

000 001 ALPHAAGENTEVO: EVOLUTION-ORIENTED ALPHA 002 MINING VIA SELF-EVOLVING AGENTIC REINFORCE- 003 MENT LEARNING 004 005

006 **Anonymous authors**

007 Paper under double-blind review
008
009
010

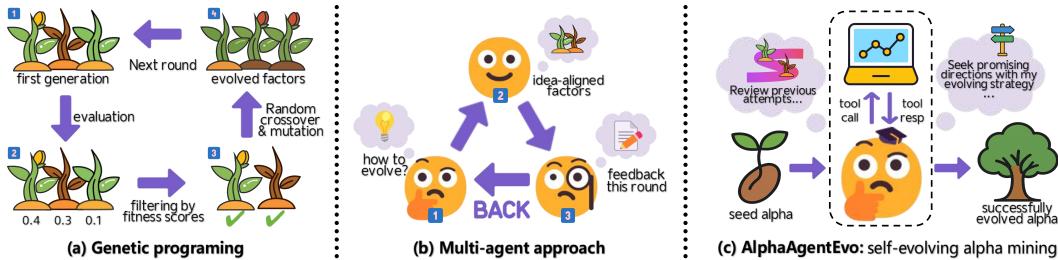
011 ABSTRACT 012

013 *Alpha mining* seeks to identify predictive alpha factors that generate excess re-
014 turns beyond the market from a vast and noisy search space; however, existing
015 *evolution-based* approaches struggle to facilitate the systematic *evolution* of al-
016 phas. Traditional methods, such as genetic programming, cannot interpret hu-
017 man natural-language instructions and often fail to extract valuable insights from
018 unsuccessful attempts, leading to low interpretability and inefficient exploration.
019 Analogously, without mechanisms for systematic *evolution*, e.g., long-term plan-
020 ning and reflection, existing multi-agent approaches may easily fall into repetitive
021 evolutionary routines, ending up with inefficient *evolution*. To overcome these
022 limitations, we introduce **AlphaAgentEvo**, a self-evolving Agentic Reinforce-
023 ment Learning (ARL) framework for *alpha mining*, which moves *alpha mining*
024 beyond the brittle “search–backtest–restart” cycle toward a continuous trajectory
025 of *evolution*. Instructed by a hierarchical reward function, our agent engages in
026 self-exploration of the search space, progressively learning basic requirements
027 (e.g., valid tool calls) and then harder objectives (e.g., continuous performance
028 improvements). Through this process, the agent acquires advanced behaviors such
029 as long-horizon planning and reflective reasoning, which enable it to actively re-
030 act to the underlying state (e.g., market regime shifting) and realize a self-evolving
031 agent, taking a step toward more principled and scalable *alpha mining*. Extensive
032 experiments demonstrate that AlphaAgentEvo achieves more efficient *alpha evo-
033 lution* and generates more diverse and transferable *alphas*, consistently surpassing
034 a wide range of baselines. Notably, with only 4B parameters, it outperforms LLM-
035 driven evolution methods configured with state-of-the-art close-source reasoning
036 models, highlighting the promise of ARL for next-generation *alpha mining*.
037
038

039 1 INTRODUCTION 040

041 *Alpha mining*, which refers to uncovering quantitative signals that generate excess returns beyond the
042 market, remains highly challenging due to its vast search space, high-variance feedback signals, spu-
043 rious correlations, and high computational cost. These issues make the process both computa-
044 tionally intensive and prone to false discoveries, highlighting the need for more systematic approaches.
045 In this context, the notion of *alpha evolution* becomes central: instead of viewing each candidate
046 *alpha* as an independent trial, the process emphasizes progressively transforming an initial *alpha*
047 into a stronger one through multi-turn interaction that incorporates evaluative feedback, refines its
048 structure, and enhances its performance. This evolution-oriented perspective not only increases the
049 likelihood of uncovering genuinely effective *alpha factors* (hereafter, *alphas* for simplicity) but also
050 helps retain their internal logic and interpretability across interaction turns, offering a principled al-
051 ternative to ad hoc trial-and-error exploration. Yet despite this promise, existing approaches struggle
052 to effectively realize such systematic *alpha evolution*.
053

054 Traditional evolution-oriented approaches, such as Genetic Programming (GP) (Lin et al., 2019;
055 Schmidt & Lipson, 2010; Zhaofan et al., 2022), rely heavily on heuristic search and random muta-
056 tion Ren et al. (2024), without leveraging valuable insights from negative feedback, thereby missing
057 opportunities to learn from failures and refine *alpha* design. Moreover, these methods cannot inter-
058

Figure 1: Comparison of evolution-oriented approaches in *alpha mining*.

pret their evolution process with human-readable language, which not only limits their usability but also increases the risk of generating *alphas* that capture spurious correlations (Shi et al., 2025b).

Emerging Large Language Models (LLMs) (Yang et al., 2025; DeepSeek-AI et al., 2025; OpenAI, 2025) and multi-agent frameworks (Tang et al., 2025; Li et al., 2025) bring richer human instructions but often lack mechanisms for *self-evolution*, such as long-term planning and reflective reasoning from past outcomes. As a result, they tend to fall into repetitive local modifications, leading to inefficient exploration. Consequently, current *alpha mining* workflows remain myopic: they *search, backtest, and restart* rather than systematically evolve *alphas*. This gap calls for an evolution-oriented paradigm that couples deliberate planning and reflective reasoning to refine *alphas* over multi-turn trajectories.

To overcome these limitations, we introduce **AlphaAgentEvo**, the first self-evolving Agentic Reinforcement Learning (ARL) framework for *alpha mining*. First, AlphaAgentEvo aims to move *alpha mining* beyond the brittle “search–backtest–restart” cycle toward a continuous interaction trajectory, dubbed as *evolution*, encouraging progressively refined *alphas* by searching diverse *alphas* grounded in a given seed. During the ARL process, an LLM-driven agent engages in large-scale exploration of the *alpha* search space, interacting with an evaluation tool under the guidance of a hierarchical reward structure. With this reward, the agent explicitly learns to satisfy basic requirements (e.g., constructing valid *alphas*) and then moves on to harder objectives (e.g., performance and streak improvements). More importantly, the agent acquires advanced behaviors, such as long-horizon planning and reflective reasoning, that enable it to actively react to the underlying state (e.g., market regime shifting). Ultimately, AlphaAgentEvo gives rise to a self-evolving agent, taking a step toward more principled and scalable *alpha mining*.

Results show that AlphaAgentEvo, at a relatively lightweight scale of 4B parameters, consistently outperforms GP, multi-agent, tool-use RL, and LLM-driven evolution baselines by a significant margin. Moreover, AlphaAgentEvo generates diverse and transferable *alphas*, avoiding over-exploitation of specific patterns and showing stronger out-of-sample robustness. These results underscore the superiority of our *self-evolving* ARL paradigm for evolution-oriented *alpha mining*.

Our key contributions are summarized as follows:

- We propose a novel framework, **AlphaAgentEvo**, that reformulates *alpha mining* from a brittle “search–backtest–restart” loop into an evolution-oriented paradigm. To the best of our knowledge, this is the **first** self-evolving agentic reinforcement learning framework for quantitative *alpha mining*.
- In this paradigm, an LLM-driven agent performs multi-turn exploration of the *alpha* search space under a hierarchical reward function, gradually acquiring advanced behaviors, such as long-horizon planning and reflective reasoning, that enable it to evolve *alphas* across changing market regimes, thereby overcoming performance bottlenecks in dynamic markets and realizing a self-evolving agent.
- Through extensive experiments, our method demonstrates strong evolution efficiency and the generated *alphas*’ validity, diversity, and transferability, demonstrating strong generalization in the self-evolving agent. Remarkably, even with only 4B parameters, AlphaAgentEvo surpasses strong baselines with state-of-the-art LLMs on several metrics.

