AlphaPareto: Formulaic Alpha Discovery with LLM-Guided Multi-Objective Reinforcement Learning

Yingbo Zhao
School of Economics
Xiamen University
Fujian, China
15620241152783@stu.xmu.edu.cn

Zeyu Yang
Paula and Gregory Chow Institute for Studies in Economics
Xiamen University
Fujian, China
yangzeyu@stu.xmu.edu.cn

Zhoufan Zhu *
School of Economics & Wang Yanan Institute for Studies in Economics
Xiamen University
Fujian, China
tylerzzf@xmu.edu.cn

Abstract

Formulaic alpha discovery is a core challenge in quantitative trading, as identifying alphas that work well together remains difficult. Recent reinforcement learning (RL) methods formulate this task as a Markov decision process (MDP), but two important issues remain unresolved. First, as the alpha pool evolves, the reward function changes accordingly, making the MDP inherently non-stationary. Second, most existing methods optimize a single objective, typically predictive power, while ignoring other important properties of a high-quality alpha pool. Motivated by these challenges, we propose AlphaPareto, an RL method for formulaic alpha discovery. To address non-stationarity, AlphaPareto augments the state to include both the alpha under construction and the current alpha pool, and applies a large language model (LLM) to encode the pool. This design allows the agent to adapt to the evolving search environment. To overcome the limitation of single-objective reward design, AlphaPareto replaces the scalar reward with a multi-objective vector-valued reward that simultaneously captures predictive power, temporal stability, perturbation robustness, and diversity, and optimizes these objectives through a Pareto-regularized learning procedure. Empirical applications to real-world datasets show that our AlphaPareto method outperforms its competitors.

*Correspondence to: Zhoufan Zhu <tylerzzf@xmu.edu.cn>.

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

1 Introduction

In quantitative investing, predictive signals derived from historical market data, known as alphas (or alpha factors), form the foundation of automated trading strategies. Among the various forms of alphas, *formulaic alphas*, which transform stock features (e.g., historical prices and volume) into signals via concise mathematical formulas, are particularly valued for their interpretability, auditability, and ease of deployment. Since a single alpha is typically unstable across dynamic markets, the final trading signal is usually a *mega-alpha*, which combines a set of formulaic alphas (i.e., an *alpha pool*) through an interpretable model (say, a linear model). Intuitively, the quality of this mega-alpha depends critically on whether the included alphas are *synergistic*, i.e., whether the alpha pool captures diverse and complementary aspects of market dynamics.

AlphaGen (Yu et al., 2023) introduces a powerful paradigm for formulaic alpha discovery, which formulates the discovery process as a Markov decision process (MDP) and applies reinforcement learning (RL) to generate a synergistic alpha pool. This RL-based approach has been further advanced by Zhu & Zhu (2025), Shi et al. (2025), Chen et al. (2026), among others, yet two important issues remain unresolved, limiting the efficiency and stability of the discovery process in practice.

The first issue is *non-stationarity* (Lecarpentier & Rachelson, 2019): the reward function of the alpha discovery MDP changes across episodes. This stems from a structural misalignment between state and reward. The reward measures the contribution of the generating alpha to the current alpha pool, and therefore depends on the pool's evolving composition. However, the state encodes only the generating alpha, and no information about the pool is included. As stronger alphas enter the pool and replace weaker ones, the same state-action trajectories can yield different rewards at different stages of alpha discovery. Most existing methods simply ignore this non-stationarity (Yu et al., 2023; Ren et al., 2024; Shi et al., 2025; Chen et al., 2026), except Zhu & Zhu (2025). So far, none of the existing methods inform the RL agent about what the current alpha pool already contains or what complementary signals it should seek next; the agent must infer this indirectly from rewards alone.

The second issue is *single-objective reward design*. All existing RL-based methods optimize a single scalar reward, typically the Information Coefficient (IC) of the mega-alpha formed by the alpha pool. While IC measures average predictive power, the quality of an alpha pool is inherently multi-dimensional. For example, a pool with high IC may still exhibit large temporal fluctuation or prove fragile under market regime shifts, leading to poor out-of-sample performance. As noted by Ding et al. (2025), other criteria, including temporal stability, perturbation robustness, and diversity, are also important for a reliable alpha pool. However, no existing method incorporates these criteria into the alpha discovery process.

To address both issues, we propose **AlphaPareto**, an RL method that integrates alpha pool semantic reasoning and Pareto-regularized multi-objective RL (MORL). Specifically, to handle non-stationarity, we redefine the state to include both the generating alpha and the evolving alpha pool. Under this augmented state design, the original non-stationary MDP becomes a stationary one. However, the alpha pool consists of complex mathematical expressions that are difficult to understand directly. This motivates us to use a large language model (LLM), together with a structured prompt, to encode the pool's semantic content. To address the single-objective reward issue, we define a multi-dimensional reward function that jointly characterizes the quality of alpha pool along four axes: predictive power (IC of the mega-alpha), temporal stability (consistency of mega-alpha over time), perturbation robustness (robustness of mega-alpha under market perturbations), and diversity (complementarity relative to existing pool). Rather than collapsing these rewards into a single scalar via manual weighting, we employ a Pareto-regularized learning method to generate alphas along the efficient frontier, producing principled trade-offs among all objectives.

In summary, our main contributions are three-fold:

1. Conceptually, we reformulate the non-stationary alpha discovery MDP with a single-objective reward into a context-conditional stationary MDP with a multi-objective reward, better aligning the framework with practical challenges.

2. Methodologically, we propose AlphaPareto, which uses an LLM to encode the evolving alpha pool as a part of state, thereby stationarizing the alpha discovery MDP, and introduces a Pareto-regularized learning procedure to obtain preference-conditioned optimal policies.

3. Empirically, we conduct extensive experiments on large-scale real-world datasets from Chinese stock markets, demonstrating that AlphaPareto consistently outperforms state-of-the-art baselines. Ablation studies further confirm the complementary contributions of each proposed component.

2 Background and Related Work

GP-based alpha discovery. Over the past decade, genetic programming (GP) has been the dominant approach to formulaic alpha discovery (Lin et al., 2019a,b; Zhang et al., 2020; Cui et al., 2021). It evolves expression trees using IC as the fitness measure. However, GP scales poorly as the number of operators and features grows. More importantly, it discovers alphas independently and does not directly optimize for synergy within the alpha pool, leading to poor practical performance.

RL-based alpha discovery. AlphaGen (Yu et al., 2023) is the first to model formulaic alpha discovery as an MDP and to use MaskPPO to train an agent that generates a set of synergistic formulaic alphas. This marks an important step beyond GP, since it makes the generator aware of the performance of alpha pool via reward. Zhu & Zhu (2025) point out the non-stationarity in this MDP and propose AlphaQCM, a distributional RL method that uses quantiled conditional variance as an exploration bonus to reduce the impact of non-stationarity. AlphaForge (Shi et al., 2025) improves the combination model by incorporating alpha's temporal performance into selection and dynamically adjusting weight for each alpha. Despite these advances, all existing RL-based methods share the aforementioned two limitations: (i) none analyze alpha pool to handle non-stationarity; (ii) all optimize a single scalar reward, and therefore ignore the inherently multi-dimensional nature of alpha pool quality.

LLM-based alpha discovery. The emergence of LLMs has opened a new direction for alpha discovery (Wang et al., 2025a; Yuan et al., 2024; Tang et al., 2025; Li et al., 2025; Luo et al., 2026). Most of these methods use LLMs as *factor generators* that directly produce formulaic alpha expressions. For example, AlphaAgent (Tang et al., 2025) uses LLMs to propose market hypotheses and construct factors with regularization on originality and complexity. R&D-Agent-Quant (Li et al., 2025) separates this process into a Research stage, forming hypotheses from domain priors, and a Development stage, using code-generation agents for backtesting. However, Luo et al. (2026) highlight that the robustness and search efficiency of LLM-based alpha discovery remain significant challenges, warranting further investigation. Rather than using the LLM as a direct alpha generator, AlphaPareto uses it as a semantic encoder of the alpha pool within an RL loop.

3 Problem Setup

We consider a market with N stocks observed over S trading days. Our ultimate goal is to discover an alpha pool $\mathcal{F} = \{f^1, \ldots, f^P\}$ with best out-of-sample predictive power, containing P formulaic alphas. Each alpha f^p maps historical market data $\boldsymbol{H}_{s-1}$ to a cross-sectional signal $\boldsymbol{\alpha}_s^p = f^p(\boldsymbol{H}_{s-1}) \in \mathbb{R}^N$. These signals are then combined into a mega-alpha via a linear model:

$$\widehat{\boldsymbol{\alpha}}_s(\mathcal{F}) = \sum_{p=1}^{P} \widetilde{\boldsymbol{\alpha}}_s^p \, \widehat{\beta}^{\,p} \in \mathbb{R}^N, \tag{1}$$

where $\widetilde{\boldsymbol{\alpha}}_s^p = \big(\boldsymbol{\alpha}_s^p - \mathrm{Mean}(\boldsymbol{\alpha}_s^p)\big) / \mathrm{Std}(\boldsymbol{\alpha}_s^p)$ is the cross-sectionally standardized alpha signal, and $\widehat{\boldsymbol{\beta}} = (\widehat{\beta}^{\,1}, \ldots, \widehat{\beta}^{\,P})^\top$ is estimated by minimizing $\sum_{s \in \mathcal{D}_{\mathrm{train}}} \|\boldsymbol{y}_s - \widehat{\boldsymbol{\alpha}}_s(\mathcal{F})\|^2$, with $\mathcal{D}_{\mathrm{train}}$ being the training set and $\boldsymbol{y}_s \in \mathbb{R}^N$ denoting future stock returns.

