ALPHAPROBE: Alpha Mining via Principled Retrieval and On-graph Biased Evolution

Taian Guo[1,3] **Haiyang Shen**[2] **Junyu Luo**[1] **Binqi Chen**[1,3] **Hongjun Ding**[4]
Jinsheng Huang[1,3] **Luchen Liu**[3] **Yun Ma**[2,✉] **Ming Zhang**[1,✉]

[1]State Key Laboratory of Multimedia Information Processing, School of Computer Science, PKU-Anker LLM Lab, Beijing Key Laboratory of Software and Hardware Cooperative Artificial Intelligence Systems, Peking University
[2]Institute for Artificial Intelligence, Peking University [3]Zhengren Quant [4]Baruch College

✉**Corresponding Authors.**

Abstract

Extracting signals through alpha factor mining is a fundamental challenge in quantitative finance. Existing automated methods primarily follow two paradigms: Decoupled Factor Generation, which treats factor discovery as isolated events, and Iterative Factor Evolution, which focuses on local parent-child refinements. However, both paradigms lack a global structural view, often treating factor pools as unstructured collections or isolated single-lineage paths, which leads to redundant search and inefficient allocation of optimization effort. To address these limitations, we introduce ALPHAPROBE (**Alpha** Mining via **P**rincipled **R**etrieval and **O**n-graph **B**iased **E**volution), a framework that reframes alpha mining as the strategic navigation of a Directed Acyclic Graph (DAG). By modeling factors as nodes and evolutionary links as edges, ALPHAPROBE treats the factor pool as a dynamic, interconnected ecosystem. The framework consists of two core components: a Bayesian Factor Retriever that identifies high-potential seeds by balancing exploitation and exploration through a posterior probability model, and a DAG-aware Factor Generator that leverages the full ancestral trace of factors to produce context-aware, non-redundant optimizations. Extensive experiments on major Chinese and US stock market datasets against 8 competitive baselines demonstrate that ALPHAPROBE significantly gains enhanced performance in predictive accuracy, return stability and training efficiency. Our results confirm that leveraging global evolutionary topology is essential for efficient and robust automated alpha discovery. We have open-sourced our implementation at `https://github.com/gta0804/AlphaPROBE`.

1 Introduction

Extracting predictive signals from noisy and high-dimensional market data is a fundamental challenge in quantitative finance [5, 12, 28, 18, 23]. The primary approach to this problem is *alpha factor mining*. This process involves discovering mathematical expressions, known as alpha factors, that transform raw market data into predictors of future asset returns. Rather than searching for a single perfect predictor, the objective is to build a diverse portfolio of factors that provide collective and robust predictive power. To achieve this, several automated discovery methods have recently emerged.

Automated alpha mining generally follow two paradigms. The first is *Decoupled Factor Generation (DFG)*,which generates factors independently based on a shared training distribution [30, 33, 35, 3, 21, 34]. In this paradigm, the relationship between factors remains implicit and weak because the model treats each generation attempt as an isolated event. The second is *Iterative Factor Evolution (IFE)*, which focuses on refining existing

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

factors into improved descendants [26, 15, 17, 25, 22, 14, 2, 9] and optimizing along isolated single-lineage paths. Moreover, IFE methods treat all parent factors as equally worth refining, overlooking that factors differ significantly in their remaining optimization potential.

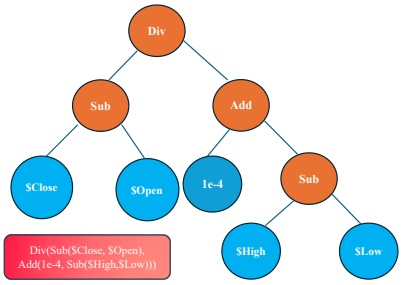


Figure 1: An example of alpha factor on the AST and expression view. All features and operators are in Appendix 8.3.

The lack of a global structural view offers a critical limitation shared by both paradigms. DFG treats the growing factor library as an unstructured collection and ignore the connections between similar expressions. IFE focuses on the factor lineage within individual branches instead of the broader evolutionary web. Consequently, existing methods prioritize the quality of individual factors and lineages but overlook the strategic information embedded in their topological relationships. Without this global view, both paradigms have limited ability to assess which factors are more promising candidates for further optimization. Factors that have been repeatedly refined may yield diminishing returns, while less-explored regions of the search space may offer greater potential. Hence, the mining process becomes redundant and inefficient. We argue that an effective discovery process requires not only organizing factors into a global structure but also strategically navigating it. The primary challenge is how to evaluate each factor's optimization potential within this topology.

Factor evolution is inherently a directed, acyclic process: each new factor is derived from one or more parents, and this derivation is irreversible. A DAG is therefore the natural representation of this process, where nodes are factors and edges encode their lineage. This structure makes two types of information accessible that flat pools and single-lineage search methods discard: the depth and branching history of each factor (indicating over-optimization risk) and the global topological relationship among all factors (revealing under-explored regions). Building on this observation, we introduce ALPHAPROBE, a closed-loop framework that reframes alpha mining as the strategic navigation of this DAG through two tightly coupled components: a *Bayesian Factor Retriever* and a *DAG-aware Factor Generator*.

The Bayesian Factor Retriever addresses the question of *where to search next*. It scores each factor in the DAG with the prior and likelihood term by balancing exploitation of high-quality, well-positioned factors against exploration of novel, under-explored regions. The DAG-aware Factor Generator then addresses *how to search effectively*. Given a selected parent, it conditions generation on the full ancestral path encoded in the DAG, enabling the LLM to avoid mutations that have already been tried and to produce more diverse offspring. New factors are validated and fed back into the DAG, thus completing the evolutionary cycle.

We conduct extensive experiments to validate ALPHAPROBE on four major Chinese and US stock market datasets including CSI 300, CSI 500, CSI 1000 and S&P 500. ALPHAPROBE is compared against 8 competitive baselines from three distinct categories. The results demonstrate that ALPHAPROBE outperforms existing methods in predictive accuracy and return stability. Furthermore, we provide detailed ablation studies to verify our design. We also conduct robustness analysis on parameters, backbone LLMs and prompts and a detailed case study to inspect the evolutionary path of generated factors. Our main contributions are summarized as follows:

- We introduce ALPHAPROBE, which models alpha factor evolution as a retrieval process on DAG. Our goal is to move beyond unstructured factor collections by capturing global relationships. This topological view enables more strategic and efficient navigation of the factor search space.

- We develop a closed-loop discovery system using a Bayesian Factor Retriever and a DAG-aware Factor Generator. The retriever identifies high-potential seeds by balancing quality and diversity through a posterior model. To avoid redundant mutations, the generator utilizes the full ancestral lineage to produce context-aware and novel factor optimizations.

- We validate ALPHAPROBE on major Chinese and US stock datasets against 8 SOTA baselines. The experiments demonstrate that ALPHAPROBE significantly improves predictive performance, return stability and training efficiency. Our analysis also confirms that this approach creates a more robust and hierarchically organized factor library.

2 Background

In trading, a factor is a predictive signal derived from various data sources. The input data for an asset at time t typically includes price-volume features like open (O_t), high (H_t), low (L_t), close (C_t), and volume (V_t). Beyond these, modern systems also incorporate fundamental data, such as earnings and cash flow from financial reports, as well as alternative data like news sentiment. An alpha factor is a mathematical expression f that transforms these data streams into a single score. As shown in Figure 1, a factor is usually represented as an expression tree. In this tree, leaf nodes represent input features, while internal nodes represent mathematical operators like `Add`, `Sub`, or `TsMean`.

The core objective of alpha mining is to discover an optimal set of factors $\mathcal{A} = \{f_1, f_2, \ldots, f_N\}$ from a massive search space $\mathcal{F}$ of all possible valid mathematical combinations, by maximizing a portfolio-level utility function: $\max_{\mathcal{A} \subset \mathcal{F}} \mathcal{J}(\mathcal{A})$. The function $\mathcal{J}$ measures the overall quality of the factor pool. In practice, $\mathcal{J}$ represents specific financial goals. For example, it can be the Sharpe Ratio of a portfolio built from these factors, which measures the return per unit of risk. Alternatively, it can be a combination of the Information Coefficient (IC) for predictive accuracy and a penalty term for inter-factor correlation to ensure diversity. By maximizing $\mathcal{J}$, we ensure that the discovered factors are powerful and complementary. This formalized goal has led to the development of various automated discovery methods, which we categorize based on their generation logic.

3 Related Work

3.1 Decoupled Factor Generation (DFG)

The first paradigm, Decoupled Factor Generation (DFG), focuses on creating factors as independent individual units. These methods typically use Reinforcement Learning [19, 30, 35, 10], Graph Neural Networks [13], or Generative Models [21, 8] to sample factors from a vast search space. Recent works have also explored enhancing sample diversity via Generative Flow Networks [3] or multi-distribution learning [33, 34]. In this paradigm, models are trained on shared data, allowing for implicit knowledge sharing. However, the generation process treats each attempt as an isolated event. Because the model ignores the specific state of the current library, the relationships between factors remain implicit and weak.

While DFG methods excel at broad exploration, their unstructured approach limits the ability to build upon previous discoveries systematically. By treating the factor pool as a simple collection rather than a connected system, these methods ignore the evolutionary logic that leads to success. This gap suggests a need for methods that can refine existing knowledge through iterative improvements. This realization leads us to the second major paradigm in the field.

3.2 Iterative Factor Evolution (IFE)

The second paradigm, Iterative Factor Evolution (IFE), treats factor discovery as a process of continuous refinement. These methods start with a "parent" factor and apply modifications to create improved "children". Early methods in this category used Genetic Programming to evolve expression trees through crossover and mutation [31, 4]. Recently, LLMs have been used to provide intelligent feedback for these refinements [26, 15, 17, 9]. Modern agents now use backtesting results to prompt LLMs for targeted formula changes [25, 2, 14]. Some frameworks even use multi-agent systems [11, 16, 27], Monte Carlo Tree Search [1, 22] and self-evolution [24] to guide local search.

The primary limitation of current IFE methods is their lack of a global view. By focusing only on single-parent-child pairs or isolated single-lineage paths, these methods overlook the broader evolutionary web of the entire factor pool. They lack a principled mechanism to assess the optimization potential of each factor and choose the most promising seed from a large library of candidates. Consequently, the discovery process often becomes repetitive or trapped in local optima. To solve these problems, ALPHAPROBE organizes the factor pool into a DAG and strategically navigates it to allocate optimization effort where it matters most.

