseudocount of one half and a one-sided binomial tail threshold of 0.05.

A 300-record waiting period follows each request, doubles after repeated requests without an admission and resets on an admission; P requires 360 remaining calls and F 160, with at least eight ordinary training examples or four under the declared fallback.

The fixed schedule uses request points at 0.2, 0.4, 0.6 and 0.8 of the total budget, chosen before any run and not tuned. The random-timing control draws four points per seed from $U(0.1, 0.9)$, rounds to two decimals and rejects draws with consecutive gaps below 0.08. The recorded points are $(0.25, 0.33, 0.52, 0.70)$, $(0.11, 0.25, 0.59, 0.79)$ and $(0.21, 0.30, 0.42, 0.58)$ for seeds 0, 1 and 2. The draws were made once, before the runs, with a generator seeded by 1000 plus the run seed. Both controls retain the alternation, feasibility checks and F floor.

C.2 Generation, prompt selection and rule trials

Adaptive dispatch can change the search-block size, task families and refinement share. The incumbent rule starts at a block of 24 proposals and halves it down to 8 while stalled, a state entered when the last 256 proposals admitted nothing and at least half of them were repeats; after a window with an admission the block grows by a factor of 1.5 back to 24. Exploration tasks follow an upper-confidence family score with exploration width 0.5; after 150 proposals, a family with at least 20 scored proposals whose mean shaped score is in the bottom third and below 0.3, with no admission, has its score halved, but no family is removed. The refinement share of a block is 0.5, 0.7 while stalled and not below 0.35 while the beam has open parents. At most 0.2 of a block repairs illegal or incomplete expressions. A candidate joins the refinement beam (capacity 64, at most four children per parent) when its absolute Sharpe reaches 0.64 of the bar, when it fails exactly one of the five quality checks, or when it is among the four candidates of its block closest to the bars. Arms with novelty handling resample repeated outputs up to three times at generation cost only, and the rule can submit, as a negate task, the sign reversal of a candidate that would pass the first five checks under that sign.

The prompt-revision step runs the GEPA optimizer (Agrawal et al., 2026) for reflection and candidate selection, followed by our paired validation with deployment-form tasks. The configured feedback, selection and holdout task counts are 16, 16 and 48; the final gate uses three paired mini-batches. Its dense score gives duplicates and pool copies zero credit and bounded credit to other near misses. Adoption requires mean dense-score gain at least zero, win rate at least 0.5, legal-rate

drop at most 0.05, skeleton-entropy drop at most 0.10, and nonnegative paired discovery-utility gain. If a candidate prompt omits a required statement of the admission thresholds, the incumbent's line stating it is appended before holdout evaluation.

Two breakers bound the prompt search: a wall-clock cap of 2,400 seconds and a cap of 40 rejected reflections. Of the 26 P attempts in the Qwen3-4B run-matrix runs, 17 ended at the wall-clock cap, 9 used their full search call budget and none ended at the rejection cap.

Dispatch-rule replay evaluates recorded candidates under a compatible evaluator and rechecks pool-dependent settlement. A rule round is due every 240 proposals, 10 blocks of the starting size, at most 12 rounds per run and only while no trial is active. The orchestrator proposes four candidate parameter sets; replay on recorded candidates screens them, and a candidate reaches a live trial only if its held-out replay score is at least 0.95 of the incumbent's. The live trial alternates blocks in challenger–incumbent–incumbent–challenger order, with at least four blocks per arm, and adopts the challenger only if its mean dense score per block exceeds the incumbent's by at least 5% with a paired t statistic of at least 1.0. A trial starts only if it fits before the next due P or F attempt and before the budget ends; an attempt that exhausts the call budget ends the run with its cost retained.

C.3 Training reservoirs and role adoption

The recorded F attempts train the subagent on admitted expressions and on filtered near misses that pass the five quality checks and are blocked only by pool correlation (Section 4.3). When a requested attempt finds fewer than eight such examples, the set adds up to eight explicitly labeled near misses ranked by Sharpe. Deduplication distinguishes exploration and refinement task forms. Targets use the recorded deployment-form user turn where available. Training uses a micro-batch of four sequences per role.