Next, following prior work, each formulaic alpha f is represented in reverse Polish notation (RPN), so that its mathematical expression is encoded as a token sequence z, consisting of operators and features.[2] Since each z is generated through a series of token selections, the discovery of f can be formulated as a sequential decision-making problem. This process is modeled as an MDP $(\mathcal{X}, \mathcal{A}, \mathcal{P}, \gamma, \mathcal{R})$, which is elaborated below.

[2]See Appendix A for more details about RPN and available tokens.

Conventionally, each state $x_t \in \mathcal{X}$ corresponds to a RPN token sequence z_t, with initial state $x_0 = z_0 \equiv$ *BEG*, where *BEG* is a special token indicating the beginning of a formula. Each action $a_t \in \mathcal{A}$ is a token selected by the RL agent. To ensure that z_t is a valid RPN, only a subset of actions is allowed at each state, enforced through invalid action masking (Yu et al., 2023).

The transition kernel $\mathcal{P}$ is deterministic: given x_t and a_t, the next state x_{t+1} is

$$x_{t+1} = \mathcal{P}(x_t, a_t) = z_{t+1} = [z_t, a_t].$$

An episode terminates when $a_t =$ *SEP* (a special token) or $|x_{t+1}| > L_{\max}$, where $L_{\max}$ is the maximum sequence length used to preserve interpretability. The discount factor is set to $\gamma = 1$, since each episode is finite in this MDP.

Let f_t denote the parsed formulaic alpha corresponding to the next token sequence z_{t+1}. The reward r_t measures the contribution of f_t to the current alpha pool $\mathcal{F}_t$. If f_t is incomplete (or invalid), $r_t = 0$ (or -1); otherwise, (i) f_t is temporarily added to $\mathcal{F}_t$, forming an extended pool $\overline{\mathcal{F}}_t$, (ii) a linear model is fitted on $\overline{\mathcal{F}}_t$ and the alpha pool is updated to $\mathcal{F}_{t+1}$ by keeping at most P alphas based on their fitted contributions, and (iii) the reward is defined as its in-sample IC:

$$\begin{aligned} r_t &= \mathrm{IC}(\{\widehat{\boldsymbol{\alpha}}_s(\mathcal{F}_{t+1}) : s \in \mathcal{D}_{\text{train}}\}) = \mathrm{Mean}(\mathrm{Corr}(\widehat{\boldsymbol{\alpha}}_s(\mathcal{F}_{t+1}), \boldsymbol{y}_s)) \\ &= \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{s \in \mathcal{D}_{\text{train}}} \frac{\sum_{i=1}^{N} \left[\widehat{\alpha}_{i,s}(\mathcal{F}_{t+1}) - \bar{\alpha}_s(\mathcal{F}_{t+1})\right] (y_{i,s} - \bar{y}_s)}{\sqrt{\sum_{i=1}^{N} [\widehat{\alpha}_{i,s}(\mathcal{F}_{t+1}) - \bar{\alpha}_s(\mathcal{F}_{t+1})]^2 \sum_{i=1}^{N} (y_{i,s} - \bar{y}_s)^2}}, \end{aligned} \tag{2}$$

where $\widehat{\boldsymbol{\alpha}}_s(\mathcal{F}_{t+1})$ is the mega-alpha defined in (1), $\boldsymbol{y}_s \in \mathbb{R}^N$ is the vector of future returns, $\widehat{\alpha}_{i,s}$ and $y_{i,s}$ are the i-th elements of $\widehat{\boldsymbol{\alpha}}_s(\mathcal{F}_{t+1})$ and $\boldsymbol{y}_s$, respectively. Here, $\bar{\alpha}_s = \frac{1}{N}\sum_{i=1}^{N} \widehat{\alpha}_{i,s}$ and $\bar{y}_s = \frac{1}{N}\sum_{i=1}^{N} y_{i,s}$ are the cross-sectional averages of $\widehat{\boldsymbol{\alpha}}_s(\mathcal{F}_{t+1})$ and $\boldsymbol{y}_s$. Clearly, the IC measures the average cross-sectional correlation between mega-alphas and target returns over all $s \in \mathcal{D}_{\text{train}}$. Thus, r_t reflects the in-sample predictive power of the updated alpha pool $\mathcal{F}_{t+1}$ in terms of correlation.

Non-stationarity. Under this classical specification of MDP, the reward can be re-written as

$$r_t = \mathcal{R}(\mathcal{F}_{t+1}) = \mathcal{R}(f_t, \mathcal{F}_t) = \mathcal{R}(\phi(z_{t+1}), \mathcal{F}_t) = \mathcal{R}(\phi(x_{t+1}), \mathcal{F}_t) = \mathcal{R}(\phi(\mathcal{P}(x_t, a_t)), \mathcal{F}_t),$$

where $\phi(\cdot)$ maps a token sequence to its formula. Hence, the reward function $\mathcal{R}$ depends not only on the state-action pair (x_t, a_t), but also on the current alpha pool $\mathcal{F}_t$. Since $\mathcal{F}_t$ evolves across episodes as new alphas are added and old ones are removed, the same state-action trajectory may receive different rewards at different stages of training. This makes the discovery process inherently non-stationary. A natural idea is to include $\mathcal{F}_t$ in the state so that the agent can also condition on the current pool $\mathcal{F}_t$. In practice, however, this is difficult because the alpha pool is a set of complex mathematical expressions whose shared structure and complementary information are hard to summarize, even for domain experts.

Single-objective reward design. The reward in (2) depends only on IC, a single scalar measuring average in-sample predictive power. However, as aforementioned, the quality of an alpha pool is inherently multi-dimensional. A pool with high IC may still exhibit large temporal variation or be fragile under market regime shifts, leading to poor out-of-sample predictive performance. To improve generalization, other criteria, such as temporal stability, perturbation robustness, and diversity, should also be incorporated into the reward of this MDP.

4 Methodology

In this section, we present the AlphaPareto method. Specifically, we first propose an alpha-pool semantic reasoning technique to augment the state representation and overcome non-stationarity. We then design a multi-objective vector-valued reward that simultaneously accounts for predictive power, temporal stability, perturbation robustness, and diversity, and introduce an optimization method to handle these competing objectives. An overview of AlphaPareto is shown in Figure 1.

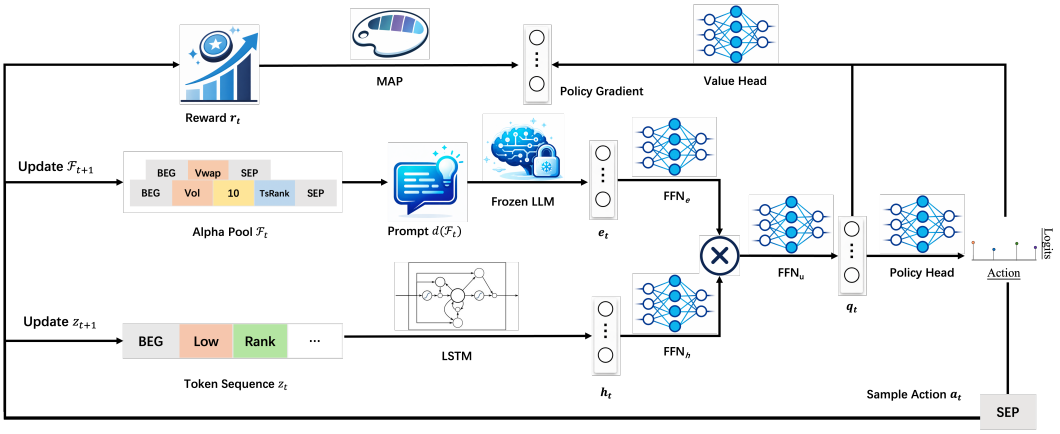


Figure 1: Overview of the AlphaPareto framework.

4.1 Alpha Pool Semantic Reasoning

A natural solution for addressing non-stationarity is to augment the state x_t with the current alpha pool $\mathcal{F}_t$ and redefine it as

$$x_t = (z_t, \mathcal{F}_t).$$

However, directly encoding $\mathcal{F}_t$ is difficult in practice, because it is a set of mathematical expressions whose synergy, redundancy, and missing aspects of market dynamics are hard to quantify explicitly.

To overcome this difficulty, AlphaPareto uses a frozen LLM as a semantic encoder of $\mathcal{F}_t$. Specifically, we construct a structured prompt $d(\mathcal{F}_t)$ from the current alpha pool to summarize the mathematical expression and assigned weight of each alpha. This prompt is designed to help the LLM capture the pool's composition, concentration, and potential blind spots.[3] Given $d(\mathcal{F}_t)$, we extract the last-layer hidden representation of the LLM:

$$\boldsymbol{e}_t = \text{LLM}(d(\mathcal{F}_t)) \in \mathbb{R}^{d_e}, \tag{3}$$

where $\text{LLM}(\cdot)$ and d_e denote the forward mapping and hidden dimension of the frozen LLM, respectively. As a result, $\boldsymbol{e}_t$ is expected to provide a compact semantic summary of $\mathcal{F}_t$, indicating which signals are already captured by the current pool and which complementary signals may still be missing. During RL training, the LLM remains frozen and no gradient is back-propagated through it.