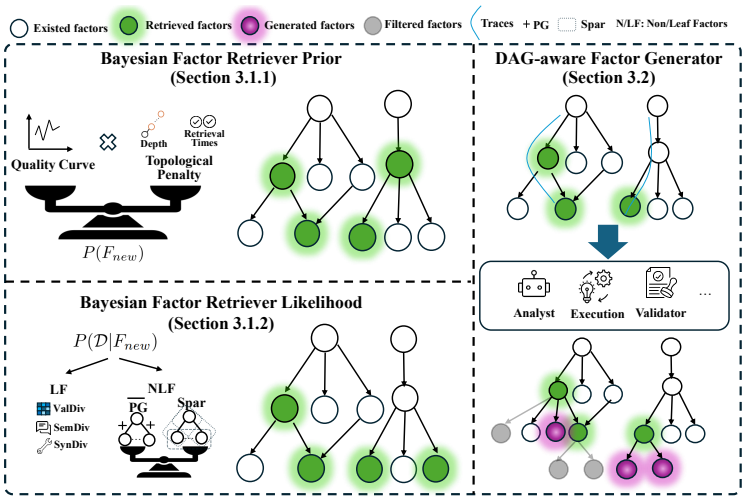


Figure 2: The overall illustration of ALPHAPROBE. ALPHAPROBE is a closed-loop framework consisting of a Bayesian Factor Retriever and a DAG-aware Factor Generator. The Bayesian Factor Retriever selects factors with higher potential by a trade-off among factors' intrinsic quality & diversity, topological structure, and lineage. The DAG-aware Factor Generator utilizes a multi-agent structure and factors' topology to generate new factors.

4 Method

We reframe the alpha factor mining problem as a strategic navigation and creation process on a DAG $\mathcal{G} = (\mathcal{F}, \mathcal{E})$, where $\mathcal{F}$ is the set of all discovered factors (nodes) and $\mathcal{E}$ is the set of directed edges representing their lineage. An edge $(F_p, F_c) \in \mathcal{E}$ indicates that factor F_c was derived from parent F_p. Within this framework, the overarching optimization problem bifurcates into two distinct, interconnected challenges:

- **Strategic Retrieval**: The first challenge is to identify which existing factor in the graph is the most promising candidate for evolution. This is a search problem on the DAG to find an optimal parent factor F_p^* that maximizes the expected quality of its yet-to-be-created descendant, F_{new}:

$$F_p^* = \arg\max_{F_p \in \mathcal{F}} \mathbb{E}[\text{Qual}(F_{new}) | \text{parent}(F_{new}) = F_p, \mathcal{G}] \tag{1}$$

  where $\text{Qual}(\cdot)$ is a measure of factor quality.

- **Targeted Generation**: The second challenge is to generate novel, high-quality offspring from the selected parent. This is a generation problem that leverages the parent's contextual information within the graph—specifically, its ancestral path—to guide the creation of new $\{F_c\}$:

$$\{F_{c,1}, \ldots, F_{c,k}\} = G(F_p^*, T(F_p^*)) \tag{2}$$

  where $\{F_{c,1}, \ldots, F_{c,k}\}$ are newly generated child factors, $T(F_p^*)$ are the generation traces (the paths from a root to F_p^* in $\mathcal{G}$), and G is the generation function.

ALPHAPROBE is a closed-loop system designed to address this dual problem. It comprises a Bayesian Factor Retriever to solve the strategic retrieval task and a DAG-aware Factor Generator to perform targeted generation. The overall illustration of ALPHAPROBE is in Figure 2.

4.1 Bayesian Factor Retriever

To effectively select parent factors, we require a framework that can reason about the multi-level structural information inherent in the factor evolution process. Inspired by Bayesian modeling, we integrate evidence from various sources—the factor's individual performance, its lineage and the topological structure it occupies, and its relationship to the broader pool.

The Retriever's task is to identify the parent factors with the greatest optimization potential. This selection problem is framed within a Bayesian framework to balance the trade-off between a factor's individual merit and its potential contribution to the collective factor pool with a topological structure

$\mathcal{G}$. We seek to rank each factor $F \in \mathcal{F}$ by its probability of producing a high-value descendant F_{new}, given our current factor pool $\mathcal{D}$:

$$\arg\max_{F \in \mathcal{F}} \mathbb{E}[\text{Qual}(F_{new}) | \text{parent}(F_{new}) = F, \mathcal{D}] \propto \frac{P(F_{new})P(\mathcal{D}|F_{new})}{P(\mathcal{D})} \propto P(F_{new})P(\mathcal{D}|F_{new}) \tag{3}$$

The prior term $P(F_{new})$ captures the intrinsic potential of a factor. The likelihood term $P(\mathcal{D}|F_{new})$ assesses the value a new descendant would add to the pool's ecosystem. The evidence term $P(\mathcal{D})$ is a constant here.

4.1.1 Prior: Trading Off Quality Against Over-Optimization Risk

The prior probability $P(F_{new})$ is approximated by the quality of its parent F. However,F with high performance alone can be misleading. A factor that is the result of a long optimization chain or has been frequently used for generation may yield diminishing returns. Our prior is therefore designed to explicitly trade off a factor's demonstrated quality against these structural risks, considering the topological structure of F in $\mathcal{G}$.

We define the prior score for a candidate parent F as:

$$P(F) = \underbrace{\sigma\left(\frac{\text{Qual}(F) - \mu_{Qual(\mathcal{F})}}{\varsigma_{Qual(\mathcal{F})}}\right)}_{\text{Normalized Quality}} \cdot \underbrace{(1-\gamma)^{\text{depth}(F)}}_{\text{Depth Penalty}} \cdot \underbrace{(1-\omega)^{k(F)}}_{\text{Retrieval Penalty}} \tag{4}$$

Here, $\text{Qual}(F)$ denotes the factor's risk-adjusted performance, normalized across the pool $\mathcal{F}$; σ is a sigmoid mapping function; $\mu_{\text{Qual}(\mathcal{F})}$ and $\varsigma_{\text{Qual}(\mathcal{F})}$ are the mean and standard deviation of $\text{Qual}(\cdot)$ over $\mathcal{F}$, respectively; $\text{depth}(F)$ is the depth of factor F in the graph $\mathcal{G}$; and $k(F)$ denotes the retrieval times of F during the retrieval phase. This quality score is tempered by two topological penalties that represent the risk side of the trade-off:

- **Depth Penalty:** Addresses the trade-off between trusting a deeply optimized factor and the risk of it being overfitted, balanced by hyperparameter γ.
- **Retrieval Penalty:** Navigates the trade-off between exploiting a known successful parent and the need to explore other, less-mined areas of the search space, balanced by hyperparameter ω.

4.1.2 Likelihood: Assessing Contribution to the Collective

The likelihood term $P(\mathcal{D}|F_{new})$ estimates the marginal utility of a new factor. It essentially assesses how much this new factor improves the collective pool's quality rather than just its own individual merit. Our estimation strategy adapts based on whether its parent factor F has a proven history of generating successful descendants. Hence we distinguish between two types of factors in the DAG:

- **Leaf Factors:** Factors that have not yet produced any successful children. They represent unexplored optimization paths.
- **Non-Leaf Factors:** Factors that have successfully parented at least one child. They have a demonstrated history of generative potential.

For Leaf Factors For a leaf factor, we have no past generation offspring. The core idea is to approximate its potential by holistically measuring its own novelty along with the whole factor pool $\mathcal{F}$, trading off among three distinct dimensions of diversity: value, semantics, and syntax. A factor that is novel across all these dimensions is more likely to spawn a truly unique descendant. Then $P(\mathcal{D}|F_{new})$ can be estimated by three components:

$$\text{ValDiv}(F, \mathcal{F}) \cdot \text{SemDiv}(F, \mathcal{F}) \cdot \text{SynDiv}(F, \mathcal{F}) \tag{5}$$

ValDiv (Value Diversity): Measure the novelty of the factor's numerical output, implemented via its average Pearson correlation with other factors in $\mathcal{F}$:

$$1 - \|\frac{1}{|\mathcal{F}|}\sum_{f \in \mathcal{F}} \|Corr(F, f)\|\| \tag{6}$$

SemDiv (Semantic Diversity): Measure the novelty of the factor's underlying financial logic, implemented via the cosine similarity (CosSim) of the embedding of its LLM-generated explanation:

$$\sigma(1 - \text{CosSim}(F, \mathcal{F})) \tag{7}$$

SynDiv (Syntactic Diversity): Measure the novelty of the factor's mathematical structure, implemented via its normalized edit distance (ED) to other factors:

$$\frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} \frac{ED(F, f)}{len(F) + len(f)} \tag{8}$$

For Non-Leaf Factors For a non-leaf factor, we can make a more informed projection by extrapolating from its past success. The trade-off here is between parents that produce marginally better but similar children, versus those that spawn a wide array of diverse descendants.

$$P(\mathcal{D}|F_{new}) \approx \overline{\text{PG}}(\mathcal{C}(F)) \cdot \text{Spar}(\mathcal{C}(F)) \tag{9}$$

Here, $\overline{\text{PG}}$ measures the average quality gain in percentage from the parent to its existing children. $\mathcal{C}(\cdot)$ denotes the children of the current factor. The Spar term embodies a crucial trade-off on children's sparsity. We reward parents whose children are not only distinct from the parent (vertical diversity) but also distinct from each other (horizontal diversity). This encourages the discovery of parents that unlock multiple, independent optimization avenues. We formalize this as the product of two distinct sparsity measures:

$$\text{Spar}(\mathcal{C}(F)) = \text{Spar}_{\text{p-c}}(F) \cdot \text{Spar}_{\text{c-c}}(F) \tag{10}$$

$\text{Spar}_{\text{p-c}}$ is the parent-child sparsity, measuring how much the children diverge from their parent:

$$1 - \frac{1}{|\mathcal{C}(F)|} \sum_{f \in \mathcal{C}(F)} \| \text{Corr}(F, f)\| \tag{11}$$

$\text{Spar}_{\text{c-c}}$ is the inter-child sparsity, measuring how much the children diverge from each other. For factors with a single child, we set $\text{Spar}_{\text{c-c}}(F) = 1$. Otherwise, it is defined as:

$$1 - \frac{1}{\binom{|\mathcal{C}(F)|}{2}} \sum_{f_i, f_j \in \mathcal{C}(F), i<j} \|\text{Corr}(f_i, f_j)\| \tag{12}$$

Finally, We rank leaf and non-leaf nodes based on their respective scores and select the global top-k candidates to pass to the Factor Generator, thus completing the loop.