108

2 METHOD

109

2.1 PROBLEM DEFINITION

110 An *alpha factor* (or simply *alpha*) is a quantitative signal designed to predict future stock returns. We consider a stock universe $\mathcal{S} = \{s_1, \dots, s_N\}$ over a time horizon $\mathcal{H} = \{h_1, \dots, h_L\}$, with a feature matrix $\mathbf{X} \in \mathbb{R}^{N \times L \times d}$ where d denotes the number of raw features. An *alpha* is a mapping $f : \mathbf{X}_h \mapsto r_{h+1}$, where \mathbf{X}_h represents the market data observed up to and including time h and r_{h+1} is the subsequent return. Beyond static alpha mining, we view *alpha evolution* as *learning an evolution policy* instead of directly optimizing a single *alpha*. Let $\mathcal{D}_{\text{seed}}$ denote a distribution over expert-designed seed alphas. For a given seed $f_{\text{seed}} \sim \mathcal{D}_{\text{seed}}$, the policy π interacts with the backtesting tool for T turns and produces evolved *alphas* $\mathcal{F}_{\pi}(f_{\text{seed}})$. We evaluate each evolved factor on two market distributions: \mathcal{D}_{evo} , which corresponds to the in-distribution regimes observed during the agent’s multi-turn evolution, and $\mathcal{D}_{\text{test}}$, which captures out-of-distribution market regimes. Formally, we learn the evolution policy by

$$111 \pi^* = \arg \max_{\pi} \mathbb{E}_{f_{\text{seed}} \sim \mathcal{D}_{\text{seed}}} \left[\max_{f \in \mathcal{F}_{\pi}(f_{\text{seed}})} \left(\mathbb{E}_{\mathbf{X} \sim \mathcal{D}_{\text{evo}}} s(f; \mathbf{X}) + \lambda \mathbb{E}_{\mathbf{X} \sim \mathcal{D}_{\text{test}}} s(f; \mathbf{X}) \right) \right] \quad (1)$$

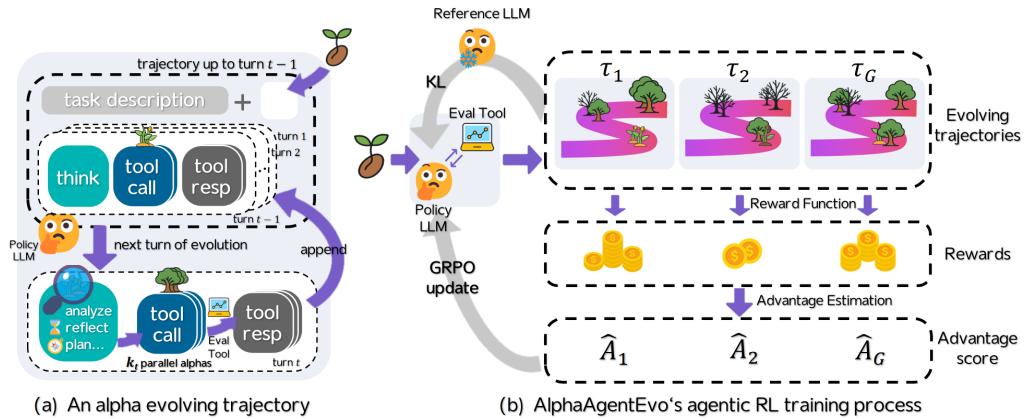
$$112 \text{ s.t. } \text{sim}(f, f_{\text{seed}}) \leq \delta \quad \text{for all } f \in \mathcal{F}_{\pi}(f_{\text{seed}}),$$

113 where $s(\cdot; \mathbf{X})$ is the performance scoring function on market features \mathbf{X} , $\lambda > 0$ trades off in-distribution fitness and out-of-distribution generalization, and $\text{sim}(\cdot, \cdot)$ is an AST-based structural 114 similarity between alphas. The constraint $\text{sim}(f, f_{\text{seed}}) \leq \delta$ leads the policy π to search in a local 115 neighborhood of each seed, producing evolved alphas that are both stronger and still interpretable, 116 rather than overfitting to noise via unconstrained global optimization. 117

118

2.2 SELF-EVOLVING AGENTIC REINFORCEMENT LEARNING

119 Existing RL-based finetuning approaches are typically designed for single-turn language modeling 120 or reasoning, where evaluation is per response and cross-turn coupling is weak. In contrast, *alpha 121 evolution* is inherently a multi-turn tool-in-the-loop process. To realize a self-evolving agent, we 122 extend GRPO (Shao et al., 2024; DeepSeek-AI et al., 2025), into an Agentic Reinforcement Learning 123 (ARL) formulation that directly optimizes the policy LLM in the *think-propose-evaluate* loop with 124 an external evaluation tool \mathcal{U} . 125



126 Figure 2: The ARL framework of AlphaAgentEvo. (a) An *alpha evolving trajectory*. In each turn 127 of the trajectory, the agent (policy LLM) generates multiple tool calls (i.e., *alpha proposals*) after 128 analyzing/reflecting on previous *alphas* and their feedback (tool resp). (b) The multi-turn on-policy 129 RL training process: each evolving trajectory is assigned a reward through the reward function. 130 Trajectories that originate from the same seed *alpha* are grouped, and their rewards are jointly used 131 for advantage estimation and agent update.

132 In our formulation, each turn consists of policy-generated reasoning tokens and tool call tokens that 133 trigger the tool, followed by tool response tokens (tool resp in Fig.2); all are appended to the trajec- 134 tory, but only policy-generated tokens (indicated by $M_{i,t}$) contribute to gradients. To broaden explo- 135

ration, at turn t , the policy LLM produces a set of k_t parallel offspring $\mathcal{F}^{(t)}$ as candidates, which are jointly evaluated by \mathcal{U} . Note that when generating turn t 's tokens, the policy LLM conditions on the entire past trajectory $\tau_{1:t-1}$ to refine *alphas*, enabling reflective reasoning across previous attempts.

For each input x sampled from the dataset D , a group of trajectories $\mathcal{T} = \{\tau_1, \dots, \tau_G\}$ are rolled out by the old policy π_{old} . Their rewards are normalized within the group to estimate their relative advantages $\{\hat{A}_1, \dots, \hat{A}_G\}$ using the group average reward as a baseline, i.e., $\hat{A}_g = \frac{R(\tau_g) - \mu_{\mathcal{T}}}{\sigma_{\mathcal{T}}}$, where $\mu_{\mathcal{T}}$ and $\sigma_{\mathcal{T}}$ denote the mean and standard deviation of $\{R(\tau_j)\}_{j=1}^G$. The optimization objective is

$$\begin{aligned} J_{\text{GRPO}}(\theta) = \mathbb{E}_{x \sim D, \mathcal{T} = \{\tau_1, \dots, \tau_G\} \sim \pi_{\text{old}}} \left[\frac{1}{G} \sum_{i=1}^G \frac{1}{\sum_t M_{i,t}} \sum_t M_{i,t} \min \left(\frac{\pi_{\theta}(\tau_{i,t} | x, \tau_{i,<t}, \mathcal{U})}{\pi_{\text{old}}(\tau_{i,t} | x, \tau_{i,<t}, \mathcal{U})} \hat{A}_{i,t}, \right. \right. \\ \left. \left. \text{clip} \left(\frac{\pi_{\theta}(\tau_{i,t} | x, \tau_{i,<t}, \mathcal{U})}{\pi_{\text{old}}(\tau_{i,t} | x, \tau_{i,<t}, \mathcal{U})}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_{i,t} \right) - \beta \mathbb{D}_{\text{KL}}[\pi_{\theta} \| \pi_{\text{ref}}] \right], \end{aligned} \quad (2)$$

where τ_i is a complete evolution trajectory, $\tau_{i,t}$ its t -th token, and $\pi_{\theta}(\cdot) / \pi_{\text{old}}(\cdot)$ are the current/old policies conditioned on x , past tokens $\tau_{i,<t}$, and the tool \mathcal{U} . The mask $M_{i,t}$ excludes tool-emitted tokens from gradients, $\frac{1}{\sum_t M_{i,t}}$ normalizes for effective length, $\text{clip}(\cdot)$ is ratio clipping with parameter ϵ , and \mathbb{D}_{KL} is a KL penalty to a reference policy π_{ref} with weight β .

In summary, our ARL formulation adapts GRPO from single-turn text optimization to interactive, multi-turn *alpha evolution*. This enables the model to plan, analyze, and reflect throughout a long trajectory that progressively evolves *alphas* beyond the heuristic “search–backtest–restart” cycle.

2.3 ALPHA EVOLUTION REWARD FUNCTION

Unlike domains such as mathematics or generic tool use, *alpha mining* is central to quantitative investment but remains highly challenging due to its vast search space, noisy data with spurious correlations, and high computational cost, making it infeasible to rely on a single scalar reward. Moreover, the logical semantics of *alphas* and the need for efficient search are critical for effective alpha mining but rarely considered in other reasoning tasks. These challenges motivate a hierarchical reward function that enables principled *alpha evolution* throughout multiple turns, where AlphaAgentEvo integrates multiple objectives: ensuring valid *alpha* candidates, preventing excessive deviation from the seed *alpha*, promoting diversity exploration, rewarding performance improvements, and sustaining progress across turns.

