EVOMIND: A SELF-EVOLVING ALPHA FACTOR MINING AGENT VIA PROCESS-REWARD REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Alpha factor discovery seeks predictive, robust, and interpretable signals in a vast expression space, guided by market feedback that is noisy and non-stationary. Large language model (LLM) agents can propose economically motivated factors and revise them from backtests, but most keep the LLM frozen and improve only through prompting, while agents that do train the LLM reward an entire mining trajectory with a single scalar that cannot tell which revision helped. We propose EvoMind, a self-evolving alpha factor mining agent trained with process-reward reinforcement learning. EvoMind formulates factor discovery as a multi-turn loop of hypothesis generation, factor implementation, tool-based evaluation, feedback reasoning, and revision. Its hierarchical process rewards score every turn along a gated chain of criteria: executability without look-ahead, predictive power on the feedback window, novelty with respect to the current factor pool, and stability on a verification window that the agent never observes. These rewards are converted into turn-level advantages, so each revision is credited for the outcomes that follow it rather than for the trajectory it belongs to. EvoMind further diagnoses both successful and failed attempts into an experience bank whose failure types are read off the reward hierarchy, and conditions later exploration on retrieved validated mechanisms and falsified directions. On CSI 300, CSI 500, and S&P 500, with agent-facing windows disjoint from the test period and a shared evaluation budget, EvoMind raises the mean test RankIC from 0.0447 for the strongest baseline to 0.0493 and the mean annualized excess return from 7.5% to 9.2%. Ablations isolate the contributions of turn-level credit, the reward hierarchy, hidden-window stability, and failure-aware experience.

1 INTRODUCTION

Formulaic alpha factors map historical market data to cross-sectional scores that predict future returns. They remain central to quantitative investment because they are interpretable, auditable, and cheap to deploy (Kakushadze, 2016). Discovering new ones is hard: the expression space is combinatorial, backtest feedback is noisy and non-stationary, and a factor that fits one period often decays in the next (Harvey et al., 2016; Bailey et al., 2017). Genetic programming and reinforcement learning (RL) over expression trees search this space efficiently (Koza, 1992; Yu et al., 2023; Zhu & Zhu, 2025), but they optimize backtest fit without an economic rationale and tend to produce complex, fragile expressions.

LLM agents offer a complementary route. They state a market hypothesis, translate it into a factor expression and code, backtest it, and revise it from the result (Yao et al., 2023; Tang et al., 2025; Li et al., 2025). Recent systems evolve complete mining trajectories through mutation and crossover (Han et al., 2026) or allocate a search budget with a controller and a shared memory (Li et al., 2026). In these systems the LLM is frozen: improvements come from better prompts, schedules, and retrieval, while the policy that proposes factors never changes. AlphaAgentEvo (Tang et al., 2026) instead trains a small LLM with agentic RL and a hierarchical reward, showing that the mining policy itself can be learned.

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.

Learning a mining policy from market feedback raises three problems that a trajectory-level reward leaves open. *Credit assignment:* a trajectory that ends with a strong factor usually contains invalid, redundant, or weak proposals along the way; one scalar per trajectory reinforces all of them equally and penalizes the good revisions inside trajectories that end poorly. *Overfitting to the feedback:* if the policy is rewarded on the same window whose metrics it reads, RL turns backtest overfitting into a training objective, and auxiliary terms such as diversity can be collected by invalid or uninformative variants. *Wasted failures:* failed hypotheses are discarded or kept only in the current context, so the agent keeps revisiting directions that have already been falsified.

We present EvoMind, which addresses these problems with three components (Figure 1). First, *hierarchical process rewards* score every turn along a gated chain: a factor earns predictive reward only if it executes without look-ahead, novelty reward only if it is also minimally predictive, and the largest reward, for stability, only if it additionally retains its predictive power on a verification window hidden from the agent. Second, *turn-level credit assignment* converts these rewards into advantages computed from the return that follows each turn and normalized among trajectories at the same depth. Third, a *failure-aware experience bank* stores successful and failed attempts, types each failure by the first reward level it violates, and conditions later episodes on retrieved validated mechanisms and falsified directions. EvoMind therefore evolves on two time scales: its weights improve through RL, and its experience keeps growing during mining without further gradient updates.

Our contributions are as follows.

- We formulate LLM-based alpha mining as a multi-turn decision process with an explicit information boundary: metrics on a search window are visible to the agent, a later verification window enters only the reward and pool admission, and the test period is evaluated once (Section 3).
- We introduce gated hierarchical process rewards, a turn-level advantage estimator that reduces to group-relative policy optimization in the single-turn case, and an experience bank whose failure types are verifiable rather than LLM-judged (Section 4).
- We evaluate on three markets against factor libraries, symbolic search, frozen-LLM agents, and an RL-trained agent under matched evaluation budgets, with five independent end-to-end runs per configuration and ablations that isolate each component (Section 5). EvoMind improves the mean test RankIC by 10.2% over the strongest baseline; replacing turn-level with trajectory-level credit lowers CSI 300 RankIC from 0.0528 to 0.0490, and removing the experience bank more than doubles the rate of repeated failures.

2 Related Work

Formulaic alpha mining. Hand-crafted libraries such as the 101 formulaic alphas (Kakushadze, 2016) and Alpha158/Alpha360 in Qlib (Yang et al., 2020) encode expert knowledge but grow slowly. Genetic programming (Koza, 1992) and RL over expression trees, such as AlphaGen (Yu et al., 2023) and AlphaQCM (Zhu & Zhu, 2025), automate the search and optimize the joint performance of a factor set, but they reward backtest fit alone and cannot use natural-language priors.