4.2 DAG-aware Factor Generator

Once the Retriever selects a parent factor F_p, the Generator's task is to produce novel and valid offspring. Rather than naively prompting an LLM for improvements—a process prone to syntactic errors and trivial variations—we structure generation as a three-stage, DAG-aware workflow that emulates a sequence of specialized analytical roles.

First, an Analyst agent leverages the parent's entire evolutionary path, its trace $T(F_p)$, to formulate a set of diverse and context-aware modification strategies, $\{\mathcal{S}\}$. This initial step transforms the historical optimization trajectory encoded in the DAG into actionable plans for future mutations, directly addressing the core of ALPHAPROBE. This process is formalized as:

$$\{\mathcal{S}_1, \ldots, \mathcal{S}_m\} = G_{\text{strategy}}(F_p, T(F_p)) \tag{13}$$

Next, an Execution agent translates each abstract strategy $\mathcal{S}_i$ into a concrete candidate factor expression, $F'_{c,i}$. This stage separates the high-level financial and structural reasoning from the precise task of expression synthesis:

$$F'_{c,i} = G_{\text{synth}}(F_p, \mathcal{S}_i) \quad \forall i \in \{1, \ldots, m\} \tag{14}$$

Finally, a Validator agent subjects each candidate $F'_{c,i}$ to a rigorous check for syntactic correctness and adherence to predefined constraints, filtering out invalid expressions. This entire workflow constitutes the generation function G. The set of newly validated factors $\{F_c\} \subseteq \{F'_c\}$ is then added to the DAG with new edges from F_p, completing the evolutionary loop and expanding the graph for the next retrieval cycle. The detailed design of Factor Generator is provided in the Appendix 8.2.1.

Table 1: **Performance Comparison** on the CSI 300, CSI 500 and CSI 1000. *Bold* and underlined numbers denote the best and second-best results. ↑/↓ indicate higher/lower is better. Results on the S&P 500 are in Table 4 in the Appendix.

Predictive Power (%)

Method	CSI300				CSI500				CSI1000			
	$IC^{\uparrow}$	$ICIR^{\uparrow}$	$RIC^{\uparrow}$	$RICIR^{\uparrow}$	$IC^{\uparrow}$	$ICIR^{\uparrow}$	$RIC^{\uparrow}$	$RICIR^{\uparrow}$	$IC^{\uparrow}$	$ICIR^{\uparrow}$	$RIC^{\uparrow}$	$RICIR^{\uparrow}$
Alpha158	3.91	25.80	5.77	37.61	5.24	38.16	7.72	54.78	7.85	51.42	10.10	65.85
GP	1.36	9.97	1.96	14.63	2.83	20.76	5.82	41.67	4.85	46.77	6.98	69.37
AlphaGen	4.93	29.41	6.32	38.95	4.77	38.13	6.35	49.54	7.44	54.85	9.68	71.99
AlphaForge	4.56	29.16	5.17	33.14	5.17	31.63	7.85	49.90	8.24	57.84	10.30	68.90
AlphaQCM	4.03	32.99	4.53	36.13	5.06	34.17	7.73	52.09	7.90	56.31	10.63	76.79
AlphaSAGE	5.02	35.82	6.31	44.58	4.46	35.62	6.58	54.89	7.27	58.45	9.41	76.77
AlphaAgent	4.27	25.66	5.45	30.94	5.50	39.90	7.45	54.90	8.01	54.00	10.95	60.86
R&D-Agent(Q)	4.88	29.39	6.30	36.73	5.81	37.72	7.94	52.35	8.58	61.00	11.25	76.03
ALPHAPROBE	**5.84**	**39.02**	**7.20**	**46.94**	**6.26**	**52.39**	**8.78**	**73.18**	**9.04**	**70.49**	**11.35**	**88.02**

Portfolio Construction

Method	CSI300			CSI500			CSI1000		
	$AR^{\uparrow}$	$MDD^{\downarrow}$	$SR^{\uparrow}$	$AR^{\uparrow}$	$MDD^{\downarrow}$	$SR^{\uparrow}$	$AR^{\uparrow}$	$MDD^{\downarrow}$	$SR^{\uparrow}$
Alpha158	6.03%	25.29%	0.3925	10.39%	25.78%	0.5682	9.32%	34.10%	0.4372
GP	3.46%	39.06%	0.1882	6.52%	23.46%	0.3421	10.23%	38.16%	0.4078
AlphaGen	6.19%	31.40%	0.3425	7.49%	31.85%	0.3429	10.96%	36.12%	0.4416
AlphaForge	1.01%	35.34%	0.0694	8.09%	23.09%	0.4257	13.72%	**31.86%**	0.5945
AlphaQCM	3.42%	24.13%	0.1973	8.86%	30.13%	0.4118	13.98%	32.69%	0.5786
AlphaSAGE	4.15%	30.79%	0.1832	10.46%	23.48%	0.5220	14.98%	32.03%	0.6099
AlphaAgent	3.43%	31.74%	0.1883	6.91%	31.57%	0.3588	12.29%	32.18%	0.5861
R&D-Agent(Q)	6.06%	31.38%	0.3456	12.36%	30.74%	0.5165	13.38%	33.53%	0.5968
ALPHAPROBE	**7.50%**	**22.25%**	**0.4411**	**17.45%**	**22.98%**	**0.8262**	**16.68%**	31.95%	**0.6475**

4.3 Dynamic Factor Integrator

We follow the approach proposed by AlphaForge [21] to dynamically re-select recently effective factors and integrate them into a "Mega" factor $F_{\hat{y}}$ for portfolio construction.

5 Empirical Experiments

5.1 Experiment Setup

Evaluation metrics: Following [25, 3], we employ two evaluation types. (1)**Predictive Power**: Information Coefficient (IC), IC Information Ratio (ICIR), Rank Information Coefficient (RIC), RIC Information Ratio (RICIR). These metrics are all computed against the 20-day forward returns. (2) **Portfolio Construction**: including Annualized Return (AR), Maximum Drawdown (MDD), Sharpe Ratio (SR). Details about evaluation metrics and backtest settings are provided in the Appendix 8.1.1.

Datasets: Following [25, 3], we conduct experiments on four stock pools: the CSI 300 (large cap), the CSI 500 (mid cap), the CSI 1000 (small cap) and S&P 500. The split of the training/validation/test set is defined as 2010-01-01 to 2020-12-31 / 2021-01-01 to 2022-06-30 / 2022-07-01 to 2025-06-30.

Baselines: We compare ALPHAPROBE with various baselines: (1) Expert-Collected factor pool: Alpha158 [29] (2) DFG methods: AlphaGen [30], AlphaForge [21], AlphaQCM [35], AlphaSAGE [3]. (3) IFE methods: GP [4], AlphaAgent [25], R&D-Agent(Q) [14]. More details about baselines and implementation are in the Appendix 8.1.2.

Implementation details: We use Deepseek V3.1 [6] as the backbone LLM for ALPHAPROBE and all LLM-based baselines (AlphaAgent, R&D-Agent(Q)). We use Qwen3 Embedding-4B [32] as the embedding model in Equation 7. We use the absolute value of the factor's ICIR during the training period as the quality measurement in Equation 4. The number of generated factors in Equation 13,14 is set to 5. To ensure a fair comparison, we standardize key settings across all methods: the pool capacity is 50 and the maximum expression length is 40 for ALPHAPROBE and all baselines, following [30, 21, 3]. All baselines use the same dynamic factor integrator in Section 4.3 and the same feature and operator set. Following [30], when the factor pool hits the size limit, we filter out the factor with the lowest quality. The depth penalty γ and the retrieval penalty ω are 0.05 and 0.10.

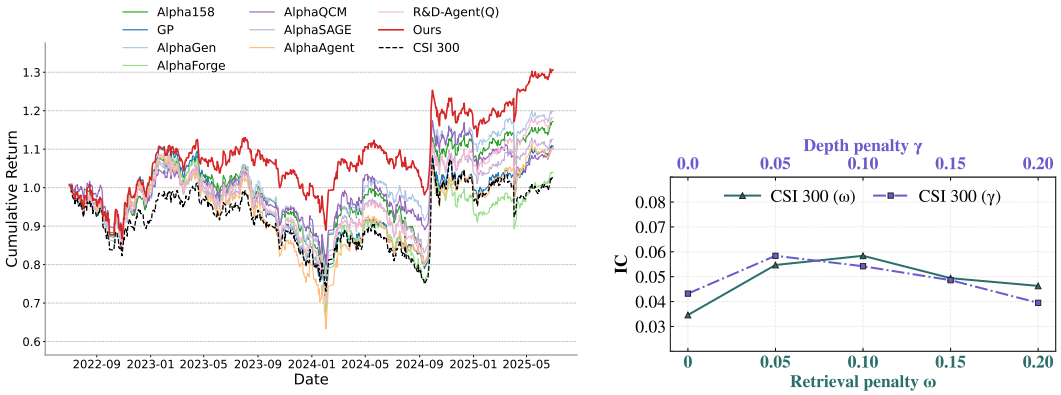
Table 2: Ablation study of ALPHAPROBE's core components. The results demonstrate the effectiveness of the Bayesian Factor Retriever and the DAG-aware Factor Generator. All values are in percent.