Meanwhile, following Yu et al. (2023), the token sequence z_t is encoded by an LSTM:

$$\boldsymbol{h}_t = \text{LSTM}(z_t) \in \mathbb{R}^{d_h}, \tag{4}$$

where d_h is the hidden dimension. We then project both $\boldsymbol{h}_t$ and $\boldsymbol{e}_t$ into a shared latent space through two feed-forward networks (FFNs), and combine them through element-wise product followed by another FFN:

$$\boldsymbol{q}_t = \text{FFN}_u(\boldsymbol{u}_t) = \text{FFN}_u\Big(\text{FFN}_h(\boldsymbol{h}_t) \odot \text{FFN}_e(\boldsymbol{e}_t)\Big) \in \mathbb{R}^d. \tag{5}$$

Finally, the resulting representation $\boldsymbol{q}_t$ is fed into the policy and value heads of MaskPPO, as in Yu et al. (2023), to produce action probabilities and value estimates. As a result, the policy $\pi(a_t \mid x_t) = \pi(a_t \mid z_t, \mathcal{F}_t)$ is conditioned on both the token sequence and the alpha pool, which helps resolve the non-stationarity caused by state–reward misalignment. See Figure 1 for an illustration.

4.2 Pareto-Regularized MORL

4.2.1 Multi-objective Reward

To overcome the limitation of single-objective design, AlphaPareto replaces the scalar reward in (2) with a vector-valued reward and adopts the Multi-Human-Value Alignment Palette (MAP; Wang et al., 2025b) framework to optimize along the Pareto efficient frontier of competing objectives.

[3]The full prompt template is provided in Appendix B.

Specifically, we measure the four-dimensional contribution of f_t to the current alpha pool $\mathcal{F}_t$: predictive power, temporal stability, perturbation robustness, and diversity. The predictive-power component is defined as

$$\mathrm{IC}_t = \mathrm{IC}(\{\widehat{\boldsymbol{\alpha}}_s(\mathcal{F}_{t+1}) : s \in \mathcal{D}_{\mathrm{train}}\}),$$

which is the same as the reward used in the classical MDP, defined in (2).

Then, temporal stability measures the consistency of the cross-sectional rankings produced by the mega-alpha over time. Its reward form is defined as

$$\mathrm{RRE}_t = \mathrm{RRE}(\{\widehat{\boldsymbol{\alpha}}_s(\mathcal{F}_{t+1}) : s \in \mathcal{D}_{\mathrm{train}}\}) = \frac{1}{|\mathcal{D}_{\mathrm{train}}|}\left(1 + \sum_{s=2}^{|\mathcal{D}_{\mathrm{train}}|} \frac{1}{1 + D_{\mathrm{KL}}(\boldsymbol{p}_s \,\|\, \boldsymbol{p}_{s-1})}\right),$$

where $\mathrm{RRE}(\cdot)$ denotes the Relative Rank Entropy (RRE), $D_{\mathrm{KL}}(\cdot \,\|\, \cdot)$ denotes the Kullback–Leibler (KL) divergence, and $\boldsymbol{p}_s = (p_{1,s}, \ldots, p_{N,s})^\top$ is the cross-sectional ranking vector at time s, with

$$p_{i,s} = \frac{\mathrm{Rank}(\widehat{\alpha}_{i,s}(\mathcal{F}_{t+1}))}{\sum_{j=1}^{N} \mathrm{Rank}(\widehat{\alpha}_{j,s}(\mathcal{F}_{t+1}))}.$$

Clearly, a higher RRE_t indicates more stable stock rankings over time and therefore better temporal stability of the updated alpha pool $\mathcal{F}_{t+1}$.

Next, perturbation robustness quantifies how robust the mega-alpha remains under perturbations. We define its reward component as

$$\mathrm{PFS}_t = \mathrm{PFS}(\{\widehat{\boldsymbol{\alpha}}_s(\mathcal{F}_{t+1}) : s \in \mathcal{D}_{\mathrm{train}}\}) = \frac{1}{|\mathcal{D}_{\mathrm{train}}|} \sum_{s \in \mathcal{D}_{\mathrm{train}}} \rho\Big(\widehat{\boldsymbol{\alpha}}_s(\mathcal{F}_{t+1}),\ \widehat{\boldsymbol{\alpha}}^*_s(\mathcal{F}_{t+1})\Big),$$

where $\mathrm{PFS}(\cdot)$ denotes the perturbation fidelity score (PFS), $\widehat{\boldsymbol{\alpha}}^*_s(\mathcal{F}_{t+1}) = (\mathbf{1} + \boldsymbol{\varepsilon}_s) \odot \widehat{\boldsymbol{\alpha}}_s(\mathcal{F}_{t+1})$ is the perturbed mega-alpha at time s, and $\rho(\cdot, \cdot)$ denotes the Spearman rank correlation. Here, the perturbation noise $\boldsymbol{\varepsilon}_s$ is independently sampled from a predefined distribution $\mathcal{E}$, where $\mathcal{E}$ is chosen to be either a Gaussian distribution or a Student-t_3 distribution with equal probability. Particularly, the Gaussian distribution is used to simulate random market microstructure fluctuations, while the heavy-tailed t_3 distribution captures structural shocks such as policy changes. These two distributions are rescaled so that their variances match the empirical cross-sectional variance of stock returns. Therefore, a higher PFS_t indicates that the alpha pool is more robust to disturbances and thus more likely to generalize well under changing market conditions.[4]

Lastly, diversity captures whether the alpha pool contains varied and complementary signals. We quantify the reward component of diversity through the diversity entropy (DH):

$$\mathrm{DH}_t = \mathrm{DH}(\{\widehat{\boldsymbol{\alpha}}_s(\mathcal{F}_{t+1}) : s \in \mathcal{D}_{\mathrm{train}}\}) = -\frac{\sum_{i=1}^{|\mathcal{F}_{t+1}|} \tilde{p}_i \log \tilde{p}_i}{\log |\mathcal{F}_{t+1}|},$$

where $\tilde{p}_i = \lambda_i / \sum_{j=1}^{|\mathcal{F}_{t+1}|} \lambda_j$, and $\lambda_1, \ldots, \lambda_{|\mathcal{F}_{t+1}|}$ are the eigenvalues of the covariance matrix $\boldsymbol{C} \in \mathbb{R}^{|\mathcal{F}_{t+1}| \times |\mathcal{F}_{t+1}|}$ computed from the individual alpha signals $\{\widehat{\boldsymbol{\alpha}}^p_s : p = 1, \ldots, |\mathcal{F}_{t+1}|, s \in \mathcal{D}_{\mathrm{train}}\}$. Concretely, a higher DH_t indicates more evenly distributed variance across orthogonal directions, and thus $\mathcal{F}_{t+1}$ contains more complementary information and less redundancy among its constituents.

Combining all these components, we obtain the vector-valued reward for multi-objective learning:

$$\boldsymbol{r}_t = (\mathrm{IC}_t, \mathrm{RRE}_t, \mathrm{PFS}_t, \mathrm{DH}_t)^\top. \tag{6}$$

Conventionally, for incomplete (or invalid) formulas, we set $\boldsymbol{r}_t = \mathbf{0}$ (or $-\mathbf{1}$) and evaluate the above quantities only when a valid formula is terminated.

[4] Ding et al. (2025) define PFS by perturbing the stock features $\boldsymbol{H}_{s-1}$ and then taking the minimum over the two perturbation types, which makes the resulting measure more fragile and indirect.

4.2.2 MAP-based Optimization

Realistically, the four objectives in (6) may conflict. For example, improving predictive power may reduce diversity, since focusing on a narrow class of alphas can work better in a given market condition. It is therefore essential to consider the Pareto frontier of this multi-objective optimization. The MAP framework shows that a linear combination of individual reward functions is sufficient to reach the Pareto frontier under mild assumptions. Motivated by this finding, we collapse the competing rewards into a single scalar with data-driven weights, under some realistic constraints.

Specifically, let $\boldsymbol{c} = (c_{\mathrm{IC}}, c_{\mathrm{RRE}}, c_{\mathrm{PFS}}, c_{\mathrm{DH}})^\top$ denote a vector of preferred levels, where each entry specifies the desired minimum level for one objective. We refer to $\boldsymbol{c}$ as the *value palette*. By Theorem 2 of Wang et al. (2025b), the Pareto-regularized optimization under vector-valued return $\boldsymbol{r}_t$ is equivalent to optimizing the linear-scalarized reward:

$$r_t^* = \boldsymbol{\lambda}(\boldsymbol{c})^\top \boldsymbol{r}_t, \tag{7}$$

where the weight vector $\boldsymbol{\lambda}(\boldsymbol{c}) \in \mathbb{R}^4$ is the unique maximizer of the dual objective:

$$\operatorname*{argmax}_{\boldsymbol{\lambda} \geq \mathbf{0}} g(\boldsymbol{\lambda}; \boldsymbol{c}) = \operatorname*{argmax}_{\boldsymbol{\lambda} \geq \mathbf{0}} -\log \mathbb{E}_\pi\big[\exp\big(\boldsymbol{\lambda}^\top \boldsymbol{r}_t\big)\big] + \boldsymbol{\lambda}^\top \boldsymbol{c}. \tag{8}$$

Clearly, $\boldsymbol{\lambda}(\boldsymbol{c})$ depends on both the multi-objective reward $\boldsymbol{r}_t$ under policy π and the chosen value palette $\boldsymbol{c}$. In this sense, $\boldsymbol{c}$ acts as a regularizer that encodes the user's preferences over the multiple objectives, while $\boldsymbol{\lambda}(\boldsymbol{c})$ adaptively balances them according to the current dynamics. This formulation requires only a minimum target level for each objective in $\boldsymbol{c}$, rather than fully subjective scalarization weights, and thus provides a principled way to trade off these competing objectives.