LLM agents for quantitative research. AlphaAgent regularizes LLM-driven mining with originality, hypothesis alignment, and complexity constraints (Tang et al., 2025); R&D-Agent-Quant coordinates factor and model development (Li et al., 2025); QuantaAlpha evolves complete mining trajectories with mutation and crossover and gates generation by semantic consistency, complexity, and redundancy (Han et al., 2026). The concurrent AutoScientist-Quant treats research as one budgeted search in which a controller chooses to improve, combine, pivot, or stop, shares memory across branches, and separates the feedback window from the test window after fixing look-ahead problems in widely reused evaluation code (Li et al., 2026). All of these keep the LLM frozen. EvoMind adopts their hypothesis-to-code loop, generation gates, and disjoint evaluation windows as infrastructure, and asks how the proposing policy itself can be trained from process-level feedback. LLM-guided program search in other domains (Romera-Paredes et al., 2024; Novikov et al., 2025; Chen et al., 2023) and concept libraries distilled from strong and weak hypotheses (Grayeli et al., 2024) likewise rely on frozen models.

Agentic RL and process rewards. Group-relative policy optimization (GRPO) and its variants train LLMs from verifiable outcomes (Shao et al., 2024; DeepSeek-AI et al., 2025; Yu et al., 2025). Process supervision rewards intermediate steps using human or automatic labels (Uesato et al., 2022;

Lightman et al., 2024; Wang et al., 2024) or implicit process rewards (Cui et al., 2025); VinePPO estimates step values with Monte Carlo rollouts (Kazemnejad et al., 2024); ArCHer and GiGPO assign credit at the turn or step level in multi-turn agents (Zhou et al., 2024; Feng et al., 2025). Closest to our work, AlphaAgentEvo trains an LLM for alpha evolution with GRPO and a hierarchical reward over tool validity, seed consistency, exploration, performance, and improvement streaks (Tang et al., 2026); its components are aggregated into one scalar per trajectory and computed on the window the agent observes. EvoMind differs in where the reward is assigned (every turn, with gated levels), what it measures (pool-relative novelty and stability on a hidden window), and what is carried across episodes (a typed experience bank).

Learning from experience. Reflexion stores verbal feedback across attempts (Shinn et al., 2023), ExpeL distills insights from successful and failed trajectories (Zhao et al., 2024), and Voyager accumulates a library of skills (Wang et al., 2023). These methods keep the weights fixed and ask an LLM why an attempt failed. EvoMind derives failure types from the reward hierarchy, which makes them verifiable, and retrieves experience inside RL rollouts, so the policy is trained to use it.

3 Problem Formulation

Factors and data. Let $\mathcal{S}_t$ be the point-in-time universe on day t and $\boldsymbol{X}_{\leq t}$ the price–volume history available after the close of day t. A factor is an expression f in a domain-specific language (DSL) of arithmetic, cross-sectional, and time-series operators that maps $\boldsymbol{X}_{\leq t}$ to a score for every stock in $\mathcal{S}_t$. It is judged by how well these scores rank the forward return y_t from the open of day $t{+}1$ to the open of day $t{+}2$. For a set of trading days $\mathcal{D}$, $\mathrm{RIC}_{\mathcal{D}}(f)$ denotes the mean daily Spearman correlation between f and y, and $\rho_{\mathcal{D}}(f, g)$ the mean daily Spearman correlation between two factors.

Information boundary. Time is divided into five consecutive, disjoint windows. $\mathcal{D}_{\mathrm{tr}}$ and $\mathcal{D}_{\mathrm{va}}$ fit and early-stop the downstream predictor. The agent-facing period is split into a search window $\mathcal{D}_{\mathrm{srch}}$, whose metrics appear in tool responses, and a later verification window $\mathcal{D}_{\mathrm{ver}}$, which enters only the reward and pool admission and is never shown to the agent. The test window $\mathcal{D}_{\mathrm{te}}$ is evaluated once, after all factors and models are frozen. During RL training, rewards are computed only on $\mathcal{D}_{\mathrm{srch}}$ and $\mathcal{D}_{\mathrm{ver}}$, both of which precede $\mathcal{D}_{\mathrm{te}}$.

Mining as a multi-turn decision process. An episode starts from a task x, a research direction stated in natural language or, optionally, a seed factor, together with retrieved experience m. At turn $k = 1, \ldots, K$ the policy π_θ, an LLM, observes the state $s_k = (x, m, a_{1:k-1}, o_{1:k-1})$ and emits an action a_k that contains a reflection on the previous feedback, a hypothesis h_k describing an economic mechanism, and a set F_k of at most B factor expressions. The evaluation tool returns an observation o_k with execution errors, $\mathcal{D}_{\mathrm{srch}}$ metrics, and the maximum correlation of each candidate with the current factor pool $\mathcal{P}$. We seek a policy that, within a fixed evaluation budget, discovers many factors that are valid, predictive, non-redundant, and stable beyond the window it observes. Admitted factors form a library that a fixed predictor and portfolio rule consume.

4 EvoMind

EvoMind consists of the multi-turn mining environment of Section 3, a reward that scores each turn (Section 4.1), an estimator that turns these rewards into turn-level advantages (Section 4.2), and an experience bank that carries information across episodes (Section 4.3). Section 4.4 describes how they are combined for training and for mining.

4.1 Hierarchical Process Rewards

For each candidate $f \in F_k$ the evaluation tool computes four quantities, all available when turn k ends (Table 1).

- *Validity* $v(f) \in \{0, 1\}$: f parses, every operator looks only backward in time (checked statically on the syntax tree), its size stays within $C_{\max}$ nodes, and it executes on $\mathcal{D}_{\mathrm{srch}}$ with sufficient cross-sectional coverage.
- *Predictive strength* $q(f) = |\mathrm{RIC}_{\mathcal{D}_{\mathrm{srch}}}(f)|$ with orientation $\varsigma(f) = \mathrm{sign}(\mathrm{RIC}_{\mathcal{D}_{\mathrm{srch}}}(f))$, normalized as $\phi(q) = \min(q/q_{\mathrm{ref}}, 1)$, where q_{ref} is a fixed reference scale.