Starting from an individual component, **Tool Call Reward** (R_{tool}) provides feedback on correct tool usage and penalizes failed attempts, defined as $R_{\text{tool}}(\tau) = \alpha_{\text{succ}} \cdot N_{\text{succ}} - \alpha_{\text{fail}} \cdot N_{\text{fail}}$, where N_{succ} and N_{fail} denote the number of successful and failed tool calls, respectively. Here, each α denotes the corresponding weighting coefficient (e.g., α_{succ}). Next, for direction-aware *alpha* generation, **Consistency Reward** (R_{cons}) prevents excessive deviation from the seed *alpha* by penalizing candidates whose structural similarity $\text{sim}(f_i, f_{\text{seed}})$ falls below a lower threshold h_{low} (we set to 0.1 in our experiments), i.e., $R_{\text{cons}}(\tau) = \sum_{f_i \in \mathcal{F}_{\text{succ}}(\tau)} \alpha_{\text{cons}} \cdot \mathbb{1}[\text{sim}(f_i, f_{\text{seed}}) > h_{\text{low}}]$, where $\mathbb{1}[\cdot]$ is an indicator function. This serves as a soft constraint, preventing random drifts that may harm interpretability. **Exploration Reward** (R_{expl}) encourages diversified exploration by rewarding *alphas* whose similarity to previously proposed ones remains low:

$$R_{\text{expl}}(\tau) = \sum_{f_i \in \mathcal{F}_{\text{succ}}(\tau)} \alpha_{\text{expl}} \cdot \left(1 - \max_{f_j \in \mathcal{F}_{<i}(\tau)} \text{sim}(f_i, f_j) \right), \quad (3)$$

where $\mathcal{F}_{\text{succ}}(\tau)$ is the set of successfully runnable offspring *alphas* in trajectory τ , $\mathcal{F}_{<i}(\tau)$ denotes all factors proposed before f_i (including the seed). The structural similarity score $\text{sim}(\cdot, \cdot)$ measured by Abstract Syntax Tree (AST) overlap (Tang et al., 2025) is written as:

$$\text{sim}(f_i, f_j) = \frac{|\text{AST}(f_i) \cap \text{AST}(f_j)|}{\max(|\text{AST}(f_i)|, |\text{AST}(f_j)|)}. \quad (4)$$

To encourage performance improvement while handling noisy metrics, **Performance Reward** (R_{perf}) uses a logarithmic scaling $\alpha_{\text{perf}} \cdot \log(1 + \exp(s(f^*) - \max(0, s(f_{\text{seed}}))))$. Finally, **Streak**

216 **Reward** (R_{streak}) provides an additional bonus $\alpha_{\text{streak}} \cdot N_{\text{streak}}$, where N_{streak} denotes the length of
 217 the longest sequence of progressive performance improvements within a trajectory, serving as a
 218 booster towards efficient *alpha evolution*.

219 Totally, the hierarchical reward of a trajectory τ is
 220

$$R(\tau) = \frac{\min(R_{\text{cons}}(\tau), C_{\text{cons}}) + \min(R_{\text{expl}}(\tau), C_{\text{expl}})}{\min(R_{\text{tool}}(\tau), C_{\text{tool}})} + \min(R_{\text{perf}}(\tau), C_{\text{perf}}) \cdot \min(R_{\text{streak}}(\tau), C_{\text{streak}}), \quad (5)$$

223 where each component reward $R_j(\tau)$ is capped by its corresponding C_j (e.g., C_{tool}) to avoid any
 224 single term from overwhelming the total reward. The tool-use term treats each tool call as a cost,
 225 preventing brute-force search through frequent tool calls and encouraging meaningful and efficient
 226 alpha evolution. This hierarchical reward structure transforms the sparse and noisy feedback from
 227 financial backtesting into dense, multi-dimensional signals. By balancing direction-aware consist-
 228 ency and exploration (normalized by tool usage) and integrating performance with sustained im-
 229 provement through a multiplicative term, it progressively guides the agent from basic compliance to
 230 higher-level objectives, ultimately preventing collapse into repetitive patterns and enabling efficient
 231 *alpha evolution* Wang et al. (2025).

233 3 EXPERIMENTS

235 In this section, we present the main experimental results, while **we strongly encourage readers to**
 236 **refer to Sec. E in the Appendix for a detailed case analysis** that provides a deeper understanding
 237 of the agent’s *self-evolution* process. For details on the training statistics, please refer to Sec. C; for
 238 training configurations and the evaluation tool, please refer to Sec. D and Sec. F in the Appendix.

240 3.1 EXPERIMENT SETTINGS

242 **Datasets.** To systematically evaluate *alpha evolution* ability, we construct an expert-curated
 243 dataset, referred to as *AlphaEvo500*, which serves as an *alpha evolution* benchmark in this study. It
 244 consists of 350 seed *alphas* for training, 50 for validation, and 100 for testing, enabling a controlled
 245 yet diverse setting for evolutionary experiments. To further assess the generalization ability, we
 246 additionally incorporate *Alpha158* (Yang et al., 2020) as an extra test set.

247 **Backtesting settings.** Backtesting is conducted on the HS300 and CSI500 markets, spanning from
 248 January 2023 to November 2025, and covering both bearish and bullish market conditions. For
 249 model training, only one year of market data (2023-01-01 to 2024-01-01) is used to accelerate
 250 iterations. For evaluation, *alpha evolution* experiments are performed on two distinct periods: 2023-
 251 01-01 to 2024-01-01 (bearish) and 2024-01-01 to 2025-01-01 (bullish), with test split’s *alphas* as
 252 the seeds. It should be noted that no data information is disclosed from the evaluation tool or the
 253 prompt, covering both the market and the time range. We adopt a single-factor evaluation protocol,
 254 where the cross-sectional values of each *alpha* are treated as signals, without extra processing. In
 255 each rebalancing period, we long at most the top 10% of stocks in the universe. The rebalancing
 256 frequency is set to every 5 trading days. Unless otherwise specified, other experimental settings are
 257 described at the beginning of each subsection. For detailed data and operators that we use in this
 258 paper, please see Sec. G in the Appendix.

259 **Evaluation metrics.** We evaluate the capability of AlphaAgentEvo to evolve *alphas* by com-
 260 puting the pass rate at the third and fifth turns, denoted as *pass@3* and *pass@5*, respectively. For
 261 performance measurement, we adopt the Information Ratio (IR) as $s(f)$, which quantifies risk-
 262 adjusted excess return. Given a seed *alpha* f_{seed} with score $s(f_{\text{seed}})$, a generated *alpha* is regarded
 263 as successful if its score is higher than the seed and non-negative. Formally, the pass rate at turn
 264 T is defined as:

$$\text{pass}@T = \frac{1}{N} \sum_{j=1}^N \mathbb{1}_{\left[\max_{f \in \bigcup_{t=1}^T \mathcal{F}^{(t)}} s(f) > \max(0, s(f_{\text{seed}}^{(j)})) \right]}, \quad (6)$$

265 where N is the number of test cases, $\mathcal{F}^{(t)}$ is the set of evolved *alphas* at turn t , and the indicator $\mathbb{1}[\cdot]$
 266 equals 1 if the success condition is satisfied and 0 otherwise. In addition, we report the *valid ratio*

(VR), which measures the percentage of generated *alphas* that are syntactically valid and executable in backtesting, reflecting the reliability of the generation process.

For *alpha* performance measurement, we adopt the Annualized Excess Return (AER) to quantify the yearly excess investment return relative to the benchmark index and the Information Ratio (IR) to measure the risk-adjusted performance, which calculates the ratio between the AER and the annualized standard deviation of an *alpha*’s daily excess returns. With these evaluation metrics, we can comprehensively assess an *alpha*’s profitability and risk-bearing capacity. We do not rely on cross-sectional correlation-based metrics (such as information coefficient or IC), since some of the seed *alphas* are designed as stock-selection signals in the form of Boolean expressions, where their values for unselected stocks are set to NaN, making these measures unreliable.

Compared methods. We compare against four categories of baselines: (i) Genetic Programming (GP) with (Lin et al., 2019; Schmidt & Lipson, 2010; Zhaoan et al., 2022; Patil, 2023) 4, 20, and 50 offspring per generation, representing traditional heuristic search; (ii) **LLM-driven evolutionary frameworks**, AlphaAgent (Tang et al., 2025) (multi-agent evolution) and GEPA Agrawal et al. (2025) (reflective prompt evolution); (iii) a series of **reasoning LLMs** that evolve *alphas* via our unified pipeline, incorporating our base models (i.e., Qwen3-1.7B and Qwen3-4B-thinking (Yang et al., 2025)) and state-of-the-art models (GPT-5-mini (OpenAI, 2025), DeepSeek-R1 (Guo et al., 2025)); (iv) a **tool-use RL approach** ToolRL Qian et al. (2025). All methods share the same back-testing tool, training tool call budgets, and inference tool call budget of 4 offspring per turn (unless otherwise noted), ensuring a fair comparison. We also compare multi-factor portfolio performance with global optimization approaches Zhu & Zhu (2025); Fan & Shen (2024); Ke et al. (2017) (Sec. B, Appendix).