Method	CSI 300				CSI 500				CSI 1000			
	IC	ICIR	RIC	RICIR	IC	ICIR	RIC	RICIR	IC	ICIR	RIC	RICIR
Retriever Ablation												
w/ Random Retriever	2.95	18.71	3.18	21.65	3.82	21.44	4.17	26.93	5.64	32.45	6.88	43.30
w/ KNN Retriever	4.01	28.54	5.14	34.62	4.39	34.71	5.97	50.88	6.21	48.32	7.82	60.77
w/ Heuristic Retriever	4.54	35.15	5.92	35.71	5.11	41.86	6.08	39.03	7.74	50.09	8.15	58.33
w/ MCTS Retriever	4.75	35.20	5.94	36.86	5.09	43.25	6.11	38.71	7.85	49.94	8.37	59.27
w/o Prior	4.13	34.88	5.68	34.42	4.82	40.07	5.99	36.87	7.60	47.72	7.95	58.01
w/o Topology penalty	5.06	36.37	6.27	40.05	5.80	44.38	8.27	69.24	8.37	54.62	9.68	64.89
w/o Likelihood	4.09	34.99	5.66	34.39	4.86	40.02	6.01	36.98	7.49	46.96	7.67	57.70
w/o NLF	5.15	37.01	6.40	42.17	5.93	43.88	8.29	68.94	8.45	56.08	10.13	67.96
Generator Ablation												
w/ CoT Generator	5.11	31.79	6.08	39.17	5.41	39.70	8.35	63.00	8.03	60.12	9.75	66.54
ALPHAPROBE	**5.84**	**39.02**	**7.20**	**46.94**	**6.26**	**52.39**	**8.78**	**73.18**	**9.04**	**70.49**	**11.35**	**88.02**

5.2 Main Experiments

Table 1 summarizes the empirical results across three distinct datasets, from which two primary conclusions can be drawn. First, ALPHAPROBE exhibits superior predictive power for future stock returns, consistently achieving the highest IC, RIC, and AR. Recent research like [25, 14] also exhibits promising results, underscoring the significant potential of leveraging LLMs for alpha factor discovery. Second, ALPHAPROBE demonstrates enhanced stability against market regime shifts. This is substantiated by its significantly higher ICIR, RICIR, and SR, coupled with a lower MDD.

5.3 Backtest Experiments



(a) Backtest curve on the CSI 300.

(b) Sensitivity study of ALPHAPROBE.

Figure 3: (a) ALPHAPROBE maintains a leading edge in cumulative returns and demonstrates superior resilience during market stress events. (b) Parameter sensitivity on γ and ω.

As shown in Figure 3a, ALPHAPROBE not only maintains a leading edge in returns throughout the vast majority of the backtest period but also demonstrates superior resilience. Specifically, it exhibits more controlled drawdowns and faster recoveries during significant market stress events, such as the bear market from late 2023 to early 2024 and the tariff-induced turmoil in April 2025. We further stress-test robustness under doubled transaction costs and liquidity filters in Appendix 8.1.3.

5.4 Ablation Study

Table 2 summarizes the ablation study, evaluating components across the retriever and generator:

- **w/ Random Retriever**: Replacing our retriever with random selection leads to a sharp performance drop, proving that ignoring factor relationships results in noisy optimizations.
- **w/ KNN Retriever**: Using a flat factor pool with KNN-based retrieval (without any DAG structure) also underperforms ALPHAPROBE, confirming that the performance gains stem from topology-aware retrieval rather than simply maintaining a memory buffer.
- **w/ Heuristic Retriever**: Ranking factors by individual metrics [7] without structural context yields sub-optimal results, proving the necessity of strategic information encoded in the DAG.

- **w/ MCTS Retriever**: Local-view MCTS [22] falls short of AlphaPROBE, confirming a global topological view is superior to local lineage tracking.
- **w/o Prior / Topology Penalty**: Removing the prior term or the topology penalty degrades performance, demonstrating that intrinsic factor quality are essential for effective retrieval and topology penalty mitigates overfitting.
- **w/o Likelihood / NLF**: The decline in performance upon removing these components indicates that pool-level context and ancestral lineage provide critical, non-redundant information for discovery.
- **w/ CoT Generator**: Substituting the DAG-aware generator with simple alpha chains confirms that the DAG structure encodes richer interactions and better guides the evolutionary search.

5.5 Robustness Analysis

We investigate the sensitivity of AlphaPROBE to the hyperparameters γ and ω in Equation 4 on CSI 300. As shown in Figure 3b, AlphaPROBE demonstrates considerable robustness, with performance remaining stable across a wide range, especially within the 0.05 to 0.15 interval.

Extreme parameter values degrade performance by disrupting the balance between exploitation and exploration. An overly small γ or ω leads to over-exploitation, risking the selection of over-fitted factors or repeatedly using the same parents. Conversely, excessively large values cause under-exploitation, with a high γ prematurely penalizing deep, promising factors and a high ω forcing exploration away from high-potential areas.

Beyond hyperparameters, we further evaluate robustness to key infrastructure choices in the Appendix. Table 5 reports results across four backbone LLM and embedding model combinations. AlphaPROBE consistently outperforms baselines regardless of the configuration, confirming that performance gains stem from algorithmic design rather than a specific model choice. Table 6 examines sensitivity to prompt templates and LLM temperature. The performance variations remain minimal, indicating that AlphaPROBE is robust to these practical deployment choices.

5.6 Visualization

To illustrate the factor discovery process, we present a visualization on factors for interday price movements, with the resulting DAG shown in Figure 5. The process can originate from a simple, human-designed seed factor such as `Div(Sub(Less($open, $close), $low), $open)` about interday price movements. Our framework identifies this as a promising parent and generates a more complex, high-quality descendant, `TsCorr(Sub($close, $open), Sub($low, TsMin($low, 5)), 5)`, by introducing a time-series correlation. Besides, we can observe that AlphaPROBE can grab different potential of candidate factors. This evolutionary step highlights how AlphaPROBE establishes clear optimization pathways. Further details on all factors are in Table 9 the Appendix.

5.7 Efficiency Analysis

We further investigate the training efficiency of AlphaPROBE by comparing its convergence speed with LLM-based baselines. As shown in Figure 4, by leveraging global information with structural topology, the retriever in AlphaPROBE identifies the most promising factors, enabling it to achieve better results with fewer training iterations and thus demonstrating its efficiency. In addition to convergence speed, we compare the API cost. As shown in Table 7 in the Appendix, AlphaPROBE incurs roughly 40% lower cost than both AlphaAgent and R&D-Agent(Q), because its topology-aware retrieval reduces the number of LLM calls needed to reach the same pool quality.

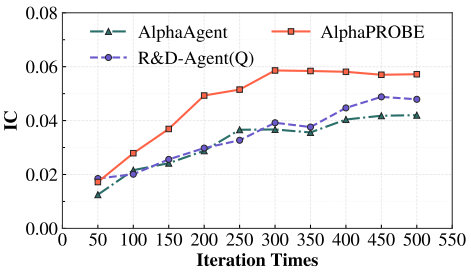


Figure 4: IC dynamics of AlphaPROBE and two LLM-based methods on CSI 300 over training iterations. For fair comparison, one iteration means the backbone LLM generates a new factor to be evaluated.

6 Conclusion

In this paper, we introduce ALPHAPROBE, a closed-loop retriever & generator framework designed for alpha mining. ALPHAPROBE leverages a Bayesian Factor Retriever to capture factors' exploitation-exploration trade-off beyond factor topology, and a DAG-aware Factor Generator to perform context-aware optimization. This approach offers enhanced predictive performance, stability, hierarchy and robustness on Chinese and US stock pools with distinct styles.

Looking ahead, we anticipate ALPHAPROBE will extend beyond traditional daily-frequency scenarios to high-frequency and fundamental factor mining.

7 Acknowledgements

This paper is partially supported by the National Key Research and Development Program of China with Grant No. 2023YFC3341203 as well as the National Natural Science Foundation of China with Grant Number 62276002. This paper is also partially supported by Zhengren Quant.

References

[1] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[2] Lang Cao, Zekun Xi, Long Liao, Ziwei Yang, and Zheng Cao. Chain-of-alpha: Unleashing the power of large language models for alpha mining in quantitative trading. *arXiv preprint arXiv:2508.06312*, 2025.

[3] Binqi Chen, Hongjun Ding, Ning Shen, Jinsheng Huang, Taian Guo, Luchen Liu, and Ming Zhang. Alphasage: Structure-aware alpha mining via gflownets for robust exploration. *arXiv preprint arXiv:2509.25055*, 2025.

[4] Tianxiang Chen, Wei Chen, and Luyao Du. An empirical study of financial factor mining based on gene expression programming. In *2021 4th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*, pages 1113–1117. IEEE, 2021.

[5] Keith Cuthbertson and Dirk Nitzsche. *Quantitative financial economics: stocks, bonds and foreign exchange*. John Wiley & Sons, 2005.

[6] DeepSeek-AI. Deepseek-v3 technical report, 2024. URL `https://arxiv.org/abs/2412.19437`.

[7] Hongjun Ding, Binqi Chen, Jinsheng Huang, Taian Guo, Zhengyang Mao, Guoyi Shao, Lutong Zou, Luchen Liu, and Ming Zhang. Alphaeval: A comprehensive and efficient evaluation framework for formula alpha mining, 2025. URL `https://arxiv.org/abs/2508.13174`.

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[9] Jun Han, Shuo Zhang, Wei Li, Zhi Yang, Yifan Dong, Tu Hu, Jialuo Yuan, Xiaomin Yu, Yumo Zhu, Fangqi Lou, et al. Quantaalpha: An evolutionary framework for llm-driven alpha mining. *arXiv preprint arXiv:2602.07085*, 2026.

[10] Zuoyou Jiang, Li Zhao, Rui Sun, Ruohan Sun, Zhongjian Li, Jing Li, Daxin Jiang, Zuo Bai, and Cheng Hua. Alpha-r1: Alpha screening with llm reasoning via reinforcement learning. *arXiv preprint arXiv:2512.23515*, 2025.

[11] Zhizhuo Kou, Holam Yu, Junyu Luo, Jingshu Peng, Xujia Li, Chengzhong Liu, Juntao Dai, Lei Chen, Sirui Han, and Yike Guo. Automate strategy finding with llm in quant investment. In *In Findings of the Association for Computational Linguistics: EMNLP 2025, pages 18517–18533, Suzhou, China. Association for Computational Linguistics.*, 2025.

[12] Cheng F Lee, Alice C Lee, and John C Lee. *Handbook of quantitative finance and risk management*, volume 1. Springer, 2010.

[13] Shicheng Li, Jinshan Zhang, and Feng Wang. Alphagat: A two-stage learning approach for adaptive portfolio selection. In James Kwok, editor, *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 7500–7508. International Joint Conferences on Artificial Intelligence Organization, 8 2025. doi: 10.24963/ijcai.2025/834. URL `https://doi.org/10.24963/ijcai.2025/834`. Main Track.