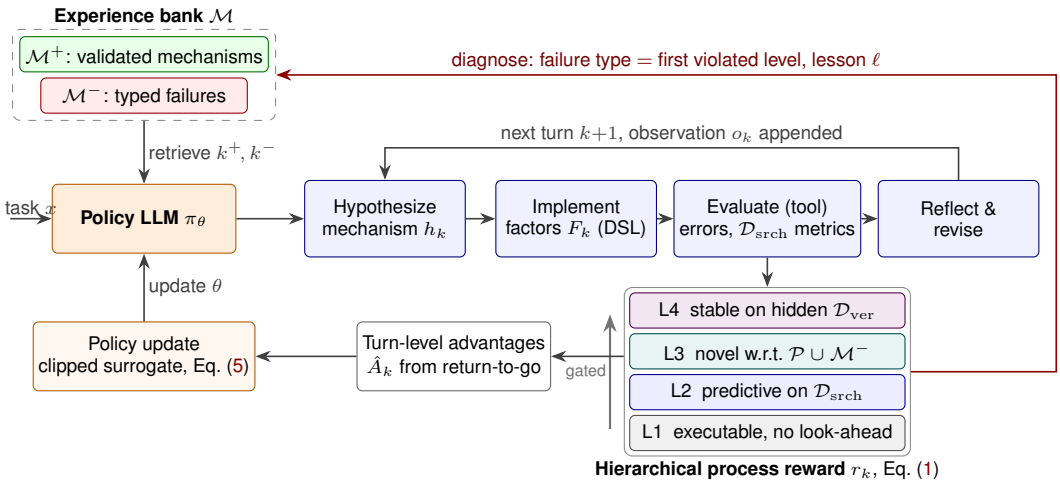

Figure 1: **Overview of EvoMind.** The policy LLM runs a multi-turn loop of hypothesis, implementation, tool evaluation, and revision, conditioned on experience retrieved from a bank of validated mechanisms and typed failures. Each turn receives a gated hierarchical process reward; level L4 is computed on a verification window that never appears in the agent's observations. Returns that follow each turn yield turn-level advantages for the policy update, and every attempt is diagnosed back into the experience bank. Admissible factors enter the factor pool $\mathcal{P}$; the final library is evaluated once on the test window.

Table 1: **Reward hierarchy.** Each level is paid only if all lower levels are satisfied (Equation (1)). "Visible" indicates whether the underlying quantity is shown to the policy in the tool response.

Level	Criterion	Signal	Window	Visible
L1	executable, backward-looking, bounded size	$v(f) \in \{0, 1\}$	$\mathcal{D}_{\text{srch}}$	yes (errors)
L2	predictive	$\phi(q(f))$	$\mathcal{D}_{\text{srch}}$	yes
L3	novel w.r.t. pool and falsified factors	$n(f)$	$\mathcal{D}_{\text{srch}}$	yes (max. corr.)
L4	stable out of sample	$u(f)$	$\mathcal{D}_{\text{ver}}$	no
Outcome	marginal pool gain of admissible factors	$R_{\text{out}}(\tau)$	$\mathcal{D}_{\text{srch}} \to \mathcal{D}_{\text{ver}}$	no

- *Novelty* $n(f) = 1 - \max_{g \in \mathcal{P} \cup \mathcal{M}^- \cup F_{<k}} |\rho_{\mathcal{D}_{\text{srch}}}(f, g)|$, measured against the factor pool, the falsified factors stored in the experience bank, and the episode's earlier proposals $F_{<k}$.
- *Stability* $u(f) = \text{clip}\big(\varsigma(f)\, \text{RIC}_{\mathcal{D}_{\text{ver}}}(f) / \max(q(f), \epsilon),\, 0,\, 1\big)$, the fraction of the search-window predictive strength that survives, with the same sign, on the hidden window.

The candidate reward is gated level by level:

$$r(f) \;=\; \lambda_1\big(2v - 1\big) \;+\; v\Big[\lambda_2\, \phi(q) \;+\; \mathbf{1}[q \geq q_{\min}]\big(\lambda_3\, n \;+\; \mathbf{1}[n \geq \delta]\, \lambda_4\, u\big)\Big], \tag{1}$$

where the argument f is dropped on the right-hand side, and the turn reward is the mean over candidates, $r_k = |F_k|^{-1} \sum_{f \in F_k} r(f)$. A factor is *admissible* if it passes every level: $v = 1$, $q \geq q_{\min}$, $n \geq \delta$, and $u \geq u_{\min}$. At the end of an episode, an outcome reward measures what the episode contributes to the pool on the hidden window,

$$R_{\text{out}}(\tau) \;=\; \lambda_5\big[V_{\text{ver}}(\mathcal{P} \cup \mathcal{A}(\tau)) - V_{\text{ver}}(\mathcal{P})\big], \tag{2}$$

where $\mathcal{A}(\tau)$ is the set of admissible factors produced in episode τ and $V_{\text{ver}}(\cdot)$ is the $\mathcal{D}_{\text{ver}}$ RankIC of a ridge combination of the given factors fitted on $\mathcal{D}_{\text{srch}}$.

Why gate the levels. Summing the four criteria admits two undesirable shortcuts: random or near-constant expressions are maximally novel, and small rewrites of a known strong factor are predictive. The gate $\mathbf{1}[q \geq q_{\min}]$ withholds novelty reward from uninformative expressions, and $\mathbf{1}[n \geq \delta]$ withholds stability reward from redundant ones. The largest weight, λ_4, is placed on stability, which

the policy can earn only by proposing factors whose predictive power transfers to a window it never reads. The order of the levels follows their dependence: stability is undefined for a factor that does not execute and meaningless for one that does not predict.