3.2 ALPHA EVOLUTION PERFORMANCE

To evaluate whether AlphaAgentEvo can consistently outperform existing evolution-oriented baselines in terms of *alpha evolution* capability, we first focus on VR and the pass rate, as illustrated previously. By evaluating on *AlphaEvo500* and *Alpha158* *alpha* libraries across two periods, we further test the generalization ability of our approach under distinct market regimes. Note that AlphaAgentEvo models are trained with at most 3 turns.

Results on *AlphaEvo500*. Table 1 shows the results in the HS300 and CSI500 markets. While GP’s expression system is incompatible with *AlphaEvo500*, we are unable to test GP here. In terms of pass rates, Qwen3 and GPT-5-mini offer only limited improvements, and DeepSeek-R1 performs more strongly but inconsistently. For an agentic RL baseline, ToolRL, its Pass@3 remains the same level as GPT-5-mini, but fails to generalize to a longer horizon due to the shortcomings in multi-turn planning. By contrast, AlphaAgentEvo achieves clear superiority: even the 1.7B version surpasses GPT-5-mini, while the 4B model outperforms the strongest baseline GEPA and attains the best overall results. These findings demonstrate that our *self-evolving* agent not only effectively generalizes to different market regimes but also to longer evolution trajectories.

Table 1: Performance comparison on *AlphaEvo500* across two markets during 2024-2025.

Method	HS300			CSI500		
	VR	Pass@3	Pass@5	VR	Pass@3	Pass@5
Qwen3-1.7B (Yang et al., 2025)	0.676	0.08	0.11	0.657	0.35	0.43
Qwen3-4B-thinking (Yang et al., 2025)	0.942	0.36	0.47	0.951	0.68	0.78
GPT-5-mini (OpenAI, 2025)	0.970	0.75	0.88	0.972	0.73	0.82
DeepSeek-R1 (Guo et al., 2025)	0.872	0.68	0.71	0.886	0.71	0.86
ToolRL-1.7B Qian et al. (2025)	0.864	0.74	0.78	0.851	0.66	0.74
ToolRL-4B Qian et al. (2025)	0.954	0.75	0.81	0.961	0.73	0.76
GEPA (Agrawal et al., 2025) (GPT-5-mini)	0.992	<u>0.87</u>	<u>0.90</u>	0.971	<u>0.86</u>	<u>0.91</u>
GEPA (Agrawal et al., 2025) (DeepSeek-R1)	0.977	0.83	0.87	0.978	0.82	0.88
AlphaAgentEvo-1.7B (<i>ours</i>)	0.940	0.77	<u>0.90</u>	0.923	0.76	0.78
AlphaAgentEvo-4B (<i>ours</i>)	0.979	0.97	0.97	0.977	0.93	0.95

Results on *Alpha158*. Table 2 reports the results on the external *Alpha158* library, which serves as an additional test set. GP reveals poor results with limited offspring size, while the 4-offspring setting is aligned with all other approaches. This highlights the inefficiency of purely heuristic

search. The multi-agent framework AlphaAgent improves upon GP, yielding higher VR and reasonable pass rates, particularly with stronger backbone models. However, AlphaAgentEvo again achieves the most consistent improvements. AlphaAgentEvo-1.7B’s VR remains above 0.91 across both market periods, while the 4B version shows a striking advantage in pass rates, with pass@5 exceeding 0.72 in the bearish period and reaching 0.994 in the bullish period, nearly saturating the success rate. This demonstrates not only superior evolutionary efficiency but also strong adaptability to different market styles and factor libraries.

Table 2: Performance comparison on *Alpha158* across two periods.

Method	2023-01 – 2024-01			2024-01 – 2025-01		
	VR	Pass@3	Pass@5	VR	Pass@3	Pass@5
GP (4 offspring)	0.766	0.000	0.074	0.823	0.003	0.003
GP (20 offspring)	0.714	0.000	0.058	0.713	0.125	0.132
GP (50 offspring)	0.619	0.022	0.024	0.633	0.094	0.107
AlphaAgent (Tang et al., 2025) (GPT-3.5-turbo)	0.905	0.236	0.495	0.900	0.643	0.783
AlphaAgent (Tang et al., 2025) (DeepSeek-R1)	0.975	0.294	0.550	0.966	0.750	0.848
Qwen3-1.7B (Yang et al., 2025)	0.714	0.100	0.113	0.674	0.500	0.543
Qwen3-4B-thinking (Yang et al., 2025)	0.792	0.350	0.450	0.974	0.848	0.856
DeepSeek-R1 (Guo et al., 2025)	0.889	0.327	0.519	0.874	0.872	0.943
GPT-5-mini (OpenAI, 2025)	0.988	0.156	0.293	0.975	0.828	0.903
AlphaAgentEvo-1.7B (<i>ours</i>)	0.952	<u>0.506</u>	0.613	0.917	<u>0.909</u>	0.926
AlphaAgentEvo-4B (<i>ours</i>)	0.982	0.581	0.725	0.982	0.963	0.994

Summary. Across both datasets and market conditions, AlphaAgentEvo exhibits clear superiority over a wide range of strong baselines. Its ability to sustain both high valid ratios and high pass rates underscores the effectiveness of our *self-evolving* agent paradigm. Importantly, these results validate that the proposed framework not only accelerates the discovery of profitable *alphas* but also reduces invalid generations and enhances robustness under dynamic market environments.

3.3 EVOLUTION ANALYSIS.

To probe into how AlphaAgentEvo’s agent and its generated *alphas* evolve across turns, in Fig. 3, we visualize its IR trajectory and per-turn standard deviation (Std.) along with average consistency score against ToolRL Qian et al. (2025) on AlphaEvo500. For better visual clarity, we display the uncertainty band as $\pm 0.25 \times \text{std.}$ In Fig. 3 (a), our approach’s mean IR increases more rapidly than ToolRL, leading to a continually widening gap. Fig. 3 (b) shows that AlphaAgentEvo more aggressively explores in the search space as the agent continues to mine *alphas*, resulting in a higher standard deviation. Moreover, AlphaAgentEvo stays anchored to seed *alphas*, ensuring that improvements accumulate progressively rather than degenerating into uncontrolled search. These patterns cannot be explained by alpha-level evolution alone. The accelerating IR gains across turns, together with the simultaneous rise in exploration and stabilized consistency, indicate that the agent’s strategy in each turn is evolving (improving over time). These experience-dependent behaviors constitute clear evidence of agent-level self-evolution, rather than merely evolving individual *alphas*.

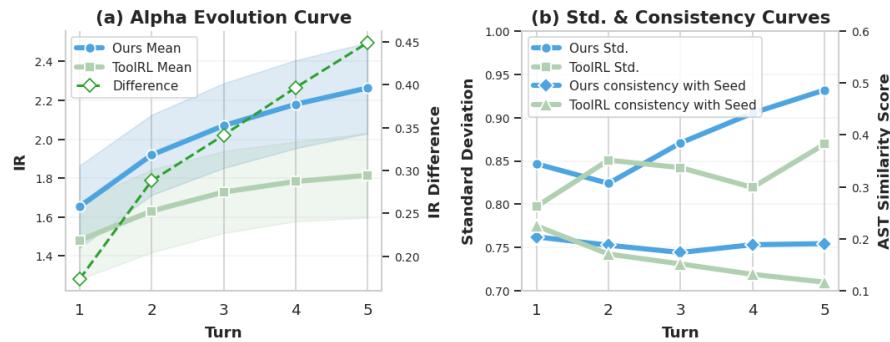


Figure 3: Comparison between AlphaAgentEvo and ToolRL’s evolution trajectories.