The orchestrator trains on rule-amendment examples from live trials or held-out replay. For a positive relative advantage d, the default weight is $\min(1, \max(0.1, d/0.10))$. A candidate within 5% of the incumbent on replay can supply a format example at weight 0.1. Up to four extra replay rounds can supply examples when fewer than six are available.

The subagent gate uses eight paired minibatches. It requires mean dense-score gain at least zero, win rate at least 0.5, legal-rate drop at most 0.10, skeleton-entropy drop at most 0.50, nonnegative discovery-utility gain and reference-score degradation at most 0.05. Strict joint adoption requires both roles to train and pass their gates, failure restores both incumbent adapters, and a declared subagent-only fallback uses its own gate.

C.4 Run matrix

For each arm and seed, the reporting rule uses the one run that reaches the full budget; Appendix D.2 records when it was added. Runs interrupted by loss of their compute instance were continued from checkpoints holding the pool, history, prompt, adapters, rule state and trigger state. Three runs that could not be continued, the full configuration at seeds 1 and 2 and memory seed 1, were restarted from an empty pool with the same seed; their incomplete attempts are not reported (Appendix D.2).

NVIDIA's pipeline enters through three routes. The first re-implements the blueprint's generate, evaluate and advise loop as a dispatch rule on Qwen3-4B-Instruct-2507. It uses the code default of five signals per round and seeds 0–2. The same model plays the signal agent and the advisor. Our DSL parser replaces the code agent, which only translates a formula into Python. Admission by the six checks replaces the rank information-coefficient test. The best signal so far is ranked by the bounded shaped score that the other arms use instead of the absolute information coefficient. Several requests run side by side to fill a generation batch. The generator samples at the temperature of 1.0 used by every arm, and the advisor decodes greedily.

The second route runs this re-implementation with the configuration shipped in the blueprint repository (NVIDIA, 2026c): nvidia/nemotron-3.5-lightning-30b-a3b in every role, served from the published weights nvidia/NVIDIA-Nemotron-3.5-Lightning-30B-A3B-BF16 (NVIDIA, 2026a), with thinking disabled, two signals per round and three iterations. At seed 0, the generator and advisor temperatures are 0.8 and 0.5.

In the shipped-configuration run, 2,153 of 3,000 proposals (71.8%) were legal: 91.0% of first-round tasks and 61.9% of tasks that carried the advisor's or structural feedback, because the advisor names operators and fields outside the DSL and the generator follows it. Illegal proposals count against its budget as in every arm.

The third route runs the original blueprint code, copied with every file byte-identical except one workflow module, on the same published weights. It keeps the blueprint's agents and code generation. It changes the data to the evaluation world's open, close, high, low and volume fields, the only fields its prompt names. It changes the acceptance test to the six checks with pool self-correlation and the settings search, and model serving to a local server. Within the workflow module, the advisor prompt and feedback name our check values in place of the information coefficient, and the best signal so far is compared by signed Sharpe rather than absolute information coefficient. The original-code run made 2,504 legal proposals (83.5%) and 1,526 distinct legal proposals, the denominator of its rate per 1,000 distinct legal proposals in Table 1.

Three further differences follow from our evaluator. The blueprint's acceptance is sign-agnostic, whereas our checks evaluate a signal as written. A cross-sectional operator nested inside a generated formula ranks over every priced security of the day rather than over the universe members. Negative lags and backward fills are illegal under our causality check.