4.2 Turn-Level Credit Assignment

For each task x we sample a group of G episodes $\{\tau^{(i)}\}_{i=1}^{G}$ from the current policy π_{old}; episode i lasts $K_i \le K$ turns. The return that follows turn k is

$$G_k^{(i)} \;=\; \sum_{j=k}^{K_i} \gamma^{\,j-k}\, r_j^{(i)} \;+\; \gamma^{\,K_i-k}\, R_{\text{out}}\big(\tau^{(i)}\big), \tag{3}$$

with discount $\gamma \in (0, 1]$. Let $\mathcal{I}_k = \{i : K_i \ge k\}$ be the episodes that reach turn k. The advantage of turn k is normalized among them,

$$\hat{A}_k^{(i)} \;=\; \frac{G_k^{(i)} - \mu_k}{\sigma_k + \epsilon}, \qquad \mu_k = \operatorname*{mean}_{j \in \mathcal{I}_k} G_k^{(j)}, \quad \sigma_k = \operatorname*{std}_{j \in \mathcal{I}_k} G_k^{(j)}, \tag{4}$$

and when $|\mathcal{I}_k| < 2$ we use the mean and standard deviation of all returns in the group. Every token generated by the policy in turn k receives $\hat{A}_k^{(i)}$; tokens returned by the tool are masked. With importance ratio $\varrho_t(\theta) = \pi_\theta(y_t \mid y_{<t}) / \pi_{\text{old}}(y_t \mid y_{<t})$ for token y_t, we minimize the clipped surrogate with token-level aggregation (Schulman et al., 2017; Yu et al., 2025),

$$\mathcal{L}(\theta) = -\frac{1}{Z} \sum_{i,k} \sum_{t \in a_k^{(i)}} \min\Big(\varrho_t \hat{A}_k^{(i)},\; \text{clip}\big(\varrho_t, 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}}\big) \hat{A}_k^{(i)} \Big) + \beta\, \text{KL}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big), \tag{5}$$

where $Z = \sum_{i,k} |a_k^{(i)}|$ is the number of policy tokens in the group.

Properties. (i) Rewards collected before turn k do not depend on the action taken at turn k, so dropping them from its weight leaves the policy gradient unbiased while reducing its variance (Schulman et al., 2016). (ii) With a single turn, Equation (4) is the GRPO advantage of $r_1 + R_{\text{out}}$; with all process rewards set to zero and $\gamma = 1$ it is outcome-only GRPO, which we use as an ablation. (iii) Compared with a trajectory-level advantage, a turn is no longer credited for what happened before it, and for $\gamma < 1$ its own process reward outweighs rewards that arrive many turns later. The estimator requires no learned critic and no additional rollouts; normalizing per depth compares turns that have consumed the same share of the budget.

4.3 Failure-Aware Experience Bank

Records and verifiable failure types. Every evaluated candidate yields a record $e = (x, h, f, \varsigma, \text{metrics on } \mathcal{D}_{\text{srch}}, c, \ell)$. Its type c is the first level of Table 1 that the candidate violates, which the tool determines without any model judgment: Exec (fails to parse, looks forward, exceeds $C_{\max}$, or raises an error; L1), Weak ($q < q_{\min}$; L2), Redundant ($n < \delta$, with the nearest pool or falsified factor recorded; L3), Unstable ($u < u_{\min}$; L4), or Admissible. The frozen reference model π_{ref} then writes a lesson ℓ of at most two sentences, conditioned on the record and its type, that states which part of the mechanism or construction to change. Because the type is fixed by the tool, a lesson can explain a failure but cannot misclassify it.

Storage and retrieval. Admissible records form $\mathcal{M}^+$ and failed records form $\mathcal{M}^-$. Records are deduplicated by the canonical form of the syntax tree and by factor correlation, and each failure type keeps at most N_c records, replaced first-in first-out. At the start of an episode we embed the task and retrieve k^+ records from $\mathcal{M}^+$ and k^- from $\mathcal{M}^-$ by maximal marginal relevance (Carbonell & Goldstein, 1998), which trades similarity to the task against diversity among the retrieved records. They are inserted into the prompt as two lists, validated mechanisms and falsified directions.

How experience enters learning. Experience affects EvoMind in three ways. During RL, rollouts are conditioned on retrieved records, so the policy is optimized to exploit them rather than merely shown them at test time. The novelty level includes $\mathcal{M}^-$ in its reference set, so re-proposing a falsified factor is penalized as redundancy and ignoring negative experience has a direct cost. During mining, the bank keeps growing without gradient updates, which adapts the agent to the target market. To respect the information boundary, records store $\mathcal{D}_{\text{srch}}$ metrics only; for Unstable records the bank keeps the type but no $\mathcal{D}_{\text{ver}}$ values, and nothing from $\mathcal{D}_{\text{te}}$ is ever stored.

4.4 Training and Mining

Training. The policy is initialized from an instruction-tuned open-weight LLM and warm-started by supervised fine-tuning on a small set of trajectories in the agent's output format (Section 5.1). Each RL iteration samples a batch of tasks, retrieves experience, rolls out G episodes of at most K turns with B candidates per turn, computes Equations (1) and (2), forms the advantages of Equation (4), updates θ with Equation (5), and writes all records to the bank. Algorithm 1 in Appendix A gives the full procedure.

Mining. On a target market the trained policy runs R rounds of episodes against a growing pool and bank. After each round, admissible factors are sorted by $\mathcal{D}_{\text{ver}}$ RankIC and admitted greedily if their absolute correlation with every pool member is at most $1 - \delta$. The final library is fed to a fixed LightGBM predictor (Ke et al., 2017) trained on $\mathcal{D}_{\text{tr}}$ and early-stopped on $\mathcal{D}_{\text{va}}$, whose scores drive a fixed portfolio rule on $\mathcal{D}_{\text{te}}$.