3.4 ABLATION STUDY

To verify the effectiveness of our ARL training and reward design, we compare validity rates before and after training and ablate two key reward components. As shown in Fig. 4(a), training markedly improves validity, confirming the model’s ability to generate well-formed *alphas*. In Fig. 4(b)–(c), removing either the exploration or direction-aware reward lowers pass rates on both datasets, with

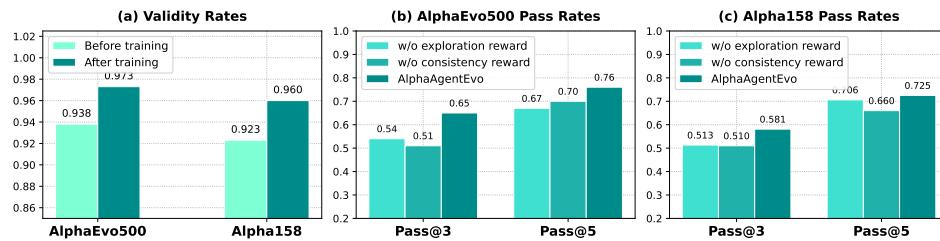


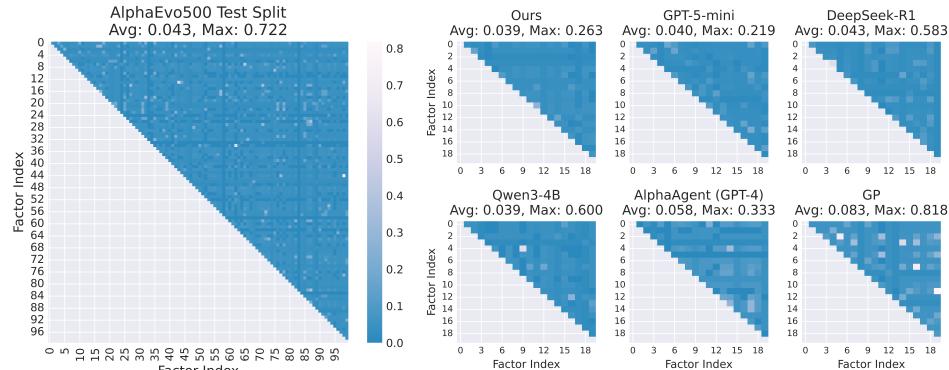
Figure 4: Ablation study on reward components.

the largest drops at pass@3 (AlphaEvo500: 0.65→0.54/0.51; Alpha158: 0.581→0.513/0.510). These results show that exploration and direction-awareness are both critical and complementary for efficient *alpha evolution*.

3.5 DIVERSITY AND TRANSFERABILITY OF ALPHAS

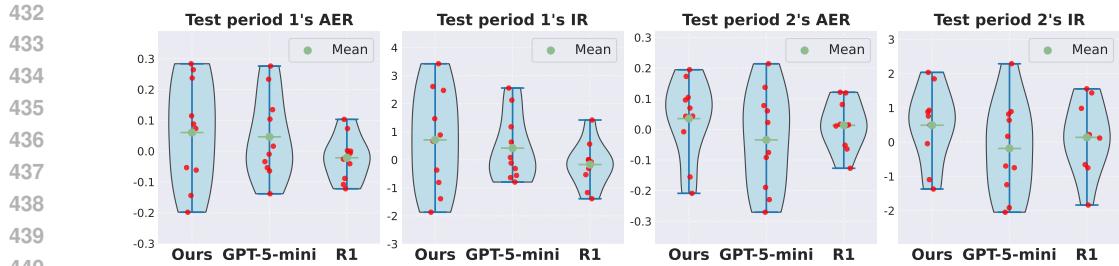
To investigate whether models suffer from reward hacking (Wang et al., 2025) by overexploiting specific patterns, we analyze the structural similarity distribution of the top 20 generated *alphas* (evaluated on the *alpha evolution* period) using Eq. 4. Note that for GP, *Alpha158* serves as seeds due to an incompatible *alpha* calculation system; therefore, its results are provided solely for reference. For AlphaAgent, we reset its *alpha zoo* for each seed *alpha*.

In Fig. 5, the seed *alpha* library (*AlphaEvo500* test split) exhibits a broad similarity distribution, with an average pairwise similarity of 0.043 and a relatively high maximum similarity of 0.722, reflecting the presence of clusters of closely related *alphas* despite overall diversity. When comparing different models, our method achieves both a low average similarity (0.039) and a low maximum similarity (0.263), indicating that the generated *alphas* are more diverse. By contrast, models such as DeepSeek-R1 and Qwen3-4B tend to produce *alphas* with higher maximum similarity (0.583 and 0.600, respectively), suggesting partial over-concentration on specific patterns. AlphaAgent (GPT-4) also shows an elevated average similarity (0.058), suggesting that for different seed *alphas*, it may repeatedly fall into local optima with limited diversity. These statistics highlight a key advantage of our approach: **it does not overexploit narrow or spurious patterns and instead learns genuinely generalizable evolutionary strategies.**

Figure 5: Similarity scores of top 20 *alphas* generated by different models.

To assess the out-of-sample performance of *alphas* from different LLMs, we collect two groups of evolved *alphas* from each LLM. While one group uses market data from 2023-01-01 to 2024-01-01 as an evolution period, the other uses 2024-01-01 to 2025-01-01 to evolve *alphas*. Then, the first group undergoes backtesting from 2024-01-01 to 2025-01-01 (test period 1), and the second group from 2025-01-01 to 2025-06-01 (until the datasets are created), noted as test period 2. These evolved *alphas* are sampled from each model’s top-20 candidates without altering their original order **such that the selected subsets share the same average IR during the evolution period**. Specifically, the average evolution-period IR is 1.05 for test period 1 and 2.72 for test period 2.

Compared to two state-of-the-art LLMs, GPT-5-mini and DeepSeek-R1 (R1), our AlphaAgentEvo-4B demonstrates highly competitive performance, consistently achieving a higher average AER and

Figure 6: *Alpha* performance comparison with state-of-the-art LLMs on out-of-sample periods.

average IR across test periods 1 and 2. This indicates that its evolution strategy exhibits favorable out-of-sample generalization and superiority under diverse market conditions. Meanwhile, its violin plots of test period 2 reveal a more evident top-heavy distribution, with a wider upper and narrower lower tail, suggesting that a greater share of its evolved *alphas* achieve positive predictive power.

4 RELATED WORK

Due to the vast number of available operators and features, the search space for *alphas* is astronomically large. A traditional category is Genetic Programming (GP), which generate candidates through random mutation and crossover (Lin et al., 2019; Schmidt & Lipson, 2010; Zhaofan et al., 2022; Patil, 2023), or introduce hierarchical mechanisms to identify reusable components for efficient search (Zhang et al., 2020). While GP can recycle partial structures from existing *alphas*, they are largely heuristic, fail to exploit feedback from failed candidates, and offer limited interpretability. Some Reinforcement Learning (RL)-based attempts (Yu et al., 2023; Shi et al., 2025a; Zhu & Zhu, 2025) further guide *alpha mining* with reward signals, but they still operate at the operator level and rely heavily on trial-and-error. In non-stationary markets, such incremental search is easily misled by spurious correlations and struggles to discover robust *alphas*.

Large language models (LLMs) provide a promising alternative by leveraging semantic reasoning and domain knowledge to construct more interpretable *alphas* (Wang et al., 2023; Haluptzok et al., 2023; Weng, 2023; Sumers et al., 2024; Shi et al., 2025b). Several recent studies (Luo et al., 2025; Wang et al., 2024) integrate LLMs into *alpha mining*, such as FAMA for dynamic factor combination (Li et al., 2024) and AlphaAgent (Tang et al., 2025) for a multi-agent architecture for decay-resistant *alphas*. However, these approaches remain essentially prompt-driven, lacking mechanisms for long-horizon planning, systematic reflection, and self-evolution. In parallel, RL-based LLM post-training (Wang et al., 2025; Jin et al., 2025; Chen et al., 2025) has made progress in mathematics, games, and tool use. Self-evolving LLM systems (Gao et al., 2025; Agrawal et al., 2025; Romera-Paredes et al., 2024; Chen et al., 2023) demonstrate great potential in solving complex problems through progressive refinement, but their application to alpha mining remains problematic, due to their vulnerability to market regime shifts or the inherent inability of text-based experience to fully encode desired evolutionary patterns. In this work, we bridge this gap with a self-evolving agentic RL paradigm.

5 CONCLUSION

In this work, we introduced AlphaAgentEvo, a novel agentic reinforcement learning paradigm for *alpha mining*. By reformulating alpha mining from a brute-force searching problem into a multi-turn evolution-driven paradigm, our framework endows LLM-driven agents with stronger self-evolution capabilities, enabling them to dynamically extract structure from noisy and high-variance financial tool feedback through hierarchical reward signals. Extensive experiments on *AlphaEvo500* and *Alpha158* confirm that our method not only delivers consistently higher valid ratios and pass rates, but also generalizes effectively across market regimes and longer evolutionary trajectories, surpassing modern self-evolution approaches associated with state-of-the-art close-source LLMs with only 4B parameters. These results highlight *self-evolving ARL* as a principled and generalizable paradigm for next-generation quantitative investment.

486
487

REPRODUCIBILITY STATEMENT

488

In this submission, we have made extensive efforts to ensure the reproducibility of our work. Specifically, the dataset files used in this paper are enclosed in the supplementary materials, including training, validation, and test splits. The evaluation tool's parameters are described in Sec. 3.1 and the tool schema in Sec. F. Available data variables and representative functions are listed in Sec. G, Appendix. Training configurations are documented in Sec. D of the Appendix.