Moreover, MAP is originally developed for an offline setting with fixed transitions. We slightly extend it to our online alpha-discovery setting by recomputing the palette-dependent multiplier $\boldsymbol{\lambda}(\boldsymbol{c})$ in the online manner. Particularly, given T sampled transitions $\{(x_t, a_t, \boldsymbol{r}_t, x_{t+1})\}$ from the replay buffer, we solve the empirical dual

$$\widehat{\boldsymbol{\lambda}}(\boldsymbol{c}) = \operatorname*{argmax}_{\boldsymbol{\lambda} \geq \mathbf{0}} \left\{ -\log \frac{1}{T} \sum_{t=1}^{T} \exp\big(\boldsymbol{\lambda}^\top \boldsymbol{r}_t\big) + \boldsymbol{\lambda}^\top \boldsymbol{c} \right\}. \tag{9}$$

We then use the online scalarized reward $r_t^*(\boldsymbol{c}) = \widehat{\boldsymbol{\lambda}}(\boldsymbol{c})^\top \boldsymbol{r}_t$ inside MaskPPO and update the policy and value networks in the standard way.

Overall, the complete AlphaPareto algorithm is summarized in Algorithm 1.

5 Experiments

Conventionally, we conduct experiments on Chinese A-share stock market data, with the prediction target defined as the 20-day future stock return. To examine how market complexity affects alpha discovery, we consider three stock universes of increasing difficulty: (1) the largest 300 stocks (CSI300), (2) the largest 800 stocks (CSI800), and (3) the full market universe (Market). As the stock universe expands, discovering a synergistic alpha pool becomes more challenging because the underlying market dynamics become more complex. Each dataset is split chronologically into a training period (2013/07/01–2023/06/30), a validation period (2023/07/01–2024/06/30), and a test period (2024/07/01–2025/06/30).

We compare AlphaPareto with the following four categories of baselines:

(1) **Alpha158** and **MLP** (handcrafted formulaic and end-to-end non-formulaic baselines): **Alpha158** fixes the alpha pool to the 158 predefined formulaic alphas in Qlib (Yang et al., 2020) and combines them with a linear model to form a mega-alpha, whereas **MLP** replaces the linear model with a multi-layer perceptron network.

(2) **GP** (GP-based formulaic alpha discovery): **GP** uses genetic programming to generate candidate alphas and builds the mega-alpha from the top-P alphas.

(3) **AlphaAgent** and **R&D-Agent-Quant** (LLM-based formulaic alpha discovery): These methods use LLMs as alpha generators to propose candidate factors or formulas, from which the final alpha pool is constructed.

(4) **AlphaGen**, **AlphaForge** and **AlphaQCM** (RL-based formulaic alpha discovery): These methods use RL to search for a synergistic alpha pool and then fit a linear mega-alpha based on the discovered formulas.

Each stochastic experiment is repeated with 5 random seeds, and the alpha-pool size P is tuned on the validation set. Since LLM-based alpha discovery methods heavily rely on the underlying LLM's capability, AlphaAgent and R&D-Agent-Quant use the largest DeepSeek-V3.2 (671B) model as the alpha generator. In contrast, AlphaPareto uses the LLM as a semantic encoder to obtain the hidden state, and we choose the Qwen-Embedding-4B model for this purpose. More details on the baseline methods are provided in Appendix D, and hyperparameters for AlphaPareto are listed in Appendix E. Additional experiments on the impact of MAP, random-noise control, and cross-market generalization are provided in the Appendices H–J. The source code for reproducing the experiments is available at `https://github.com/BiQiBaoWinner/AlphaPareto`.

5.1 Main Comparison

We first assess how different alpha discovery methods affect out-of-sample predictive performance, since predictive power is the most important criterion in practice. Table 1 reports the out-of-sample ICs on the CSI300, CSI800, and Market datasets. From this table, AlphaPareto achieves the highest IC on all three datasets, reaching 3.92% on CSI300, 5.70% on CSI800, and 10.10% on Market, outperforming all competing methods: 3.69%, 5.07% and 9.16% earned by the second best method. The paired t-tests reported in Appendix F provide further statistical support for the advantages of AlphaPareto.

The gain of AlphaPareto over the strongest RL-based baseline is moderate on CSI300 and CSI800 (0.23% and 0.63%), but becomes much larger on the Market dataset (0.94%). This pattern suggests that conditioning on the evolving alpha pool and optimizing multiple objectives may be especially useful in complex markets, where discovering synergistic alphas is more difficult.

Meanwhile, the LLM-based baselines do not perform well in this setting. A possible reason is that out-of-sample market conditions can differ substantially from the in-sample period, making it difficult for direct LLM generation to produce stable and effective alphas. In contrast, AlphaPareto uses the LLM only to encode the alpha pool and support decision making within the RL loop.

Table 1: Out-of-sample ICs of different methods across different datasets.

	CSI300		CSI800		Market	
Method	Mean	Std	Mean	Std	Mean	Std
Alpha158	2.99%	-	4.77%	-	4.04%	-
MLP	1.51%	(2.33%)	4.63%	(0.90%)	7.00%	(0.18%)
GP	1.09%	(0.82%)	2.74%	(1.40%)	7.52%	(1.53%)
AlphaAgent	0.51%	(0.34%)	0.48%	(0.46%)	1.10%	(0.59%)
R&D-Agent-Quant	2.39%	(0.96%)	2.45%	(0.73%)	1.48%	(0.91%)
AlphaGen	3.69%	(0.66%)	5.07%	(0.63%)	8.44%	(1.09%)
AlphaQCM	0.50%	(0.95%)	4.79%	(0.34%)	9.16%	(4.22%)
AlphaForge	2.03%	(0.65%)	4.17%	(1.96%)	5.37%	(1.81%)
AlphaPareto	**3.92%**	**(0.46%)**	**5.70%**	**(0.87%)**	**10.10%**	**(1.01%)**

Beyond predictive performance, Figure 2 compares the out-of-sample results in terms of temporal stability, perturbation robustness, and diversity. The figure shows that, together with having the highest out-of-sample predictive power, AlphaPareto also performs strongly across these three dimensions. In particular, on CSI300 and CSI800, it achieves the highest perturbation robustness (PFS) and the second-highest diversity (DH), while maintaining competitive temporal stability (ranking third or fourth in RRE). On the Market dataset, AlphaPareto still attains the highest PFS, though its DH and RRE are relatively moderate. Overall, these results further support the design of AlphaPareto: by employing multi-objective optimization, it discovers alpha pools that are not only more predictive, but also more stable, robust, and diverse.

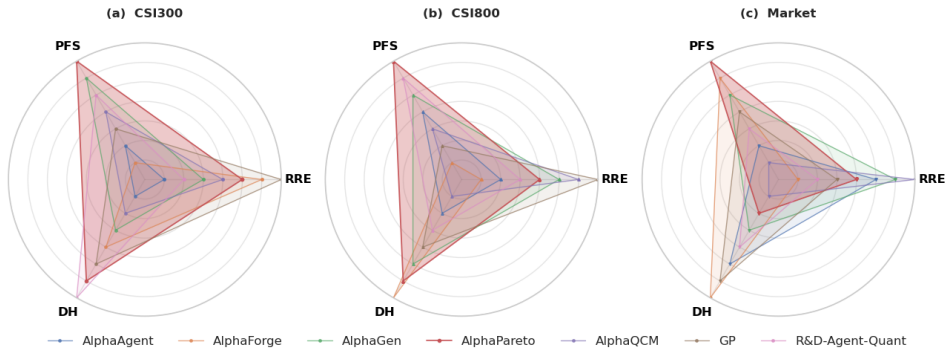


Figure 2: Radar chart comparison in terms of temporal stability, perturbation robustness, and diversity.

We further evaluate the out-of-sample economic performance of AlphaPareto through portfolio backtesting.[5]Table 2 summarizes the primary portfolio results. From this table, AlphaPareto achieves the highest annualized return, information ratio, and Sharpe ratio among all evaluated methods, with an annualized return of 58.92%, an information ratio of 2.62, and a Sharpe ratio of 2.59. Its maximum drawdown is -18.12%, which is slightly worse than those of GP and AlphaGen but better than remaining methods. Overall, AlphaPareto provides the strongest portfolio-level performance, suggesting that its improved alpha discovery translates into more effective portfolio construction.

Table 2: Out-of-sample monthly portfolio performance. AV denotes the annualized return, MDD denotes the maximum drawdown, IR denotes the information ratio, and SR denotes the Sharpe ratio.