5 Experiments

We ask three questions. **Q1:** Under matched budgets, does EvoMind discover libraries that predict and trade better out of sample than factor libraries, symbolic search, frozen-LLM agents, and an RL-trained agent? **Q2:** Which components are responsible: turn-level credit, the reward hierarchy, hidden-window stability, or failure-aware experience? **Q3:** How do these components change the search itself, including validity, redundancy, repeated failures, and the gap between search and verification performance?

5.1 Setup

Markets and data. We use CSI 300, CSI 500, and S&P 500 with daily open, high, low, close, volume, and VWAP from Qlib (Yang et al., 2020) and point-in-time index membership. The windows are $\mathcal{D}_{\text{tr}}$ = 2016-01-01 to 2019-12-31, $\mathcal{D}_{\text{va}}$ = 2020, $\mathcal{D}_{\text{srch}}$ = 2021, $\mathcal{D}_{\text{ver}}$ = 2022, and $\mathcal{D}_{\text{te}}$ = 2023-01-01 to 2025-12-31. Signals are computed after the close of day t and traded at the open of day $t{+}1$. Every search-time metric is computed only on its declared window, and no stage of training or mining observes $\mathcal{D}_{\text{te}}$ (Appendix D).

Policy and training. The policy is Qwen3-4B-Instruct-2507 (Yang et al., 2025), warm-started on 1K trajectories in the agent's output format and trained for 600 RL steps with 8 tasks per step, group size $G = 4$, at most $K = 4$ turns, and $B = 2$ candidates per turn. Reward weights, thresholds, and optimizer settings are listed in Table 5; all of them were fixed before the single evaluation on $\mathcal{D}_{\text{te}}$.

Baselines. (i) Factor libraries Alpha158 and Alpha360 with LightGBM (Yang et al., 2020; Ke et al., 2017); (ii) symbolic search: genetic programming (Koza, 1992), AlphaGen (Yu et al., 2023), and AlphaQCM (Zhu & Zhu, 2025); (iii) frozen-LLM agents: AlphaAgent (Tang et al., 2025), QuantaAlpha (Han et al., 2026), and AutoScientist-Quant (Li et al., 2026), all reimplemented with Qwen3-4B-Instruct-2507 as the frozen LLM backbone; (iv) the RL-trained agent AlphaAgentEvo (Tang et al., 2026) on the same backbone as ours; and (v) two same-backbone controls: *Frozen-EvoMind*, which runs our loop and experience bank with the warm-started backbone but without RL, and *Outcome-GRPO*, which trains with R_{out} only. All baselines are re-run under our data, windows, and portfolio protocol. All agentic methods share the DSL, the evaluation tool, the number of factor evaluations per run, the pool-admission rule, and the downstream predictor and portfolio. For AutoScientist-Quant, we evaluate its alpha-discovery component under the common downstream LightGBM predictor, so that differences reflect factor-mining quality rather than downstream model selection.

Metrics. We report RankIC and RankICIR of the predictor's scores and, from a strategy that holds the top 50 stocks with at most 5 replacements per trading day and a one-way transaction cost of 10 bps, the annualized excess return over the benchmark index (ARR), information ratio (IR), maximum drawdown of cumulative excess return (MDD), and Calmar ratio. Every configuration is run end to end five times with different seeds; we report means, and Appendix E gives standard deviations of all eight metrics on CSI 300.

Table 2: **Test-period results.** RankIC (RIC), annualized excess return (ARR, %), and maximum drawdown (MDD, %, lower is better), mean over five runs; standard deviations and all eight metrics for CSI 300 are in Appendix E. Agentic methods share the evaluation budget. Best value per column in bold.

	CSI 300			CSI 500			S&P 500		
Method	RIC	ARR	MDD	RIC	ARR	MDD	RIC	ARR	MDD
Factor libraries + LightGBM									
Alpha158	0.0301	2.8	14.7	0.0352	4.6	16.2	0.0186	2.2	11.2
Alpha360	0.0327	3.1	14.1	0.0378	5.0	15.4	0.0201	2.5	10.8
Symbolic search									
Genetic programming	0.0348	3.4	13.9	0.0395	5.3	15.1	0.0194	2.3	11.0
AlphaGen	0.0382	4.1	12.9	0.0436	6.4	13.8	0.0228	3.0	10.2
AlphaQCM	0.0401	4.6	12.2	0.0453	7.0	13.2	0.0241	3.4	9.8
Frozen-LLM agents									
AlphaAgent	0.0417	5.1	11.5	0.0471	7.5	12.5	0.0264	4.1	9.2
QuantaAlpha	0.0456	6.0	10.7	0.0508	8.8	11.6	0.0308	5.5	8.5
AutoScientist-Quant	0.0479	6.7	9.8	0.0532	9.7	10.7	0.0331	6.2	**7.4**
Frozen-EvoMind	0.0461	6.2	10.2	0.0510	9.0	11.2	0.0309	5.6	8.1
RL-trained agents									
AlphaAgentEvo	0.0472	6.5	10.0	0.0523	9.5	10.9	0.0318	5.9	7.9
Outcome-GRPO	0.0454	6.1	10.5	0.0496	8.6	11.8	0.0297	5.2	8.4
EvoMind (ours)	**0.0528**	**8.4**	**8.6**	**0.0584**	**12.1**	**9.3**	**0.0367**	**7.1**	7.6

5.2 Main Results (Q1)

Overall comparison. Table 2 shows that EvoMind attains the highest test RankIC in all three markets: 0.0528, 0.0584, and 0.0367 on CSI 300, CSI 500, and S&P 500, against 0.0479, 0.0532, and 0.0331 for the strongest baseline, AutoScientist-Quant, which are relative improvements of 10.2%, 9.8%, and 10.9%. The gains carry over to the portfolio, with annualized excess returns of 8.4%, 12.1%, and 7.1% against 6.7%, 9.7%, and 6.2%. EvoMind has the lowest drawdown on CSI 300 and CSI 500 but not on S&P 500 (7.6% against 7.4% for AutoScientist-Quant), so a stronger ranking signal does not uniformly translate into a lower drawdown.