493

In addition, we provide the full source code (including training pipelines, evaluation scripts, and evaluation tools) as supplementary files. We will publicly release the source code associated with the used data once the paper is accepted to facilitate further research.

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540 REFERENCES
541

542 Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong,
543 Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik
544 Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. Gepa:
545 Reflective prompt evolution can outperform reinforcement learning, 2025. URL <https://arxiv.org/abs/2507.19457>.

546 Huan ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong
547 Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang,
548 Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Qihan Ren, Cheng Qian,
549 Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A
550 survey of self-evolving agents: On path to artificial super intelligence, 2025. URL <https://arxiv.org/abs/2507.21046>.

551 Angelica Chen, David Dohan, and David So. Evoprompting: Language models for code-level neural
552 architecture search. *Advances in neural information processing systems*, 36:7787–7817, 2023.

553 Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan,
554 Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. Research: Learning to
555 reason with search for llms via reinforcement learning, 2025. URL <https://arxiv.org/abs/2503.19470>.

556 DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, and Ruoyu Zhang et al.
557 Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL
558 <https://arxiv.org/abs/2501.12948>.

559 Jinyong Fan and Yanyan Shen. Stockmixer: A simple yet strong mlp-based architecture for stock
560 price forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(8):8389–
561 8397, Mar. 2024. doi: 10.1609/aaai.v38i8.28681. URL <https://ojs.aaai.org/index.php/AAAI/article/view/28681>.

562 D. Guo, D. Yang, H. Zhang, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement
563 learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL <https://doi.org/10.1038/s41586-025-09422-z>.

564 Patrick Haluptzok, Matthew Bowers, and Adam Tauman Kalai. Language models can teach them-
565 selves to program better. In *The Eleventh International Conference on Learning Representations*,
566 2023. URL <https://openreview.net/forum?id=SaRj2ka1XZ3>.

567 Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and
568 Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement
569 learning. *arXiv preprint arXiv:2503.09516*, 2025.

570 Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-
571 Yan Liu. Lightgbm: a highly efficient gradient boosting decision tree. *NIPS’17*, pp. 3149–3157,
572 Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

573 Yuante Li, Xu Yang, Xiao Yang, Minrui Xu, Xisen Wang, Weiqing Liu, and Jiang Bian. R&d-agent-
574 quant: A multi-agent framework for data-centric factors and model joint optimization, 2025.

575 Zhiwei Li, Ran Song, Caihong Sun, Wei Xu, Zhengtao Yu, and Ji-Rong Wen. Can large language
576 models mine interpretable financial factors more effectively? a neural-symbolic factor mining
577 agent model. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Asso-
578 ciation for Computational Linguistics: ACL 2024*, pp. 3891–3902, Bangkok, Thailand, August
579 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.233. URL
580 <https://aclanthology.org/2024.findings-acl.233>.

581 X. Lin, Y. Chen, Z. Li, and K. He. Revisiting stock alpha mining based on genetic algorithm.
582 Technical report, Huatai Securities Research Center, 2019. URL <https://crm.htsc.com>.

583 Haochen Luo, Yuan Zhang, and Chen Liu. Efs: Evolutionary factor searching for sparse portfolio
584 optimization using large language models, 2025. URL <https://arxiv.org/abs/2507.17211>.

594 OpenAI. Gpt-5 system card. Technical report, OpenAI, August 2025. URL <https://cdn.openai.com/gpt-5-system-card.pdf>. Retrieved from OpenAI.

595

596

597 RR Patil. Ai-infused algorithmic trading: genetic algorithms and machine learning in high-
598 frequency trading. *International Journal for Multidisciplinary Mining profitable alpha factors via con-*
599 *volution kernel learningiplinary Research*, 5(5), 2023.

600 Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan
601 Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*,
602 2025.

603 Weizhe Ren, Yichen Qin, and Yang Li. Alpha mining and enhancing via warm start genetic program-
604 ming for quantitative investment, 2024. URL <https://arxiv.org/abs/2412.00896>.

605

606 Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog,
607 M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang,
608 Omar Fawzi, et al. Mathematical discoveries from program search with large language models.
609 *Nature*, 625(7995):468–475, 2024.

610 Michael D. Schmidt and Hod Lipson. Age-fitness pareto optimization. In *Proceedings of the 12th*
611 *Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pp. 543–544, New
612 York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300728. doi: 10.
613 1145/1830483.1830584. URL <https://doi.org/10.1145/1830483.1830584>.

614

615 Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang,
616 Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathemati-
617 cal reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

618

619 Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng,
620 Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint*
621 *arXiv: 2409.19256*, 2024.

622

623 Hao Shi, Weili Song, Xinting Zhang, Jiahe Shi, Cuicui Luo, Xiang Ao, Hamid Arian, and Luis Angel
624 Seco. Alphaforge: A framework to mine and dynamically combine formulaic alpha factors. In
625 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 12524–12532,
626 2025a.

627

628 Yu Shi, Yitong Duan, and Jian Li. Navigating the alpha jungle: An llm-powered mcts framework
629 for formulaic factor mining. *arXiv preprint arXiv:2505.11122*, 2025b.

630

631 Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive archi-
632 tectures for language agents, 2024. URL <https://arxiv.org/abs/2309.02427>.

633

634 Ziyi Tang, Zechuan Chen, Jiarui Yang, Jiayao Mai, Yongsen Zheng, Keze Wang, Jinrui Chen, and
635 Liang Lin. Alphaagent: Llm-driven alpha mining with regularized exploration to counteract alpha
636 decay. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data*
637 *Mining V. 2*, pp. 2813–2822, 2025.

638

639 D. (waditu) Wang. Tushare. GitHub Repository, 2024. URL <https://github.com/waditu/tushare>.

640

641 Meiyun Wang, Kiyoshi Izumi, and Hiroki Sakaji. Llmfactor: Extracting profitable factors through
642 prompts for explainable stock movement prediction, 2024. URL <https://arxiv.org/abs/2406.10811>.

643

644 Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang,
645 Xin Jiang, and Qun Liu. Aligning large language models with human: A survey, 2023. URL
646 <https://arxiv.org/abs/2307.12966>.

647

648 Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin,
649 Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu,
650 Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in
651 llm agents via multi-turn reinforcement learning, 2025. URL <https://arxiv.org/abs/2504.20073>.

648 Lilian Weng. Llm-powered autonomous agents. *lilianweng.github.io*, Jun 2023. URL <https://lilianweng.github.io/posts/2023-06-23-agent/>.

649

650

651 An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang

652 Gao, Chengan Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu,

653 Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin

654 Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang,

655 Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui

656 Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang

657 Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger

658 Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan

659 Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

660

661 Xiao Yang, Weiqing Liu, Dong Zhou, Jiang Bian, and Tie-Yan Liu. Qlib: An ai-oriented quantitative

662 investment platform. *arXiv preprint arXiv:2009.11189*, 2020.

663

664 Shuo Yu, Hongyan Xue, Xiang Ao, Feiyang Pan, Jia He, Dandan Tu, and Qing He. Generating

665 synergistic formulaic alpha collections via reinforcement learning. In *Proceedings of the 29th*

666 *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 5476–5486,

667 New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701030. doi:

668 10.1145/3580305.3599831. URL <https://doi.org/10.1145/3580305.3599831>.

669

670 Tianping Zhang, Yuanqi Li, Yifei Jin, and Jian Li. Autoalpha: an efficient hierarchical evolutionary

671 algorithm for mining alpha factors in quantitative investment, 2020. URL <https://arxiv.org/abs/2002.08245>.

672

673 Su Zhaofan, Lin Jianwu, and Zhang Chengshan. Genetic algorithm based quantitative factors con-

674 struction. In *2022 IEEE 20th International Conference on Industrial Informatics (INDIN)*, pp.

675 650–655. IEEE, 2022.

676

677 Zhoufan Zhu and Ke Zhu. AlphaQCM: Alpha discovery in finance with distributional reinforcement

678 learning. In *Forty-second International Conference on Machine Learning*, 2025. URL <https://openreview.net/forum?id=3sXMH1hBSs>.

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702 A DISCLOSURE OF LARGE LANGUAGE MODEL USAGE

703
 704 In this paper, Large Language Models (LLMs) are only used for polishing paragraph content, check-
 705 ing and revising grammar, and writing visualization code for some experimental results during the
 706 paper writing process. All other parts of this paper were completed by human authors, in particular,
 707 the conception of research ideas, the creation of figures in the paper, the conduct of experiments,
 708 and the recording of experimental data.