Model	AV (↑)	MDD (↑)	IR (↑)	SR (↑)
Alpha158	33.17%	-18.71%	1.08	1.34
MLP	47.09%	-25.21%	1.57	1.58
GP	24.49%	**-13.29%**	0.77	1.18
AlphaGen	40.03%	-15.91%	1.65	1.61
AlphaAgent	35.07%	-29.94%	0.71	0.89
R&D-Agent-Quant	37.30%	-30.18%	0.76	0.96
AlphaQCM	19.45%	-19.74%	0.20	1.21
AlphaPareto	**58.92%**	-18.12%	**2.62**	**2.59**

5.2 Scaling Performance

The choice of LLM within AlphaPareto tends to affect its performance. To examine this effect, we vary the parameter scale of the frozen LLM while keeping the rest of the framework unchanged. We use the DeepSeek-R1 model family to provide LLMs of different parameter scales. Table 3 reports the out-of-sample ICs of AlphaPareto with different LLM scales. The results show that AlphaPareto presents strong performance over a wide range of scales, but the pattern is clearly non-monotonic. Notably, the Qwen-Embedding-4B model achieves the best ICs on all datasets.

This finding suggests that the role of the LLM in AlphaPareto differs from that in direct generation-based methods. Here, the LLM serves only as a frozen semantic encoder of the alpha pool rather than a generator of new alphas. In this setting, a small or medium-sized model may already provide sufficient semantic information, whereas excessively large models may introduce unnecessary domain knowledge or noise into the representation of alpha pool. From a practical perspective, these scaling results are encouraging, as they make AlphaPareto more cost-effective for real-world deployment.

[5]Details of the trading strategy and additional portfolio results are provided in Appendix G.

Table 3: Out-of-sample ICs of AlphaPareto across different LLM sizes.

Model	CSI300		CSI800		Market	
	Mean	Std	Mean	Std	Mean	Std
DeepSeek-1.5B	2.65%	(0.99%)	5.39%	(0.44%)	9.33%	(0.92%)
Qwen-Embedding-4B	**3.92%**	**(0.46%)**	**5.70%**	**(0.87%)**	**10.10%**	**(1.01%)**
DeepSeek-7B	3.38%	(0.56%)	5.50%	(1.23%)	9.50%	(0.56%)
DeepSeek-14B	3.71%	(1.73%)	5.59%	(0.46%)	9.74%	(0.64%)
DeepSeek-32B	3.26%	(1.43%)	4.76%	(0.52%)	9.88%	(1.13%)

5.3 Ablation Study

To examine the contribution of the two key components in AlphaPareto, we conduct an ablation study with four variants: the baseline without either component (i.e., AlphaGen), the model with only LLM-based alpha pool semantic reasoning, the model with only MAP-based MORL, and the full AlphaPareto method with both components. Table 4 reports the out-of-sample IC performance in this ablation study.

The results show that both components are useful, but they contribute in different ways. The LLM-based alpha pool semantic reasoning mainly improves performance on the Market dataset, where the underlying financial system is more complex. In contrast, the MAP-based MORL component improves performance on all three datasets, highlighting the value of improving the optimization target beyond predictive power alone. When the two components are combined, the resulting AlphaPareto method achieves the best performance on CSI800 and Market, while remaining very close to the best mean result on CSI300 and exhibiting substantially smaller variation (only a 0.07% loss in mean, but a 0.81% reduction in standard deviation). This slight decline in IC on CSI300 may arise from additional domain noise introduced by the LLM. The general market knowledge from the LLM can be useful for a broad stock universe, but may be less informative for the large, relatively efficiently priced stocks in CSI300. Consistent with this hypothesis, using the LLM component alone leads to a more pronounced performance decline on CSI300.

Table 4: Ablation study of AlphaPareto, where "LLM" denotes the LLM-based alpha pool semantic reasoning, and "MORL" denotes the MAP-based multi-objective reinforcement learning.

Included Components		CSI300		CSI800		Market	
LLM	MORL	Mean	Std	Mean	Std	Mean	Std
		3.69%	(0.66%)	5.07%	(0.63%)	8.44%	(1.09%)
✓		2.37%	(0.68%)	5.04%	(0.42%)	9.68%	(0.83%)
	✓	**3.99%**	**(1.27%)**	5.54%	(1.03%)	9.63%	(0.80%)
✓	✓	3.92%	(0.46%)	**5.70%**	**(0.87%)**	**10.10%**	**(1.01%)**

6 Conclusion

We propose AlphaPareto, an RL framework for formulaic alpha discovery that addresses two key limitations of existing RL-based methods: non-stationarity and single-objective reward design. The method conditions the policy on an LLM-based representation of the evolving alpha pool and optimizes predictive power, temporal stability, perturbation robustness, and diversity simultaneously through Pareto-regularized multi-objective learning. Experiments on Chinese and U.S. data show that AlphaPareto outperforms existing baselines across multiple dimensions, while the scaling and ablation studies further indicate that strong performance does not require the largest available LLMs and that both proposed components play complementary roles.

Several limitations are worth noting. First, performance may depend on the prompt template, which has not been systematically studied. Second, we use a linear mega-alpha combiner, which may exclude useful nonlinear structures that could further improve performance.

Acknowledgements and Disclosures

The authors are grateful to the reviewers, Area Chairs, and Program Committee members for their constructive comments and suggestions, which have significantly improved the paper. Zhoufan Zhu's work was supported by the National Natural Science Foundation of China (nos. 72503203 and 72573142), the Natural Science Foundation of Fujian Province (nos. 2025J08009 and 2025J01036), and the Fundamental Research Funds for the Central Universities (no. 20720251053). To the best of the authors knowledge, no competing interests require disclosure.

References

Chen, B., Ding, H., Shen, N., Guo, T., Huang, J., Liu, L., and Zhang, M. AlphaSAGE: Structure-aware alpha mining via GFlownets for robust exploration. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=zRKF4ln2VE.

Cui, C., Wang, W., Zhang, M., Chen, G., Luo, Z., and Ooi, B. C. Alphaevolve: A learning framework to discover novel alphas in quantitative investment. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, pp. 22082216. ACM, 2021.

Ding, H., Chen, B., Huang, J., Guo, T., Mao, Z., Shao, G., Zou, L., Liu, L., and Zhang, M. Alphaeval: A comprehensive and efficient evaluation framework for formula alpha mining. *arXiv preprint arXiv:2508.13174*, 2025. URL https://arxiv.org/abs/2508.13174.

Kakushadze, Z. 101 Formulaic Alphas. *arXiv preprint arXiv:1601.00991*, March 2016.

Lecarpentier, E. and Rachelson, E. Non-stationary markov decision processes a worst-case approach using model-based reinforcement learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc.

Li, Y., Yang, X., Yang, X., Wang, X., Liu, W., and Bian, J. 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 Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=9VxTXAUH7G.

Lin, X., Chen, Y., Li, Z., and He, K. Stock alpha mining based on genetic algorithm. Technical report, Huatai Securities Research Center, 2019a. URL https://crm.htsc.com.cn/doc/2019/10750101/f75b4b6a-2bdd-4694-b696-4c62528791ea.pdf.

Lin, X., Chen, Y., Li, Z., and He, K. Revisiting stock alpha mining based on genetic algorithm. Technical report, Huatai Securities Research Center, 2019b. URL https://crm.htsc.com.cn/doc/2019/10750101/3f178e66-597a-4639-a34d-45f0558e2bce.pdf.

Luo, H., Ko, H. T., Chen, J., Sun, D., Zhang, Y., and Liu, C. Alphabench: Benchmarking large language models in formulaic alpha factor mining. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=d97Q8r7ZKZ.

Ren, T., Zhou, R., Jiang, J., Liang, J., Wang, Q., and Peng, Y. Riskminer: Discovering formulaic alphas via risk seeking monte carlo tree search. In *Proceedings of the 5th ACM International Conference on AI in Finance*, ICAIF '24, pp. 1–9, Brooklyn, NY, USA, 2024. ACM. doi: 10.1145/3677052.3698613.

Shi, H., Song, W., Zhang, X., Shi, J., Luo, C., Ao, X., Arian, H., and Seco, L. A. Alphaforge: a framework to mine and dynamically combine formulaic alpha factors. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 2025. ISBN 978-1-57735-897-8.

Tang, Z., Chen, Z., Yang, J., Mai, J., Zheng, Y., Wang, K., Chen, J., and Lin, L. Alphaagent: Llm-driven alpha mining with regularized exploration to counteract alpha decay. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 28132822. Association for Computing Machinery, 2025.

Wang, S., Yuan, H., Zhou, L., Ni, L. M., Shum, H.-Y., and Guo, J. Alpha-gpt: Human-ai interactive alpha mining for quantitative investment. *arXiv preprint arXiv:2308.00016*, 2025a. URL `https://arxiv.org/abs/2308.00016`.

Wang, X., Le, Q., Ahmed, A., Diao, E., Zhou, Y., Baracaldo, N., Ding, J., and Anwar, A. MAP: Multi-human-value alignment palette. In *Proceedings of the Thirteenth International Conference on Learning Representations*, 2025b.

Yang, X., Liu, W., Zhou, D., Bian, J., and Liu, T. Qlib: An ai-oriented quantitative investment platform. *arXiv preprint arXiv:2009.11189*, 2020.

Yu, S., Xue, H., Ao, X., Pan, F., He, J., Tu, D., and He, Q. Generating synergistic formulaic alpha collections via reinforcement learning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 54765486. Association for Computing Machinery, 2023.

Yuan, H., Wang, S., and Guo, J. Alpha-gpt 2.0: Human-in-the-loop ai for quantitative investment. *arXiv preprint arXiv:2402.09746*, 2024. URL `https://arxiv.org/abs/2402.09746`.