[14] 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. In *The Thirty-Ninth Annual Conference on Neural Information Processing Systems*, 2025.

[15] Zhiwei Li, Ran Song, Caihong Sun, Wei Xu, Zhengtao Yu, and Ji-Rong Wen. Can large language models mine interpretable financial factors more effectively? a neural-symbolic factor mining agent model. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 3891–3902, 2024.

[16] Fengyuan Liu, Huang Yi, Sichun Luo, Yuqi Wang, Yazheng Yang, Xinye Li, Zefa Hu, Junlan Feng, and Qi Liu. Cognitive alpha mining via llm-driven code-based evolution. *arXiv preprint arXiv:2511.18850*, 2025.

[17] Haochen Luo, Yuan Zhang, and Chen Liu. Efs: Evolutionary factor searching for sparse portfolio optimization using large language models, 2025. URL `https://arxiv.org/abs/2507.17211`.

[18] Francesco Rundo, Francesca Trenta, Agatino Luigi Di Stallo, and Sebastiano Battiato. Machine learning for quantitative finance applications: A survey. *Applied Sciences*, 9(24):5574, 2019.

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

[20] William F Sharpe. The sharpe ratio. *Streetwise–the Best of the Journal of Portfolio Management*, 3(3):169–85, 1998.

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

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

[23] Shuo Sun, Rundong Wang, and Bo An. Reinforcement learning for quantitative trading. *ACM Transactions on Intelligent Systems and Technology*, 14(3):1–29, 2023.

[24] Ziyi Tang, Xuexiong Yin, Weixing Chen, Zechuan Chen, Yongsen Zheng, Wenxuan Ye, Keze Wang, and Liang Lin. Alphaagentevo: Evolution-oriented alpha mining via self-evolving agentic reinforcement learning. In *The Fourteenth International Conference on Learning Representations*.

[25] 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 V. 2*, pages 2813–2822, 2025.

[26] Saizhuo Wang, Hang Yuan, Leon Zhou, Lionel M Ni, Heung-Yeung Shum, and Jian Guo. Alpha-gpt: Human-ai interactive alpha mining for quantitative investment. *arXiv preprint arXiv:2308.00016*, 2023.

[27] Yanlong Wang, Jian Xu, Hongkang Zhang, Shao-Lun Huang, Danny Dongning Sun, and Xiao-Ping Zhang. Factorminer: A self-evolving agent with skills and experience memory for financial alpha discovery. *arXiv preprint arXiv:2602.14670*, 2026.

[28] Paul Wilmott. *Paul Wilmott on quantitative finance*. John Wiley & Sons, 2013.

[29] Xiao Yang, Weiqing Liu, Dong Zhou, Jiang Bian, and Tie-Yan Liu. Qlib: An ai-oriented quantitative investment platform, 2020. URL https://arxiv.org/abs/2009.11189.

[30] 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*, pages 5476–5486, 2023.

[31] Tianping Zhang, Yuanqi Li, Yifei Jin, and Jian Li. Autoalpha: an efficient hierarchical evolutionary algorithm for mining alpha factors in quantitative investment. *arXiv preprint arXiv:2002.08245*, 2020.

[32] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

[33] Junjie Zhao, Chengxi Zhang, Min Qin, and Peng Yang. Quantfactor reinforce: mining steady formulaic alpha factors with variance-bounded reinforce. *IEEE Transactions on Signal Processing*, 2025.

[34] Junjie Zhao, Chengxi Zhang, Chenkai Wang, and Peng Yang. Learning from expert factors: Trajectory-level reward shaping for formulaic alpha mining, 2025. URL https://arxiv.org/abs/2507.20263.

[35] Zhoufan Zhu and Ke Zhu. Alphaqcm: Alpha discovery in finance with distributional reinforcement learning. In *Forty-second International Conference on Machine Learning*, 2025.

8 Appendix

8.1 Experiment details

8.1.1 Details about evaluation metrics

In this section, we define the metrics used to evaluate the performance of a predictive model (or alpha factor), denoted by α. On any given day d, the model processes a set of features for all assets in the universe, X_d, to generate a vector of predictive scores, $\alpha(X_d)$. These scores are evaluated against the vector of actual next-period outcomes, y_d (e.g., forward returns). All statistical moments, such as covariance (Cov) and variance (Var), are computed cross-sectionally across the asset universe for that specific day.

The foundational metric is the daily cross-sectional Pearson correlation between predictions and outcomes:

$$\rho_d = \frac{\mathrm{Cov}(\alpha(X_d), y_d)}{\sqrt{\mathrm{Var}(\alpha(X_d))\,\mathrm{Var}(y_d)}}.$$

For metrics based on portfolio simulation, let R_d denote the portfolio return constructed from the factor scores $\alpha(X_d)$ on day d. Let K be the number of periods per year (e.g., $K = 252$ for daily data), $r_{f,d}$ be the risk-free rate, and W_t be the cumulative wealth at time t:

$$W_t = \prod_{u=1}^{t} (1 + R_u).$$

Information Coefficient (IC) The Information Coefficient is the time-series average of the daily cross-sectional correlations: $\mathrm{IC} = \mathrm{E}_d[\rho_d]$. It measures the cross-sectional predictive power of the factor—how well its predictions $\alpha(X_d)$ align with outcomes y_d. Using the absolute value isolates the magnitude of the correlation from its sign (as a consistently negative correlation is also predictive).

Information Ratio of IC (ICIR)

$$\mathrm{ICIR} = \frac{\mathrm{E}_d[\rho_d]}{\sqrt{\mathrm{Var}_d(\rho_d)}}.$$

Time-series consistency of cross-sectional predictability; the mean of daily correlations relative to its standard deviation. ICIR approximates a signal-to-noise measure (akin to a t-statistic for $\mathrm{E}[\rho_d]$), favoring factors that perform consistently over time.

Rank Information Coefficient (RIC)

$$\mathrm{RIC} = \mathrm{E}_d[\rho_d^{\text{rank}}], \qquad \rho_d^{\text{rank}} = \frac{\mathrm{Cov}(\mathrm{rank}(\alpha(X_d)), \mathrm{rank}(y_d))}{\sqrt{\mathrm{Var}(\mathrm{rank}(\alpha(X_d)))\,\mathrm{Var}(\mathrm{rank}(y_d))}}.$$

The Spearman correlation counterpart to IC, which evaluates if higher-ranked predictions correspond to higher-ranked outcomes. RankIC is robust to outliers and non-linear monotonic relationships.

Information Ratio of RankIC (RICIR)

$$\mathrm{RICIR} = \frac{\mathrm{E}_d[\rho_d^{\text{rank}}]}{\sqrt{\mathrm{Var}_d(\rho_d^{\text{rank}})}}. \tag{30}$$

Measure the time-series stability of the rank-based predictive power, prioritizing factors whose cross-sectional ordering of assets remains reliable over time.

Annualized Return (AR)

$$\mathrm{AR} = K \cdot \mathrm{E}_d[R_d].$$

The average economic value produced by the portfolio strategy induced by α. When compounding is significant, geometric annualization is often preferred.

Maximum Drawdown (MDD)

$$\mathrm{MDD} = \max_{t} \left(1 - \frac{W_t}{\max_{u \le t} W_u} \right).$$

The largest peak-to-trough loss in cumulative wealth. It is a critical tail-risk metric not captured by variance alone, relevant for risk management and assessing investor experience.

Sharpe Ratio (SR) [20] (annualized, excess over risk-free)

$$\mathrm{SR} = \frac{\sqrt{K}\,\mathrm{E}_d[R_d - r_{f,d}]}{\sqrt{\mathrm{Var}_d(R_d - r_{f,d})}}.$$

The risk-adjusted return per unit of volatility for the α-induced portfolio, enabling fair comparison across different strategies, asset universes, and rebalancing frequencies.

8.1.2 Details about baselines

We conduct experiments on various competitive baselines. They can be separated into three categories:(1) Factors collected by human experts. (2) Decoupled Factor Generation(DFG). (3) Iterative Factor Evolution(IFE).

Expert-collected Factors :

Alpha158 [29] is a well-established factor pool constructed by Microsoft Qlib [1] platform.

DFG methods :

- AlphaGen [30]: AlphaGen leverages the strong exploratory capabilities of reinforcement learning (RL) to better explore the vast search space based on policy gradient algorithms. We use the open-source link [2] to implement AlphaGen. of formulaic alphas.
- AlphaForge [21]: AlphaForge is a novel deep learning framework designed for formulaic alpha mining by a surrogate model to predict fitness score and a composite model to dynamically combine factors during test time combination. We use the open-source link [3] to implement AlphaForge.
- AlphaQCM [35]: AlphaQCM is a powerful distributional reinforcement learning framework for alpha discovery relying on the unbiased estimation of variance derived from potentially biased quantiles. We use the open-source link [4] to implement AlphaQCM.
- AlphaSAGE [3]: AlphaSAGE is a recently proposed framework designed for alpha mining by unifying a GNN encoder for symbolic expressions, a GFlowNet generator that explores multiple high-reward modes, and a multi-signal training objective coupling predictive quality.We use the open-source link [5] to implement AlphaSAGE.

IFE methods :

- GP [4]: Genetic programming performs symbolic regression by evolving expression trees via mutation and crossover, yielding human-readable formulas. We use the gplearn package to implement this baseline [6].
- AlphaAgent [25]: AlphaAgent is a novel LLM-driven alpha mining framework with three key regularization mechanisms: originality enforcement,complexity control, and hypothesis alignment. We use the open-source link [7] to implement AlphaAgent.
- R&D-Agent(Q) [14]:R&D-Agent(Quant) is a powerful LLM-driven framework for collaborative factor-model development in quantitative finance.We use the open-source link [8] to implement R&D-Agent(Q).

[1] https://github.com/microsoft/qlib
[2] https://github.com/RL-MLDM/alphagen
[3] https://github.com/DulyHao/AlphaForge
[4] https://github.com/ZhuZhouFan/AlphaQCM
[5] https://github.com/BerkinChen/AlphaSAGE
[6] https://gplearn.readthedocs.io/en/stable
[7] https://github.com/RndmVariableQ/AlphaAgent
[8] https://github.com/microsoft/RD-Agent

8.1.3 More details about backtest experiments

During backtesting, we purchase the top 20% of stocks ranked by factor scores each trading day with equal weights and sell them after 20 days (long positions only), with a round-trip transaction cost of 0.1%.