710 B MULTI-FACTOR PERFORMANCE COMPARISON.

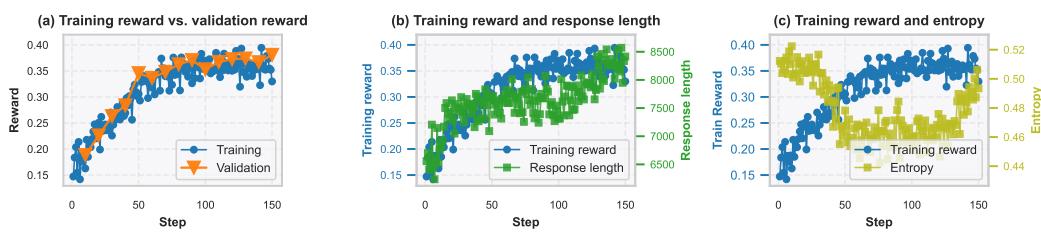
711 While some baseline approaches cannot be compared directly, to evaluate AlphaAgentEvo’s per-
 712 formance with them, we evaluate AlphaAgentEvo based on the multi-factor strategy where a group
 713 of *alphas* are weighted to generate an meta *alpha*. We report the performance comparison results
 714 against three categories of baselines, including time series models (TS model), a non-LLM RL
 715 framework, and LLM-agent-based frameworks. For fairness, the top-10 mined alphas from each
 716 LLM-agent-based framework are evenly combined for backtesting.

717 Table 3: Multi-factor portfolio performance comparison from 2024-01 to 2025-11.

718 Method	Category	719 Trainable	AER	IR	MDD
720 LightGBM Ke et al. (2017)	TS model	✓	-0.009	1.192	-0.195
721 Stock-Mixer Fan & Shen (2024)	TS model	✓	0.013	1.977	-0.182
722 AlphaQCM Zhu & Zhu (2025)	RL Framework	✓	0.027	1.815	-0.192
723 AlphaAgent Tang et al. (2025)	Multi-LLM-agent	✗	0.064	2.046	-0.196
724 GPT-5-mini OpenAI (2025)	Single-LLM-agent	✗	-0.158	0.587	-0.213
725 ToolRL-4B Qian et al. (2025)	Single-LLM-agent	✓	-0.027	1.532	-0.215
726 AlphaAgentEvo-4B (<i>ours</i>)	Single-LLM-agent	✓	0.129	2.442	-0.176

727 C TRAINING ANALYSIS

728 To analyze the policy LLM’s changes during the reinforcement learning process, we present the
 729 rewards in training and validation sets, average response length, and the output entropy during Al-
 730 phaAgentEvo’s training.



731 Figure 7: Training statistics of AlphaAgentEvo.

732 **Training reward vs. validation reward.** Fig. 7(a), both training and validation reward curves
 733 rise steadily from ~ 0.16 to ~ 0.38 – 0.39 over 150 steps, with the validation reward closely tracking
 734 the training reward and maintaining a small generalization gap (visually < 0.02 throughout). The
 735 improvement is fastest in the first 50 steps and then saturates gradually, with minor jitter but no sign
 736 of overfitting: validation continues to trend upward in tandem with training.

737 **Response length.** Fig. 7(b) shows a clear stepwise increase in average response length. Early on
 738 (0–50 steps), the model rapidly adapts to task requirements and learns to produce stable tool calls.
 739 Between 50–100 steps, growth slows as it improves tool-call quality and forms its own evolution
 740 strategy. After 100 steps, response length stabilizes at a relatively high level, reflecting the transition
 741 from basic adaptation to more complex reasoning for *alpha* mining.

756 **Output entropy.** As shown in Fig. 7(c), output entropy first decreases as the agent masters
 757 consistent reasoning, then remains stable, and finally rises again after ~ 125 steps. This rebound indicates
 758 renewed exploration, where the model diversifies its reasoning to generate richer *alphas*.
 759

760 Taken together, these results verify that our training can converge stably with good generalization,
 761 and our method is capable of effective complex reasoning to mine richer and more effective *alphas*.
 762

763 D TRAINING CONFIGURATIONS

764 We train Qwen3 (Yang et al., 2025) (1.7B and 4B) on $10 \times$ RTX4090 GPUs for 150 steps using the
 765 Verl framework (Sheng et al., 2024). The 4B model provides sufficient capability while maintaining
 766 a favorable performance-efficiency trade-off for large-scale *alpha mining*, with the 1.7B variant
 767 serving as a lighter comparison. Each batch samples 20 seed *alphas*, with 3 rollouts per seed, up
 768 to 3 turns per trajectory, and up to 4 tool calls per turn. The coefficient β for KL loss is set to
 769 0.001. The 80th-step checkpoint is used for testing. Reward caps are set as $C_{\text{tool}} = 1$, $C_{\text{cons}} = 0.2$,
 770 $C_{\text{expl}} = 0.3$, $C_{\text{perf}} = 0.5$, and $C_{\text{streak}} = 0.6$. The weighting coefficients are $\alpha_{\text{succ}} = 0.1$, $\alpha_{\text{fail}} = 0.2$,
 771 $\alpha_{\text{cons}} = 0.02$, $\alpha_{\text{exp}} = 0.02$, $\alpha_{\text{perf}} = 0.1$, and $\alpha_{\text{streak}} = 0.15$. We set the learning rate to 1×10^{-6} with
 772 a warmup ratio of 0.1, and use the Adam optimizer for training. The policy LLM is updated with a
 773 mini-batch size 20.
 774

775 E CASE ANALYSIS

776 We take a sample from *Alpha158* as a case study (only first two turns) to showcase why our model
 777 outperforms other baselines in *alpha evolution*, as shown in Fig. 8.
 778

779 In the first turn, two models demonstrate fundamentally different approaches to evaluating the seed
 780 *alpha*. The baseline model immediately suggests horizontal adjustments such as extending the look-
 781 back window or applying Z-score normalization, e.g., “*maybe using a longer window... or ZS-*
 782 *CORE...*”. This reflects a focus on technical re-scaling while leaving the underlying semantics of
 783 the *alpha* unchanged. By contrast, AlphaAgentEvo-4B begins with a critical examination of the
 784 seed *alpha*: “*I should check if the current alpha has any issues ... perhaps we can look at the*
 785 *cumulative effect of down days...*”. This reasoning explicitly questions the behavioral implication
 786 of the *alpha*—whether a higher value actually aligns with profitable trading logic. By identifying
 787 the misalignment between signal direction and trading outcome, AlphaAgentEvo-4B opens the path
 788 toward semantic reconstruction. This critical stance directly motivates subsequent refinements such
 789 as combining with RSI, normalizing by volatility, and embedding structural signals (e.g., ZigZag),
 790 which collectively transform a weak predictor into an *alpha* with strong positive alpha.
 791

792 In the second turn, this divergence becomes even clearer. The baseline model reflects only on param-
 793 eter choices from its previous trial: “*Looking at the previous attempts: the first new alpha SIGNIFI-*
 794 *CANT_DOWN_DAYS_90 ... But its IR is worse (-0.3529), so maybe the threshold is too strict or the*
 795 *window is too long.*” The analysis remains confined to surface-level tuning, without questioning the
 796 semantic suitability of the *alpha* design. In contrast, AlphaAgentEvo-4B demonstrates a stronger
 797 capacity for reflective reasoning: “*RSI_DOWN_COUNT ... giving a high IR of 0.9417 ... the key*
 798 *here is that the previous factors improved by incorporating other indicators (RSI, volatility, swing*
 799 *points) with the down count...*” Rather than merely fine-tuning thresholds, AlphaAgentEvo-4B dis-
 800 tills general patterns from past attempts and reasons about why certain designs succeed. Building
 801 on this reflection, it further proposes new factor constructions, such as adding industry neutrality
 802 or volatility adjustment, showing an ability to transform statistical feedback into structured insights
 803 that guide subsequent evolution.
 804

805 Overall, the baseline model remains locked in mechanical parameter tweaks, while AlphaAgentEvo-
 806 4B demonstrates a dialectical process: integrating empirical outcomes with theoretical reasoning to
 807 refine the *alpha* semantics. This difference explains why our model converges to successful, high-
 808 performance *alphas* while the baseline model fails.
 809