Our AlphaSAGE arm keeps AlphaSAGE's GFlowNet generator (Chen et al., 2026): its expression-token environment over open, high, low, close, volume and a typical-price proxy for the volume-weighted average price, its structure-aware graph encoder, forward and backward policies, entropy-regularized trajectory-balance objective and optimizer settings from its S&P 500 configuration, and the linear decay of its novelty and self-supervised reward weights, applied over our call budget. Two parts of the reward differ. The quality term, in the original the absolute information coefficient against a 20-day forward return, is here our discovery utility from the six checks, zero for a rejected or incomplete candidate and signed, so no sign flip is free. The capacity-limited library that the novelty and self-supervised terms compare against is our append-only admitted pool, with factor values z-scored per day over universe members and the pool settled before each episode. Six of the 36 operators have no exact DSL form and are masked out of the forward policy; the other 30 are translated exactly. The operator translation was checked cell by cell against the generator's own evaluation on 36 of 37 check expressions and on 57 random trees without cross-sectional rank. Two edge differences remain: our cross-sectional rank averages ties and is shifted by $1/n$ per day, and the generator's rolling correlation applies a small-variance guard where ours returns the correlation. Rewards are computed per batch of 64 episodes, the batch of one policy update. Each run stops at 3,000 charged evaluator calls, one per emitted valid expression tree, with repeats and incomplete rows charged as for every arm. An episode whose terminal token sequence is not a valid tree never reaches the evaluator, is not charged, receives the generator's fixed invalid reward and is trained on; seeds 0–2 used 8,478, 7,854 and 7,601 episodes for their 3,000 charged calls, 2.5 to 2.8 episodes per charged call, so an unparseable output costs a language-model arm a charged illegal row and costs the AlphaSAGE arm nothing. Per 1,000 distinct legal proposals the three runs admit 3.25, 2.08 and 4.00.

Our FactorMiner arm (Wang et al., 2026) is rebuilt from the paper, which publishes no code. It runs the frozen Qwen3-4B-Instruct-2507 backbone with thinking disabled and the sampling temperature of every arm, in place of the paper's Gemini 3.0 Flash. The experience memory holds the mining state, successful and failed templates and strategic insights; it is consolidated after every batch. It starts empty in every run, whereas the paper carries memory across sessions. Each batch of 24 candidates is written in three rounds whose turns list the candidates already written; exact within-batch duplicates are regenerated up to twice, and a surviving duplicate is evaluated and charged. One memory-distillation call to the same frozen model follows each batch, 125 per run; these calls and the regenerated completions are recorded separately and not charged. Admission is decided only by our six checks, with one charged evaluator call per candidate in place of the paper's separate fast screen and full validation, and the paper's single-incumbent replacement becomes the evaluation world's Upgrade, which keeps the incumbent in the append-only pool and retires it only in the memory's library view. The arm has no prompt revision, fine-tuning or trigger and, as in the paper, no run-level no-repeat list, so repeated expressions are charged.

The additional backbone pairs run adaptive dispatch and the full configuration at seed 0: Qwen3-8B at 3,000 calls, Qwen3.8-27B at 2,000 calls, and Gemma 4 E4B (google/gemma-4-E4B-it) at 3,000

calls with the same generation settings as the Qwen3-8B pair. All six runs reached their call limits. The Gemma full-configuration run executed two P attempts, neither adopted, each ending when its search call budget was used, and two F attempts, both adopted as joint updates, finishing at charged calls 1,832 and 2,920; the attempts charged 600 and 320 validation calls. Every run caps generation at 160 new tokens per proposal; at seed 0 about 3% of Qwen3-4B, 5% to 13% of Qwen3-8B and no Qwen3.8-27B proposals hit the cap.

C.5 Resource records and frozen inputs

The run controller limits charged evaluator calls and the numbers of P and F attempts. Raw generation, reflection, training and evaluator time are recorded separately. The call budget does not combine token use and GPU time into a common cost unit, and F shares the run's call budget. Settings-search variants and extra generation work are not individually charged as proposal-budget units.

Each run's record binds the world profile, data-array hashes, evaluator revision, model and tokenizer revisions, prompts, adapters, generation limits, seeds, hardware and checkpoint continuations to its identifier; cached evaluations and checkpoint restarts require the same evaluation identity; failed attempts, retries and rollbacks are retained, and repeated attempts are not additional seeds.