Effect of learning. Frozen-EvoMind runs the same loop and experience bank without RL; training the policy raises CSI 300 RankIC from 0.0461 to 0.0528. RL alone does not explain this gain: AlphaAgentEvo (0.0472) and Outcome-GRPO (0.0454) are trained with RL on the same backbone and remain well below EvoMind. Outcome-GRPO is even slightly below Frozen-EvoMind in all three markets, which indicates that the sparse, noisy outcome reward alone is not enough to improve the mining policy.

5.3 Ablations (Q2)

Table 3 removes one component at a time under the same budget.

Credit assignment. With the same rewards, a trajectory-level advantage lowers RankIC from 0.0528 to 0.0490 and ARR from 8.4% to 7.3%; training on the outcome reward alone lowers them further, to 0.0454 and 6.1%. Relative to outcome-only training, about half of the RankIC gain therefore comes from the process rewards themselves (0.0454 to 0.0490) and half from assigning them per turn (0.0490 to 0.0528).

Reward hierarchy. Computing stability on the visible window instead of the hidden one causes the largest RankIC loss in this group (0.0468) and the largest drawdown of all variants (11.0%), although it admits more factors (76 against 73), consistent with the policy fitting the window it observes. Summing the levels without gates costs almost as much (0.0472). Removing the novelty level also admits more factors (79) but lowers RankIC to 0.0483 and nearly doubles the repeated-failure rate (14.7% against 7.8%), so without a novelty incentive the policy drifts back to falsified directions.

Experience. Without the experience bank, the repeated-failure rate rises from 7.8% to 19.6% and RankIC falls to 0.0476. Storing successes only recovers part of the gap (0.0496 and 13.1%), and

Table 3: **Ablations** on CSI 300, mean over five runs. "Adm." is the number of factors admitted to the pool. "Rep." is the share of candidates whose absolute correlation with an earlier failed candidate of the same run is at least $1 - \delta$. All variants use the same budget.

Group	Variant	RIC	ARR (%)	MDD (%)	Adm.	Rep. (%)
	EvoMind (full)	**0.0528**	**8.4**	**8.6**	73	**7.8**
Credit	trajectory-level advantage, same rewards	0.0490	7.3	9.2	65	9.1
	outcome reward only (Outcome-GRPO)	0.0454	6.1	10.5	56	12.8
Reward	ungated sum of levels	0.0472	6.7	10.1	62	10.4
	w/o novelty level (L3)	0.0483	7.0	9.6	79	14.7
	stability on visible $\mathcal{D}_{\text{srch}}$	0.0468	6.3	11.0	76	8.4
Experience	no experience bank	0.0476	6.9	9.8	60	19.6
	successes only ($\mathcal{M}^+$)	0.0496	7.4	9.2	66	13.1
	random retrieval (length-matched)	0.0487	7.1	9.5	63	16.8
Learning	no RL (Frozen-EvoMind)	0.0461	6.2	10.2	57	13.9

Table 4: **Search behavior** on CSI 300, mean over five runs. Valid: share of candidates passing L1. Adm.: share of candidates that are admissible. Rep.: repeated-failure rate as defined in Table 3. Eval./adm.: tool evaluations per admissible factor (lower is better). Retention: $\mathcal{D}_{\text{ver}}$ RankIC divided by $\mathcal{D}_{\text{srch}}$ RankIC for admitted factors.

Method	Valid (%)	Adm. (%)	Rep. (%)	Eval./adm.	Retention (%)
QuantaAlpha	84.5	7.9	23.8	12.7	71.3
AutoScientist-Quant	88.7	9.4	18.1	10.6	76.9
AlphaAgentEvo	91.2	10.1	15.6	9.9	79.4
Frozen-EvoMind	92.5	10.8	13.9	9.3	81.8
EvoMind	**96.1**	**14.6**	**7.8**	**6.8**	**89.6**

adding typed failures brings the repeated-failure rate down to 7.8%. Random retrieval of the same prompt length gives 0.0487 and 16.8%, so the benefit comes from what is retrieved rather than from a longer prompt.

5.4 Search Behavior (Q3)

Efficiency and repeated failures. In Table 4, EvoMind produces valid factors in 96.1% of evaluations and admissible factors in 14.6%, against 10.1% for AlphaAgentEvo and 9.4% for AutoScientist-Quant. It therefore needs 6.8 evaluations per admissible factor, against 9.9 and 10.6. Its repeated-failure rate of 7.8% is less than half that of AutoScientist-Quant (18.1%), consistent with the negative experience in $\mathcal{M}^-$ steering the policy away from falsified directions.

Overfitting to the feedback window. Admitted EvoMind factors retain 89.6% of their search-window RankIC on the hidden window, against 81.8% for Frozen-EvoMind, 79.4% for AlphaAgentEvo, 76.9% for AutoScientist-Quant, and 71.3% for QuantaAlpha. EvoMind has both the highest retention and the highest test RankIC, and QuantaAlpha the lowest of both, but the methods in between are ordered differently by the two measures: AutoScientist-Quant has the second-highest test RankIC yet the second-lowest retention. Retention is therefore only a coarse indicator of test performance.

6 Limitations

Our experiments cover three equity markets with daily price–volume data; fundamental, textual, and intraday data are not used, and search behavior is analyzed on CSI 300 only. The process rewards depend on thresholds ($q_{\min}$, δ, $u_{\min}$) and weights, and a single verification window can itself be overfit if it is reused across many training iterations; rotating verification windows is a natural extension. RL training requires GPU compute for every training run, far more than prompting a frozen model. Backtests do not model market impact, capacity, or data revisions. Finally, the backbone was released in July 2025, so its pretraining data may overlap the test window (2023–2025); although

the agent never receives test-period data, it may carry prior knowledge about market behavior in that period. The results support a research workflow, not a guarantee of future returns.