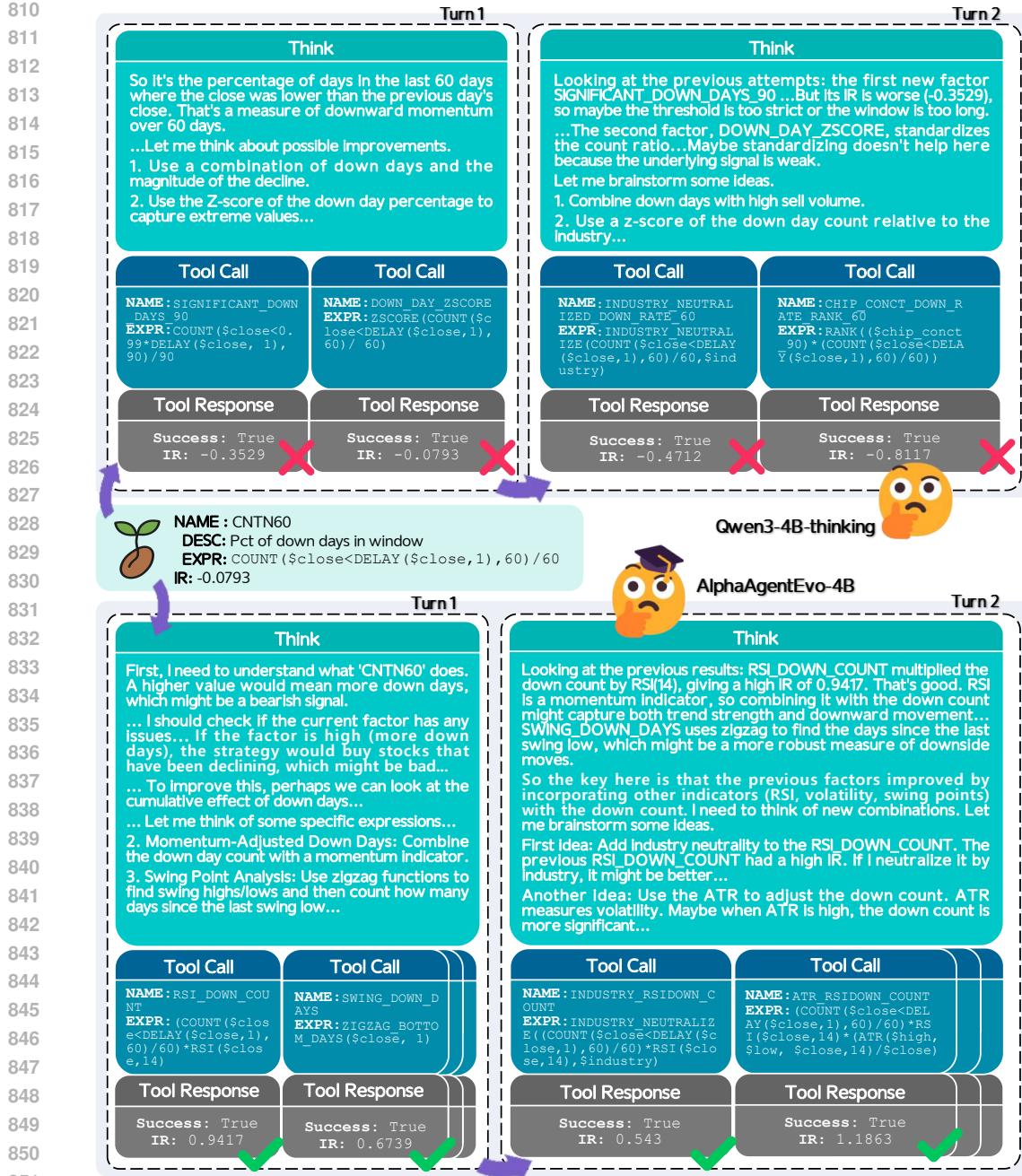


Figure 8: Case study: AlphaAgentEvo-4B vs. our base model Qwen3-4B-thinking.

F EVALUATION TOOL

As part of AlphaAgentEvo, we design a dedicated tool interface, `evaluate_factor`, to support the evaluation of *alphas* with backtesting. The simplified tool schema is shown in Listing 1, which illustrates the core arguments (`factor_name`, `factor_expr`, and `metric`). For clarity, some auxiliary parameters such as `time_range` and `market` are omitted here.

864

Listing 1: Schema of the `evaluate_factor` tool used in this paper.

```

865
866 1 {
867 2   "type": "function",
868 3   "function": {
869 4     "name": "evaluate_factor",
870 5     "description": "A tool for evaluating factors with backtesting",
871 6     "parameters": {
872 7       "type": "object",
873 8       "properties": {
874 9         "factor_name": {
875 10          "type": "string",
876 11          "description": "The name of the factor"
877 12        },
878 13        "factor_expr": {
879 14          "type": "string",
880 15          "description": "The expression of the factor"
881 16        },
882 17        "metric": {
883 18          "type": "string",
884 19          "description": "The metric to evaluate (default: 'Information_Ratio_with_cost')",
885 20          "default": "Information_Ratio_with_cost"
886 21        }
887 22      },
888 23      "required": ["factor_name", "factor_expr"]
889 24    }
890 25  }
891 26}

```

892

G AVAILABLE DATA AND FUNCTIONS

893

Data variables used to construct *alphas* are shown in Table 5, all sourced from Tushare (Wang, 2024). A representative set of functions to operate these data variables are displayed in Table 4.

894

895

Table 4: Representative functions used in our *alpha* expressions.

Function Name	Description
A. Cross-Sectional Operations	
RANK(var)	Cross-sectional percentile rank of a variable.
ZSCORE(var)	Standardizes a variable (z-score) cross-sectionally.
INDUSTRY_NEUTRALIZE(var, \$industry)	Neutralizes the variable's exposure within industries.
B. Time-Series / Rolling Window	
TS_MEAN(var, p)	Rolling mean over the past 'p' periods.
TS_MAX(var, p), TS_MIN(var, p)	Rolling maximum and minimum.
TS_RANK(var, p)	Time-series percentile rank over a window.
TS_PCTCHANGE(var, p)	Percentage change over 'p' periods.
DELTA(var, p)	Difference from 'p' periods ago ($x_t - x_{t-p}$).
EMA(var, p), SMA(var, p)	Exponential and Simple Moving Average.
C. Mathematical & Logical	
LOG(var), POW(var, exp)	Natural logarithm and power.
DELAY(var, p)	Value of the variable 'p' periods ago (lag).
COUNT(cond, p)	Count of times a condition is true over 'p' periods.
A ? B : C	Ternary operator (if condition A then B, else C).
D. Advanced & Technical	
TS_CORR(var1, var2, p)	Rolling correlation between two variables.
REGBETA(var1, var2, p)	Rolling beta from regressing var1 on var2.
RSI(var, p)	Relative Strength Index.
MACD(var, p_short, p_long)	Moving Average Convergence Divergence.

913

914

915

916

917

918
919
920
921
922
923
924
925

926
927

Table 5: Available data variables.

Variable Name	Description
Price & Market Data	
\$open	Opening price
\$high	Highest price of the day
\$low	Lowest price of the day
\$close	Closing price
\$volume	Trading volume (shares)
\$amount	Trading amount (CNY)
\$change	Price change vs. previous close
\$return	Last day's return
Chip-Distribution Data	
\$his_low	Historical low price since listing
\$his_high	Historical high price since listing
\$cost_5pct	Cost where 5% of chips lie below
\$cost_15pct	Cost where 15% of chips lie below
\$cost_50pct	Median cost of chips
\$cost_85pct	Cost where 85% of chips lie below
\$cost_95pct	Cost where 95% of chips lie below
\$weight_avg	Average cost across all chips
\$winner_rate	The chip win rate
\$chip_conct_90	Chip concentration within the densest 90%
\$chip_conct_70	Chip concentration within the densest 70%
Order-Flow / Money-Flow Data	
\$buy_sm.vol, \$sell_sm.vol	Small-lot buy/sell volume
\$buy_sm.amount, \$sell_sm.amount	Small-lot buy/sell turnover
\$buy_md.vol, \$sell_md.vol	Medium-lot buy/sell volume
\$buy_md.amount, \$sell_md.amount	Medium-lot buy/sell turnover
\$buy_lg.vol, \$sell_lg.vol	Large-lot buy/sell volume
\$buy_lg.amount, \$sell_lg.amount	Large-lot buy/sell turnover
\$buy_elg.vol, \$sell_elg.vol	Extra-large-lot buy/sell volume
\$buy_elg.amount, \$sell_elg.amount	Extra-large-lot buy/sell turnover
\$net_mf.vol	Net inflow volume (buy - sell)
\$net_mf.amount	Net inflow amount
Benchmark & Industry	
\$bench.open, \$bench_high, \$bench_low, \$bench_close	Benchmark index OHLC prices
\$bench.preclose	Benchmark previous close
\$bench.volume	Benchmark trading volume
\$bench.amount	Benchmark trading amount
\$bench.turn	Benchmark turnover ratio
\$bench.return	Benchmark last day's return
\$industry	Categorical industry label

958
959
960
961
962
963
964
965
966
967
968
969
970
971