Zhang, T., Li, Y., Jin, Y., and Li, J. Autoalpha: an efficient hierarchical evolutionary algorithm for mining alpha factors in quantitative investment. *arXic preprint arXiv:2002.08245*, 2020.

Zhu, Z. and Zhu, K. AlphaQCM: Alpha discovery in finance with distributional reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 80463–80479. PMLR, 13–19 Jul 2025.

A Reverse Polish Notation of Formulaic Alpha

In this section, we introduce the reverse Polish notation (RPN) used to represent formulaic alphas. Figure 3 provides an illustrative example. In the figure, a formulaic alpha is encoded as a token sequence, where each feature and operator is represented as a token. Moreover, special tokens *BEG* and *SEP* denote the beginning and end of the expression, respectively.

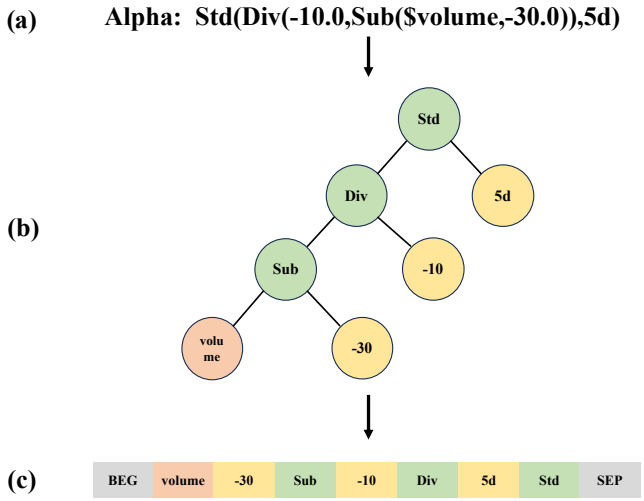


Figure 3: (a) A formulaic alpha discovered by our AlphaPareto method. (b) Its expression tree. (c) Its RPN representation.

Table 5 lists all features and operators used in this paper. Following Kakushadze (2016) and Lin et al. (2019a,b), these operators can be broadly divided into time-series operators and cross-sectional operators. Time-series operators require observations over multiple days, while cross-sectional operators act on single-day data. By combining these two types of operators, formulaic alphas can capture rich nonlinear patterns while remaining interpretable.

Table 5: Description of available features and operators.

Tokens	Description
	Features
Open/High/Low/Close/Vwap/Volume	Opening/high/low/closing/vwap price or volume of stock i at time s.
Constant	Number from $\{-30, -10, -5, -2, -1, -0.5, -0.01, 0.01, 0.5, 1, 2, 5, 10, 30\}$.
Time delta	Integer from $\{10, 20, 30, 40, 50\}$, which is used in the time-series operators.
	Time-series Operators
Ref $(u_{i,s}, d)$	Return the value of $u_{i,s-d}$, where d is the time delta and $u_{i,s}$ is a feature of stock i at time s.
TsRank $(u_{i,s}, d)$	Return the rank of $u_{i,s}$ among $\{u_{i,s}, \ldots, u_{i,s-d}\}$
Mean/*Med*/*Sum*/*Std*/*Var* $(u_{i,s}, d)$	Return the mean/median/sum/standard deviation/variance of $\{u_{i,s}, \ldots, u_{i,s-d}\}$.
Max/*Min* $(u_{i,s}, d)$	Return the maximum/minimum value of $\{u_{i,s}, \ldots, u_{i,s-d}\}$.
WMA/*EMA* $(u_{i,s}, d)$	Return the weighted/exponentially weighted moving average of $\{u_{i,s}, \ldots, u_{i,s-d}\}$.
Cov/*Corr* $(u_{i,s}, z_{i,s}, d)$	Return the covariance/correlation between $u_{i,s}$ and $z_{i,s}$ based on samples $\{(u_{i,s}, z_{i,s}) \ldots, (u_{i,s-d}, z_{i,s-d})\}$, where $u_{i,s}$ and $z_{i,s}$ are features of stock i at time s.
	Cross-sectional Operators
Sign $(u_{i,s})$	Return 1 if the value of $u_{i,s}$ is positive, otherwise return 0.
Abs $(u_{i,s})$	Return the absolute value of $u_{i,s}$.
Log $(u_{i,s})$	Return the logarithmic value of $u_{i,s}$.
Rank $(u_{i,s})$	Return the rank of $u_{i,s}$ among $\{u_{1,s}, \ldots, u_{N,s}\}$.
Add/*Sub*/*Mul*/*Div* $(u_{i,s}, z_{i,s})$	Return the result of adding/subtracting/multiplying/dividing $u_{i,s}$ and $z_{i,s}$.
Greater/*Less* $(u_{i,s}, z_{i,s})$	Return the greater/less value of $u_{i,s}$ and $z_{i,s}$.

B Prompt Template

To construct the semantic representation of the current alpha pool, we feed a structured prompt to a frozen LLM. The prompt is designed to summarize the current pool in terms of factor expressions, weights, information coverage, and potential blind spots. The placeholder "$Current_Alpha_Pool.json$" denotes a JSON file containing the expressions and fitted weights of the current alpha pool $\mathcal{F}_t$.

```
[Role Definition]
You are the Chief Investment Officer of a quantitative hedge fund managing over $5 billion in
↪  assets, with 20+ years of factor research experience. You excel at decoding market
↪  microstructure signals from mathematical expressions, precisely identifying the economic
↪  logic behind factors (e.g., liquidity premiums, behavioral bias arbitrage, volatility
↪  surface arbitrage), and evaluating their risk-return contributions in portfolios.

[Core Task]
Deeply internalize the mathematical structure and portfolio characteristics of the following
↪  factor pool. Your analysis must strictly derive from the JSON datano external assumptions
↪  permitted. Focus on:
1) Factor Style Deconstruction: Map each expression to professional factor taxonomies (e.g.,
↪  Barra style factors/alternative data factors/tail risk factors)
2) Information Topology Analysis: Detect data source blind spots (e.g., over-reliance on raw
↪  prices? missing volume-price interactions? ignoring volatility surfaces?)
3) Risk Heatmap: Identify vulnerabilities exposed by weight configuration (e.g., negative
↪  weight concentration, extreme value sensitivity, expression homogenization risk)
4) Evolution Path Design: Conceive next-generation factors that penetrate current information
↪  blind spots (specify mathematical operations and data types)

[Professional Analysis Framework]
Apply the following quantitative expertise for deep decoding:
- Price Transformation Layer: Identify market state assumptions implied by operations like
↪  Log/EMA/Greater/Less (e.g., Log($price) assumes log-normal distribution)
- Time Scale Layer: Parse window parameters (40d/20d/10d) to determine market cycle
↪  sensitivity (short-term sentiment vs. long-term trends)
- Nonlinearity Layer: Detect strategy logic formed by combinations of Add/Sub/Mul/Div with
↪  conditional functions (Greater/Less) (e.g., volatility breakout strategies)
- Weight Diagnostics: Calculate weight entropy to assess diversification; flag factors with
↪  |weight|>0.2 as risk concentration points
- Data Source Audit: Count frequencies of $high/$low/$close/$open/$vwap/$volume appearances
↪  to evaluate price dimension coverage completeness
```

```
[Critical Example Anchoring]
Below is the typical structure of the factor pool you will analyze (note expression
↪  complexity and weight distribution characteristics):
exprs = [
  'Sub(Mul(EMA(Log(Sum(Add(0.5,Div($high,0.01)),40d)),40d),-10.0),-0.01)',
  'Add(Div(Less(-30.0,$close),Add(30.0,Greater($low,-30.0))),Greater(Log($open),10.0))',
  'Sub(Less(Greater(Add(5.0,Add($close,-30.0)),-30.0),-10.0),-1.0)',
  'Div(2.0,$vwap)',
  'Sub(Div(5.0,$high),5.0)',
  'Add(Div(-10.0,$low),-5.0)',
  'Add(Sub(Abs(Log(Sub(Div(Greater(-5.0,Log($close)),0.01),10.0))),-0.5),-30.0)',
  'Greater(Div(EMA(Div(Greater(Log(Div(0.01,$high)),-30.0),-10.0),20d),$low),-10.0)',
  'Log(Sub(Greater(-30.0,$vwap),Div(Div($volume,$open),-30.0)))',
  'Ref(Log(Greater(Mul(-5.0,Mean(Mul($open,-0.5),10d)),Less(-30.0,$open))),20d)'
],
weights = [
  -0.0732,
  -0.2274,
  -0.0497,
   0.0456,
   0.2256,
  -0.0793,
  -0.1027,
  -0.4176,
  -0.1442,
   0.0939
],
>>> This example reveals critical risk: Factor #8 holds 41.76% absolute weight concentration
↪  with triple-nested nonlinear operations, highly vulnerable during market regime shifts

[Final Output Directive]
Now, channel your professional cognition entirely onto the real-world factor pool described
↪  in the JSON below.
OMIT all intermediate analysis steps, background context, and pleasantries.
Directly based on your analysis of current portfolio blind spots and risks, output 3
↪  construction style recommendations for the "Next Factor," ranked by priority (High to
↪  Low).

Output format must strictly adhere to the following:
Allowed Style List: [Trend Momentum], [Mean Reversion], [Volatility Structure], [Price-Volume
↪  Interaction], [Liquidity Premium], [Market Sentiment], [Tail Risk], [Correlation
↪  Structure], [Higher-Moment Features], [Term Structure], [Asymmetry]

1. [Selected Style Name]: <Specific mathematical construction proposal, including recommended
↪  operators and data sources>  <Expected contribution to filling current blind spots or
↪  reducing correlation>
2. [Selected Style Name]: ...
3. [Selected Style Name]: ...

[Cognitive Activation]
Begin decoding and output recommendations directly:

JSON:
$Current_Alpha_Pool.json$
```

C Algorithm

Algorithm 1 presents the pseduo code for the AlphaPareto method.