7 Conclusion

We presented EvoMind, an alpha factor mining agent whose policy is trained with process-level rewards. Gated hierarchical rewards score each turn for validity, predictive power, novelty, and stability on a hidden verification window; turn-level advantages credit each revision for what follows it; and an experience bank with verifiable failure types lets both successes and failures shape later exploration. Across three markets, EvoMind improves out-of-sample RankIC and annualized excess return over the strongest matched-budget baseline, and on CSI 300 it finds admissible factors with fewer evaluations and fewer repeated failures. Rotating verification windows, richer data sources, and portfolio-level rewards are natural next steps.

Ethics statement

EvoMind uses market-level data and no personal information. Its main risks are financial: automated discovery can amplify backtest overfitting, crowded trades, or unmonitored deployment. We mitigate these risks with point-in-time data, disjoint agent-facing and test windows, look-ahead checks, transaction costs, and risk reporting. Any practical use requires independent validation, exposure and liquidity limits, human approval, and continuous monitoring. Market data were accessed through Qlib and are used under the terms of their providers.

Reproducibility statement

Sections 4.1 to 4.3 define the rewards, the advantage estimator, and the experience bank; Algorithm 1 gives the training procedure; Table 5 lists all hyperparameters; Appendices C and D specify the factor language, data windows, labels, execution timing, costs, and metrics. We will release the code, prompts, DSL specification, and run configurations.

AI use statement

No artificial intelligence or generative AI tools were used in the preparation of this submission. All aspects of the work, including ideation, method design, implementation, experiment design, data processing, analysis, manuscript writing and revision, figure preparation, calculations, and proof development, were performed by the authors. All experimental results were obtained from the authors' own training and evaluation runs. All references were checked against their original sources. The authors take full responsibility for the content of this paper, including its text, claims, analyses, and artifacts.

References

David H. Bailey, Jonathan M. Borwein, Marcos López de Prado, and Qiji Jim Zhu. The probability of backtest overfitting. *Journal of Computational Finance*, 20(4):39–69, 2017.

Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 335–336, 1998.

Angelica Chen, David M. Dohan, and David R. So. EvoPrompting: Language models for code-level neural architecture search. In *Advances in Neural Information Processing Systems*, 2023.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, et al. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. *arXiv preprint arXiv:2505.10978*, 2025.

Arya Grayeli, Atharva Sehgal, Omar Costilla-Reyes, Miles Cranmer, and Swarat Chaudhuri. Symbolic regression with a learned concept library. In *Advances in Neural Information Processing Systems*, 2024.

Jun Han, Shuo Zhang, Wei Li, Zhi Yang, Yifan Dong, Tu Hu, et al. QuantaAlpha: An evolutionary framework for LLM-driven alpha mining. *arXiv preprint arXiv:2602.07085*, 2026.

Campbell R. Harvey, Yan Liu, and Heqing Zhu. . . . and the cross-section of expected returns. *The Review of Financial Studies*, 29(1):5–68, 2016.

Zura Kakushadze. 101 formulaic alphas. *Wilmott*, 2016(84):72–81, 2016.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. *arXiv preprint arXiv:2410.01679*, 2024.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, 2017.

John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

Yuante Li, Xu Yang, Xiao Yang, Minrui Xu, Xisen Wang, Weiqing Liu, and Jiang Bian. R&D-Agent-Quant: A multi-agent framework for data-centric factors and model joint optimization. *arXiv preprint arXiv:2505.15155*, 2025.

Zongqian Li, Yaoyiran Li, Yaohui Guo, Ming Zhang, Nigel Collier, and Eugene Ie. AutoScientist-Quant: Self-evolving coding agents for automatic research in quantitative investment. *arXiv preprint arXiv:2608.28632*, 2026.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, et al. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

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

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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*, 2023.

Ziyi Tang, Zechuan Chen, Jiarui Yang, Jiayao Mai, Yongsen Zheng, Keze Wang, Jinrui Chen, and Liang Lin. AlphaAgent: LLM-driven alpha mining with regularized exploration to counteract alpha decay. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2813–2822, 2025.

Ziyi Tang, Xuexiong Yin, Weixin 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*, 2026.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Peiyi Wang, Lei Li, Zhihong Shao, R. X. Xu, Damai Dai, Yifei Li, Deli Chen, Y. Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2024.

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.

Xiao Yang, Weiqing Liu, Dong Zhou, Jiang Bian, and Tie-Yan Liu. Qlib: An AI-oriented quantitative investment platform. *arXiv preprint arXiv:2009.11189*, 2020.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

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.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. ArCHer: Training language model agents via hierarchical multi-turn RL. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Zhoufan Zhu and Ke Zhu. AlphaQCM: Alpha discovery in finance with distributional reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

A TRAINING PROCEDURE

Algorithm 1 EvoMind training with hierarchical process rewards and turn-level credit

Require: initial policy π_θ, reference model π_{ref}, task set $\mathcal{X}$, experience bank $\mathcal{M} = \mathcal{M}^+ \cup \mathcal{M}^-$, pool $\mathcal{P}$

for each RL iteration **do**
  sample a batch of tasks $\{x\} \subset \mathcal{X}$; set $\pi_{\text{old}} \leftarrow \pi_\theta$
  for each task x and $i = 1, \dots, G$ **do**
    retrieve k^+ records from $\mathcal{M}^+$ and k^- from $\mathcal{M}^-$ by MMR; build the prompt
    for $k = 1, \dots, K$ (or until the policy stops) **do**
      generate a_k (reflection, hypothesis h_k, factors F_k) with π_{old}
      evaluate F_k with the tool: v, q, n on $\mathcal{D}_{\text{srch}}$ (returned as o_k) and u on $\mathcal{D}_{\text{ver}}$ (hidden)
      compute $r(f)$ for $f \in F_k$ (Equation (1)) and r_k; type every candidate by its first violated level
    end for
    compute $R_{\text{out}}(\tau^{(i)})$ (Equation (2))
  end for
  compute returns (Equation (3)) and turn-level advantages (Equation (4))
  update θ by minimizing Equation (5)
  write lessons with π_{ref}; insert deduplicated records into $\mathcal{M}^+$ or $\mathcal{M}^-$
end for

B HYPERPARAMETERS

Table 5: Hyperparameters used in all reported runs.