D Evidence provenance and incomplete comparisons

D.1 Result verification

Tables 1–3 and the P and F attempt counts in Section 6 are recomputed from proposal, admission and adaptation records, using one full-budget run per configuration and seed. Run summaries are checked against these counts but do not supply them. The charged calls agree with the discovery and adaptation records, with no duplicate or over-budget steps; gaps in the discovery sequence correspond exactly to adaptation attempts. The accepted pool matches the admission events, and runs continued from checkpoints preserve the expressions and decisions of their checkpoint prefixes. A separately implemented verification reproduces the admission counts for all reported runs except the Qwen3-8B adaptive-dispatch run, the three AlphaSAGE runs, the three FactorMiner runs and the Gemma pair, which completed later and were counted by the primary procedure alone. For the AlphaSAGE runs, the charged rows and the uncharged invalid-tree episodes are disjoint and together number exactly the recorded episodes. The final-500 columns of Tables 1 and 2 count the admissions recorded after charged call 2,500 in the same records.

Because P and F validation shares the budget, the full configuration evaluates 2,200 to 2,504 main-loop proposals per run against 3,000 for adaptive dispatch; per 1,000 distinct legal proposals it admits 29.4, 20.1 and 12.1 against 12.9, 14.3 and 14.0 in seeds 0–2.

D.2 Inputs inspected before the pre-specified criterion

The two conditions of the pre-specified criterion in Section 5 were fixed before any Qwen3-4B full-configuration or adaptive-dispatch run of the run matrix had completed. The three execution details, averaging over seeds 0–2, the as-proposed signs of random-pool factors and the one-run-per-seed rule, were added later, also before any such run had completed. The timing claim does not extend to the Qwen3.8-27B pair: its full-configuration run had completed, and its admission counts had been observed, before the criterion was fixed.

Three inputs had been inspected before the criterion was fixed: development runs of the compared configurations, which are not part of the reported results; partial records, namely run-matrix runs before they were continued from checkpoints, including the first 144 and 984 calls of adaptive dispatch seeds 1 and 2, and the incomplete first attempts of the three restarted runs (Appendix C); and a preliminary out-of-sample analysis. That analysis ranked the admitted pools of development runs and of the completed fixed-prompt runs of seeds 1 and 2 against the criterion's random-pool control in 2024 and 2025. Its values are not results of the run matrix.

D.3 OUT-OF-SAMPLE EVALUATION

The out-of-sample columns of Tables 1–3 score every admitted expression of the selected run on 2024 and on 2025 separately, with the decay and neutralization settings recorded at its in-sample evaluation, the settings search off, at least 200 valid days, the same universe and gross returns. The pool Sharpe is the equal-weight Sharpe of the admitted pool with signs as proposed. The percentile is the share of 1,000 random pools of the same size, drawn from a uniform sample of 200 distinct legal proposals of the same run with complete in-sample evaluations and used with their proposed signs, whose pool Sharpe is strictly below the admitted pool's. A secondary variant that flips each random factor to a positive in-sample Sharpe yields mean percentiles of 60 in 2024 and 100 in 2025 for the full configuration over seeds 0–2. The NVIDIA original-code run's four admitted signals are Python functions and are re-executed by the blueprint's own code under the same rules and random-pool seeds.

The out-of-sample evaluation covers all 44 reported runs. Over seeds 0–2, the NVIDIA re-implementation's pools have mean pool Sharpe 1.27 in 2024 and 1.17 in 2025 at mean percentiles 42 and 87, FactorMiner's 1.03 and 0.18 at 63 and 65, and AlphaSAGE's 3.27 and 3.22 at 99 and 99; at seed 0, the shipped Nemotron configuration's pool has Sharpe 1.85 and 0.88 at percentiles 64 and 73, and the original-code pool 2.58 and 2.80 at 100 and 100. Seeds 3 and 4 lie outside the seed 0–2 means: adaptive dispatch ranks at the 46th and 0th percentiles in 2024, where three of 1,000 random pools lie below seed 4, and at the 100th and 45th in 2025; the full configuration at the 95th and 4th in 2024 and the 100th and 93rd in 2025.