To further stress-test robustness, we conduct additional backtests under stricter conditions: transaction costs are doubled from the original setting, and stocks with daily trading volume below 20 million RMB are excluded as a liquidity filter. As shown in Table 3, the performance degradation is slight across all three datasets, confirming that ALPHAPROBE's discovered factors are robust to realistic market frictions.

Table 3: **Backtest under stressed conditions.** Standard: original setting (0.1% cost). Stressed: doubled cost + liquidity filter.

Setting	CSI 300			CSI 500			CSI 1000		
	$AR^{\uparrow}$	$MDD^{\downarrow}$	$SR^{\uparrow}$	$AR^{\uparrow}$	$MDD^{\downarrow}$	$SR^{\uparrow}$	$AR^{\uparrow}$	$MDD^{\downarrow}$	$SR^{\uparrow}$
Standard	7.50%	22.25%	0.4411	17.45%	22.98%	0.8262	16.68%	31.95%	0.6475
Stressed	7.47%	22.31%	0.4385	17.29%	23.16%	0.8194	16.28%	32.84%	0.6399

8.1.4 Results on the S&P 500

To evaluate the generalization of ALPHAPROBE beyond the Chinese market, we conduct additional experiments on the S&P 500 index constituents following the same experimental protocol. Table 4 reports both predictive power and portfolio construction metrics. ALPHAPROBE achieves the best or second-best performance across all metrics, confirming that the discovered factors generalize to a structurally different market.

Table 4: **Performance on the S&P 500.** *Bold* and underlined numbers denote the best and second-best results. $^{\uparrow}/^{\downarrow}$ indicate higher/lower is better. All predictive power values are in %.

Method	Predictive Power (%)				Portfolio Construction		
	$IC^{\uparrow}$	$ICIR^{\uparrow}$	$RIC^{\uparrow}$	$RICIR^{\uparrow}$	$AR^{\uparrow}$	$MDD^{\downarrow}$	$SR^{\uparrow}$
Alpha158	2.58	20.12	2.35	21.50	16.12%	19.06%	0.87
GP	3.07	27.81	3.22	29.21	19.37%	15.94%	1.08
AlphaGen	3.52	30.28	3.61	27.59	18.45%	17.51%	0.99
AlphaForge	3.88	31.29	3.67	29.58	18.36%	**15.72%**	1.13
AlphaQCM	2.96	24.70	2.81	27.89	15.61%	17.08%	1.00
AlphaSAGE	5.20	49.30	3.80	38.20	19.24%	16.53%	1.05
AlphaAgent	4.13	38.07	3.59	38.75	18.73%	17.19%	1.03
R&D-Agent(Q)	4.88	47.94	3.76	37.91	18.86%	17.40%	1.09
ALPHAPROBE	**5.43**	**49.65**	**4.02**	**39.17**	**20.06%**	15.89%	**1.16**

8.1.5 Robustness to Backbone LLM and Embedding Model

To verify that ALPHAPROBE's gains are not tied to a specific LLM or embedding model, we evaluate four configurations on the CSI 300. Each configuration is run three times and we report mean(std). As shown in Table 5, ALPHAPROBE consistently outperforms R&D-Agent(Q) across all combinations.

8.1.6 Robustness to Prompt Template and Temperature

We further test the sensitivity of ALPHAPROBE to the prompt template and generation temperature. For prompt sensitivity, we adapt AlphaAgent's feedback analysis prompt as an alternative template. For temperature sensitivity, we lower the LLM temperature from 1.0 to 0.5. As shown in Table 6, the performance variations are minimal across all datasets.

8.1.7 API Cost Comparison

Table 7 compares the total LLM API cost (in USD) of the three LLM-based methods across all datasets. ALPHAPROBE achieves substantially lower cost because its topology-aware retrieval

Table 5: **Robustness to backbone LLM and embedding model** on CSI 300. Mean(std) over 3 runs, all values in %.

Configuration	Method	$IC^{\uparrow}$	$ICIR^{\uparrow}$	$RIC^{\uparrow}$	$RICIR^{\uparrow}$
DS V3.1 + Qwen3-Emb	R&D-Agent(Q)	4.98(0.27)	29.49(5.84)	6.28(0.39)	37.08(6.08)
	ALPHAPROBE	**5.75**(0.40)	**38.70**(5.11)	**7.26**(0.41)	**47.25**(6.54)
DS V3.1 + text-emb-3-lg	R&D-Agent(Q)	5.23(0.35)	34.64(6.41)	6.35(0.49)	35.94(6.88)
	ALPHAPROBE	**5.92**(0.47)	**39.84**(5.95)	**6.92**(0.53)	**45.56**(7.21)
Qwen3.5-Plus + Qwen3-Emb	R&D-Agent(Q)	5.42(0.41)	32.18(6.74)	6.63(0.52)	40.21(7.09)
	ALPHAPROBE	**6.18**(0.51)	**41.92**(6.08)	**7.14**(0.56)	**46.72**(7.34)
Qwen3.5-Plus + text-emb-3-lg	R&D-Agent(Q)	5.39(0.38)	33.46(6.36)	6.74(0.48)	41.48(6.96)
	ALPHAPROBE	**6.02**(0.47)	**40.38**(5.82)	**7.48**(0.64)	**48.74**(7.06)

Table 6: **Robustness to prompt template and temperature.** All values are in %.

Setting	CSI 300				CSI 500				CSI 1000			
	IC	ICIR	RIC	RICIR	IC	ICIR	RIC	RICIR	IC	ICIR	RIC	RICIR
Original prompt	5.84	39.02	7.20	46.94	6.26	52.39	8.78	73.18	9.04	70.49	11.35	88.02
Adapted prompt	5.78	38.61	7.15	46.37	6.21	51.94	8.81	72.46	8.97	69.88	11.27	87.31
Temp = 1.0	5.84	39.02	7.20	46.94	6.26	52.39	8.78	73.18	9.04	70.49	11.35	88.02
Temp = 0.5	5.81	38.46	7.12	45.98	6.14	52.42	8.81	72.94	9.16	70.05	11.49	87.89

selects more promising parents, reducing the number of generation rounds needed to fill the factor pool.

Table 7: **API cost comparison** of LLM-based methods (in USD).

Method	CSI 300	CSI 500	CSI 1000
AlphaAgent	$0.91	$0.90	$0.88
R&D-Agent(Q)	$0.95	$1.02	$0.98
ALPHAPROBE	**$0.54**	**$0.57**	**$0.53**

8.2 More details about ALPHAPROBE

8.2.1 Implementation of DAG-aware Factor Generator

Strategy & Execution Agent

System & User Prompts

```
System prompts You are an expert on quantitative finance and alpha factor mining.
Strictly follow the instructions given by user below.  Make sure the output ONLY
CONTAINS A JSON FORMAT. Do not output anything else other than the JSON object,
including code block markers like ‘‘‘ or ‘‘‘json.
User prompts for features and operators definition
The available features, constants and operators are listed below.
1.  You can use the following features:
$open, $high, $low, $close:  Opening, daily highest, daily lowest, and closing
prices.
$vwap:  Daily average price, weighted by the volume of trades at each price.
$volume:  Trading number of shares.
2.  You can use int constants eg:1, 3, 5,10, 20...  etc during
rolling(time-series) calculations, and float :  0.0001 0.01, 0.0, 1.0, 2.0
during arithmetic calculations.  Other constants are not allowed.
3.  The following operators are available:
(BEGIN OF FEATURES AND OPERATORS DEFINITIONS)
Abs(x):  Absolute value of x
```

```
Log(x):  Natural logarithm of x
SLog1p(x):  Signed log transform:  sign(input) times log of (1 plus the absolute
value)
Sign(x):  Sign of x:  1 if x > 0, -1 if x < 0, 0 if x = 0
Rank(x):  Cross-sectional rank of x
Add(x,y):  x + y
Sub(x, y):  x - y
Mul(x, y):  x * y
Div(x, y):  x / y
Pow(x, y):  x raised to the power of y (x ** y) y must be a constant
Greater(x, y):  1 if x > y, else 0
Less(x, y):  1 if x < y, else 0
GetGreater(x, y):  x if x > y, else y
GetLess(x, y):  x if x < y, else y
Ref(x, d):  Value of x d days ago
TsMean(x, d):  Rolling mean of x over the past d days
TsSum(x, d):  Rolling sum of x over the past d days
TsStd(x, d):  Rolling standard deviation of x over the past d days
TsMin(x, d):  Rolling minimum of x over the past d days
TsMax(x, d):  Rolling maximum of x over the past d days
TsMinMaxDiff(x, d):  Difference between TsMax(x, d) and TsMin(x, d)
TsMaxDiff(x, d):  Difference between current x and TsMax(x, d)
TsMinDiff(x, d):  Difference between current x and TsMin(x, d)
TsIr(x, d):  Rolling Information ratio over past d days
TsVar(x, d):  Rolling variance of x over the past d days
TsSkew(x, d):  Rolling skewness of x over the past d days
TsKurt(x, d):  Rolling kurtosis of x over the past d days
TsMed(x, d):  Rolling median of x over the past d days
TsMad(x, d):  Rolling median absolute deviation over the past d days
TsRank(x, d):  Time-series rank of x over the past d days
TsDelta(x, d):  Today's value of x minus the value of x d days ago
TsRatio(x, d):  Today's value of x divided by the value of x d days ago
TsPctChange(x, d):  Percentage change in x over the past d days
TsWMA(x, d):  Weighted moving average over the past d days with linearly
decaying weights.
TsEMA(x, d):  Exponential moving average of x with span d
TsCov(x, y, d):  Time-series covariance of x and y for the past d days
TsCorr(x, y, d):  Time-series correlation of x and y for the past d days
(END OF FEATURES AND OPERATORS DEFINITIONS)
Examples of valid alpha expressions:
Div(Sub($open, $close), Add(Sub($high, $low), 0.001))
```