Group	Parameter	Value
Policy and RL	backbone	Qwen3-4B-Instruct-2507
	supervised warm start	1K trajectories in the agent's output format
	RL steps / tasks per step	600 / 8
	group size G	4
	max. turns K / candidates per turn B	4 / 2
	learning rate	1×10^{-6}
	clipping ϵ_{low} / ϵ_{high}	0.20 / 0.28
	KL coefficient β / max. response length	0.01 / 4096 tokens
Reward	weights $\lambda_1/\lambda_2/\lambda_3/\lambda_4/\lambda_5$	0.2 / 0.4 / 0.2 / 0.8 / 1.0
	reference scale q_{ref}	0.05
	thresholds $q_{\min}$ / δ / $u_{\min}$	0.02 / 0.30 / 0.50
	max. expression size $C_{\max}$ / discount γ	40 nodes / 0.90
	ridge penalty in V_{ver}	10^{-3}
Experience	retrieved records k^+ / k^-	3 / 3
	records per failure type N_c	200
	embedding model / MMR weight	BAAI/bge-small-en-v1.5 / 0.7
Mining	rounds R	10

C FACTOR LANGUAGE AND LOOK-AHEAD CHECKS

Factors are written in a DSL over the raw fields `open`, `high`, `low`, `close`, `volume`, and `vwap`, with element-wise arithmetic, cross-sectional operators (rank, z-score, demeaning), and time-series operators (lagged value, moving mean and standard deviation, rolling correlation, rolling rank, and decay-weighted averages). Every time-series operator takes a window length $d \geq 1$ and reads only days $t - d + 1, \dots, t$. The static check rejects any expression with a non-positive window or lag, any reference to a field outside the list, and any expression with more than $C_{\max} = 40$ nodes. Rejected expressions receive the EXEC type together with the parser message.

D Evaluation Protocol

Windows and labels. All three markets use the same windows: $\mathcal{D}_{\mathrm{tr}}$ from 2016-01-01 to 2019-12-31, $\mathcal{D}_{\mathrm{va}}$ from 2020-01-01 to 2020-12-31, $\mathcal{D}_{\mathrm{srch}}$ from 2021-01-01 to 2021-12-31, $\mathcal{D}_{\mathrm{ver}}$ from 2022-01-01 to 2022-12-31, and $\mathcal{D}_{\mathrm{te}}$ from 2023-01-01 to 2025-12-31. Universes use index membership as of each historical date. Signals are formed after the close of day t, and the label is the return from the open of day $t{+}1$ to the open of day $t{+}2$, which matches the execution time of the strategy.

Predictor and portfolio. For every method, the final library is fed to the same LightGBM predictor, trained on $\mathcal{D}_{\mathrm{tr}}$ with early stopping on $\mathcal{D}_{\mathrm{va}}$. On $\mathcal{D}_{\mathrm{te}}$ the strategy buys the 50 highest-scored stocks at the open of day $t{+}1$, replaces at most 5 holdings per trading day, and pays a one-way transaction cost of 10 bps. Excess returns are measured against the index of each market.

Metrics. RankIC is the mean daily Spearman correlation between predicted scores and labels, and RankICIR is its mean divided by its standard deviation; IC and ICIR are the Pearson counterparts. ARR is the annualized mean of daily excess returns after costs, IR is ARR divided by the annualized standard deviation of daily excess returns, MDD is the largest peak-to-trough decline of cumulative excess return, and the Calmar ratio is ARR divided by MDD.

Information boundary. Tool responses contain only $\mathcal{D}_{\mathrm{srch}}$ quantities. $\mathcal{D}_{\mathrm{ver}}$ enters only the stability level, the outcome reward, and pool admission, and the experience bank stores no $\mathcal{D}_{\mathrm{ver}}$ values. $\mathcal{D}_{\mathrm{te}}$ is used once, after the library, the predictor, and all hyperparameters are fixed.

E Full Results

Table 6: All eight metrics on CSI 300, mean $\pm$ standard deviation over five runs.

Method	IC	ICIR	RIC	RICIR	ARR (%)	IR	MDD (%)	Calmar
AlphaAgent	0.0430 ± 0.0018	0.62 ± 0.05	0.0417 ± 0.0016	0.60 ± 0.05	5.1 ± 0.8	0.72 ± 0.08	11.5 ± 1.1	0.44 ± 0.08
QuantaAlpha	0.0468 ± 0.0016	0.72 ± 0.04	0.0456 ± 0.0015	0.70 ± 0.04	6.0 ± 0.7	0.83 ± 0.07	10.7 ± 0.9	0.56 ± 0.08
AutoScientist-Quant	0.0491 ± 0.0015	0.79 ± 0.04	0.0479 ± 0.0014	0.77 ± 0.04	6.7 ± 0.7	0.91 ± 0.07	9.8 ± 0.8	0.68 ± 0.09
AlphaAgentEvo	0.0484 ± 0.0017	0.77 ± 0.05	0.0472 ± 0.0016	0.75 ± 0.05	6.5 ± 0.8	0.88 ± 0.08	10.0 ± 0.9	0.65 ± 0.09
EvoMind	0.0541 ± 0.0013	0.91 ± 0.04	0.0528 ± 0.0012	0.89 ± 0.03	8.4 ± 0.6	1.10 ± 0.06	8.6 ± 0.7	0.98 ± 0.08