D Details About Baseline Methods

This section supplements the summary of baseline methods in Section 5. Unless otherwise noted, all formulaic-alpha discovery baselines are evaluated through the same downstream linear mega-alpha

Algorithm 1 AlphaPareto Training Pipeline

Require: Training data $\{(\boldsymbol{H}_{s-1}, \boldsymbol{y}_s)\}_{s\in\mathcal{D}_{\text{train}}}$, frozen LLM $\text{LLM}(\cdot)$, max pool size P, max formula length $L_{\max}$, value palette $\boldsymbol{c}$, update period $\overline{n}$, and max steps n^*

Initialize alpha pool $\mathcal{F} \leftarrow \emptyset$, replay buffer $\mathcal{B} \leftarrow \emptyset$, $n \leftarrow 0$, policy π_θ, and value network V_ϕ
Collect warm-up reward vectors $\{\boldsymbol{r}_t\}$ by random policy and solve (9) to obtain $\widehat{\boldsymbol{\lambda}}(\boldsymbol{c})$
for each episode **do**
  Construct prompt $d(\mathcal{F})$ and compute pool embedding $\boldsymbol{e} \leftarrow \text{LLM}(d(\mathcal{F}))$
  $\boldsymbol{z}_0 \leftarrow$ *BEG*
  for $t = 0, 1, \ldots, L_{\max} - 1$ **do**
    $\boldsymbol{h}_t \leftarrow \text{LSTM}(\boldsymbol{z}_t)$
    $\boldsymbol{u}_t \leftarrow \text{FFN}_h(\boldsymbol{h}_t) \odot \text{FFN}_e(\boldsymbol{e})$
    $\boldsymbol{q}_t \leftarrow \text{FFN}_u(\boldsymbol{u}_t)$
    Sample $a_t \sim \pi_\theta(\cdot \mid \boldsymbol{q}_t)$ with invalid action masking
    $\boldsymbol{z}_{t+1} \leftarrow [\boldsymbol{z}_t,\, a_t]$
    if $a_t =$ *SEP* **or** $|\boldsymbol{z}_{t+1}| = L_{\max}$ **then**
      Parse $f_{\text{new}} \leftarrow \text{parse}(\boldsymbol{z}_{t+1})$
      if f_{new} is valid **then**
        Form $\overline{\mathcal{F}} \leftarrow \mathcal{F} \cup \{f_{\text{new}}\}$ and refit the linear model on $\overline{\mathcal{F}}$
        Update $\mathcal{F}$ and $\widehat{\boldsymbol{\beta}}$ by keeping at most P alphas according to their fitted contributions
        Compute $\boldsymbol{r}_t \leftarrow (\text{IC}_t, \text{RRE}_t, \text{PFS}_t, \text{DH}_t)^\top$
      else
        $\boldsymbol{r}_t \leftarrow -\mathbf{1}$
      end if
      $r_t^* \leftarrow \widehat{\boldsymbol{\lambda}}(\boldsymbol{c})^\top \boldsymbol{r}_t$
      Store the terminal transition in $\mathcal{B}$
      break
    else
      Store the transition in $\mathcal{B}$ with $\boldsymbol{r}_t \leftarrow \mathbf{0}$ and $r_t^* \leftarrow 0$
    end if
    if $n \mod \overline{n} = 0$ **then**
      Update θ and ϕ via PPO using a batch of transitions in $\mathcal{B}$
      Re-solve (9) on recent reward vectors to update $\widehat{\boldsymbol{\lambda}}(\boldsymbol{c})$
    end if
    $n \leftarrow n + 1$
  end for
  if $n > n^*$ **then**
    break
  end if
end for
return $\mathcal{F}$ and $\widehat{\boldsymbol{\beta}}$

construction, so the comparison focuses on the quality of the selected alpha pool rather than on the complexity of the combiner.

Alpha158 serves as a handcrafted formulaic benchmark. We fix the candidate pool to the 158 predefined alphas provided by Qlib (Yang et al., 2020) and fit a linear model to combine them into the final mega-alpha. In contrast, MLP is a non-formulaic end-to-end predictor that maps input directly to future returns in a data-driven approach. This pair of baselines allows us to compare AlphaPareto with both a standard handcrafted alpha pool and a representative black-box predictor.

The GP baseline follows the conventional genetic-programming paradigm. Candidate expressions are evolved one by one with IC as the fitness measure. We then construct the final alpha pool from the top-P discovered expressions. This baseline represents the classical formulaic-alpha search pipeline that does not directly optimize pool-level synergy during generation.

AlphaAgent and R&D-Agent-Quant are representative LLM-based alpha discovery methods. Specifically, AlphaAgent uses an LLM to generate market hypotheses and candidate formulas directly, whereas R&D-Agent-Quant separates this workflow into a research stage for hypothesis generation

and a development stage for implementation and backtesting. In both baselines, the LLM acts as a generator of candidate alphas rather than as a frozen semantic encoder inside an RL loop. In particular, we make several adjustments to these baselines to ensure a fair comparison with AlphaPareto. For AlphaAgent, we modify the prompt so that each run generates 10 candidate alphas per iteration over 5 iterations. For R&D-Agent-Quant, we make three changes to the default alpha generation loop: (i) we fix the number of factors per iteration at exactly 10, instead of the default range of 1–5, by modifying both the prompt template and the proposal module; (ii) we increase the number of outer loops from 5 to 10; and (iii) we replace the default free-form LaTeX expression format with the same structured expression syntax used in our method. This syntax covers the full operator set shared across all baselines and allows the generated factors to be parsed and evaluated directly, without requiring LLM-based reverse translation. All other implementation details of AlphaAgent and R&D-Agent-Quant follow their original repository defaults.

AlphaGen, AlphaForge and AlphaQCM are the closest RL-based competitors. AlphaGen formulates formulaic alpha discovery as an MDP and uses PPO to search for a synergistic alpha pool. AlphaForge modifies the static combination model to a dynamic one. AlphaQCM extends this line by using distributional RL and a variance-based exploration bonus to mitigate the non-stationarity highlighted in our related-work discussion. These methods optimize a scalar reward tied primarily to predictive performance and do not explicitly encode the semantics of the current alpha pool. Their comparison with AlphaPareto therefore isolates the value of pool-aware state augmentation and Pareto-regularized multi-objective optimization.

E Hyperparameters

E.1 Network Hyperparameters

As in Yu et al. (2023), we use a two-layer LSTM for $\mathrm{LSTM}(\cdot)$ in (4), with hidden dimension 128 and dropout rate 0.1. The LLM embedding dimension d_e depends on the chosen model. For example, $d_e = 2560$ for theQwen-Embedding-4B model; see Table 6 for more details. The hidden dimension of FFN_e and FFN_h is set to 256, and FFN_u further maps the fused representation $\boldsymbol{u}_t$ to a 128-dimensional embedding. See (5) for details.

Table 6: Embedding dimension of LLM candidates.

Chosen model	Embedding dimension d_e
DeepSeek-1.5B	1536
Qwen-Embedding-4B	2560
DeepSeek-7B	3584
DeepSeek-14B	5120
DeepSeek-32B	5120

E.2 RL Hyperparameters

Besides the network-related hyperparameters, some additional hyperparameters that our RL algorithm inherits are listed in Table 7.

F Paired Significance Tests

In addition to comparing means and standard deviations, we further compare AlphaPareto with each baseline in Table 1 using paired t-tests across five matched random seeds, where a positive t-statistic favors AlphaPareto.

Table 8 reports the t-statistics and their p-values. At 10% significance level, AlphaPareto achieves a significantly higher mean of IC than every baseline on CSI300 and CSI800, and than all baselines except AlphaQCM on Market. The comparison with AlphaQCM on Market has a positive estimated difference but is not statistically significant ($t = 0.65$, $p = 27.60\%$). Overall, the differences favor AlphaPareto and are significant in most comparisons, providing evidence that the predictive gains extend beyond seed-level noise under these tests.

Table 7: Additional hyperparameters.

Hyperparameter	Values
Instruments	CSI300 / CSI800 / Market
P (Alpha Pool Size)	10, 20
Leading days	20
Batch size	128
Optimizer	Adam
Learning rate	5×10^{-4}
Max steps / tolerance	10,000 / 500
ℓ_1 regularization for combination model α	5×10^{-3}
Lambda solver lr / steps / retries	0.01 / 500 / 1
Metrics window size / step	252 / 5
Max expression length	15
$c_{\mathrm{IC},CSI300}$	0.090
$c_{\mathrm{IC},CSI800}$	0.085
$c_{\mathrm{IC},Market}$	0.250
c_{RRE}	0.95
c_{PFS}	0.93
c_{DH}	0.80
Total step	250,000 ($P = 10$), 300,000 ($P = 20$),

Table 8: Paired t-tests of AlphaPareto against each baseline across five matched seeds. p-values are shown in parentheses as percentages, rounded to two decimal places.