User Prompts

```
User prompts for the strategy agent:
Your task is to generate a new expression based on the given expressions, the
given topic topic, the explanation of this expression, and the given generation
traces, such that:
1.  The new expression is valid(syntactically correct), and dimensionless (i.e.
Dim(expr) = 0).
2.  You can only use the features, constants and operators given above in
the (FEATURES AND OPERATORS DEFINITIONS). DO NOT MODIFY the name of any
features,operators.  All the operators should be used the same as their original
definitions, ie, the number and type of arguments should be the same as defined.
3.  As for constants, when you use it in rolling calculations, it should be
an integer annotated by %d ; when you use it in arithmetic calculations, it
should be float numbers listed in 0.0001 0.01, 0.0, 1.0, 2.0.  Do NOT use other
constants.  DO NOT use scientific notation.
4.  Besides the Ref() operator, the constants used in rolling calculations
should be the same, use "%d" for all of them.
5.  You should read the original expressions carefully, and try to understand
its semantic meaning in quantitative finance.  Then, you should try your best
to think how to express the core of meaning in a different way, or generate new
```

```
insights inspired by it.  Then, you can generate a new expression based on your
understanding.
6.  If the trace is not empty, then it contains the generation optimization
steps from the earliest to the latest.  You should learn how the expressions
are optimized before, and then generate a new expression based on the original
expressions and the generation traces.
Given the original expression, you should ONLY and STRICTLY output a JSON object
which contains the following contents:
{
"strategies":  ["the newly generated strategies for factor generation.The length
of expressions should be {num}."],
}
Given expressions:  {expressions}
Given expression explanations:  {explanations}
```

User Prompts

```
User prompts for the execution agent:
Your task is to generate a new expression based on the given expressions, the
given topic topic, the explanation of this expression, and the given generation
traces, such that:
1.  The new expression is valid(syntactically correct), and dimensionless (i.e.
Dim(expr) = 0).
2.  You can only use the features, constants and operators given above in
the (FEATURES AND OPERATORS DEFINITIONS). DO NOT MODIFY the name of any
features,operators.  All the operators should be used the same as their original
definitions, ie, the number and type of arguments should be the same as defined.
3.  As for constants, when you use it in rolling calculations, it should be
an integer annotated by %d ; when you use it in arithmetic calculations, it
should be float numbers listed in 0.0001 0.01, 0.0, 1.0, 2.0.  Do NOT use other
constants.  DO NOT use scientific notation.
4.  Besides the Ref() operator, the constants used in rolling calculations
should be the same, use "%d" for all of them.
5.  You should read the original expressions carefully, and try to understand
its semantic meaning in quantitative finance.  Then, you should try your best
to think how to express the core of meaning in a different way, or generate new
insights inspired by it.  Then, you can generate a new expression based on your
understanding.
6.  If the trace is not empty, then it contains the generation optimization
steps from the earliest to the latest.  You should learn how the expressions
are optimized before, and then generate a new expression based on the original
expressions and the generation traces.
7.  The new expression should be different from all the given expressions, and
should be novel and non-trivial, but it can share some common parts with them.
8.  The new expression should be related to the given topic topic, and you
should not generate expressions that are totally irrelevant to the topic.(EG:
generate a expression about corr between price and volume when the topic is
about volatility.  Generate a expression containing rolling operations when the
topic is about interday prices, etc.)
9.  Give {num} new expressions with different modification strategies, with
expressions soundness and explainable.  The num expressions should be different,
low correlated, looking semantically different, or using totally different
ways to specific semantics(For example, cross sectional(rank op) vs cross
time-series(Ts op) or combine them; different statistical measures(Try to
use new sound and interpretable operators that rarely or not exist in given
expressions, traces and expressions generated before.); different but related
semantic meanings, etc.).  and each of them should be valid and dimensionless.
10.  While maintaining relevance to the overarching topic{topic} and
interpretability, you are encouraged to use as diverse a range of operators /
features as possible across different expression outputs.
11.  In each generation, there is no need to make the expression more complex,
sometimes simplification is also a good way to generate novel expressions.
```

```
12.  After you generate each expressions, check whether it is valid,some INVALID
operations are:
12.1 Modify the name of operators or features.  Eg:  using "*" instead of "Mul",
using "closing_price" instead of "$close", using "TSMAD" instead of "TsMad",
etc.
12.2 The number of operands are not correct.  Eg:  Add(x), Div(x,y,z), Missing
rolling window %d in rolling operations(TsSkew, TsIr), etc.
12.3 When using constant in algorithmic calculations, using integer instead of
float, or vice versa.
12.4 Use scientific notation in constants.  You should transfer into float eg:
1e-4 -> 0.0001
12.5 Using constants other than those mentioned in 3.
12.6 Using the operators and features not mentioned in FEATURES AND OPERATORS
DEFINITIONS.
All the mentioned above in 12 are INVALID operations.  Try your best to AVOID /
FIX Them.
Given the original expression, you should ONLY and STRICTLY output a JSON object
which contains the following contents:
{
"expressions":  ["the newly generated expressions as a string list.  Each
element should contain a valid and dimensionless expression.The length of
expressions should be {num}."], "expressions_fixed":  ["for each generated
expression, if there is any invalid operations mentioned above in 12, you
should fix them and give the fixed expression here, otherwise just give the
original generated expression here.  The length of expressions_fixed should be
{num}.  Do not modify the semantics of the original generated expressions."],
"explanations":  ["brief explanations of new expressions, should be the meaning
of your given expression.  The length of explanations should be num.  And
each explanation should correspond to the expression in the same index in the
expressions list." ]
}
Given expressions:  {expressions}
Given expression explanations:  {explanations}
Generation traces:  {traces}
Given Strategies:  {strategies}
Do not output anything else other than the JSON object, including code block
markers like ‘‘‘ or ‘‘‘json.
```

Details about the validator agent A candidate factor F_c is admitted to the pool only if it passes a check. The design of this check embodies a trade-off: we accept a factor if it either significantly improves upon existing ideas (exploitation) or introduces a valuable new dimension to our pool (exploration). Formally, admission requires:

$$\Big(\text{Qual}(F_c) > \tau_q \wedge \text{Gain}(F_c, F_p) > 0\Big) \vee \Big(\text{Qual}(F_c) > \tau_q \wedge \text{Corr}(F_c, \mathcal{F}) < \tau_d\Big)$$

where $\text{Qual}(\cdot)$ is a measure of predictive power implemented by the absolute value of ICIR, $\text{Gain}(\cdot,\cdot)$ measures the percentage improvement in $\text{Qual}(\cdot)$, and $\text{Corr}(\cdot,\cdot)$ measures the novelty of the candidate relative to the existing pool by calculating maximum abs value of Pearson correlation coefficient between F_c and each factor in $\mathcal{F}$. The τ_q is set to 0.10 and τ_d is set to 0.70. Upon admission, the DAG is updated with the new factor and a corresponding edge.

8.3 Details about features and operators

The table below summarizes all available features and operators we use in ALPHAPROBE and all baselines.

Table 8: Raw features and operators. F: base market features; U/B: unary/binary operators; CS: cross-sectional operation (within-day across assets); TS: time-series rolling operation. The lookback length d denotes the past d trading days.

Name	Type	Description
Base Market Features		
Open	F	Opening price
Close	F	Closing price
High	F	Daily highest price
Low	F	Daily lowest price
Vwap	F	Daily average price, weighted by the volume of trades at each price
Volume	F	Trading volume (number of shares)
Unary and Cross-Sectional Operators		
Abs	U	Absolute value of the input
Slog1p	U	Signed log transform: $\text{sign}(\text{input}) \times \log(1 + \lvert\text{input}\rvert)$
Inv	U	Reciprocal of the input; add ϵ to avoid division by zero
Sign	U	Sign of the input, returning -1, 0, or 1
Log	U	Natural logarithm of the input; add ϵ for numerical stability
Rank	U-CS	Cross-sectional rank normalization within a day, mapped to the range [0, 1]
Binary Operators		
Add	B	Element-wise addition of two inputs
Sub	B	Element-wise subtraction: first minus second
Mul	B	Element-wise multiplication
Div	B	Element-wise division; add a small constant to the denominator for stability
Pow	B	Element-wise power: raise the first input to the power of the second
Greater	B	Element-wise comparison: 1 if first input is greater than second, else 0
Less	B	Element-wise comparison: 1 if first input is less than second, else 0
GetGreater	B	Element-wise comparison and retrieval: Retrieving the larger element
GetLess	B	Element-wise comparison and retrieval: Retrieving the smaller element
Time-Series Operators		
Ref	U-TS	Lag operator: the value from d days ago
TsMean	U-TS	Rolling mean over the past d days
TsSum	U-TS	Rolling sum over the past d days
TsStd	U-TS	Rolling standard deviation over the past d days
TsIr	U-TS	Rolling information ratio over the past d days
TsMinMaxDiff	U-TS	Rolling range over the past d days (rolling max minus rolling min)
TsMaxDiff	U-TS	Current value minus the rolling max over the past d days
TsMinDiff	U-TS	Current value minus the rolling min over the past d days
TsVar	U-TS	Rolling variance over the past d days
TsSkew	U-TS	Rolling skewness over the past d days
TsKurt	U-TS	Rolling kurtosis over the past d days
TsMax	U-TS	Rolling maximum over the past d days
TsMin	U-TS	Rolling minimum over the past d days
TsMed	U-TS	Rolling median over the past d days
TsMad	U-TS	Rolling median absolute deviation over the past d days
TsRank	U-TS	Rolling rank of the current value within the past d days, mapped to [0, 1]
TsDelta	U-TS	Change over d days: current value minus the value d days ago
TsDiv	U-TS	Ratio over d days: current value divided by the value d days ago
TsPctChange	U-TS	Percentage change over the past d days
TsWMA	U-TS	Linearly decaying weighted moving average over the past d days
TsEMA	U-TS	Exponential moving average with a decay over the past d days
TsCov	B-TS	Rolling covariance between two inputs over the past d days
TsCorr	B-TS	Rolling Pearson correlation between two inputs over the past d days

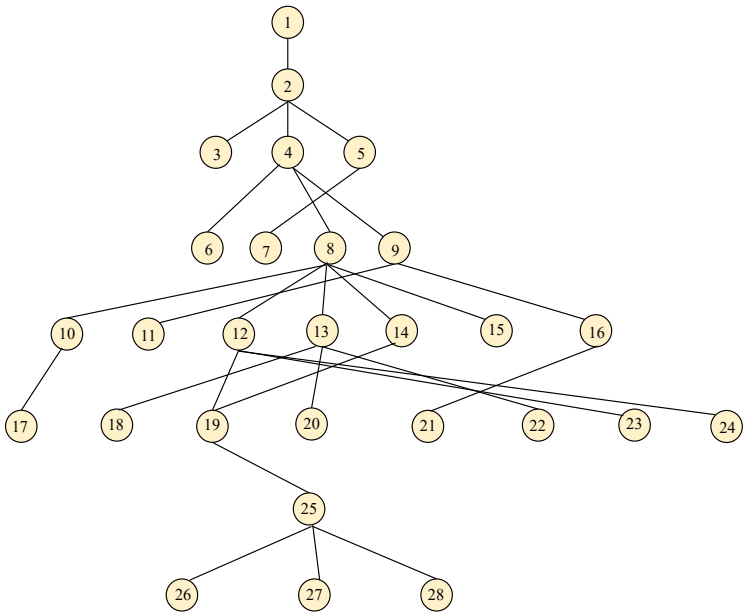


Figure 5: The topological structure of mining factors on the CSI 300 which derives from interday price movements. For more details, please refer to Table 9. The node indices in the figure correspond to the Factor ID in the table.