	CSI300		CSI800		Market	
Method	t	p-value	t	p-value	t	p-value
Alpha158	4.55	(0.52%)	2.40	(3.73%)	13.39	(0.01%)
MLP	2.73	(2.62%)	7.05	(0.11%)	8.00	(0.07%)
GP	16.11	(0.00%)	9.58	(0.03%)	10.87	(0.02%)
AlphaAgent	55.88	(0.00%)	25.41	(0.00%)	44.84	(0.00%)
R&D-Agent-Quant	6.59	(0.14%)	34.67	(0.00%)	116.93	(0.00%)
AlphaGen	1.69	(8.34%)	5.08	(0.35%)	14.96	(0.01%)
AlphaQCM	14.58	(0.01%)	3.67	(1.07%)	0.65	(27.60%)
AlphaForge	20.95	(0.00%)	2.74	(2.59%)	9.81	(0.03%)
AlphaPareto	-	-	-	-	-	-

G Additional Details and Experiments about Portfolio Backtesting

We further evaluate the out-of-sample economic performance of AlphaPareto through portfolio backtesting. Following the trading strategy in Yu et al. (2023) and Shi et al. (2025), we construct a long-only portfolio by selecting the top 50 stocks ranked by the mega-alpha at each rebalance date. At each rebalance, we partially rotate the existing holdings by trading up to 5 eligible stocks, exclude stocks subject to price-limit constraints, and execute trades at the close. Transaction costs are set to 15 basis points per side, with a minimum fee of 5 CNY. The backtest is conducted on the full market universe with monthly rebalancing, consistent with the 20-day return prediction target. The initial capital is 100 million CNY, and the Shanghai Composite Index (SH.000001) is used as the benchmark.

The cumulative returns in Figure 4 provide a consistent visual comparison: AlphaPareto delivers the strongest cumulative return over the test period while maintaining a controlled drawdown profile.

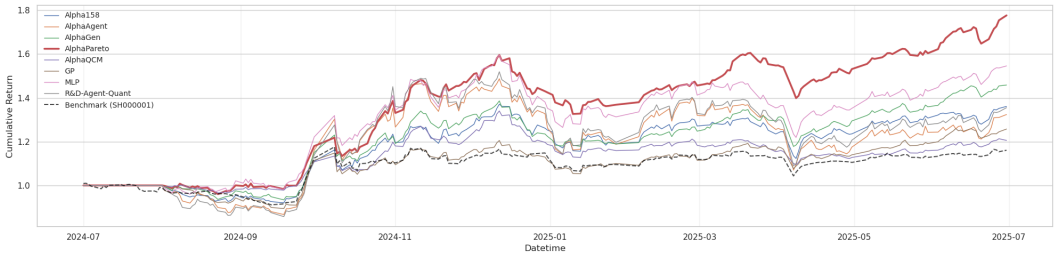


Figure 4: Cumulative returns of monthly portfolios constructed by different methods.

H Impact of MORL method

In this paper, we choose MAP, since its value palette specifies interpretable minimum target levels for predictive power, temporal stability, perturbation robustness, and diversity, while the nonnegative scalarization weights are derived from the observed reward distribution. This does not remove subjective preferences, but expresses them as thresholds in the objectives' original units rather than as a fixed weight vector whose meaning changes with objective scale.

While the MORL-only ablation in Table 4 shows the benefit of the vector reward design with MAP-based optimization, it does not establish the superiority of MAP over other MORL optimizers. To assess whether MAP's adaptive weighting provides benefits beyond merely combining the four objectives, we consider an equal-weight linear-scalarization baseline that replaces the MAP-derived weights with fixed weights while keeping all other components unchanged. Its scalar reward is the simple average of IC, RRE, PFS, and DH:

$$r_t^{\text{equal}} = \frac{1}{4}\left(\text{IC}_t + \text{RRE}_t + \text{PFS}_t + \text{DH}_t\right).$$

This baseline represents the simplest form of multi-objective scalarization.

On CSI800, the equal-weight baseline achieves a mean IC of 4.92% with a standard deviation of 0.52%, compared with 5.70% and 0.87%, respectively, for MAP-based AlphaPareto in Table 4. Thus, MAP outperforms this simple alternative in mean IC, although its standard deviation is higher. This comparison supports the benefit of MAP's adaptive weighting beyond merely combining the four objectives. Although these results establish MAP as a strong, well-motivated choice for alpha discovery, more sophisticated MORL optimizers may yield further gains.

I Random-Noise Control

To examine whether the LLM encoding acts merely as regularization noise, we replace the LLM-based pool embedding with pool-independent random noise matched in dimension and empirical norm, while keeping all other settings fixed.

Table 9 compares this control with AlphaPareto and the MORL-only variant, which uses no pool representation. On CSI800, the random-noise control achieves a mean out-of-sample IC of 4.22% with a standard deviation of 1.62%, compared with 5.70% and 0.87% for AlphaPareto and 5.54% and 1.03% for the variant without a pool representation. Replacing the LLM embedding with matched noise thus lowers the mean IC and increases its variability. These results suggest that the gain cannot be explained by matched pool-independent noise alone and are consistent with useful pool-dependent semantic information.

J Empirical Results on the S&P 500 Universe

To assess whether AlphaPareto remains effective beyond Chinese A-shares, we conduct an additional evaluation on U.S. equities using the S&P 500 stock universe. We change only the stock universe and retain the other experimental settings described in Section 5.

Table 10 reports the out-of-sample predictive performances. On S&P 500, AlphaPareto achieves the highest mean IC among the evaluated methods and the lowest standard deviation among methods

Table 9: Out-of-sample ICs on CSI800 with different pool representations.

Pool representation	CSI800	
	Mean	Std
None (MORL only)	5.54%	(1.03%)
Matched noise	4.22%	(1.62%)
LLM embedding (AlphaPareto)	**5.70%**	**(0.87%)**

with repeated runs. Compared with the strongest baseline, AlphaQCM, it improves mean IC from 7.18% to 7.65%, a gain of 0.47 percentage points, while reducing the standard deviation from 0.84% to 0.13%. These results provide additional empirical support for the applicability of AlphaPareto beyond Chinese A-shares under a different market structure.

Table 10: Out-of-sample ICs of different methods on the S&P 500 universe.

Method	S&P 500	
	Mean	Std
Alpha158	3.92%	-
MLP	5.43%	(0.31%)
AlphaAgent	-0.01%	(0.56%)
R&D-Agent-Quant	1.29%	(0.49%)
AlphaGen	7.03%	(0.30%)
AlphaQCM	7.18%	(0.84%)
AlphaForge	1.33%	(2.97%)
AlphaPareto	**7.65%**	**(0.13%)**

K Computational Resources

All experiments were conducted on internal GPU servers with 66 Intel(R) Xeon(R) Platinum 8470Q vCPUs, 330 GB RAM, and one RTX PRO 6000 GPU with 96 GB video memory. Each AlphaPareto training run used one GPU and 64 CPU workers. Table 11 summarizes the compute required to reproduce the reported experiments. All stochastic experiments were repeated over five random seeds, as described in the main text.

The full research project also involved preliminary runs for debugging, prompt design, component designs, etc. These exploratory runs are not included in the main experimental tables. They required approximately 4 additional GPU-hours. The final reported results were produced using the settings in Appendix E.

Table 11: Computational resources used for the reported experiments.

Experiment	Runs	Hardware per run	Time per run	Total compute
Main comparison	195	GPU	8h	1560h
LLM scaling study	150	GPU	8h	1200h
Ablation study	360	GPU	7h	2520h

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 the two open problems in formulaic alpha discovery, the proposed AlphaPareto framework, and its contribution and scope. These points are further developed in the Sections 4 and 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper includes a discussion about limitations in Section 6.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

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 present new theorems, lemmas, or formal proofs. It relies on theoretical results from prior work, especially the MAP framework, and cites those results in the Section 4.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

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: The paper provides the information needed to reproduce the main experimental results. In addition, the code and data required for reproduction are included in the supplementary material.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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 used to reproduce the main experimental results are provided in the supplementary material, together with the necessary instructions for reproduction.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

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 and Appendix E present all necessary details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

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 paper reports means and standard deviations across five random seeds for stochastic experiments and paired t-tests in Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

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: The compute resources, including hardware type, per-run runtime, total compute, and preliminary exploratory compute, are reported in Appendix K.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

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 historical market data and algorithmic evaluations and does not involve human subjects, intervention, or data collection from individuals. To the best of our knowledge, the work conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [No]

Justification: The current manuscript does not include a dedicated discussion of positive and negative societal impacts. It focuses on technical performance and does not yet discuss possible downstream financial impact.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

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: This paper does not release a high-risk generative model, a scraped dataset, or a similar asset that would require special safeguards. The LLMs are used as existing components within the method rather than released as new public models.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

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: The paper properly cites the prior methods, models, and data sources used in our experiments. The market data are accessed through Qlib, and the corresponding assets are used with appropriate attribution and in accordance with their terms of use.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

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 for implementing AlphaPareto is provided, together with the accompanying documentation needed to understand and reproduce the method.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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: The paper does not involve crowdsourcing experiments or research with human participants. All evaluations are conducted on market data and computational baselines.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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: The paper does not involve human subjects or crowdsourcing, so IRB approval or an equivalent review is not applicable. No participant risks or consent procedures arise in this study.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

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 part of the proposed method rather than a writing aid; see Section 4.1. The paper explains that the LLM is used as a frozen semantic encoder for the current alpha pool.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.