Table 9: Details about all factors in Figure 5. We report the factor ID, factor expressions and descriptions and their ICIR on the test period.

Factor ID	**Expression**	**Description**	**ICIR**
1	Div(Sub(Less($open, $close), $low), $open)	This expression measures the normalized daily price reversal signal by calculating the difference between a binary indicator of positive close-open return (1 if close $>$ open, else 0) and the low price, then scaling it by the opening price, which is useful for identifying potential mean-reversion opportunities.	0.2391
2	TsCorr(Sub($close, $open), Sub($low, TsMin($low, 5)), 5)	Correlates the daily return with the deviation of low price from its 5-day minimum, assessing co-movement in interday price behavior	0.2652
3	TsCorr(Sub($close, $open), TsRank(Sub($low, TsMin($low, 5)), 5), 5)	Correlates the daily return with the time-series rank of the low price deviation from its 5-day minimum, evaluating if returns associate with the relative extremity of low prices.	0.0947
4	TsCorr(TsDelta($close, 1), TsMinDiff($low, 5), 5)	Correlates the one-day price change (close delta) with the difference between current low and its 5-day minimum, evaluating momentum and support level interactions.	0.0611

Table 9: (continued)

Factor ID	Expression	Description	ICIR
5	TsCorr(Sub($close, $open), Div(Sub($low, TsMin($low, 5)), TsStd($low, 5)), 5)	Correlates the daily return with the normalized deviation of low price from its 5-day minimum (scaled by volatility), assessing if returns relate to standardized low price extremes.	0.1179
6	TsCorr(TsPctChange($close, 5), TsMinDiff($low, 5), 5)	Correlates the one-day percentage change in closing price with the deviation of low from its 5-day minimum, linking momentum to support breakout signals.	0.1319
7	TsCorr(Sub($close, $open), TsRank(Div(Sub($low, TsMin($low, 5)), TsStd($low, 5)), 5), 5)	Correlates the daily return with the time-series rank of the normalized deviation of low price from its 5-day minimum, assessing if returns relate to the relative standing of standardized low price extremes over the past 5 days.	0.1614
8	TsCorr(TsDelta($close, 5), TsMaxDiff($low, 5), 5)	Correlates the one-day change in closing price with the deviation of low from its 5-day maximum, evaluating momentum against resistance rather than support levels.	0.1507
9	TsCorr(TsDelta($close, 5), TsDelta($low, 5), 5)	Correlates the one-day change in closing price with the one-day change in low price, capturing co-movement in interday price dynamics.	0.1998
10	TsCorr(TsPctChange($close, 5), TsPctChange($low, 5), 5)	Correlates the percentage change in closing price with the percentage change in low price, assessing relative sensitivity in interday price movements.	0.1632
11	TsCorr(TsDelta($close, 5), Sub(TsMax($high, 5), $close), 5)	Correlates the one-day change in closing price with the gap between the current close and the maximum high over a period, focusing on breakout potential from resistance.	0.2639
12	TsCorr(TsDelta($close, 5), TsDelta($vwap, 5), 5)	Correlates the one-day change in closing price with the one-day change in volume-weighted average price, assessing alignment between closing trends and average traded price movements.	0.1210
13	TsCorr(TsPctChange($close, 5), TsPctChange($vwap, 5), 5)	Correlates the percentage change in closing price with the percentage change in volume-weighted average price, evaluating price efficiency and market impact.	0.0719
14	TsCorr(TsDelta($close, 5), TsDelta(Sub($high, $low), 5), 5)	Correlates the one-day change in closing price with the one-day change in the daily range (high minus low), capturing how closing momentum relates to daily volatility changes.	0.2813

Table 9: (continued)

Factor ID	Expression	Description	ICIR
15	TsCorr(TsDelta($close, 5), TsDelta($volume, 5), 5)	Correlates the one-day change in closing price with the one-day change in trading volume, capturing price-volume dynamics in interday movements.	0.2639
16	TsCorr(TsDelta($close, 5), TsPctChange($low, 5), 5)	Correlates the one-day change in closing price with the percentage change in low price over d days, measuring momentum against relative low price shifts.	0.0935
17	TsCorr(TsPctChange($close, 5), Sub($high, $low), 5)	Correlates the percentage change in closing price with the daily range (high minus low), evaluating how closing trends relate to intraday volatility.	0.2746
18	TsCorr(Sub($close, Ref($close, 1)), Sub($vwap, Ref($vwap, 1)),5)	Correlates the absolute daily change in closing price with the absolute daily change in VWAP, capturing co-movement in raw price adjustments rather than percentage changes.	0.1252
19	Rank(TsCorr(TsDelta($close, 5), TsDelta($vwap, 5), 5))	Ranks the correlation between daily changes in close and VWAP cross-sectionally, identifying stocks with strongest alignment between closing trends and average traded prices.	0.1169
20	TsCorr(TsSkew($close, 5), TsKurt($vwap, 5), 5)	Correlates the rolling skewness of closing prices with the rolling kurtosis of VWAP, measuring the association between price distribution asymmetry and tail risk in market pricing.	0.1218
21	TsCorr(TsPctChange($close, 5), TsDelta($low, 5), 5)	Correlates the percentage change in closing price over d days with the one-day change in low price, evaluating interday momentum and low price dynamics.	0.1452
22	TsCorr(TsDelta($close, 5), TsPctChange($volume, 5), 5)	Correlates one-day change in closing price with percentage change in volume, assessing relationship between price movements and trading activity shifts over interday periods.	0.1760
23	TsCorr(TsPctChange($close, 5), TsPctChange($volume, 5), 5)	Correlates the percentage change in close with the percentage change in volume, evaluating the relationship between interday price momentum and trading activity intensity.	0.2681
24	Mul(TsCorr(TsDelta($close, 5), TsDelta($vwap, 5), 5), TsSkew(TsDelta($close, 5), 5))	Multiplies the correlation between daily changes in close and VWAP with the skewness of close price changes, capturing both alignment and asymmetry in interday price movements.	0.1015

Table 9: (continued)

Factor ID	Expression	Description	ICIR
25	Rank(TsCorr(TsPctChange($close, 5), TsPctChange($vwap, 5), 5))	Ranks the correlation between percentage changes in close and VWAP cross-sectionally, assessing alignment in relative price movements rather than absolute changes.	0.0974
26	Rank(TsCorr(TsPctChange($close, 5), TsPctChange($low, 5), 5))	Ranks the correlation between percentage changes in close and low prices cross-sectionally, evaluating alignment in downward price momentum movements.	0.1731
27	Rank(TsCorr(TsPctChange($close, 5), TsDelta($volume, 5), 5))	Ranks the correlation between percentage changes in close price and absolute changes in volume, assessing price-volume relationship in interday dynamics.	0.2765
28	Sub(Rank(TsCorr(TsPctChange($close, 5), TsPctChange($vwap, 5), 5)), Rank(TsCorr(TsPctChange($close, 5), TsDelta($volume, 5), 5)))	Subtracts the ranked correlation of price change with volume change from that with VWAP change, isolating VWAP-specific interday price dynamics.	0.2281

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state three contributions: DAG-based factor modeling, the Bayesian Retriever and DAG-aware Generator, and extensive experiments on Chinese and US stock markets against 8 baselines.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The conclusion discusses future directions and the robustness analysis in Section 5.5 and the Appendix explicitly examines the framework's sensitivity to backbone LLM, embedding model, prompt template, and temperature choices.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not claim formal theoretical results. The Bayesian formulation in Section 4.1 is a modeling choice with all components explicitly defined.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 5.1 specifies datasets, splits, baselines, hyperparameters, backbone LLM, and embedding model. The Appendix provides prompt templates, operator definitions, validation criteria, and baseline implementation details with open-source links.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code and data are available at the anonymous repository linked in the abstract. All datasets are publicly available through the Qlib platform.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5.1 provides data splits, pool capacity, expression length threshold, penalty hyperparameters, backbone LLM, and embedding model. The Appendix provides additional details on backtest settings and baseline configurations.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The robustness analysis in the Appendix reports mean and standard deviation over 3 runs across four LLM and embedding configurations, demonstrating statistical consistency.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

Justification: Section 5.1 specifies the backbone LLM (DeepSeek V3.1) and embedding model (Qwen 3 Embedding-4B). The Appendix reports API costs for all LLM-based methods.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses publicly available market data and open-source tools. No human subjects or private data are involved.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper contributes to transparent, interpretable factor discovery in quantitative finance. We acknowledge that automated trading systems may amplify market volatility if widely adopted without proper risk controls.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released code generates symbolic alpha factor expressions. It does not release pre-trained models or scraped datasets that pose misuse risks.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All baseline methods are cited and their open-source repositories are referenced with footnotes. The Qlib platform and all LLM APIs are properly cited.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The code repository includes documentation for reproducing experiments. The operator set and prompt templates are fully detailed in the Appendix.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing or human subjects research.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve human subjects research.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are a core component of the proposed method. Section 4.2 and 5.1 describe the use of DeepSeek V3.1 as the backbone LLM for factor generation, and the Appendix provides full prompt templates. We also utilize LLMs to correct grammatical errors.